Control System Design Using Evolutionary Algorithms for Autonomous Shipboard Recovery of Unmanned Aerial Vehicles

by

Sergey Khantsis

BEng (Aero) 1st Class Honours

Submitted to the

School of Aerospace, Mechanical & Manufacturing Engineering

in fulfilment of the requirements for the degree of

Doctor of Philosophy

at the

Royal Melbourne Institute of Technology

August 2006

Declaration

I, Sergey Khantsis, declare that:

None of this work has been submitted previously, in whole or in part, to qualify for any other academic award. Except where due acknowledgement has been made, the work is that of the candidate alone. The content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program. Any editorial work, paid or unpaid, carried out by a third party is acknowledged.

Signed:

Acknowledgements

During the course of this research, several people have supplied assistance, and I will use this opportunity to thank them.

First I wish to thank my supervisors, Anna Bourmistrova and Cees Bil, for their support and advice in shaping the direction and concepts of this thesis. Anna's guidance in preparation of this thesis and her patience are of extreme value for me.

I am grateful to Kim Healy for her excellent and prompt support in proofreading of the final version of the text. I must say that any remaining errors are my sole responsibility.

My deepest thanks to my wife Alice for her support and love during this program.

This work would be impossible without the support from the International Postgraduate Research Scholarship program, which provided funding for this research. I would especially like to thank Elizabeth Toy for her help in administrative questions.

I wish to express my gratitude to my former supervisor, Alexander Efremov. Without his support and encouragement, this work would never have started.

Finally I thank the examiners for their time and patience in reviewing this thesis.

Contents

DECLARATION ..... **II**

ACKNOWLEDGEMENTS ..... **III**

CONTENTS ..... **IV**

GLOSSARY ..... **X**

SUMMARY **1**

CHAPTER 1. INTRODUCTION **3**

1.1 Unmanned Aerial Vehicles and shipboard recovery 3

1.2 Evolutionary Algorithms 5

1.3 The contribution of this thesis 8

1.4 Thesis outline 9

CHAPTER 2. UAV RECOVERY PROBLEM **11**

2.1 Objectives of UAV shipboard recovery 11

2.2 The environment 14

2.2.1 Atmosphere 14

2.2.1.1 Wind 14

2.2.1.2 Turbulence and gusts 15

2.2.1.3 Ship airwake 16

2.2.2 Sea and ship motion 17

2.3 Recovery methods review 17

2.3.1 Conventional runway landing 18

2.3.2 Recovery net 19

2.3.3 Parachute systems 21

2.3.3.1 Uncontrollable parachute systems 22

2.3.3.2 Gliding parachutes 24

2.3.3.3 Parasail recovery 26

2.3.3.4 Dynamic parachute deployment 27

2.3.4 Deep stall landing 28

2.3.5 Convertible airframe 31

2.3.6 In-flight arresting devices 33

2.3.6.1 Skyhook™ 34

2.3.6.2 Cable hook 35
2.4 Cable Hook recovery model 37
2.4.1 Recovery procedure 38
2.4.2 Approach corridor 40
2.4.3 Recovery equipment 42
2.4.3.1 Length of the cable and height of the arresting wire installation 42
2.4.3.2 Cable strength 44
2.4.3.3 The hook 45
2.4.3.4 Local positioning system 46
2.5 Concluding remarks 53
CHAPTER 3. SIMULATION MODELS 55
3.1 Aircraft model 55
3.1.1 The equations of motion 56
3.1.1.1 Axes systems 57
3.1.1.2 Translational motion 60
3.1.1.3 Angular motion 61
3.1.2 The force model 63
3.1.2.1 The Ariel UAV 64
3.1.2.2 The aerodynamic model 65
3.1.2.3 The propulsion model 67
3.1.2.4 Sensor and actuator models 68
3.2 Atmospheric model 70
3.2.1 Steady wind model 71
3.2.2 Turbulence model 71
3.2.3 Gust model 74
3.2.4 The airwake model 75
3.2.4.1 Discussion 75
3.2.4.2 The airwake model design 76
3.3 Ship model 81
3.3.1 Dynamic model 81
3.3.2 Physical model 83
3.4 Shipboard recovery gear 86
3.4.1 Recovery boom oscillations 87
3.5 Cable model 93
3.5.1 Model design 94

3.5.2 Kinematic equations 95
3.5.2.1 Quaternions and 3D rotation 97
3.5.3 Force equations 99
3.5.3.1 Gravity forces 100
3.5.3.2 Inertial forces 100
3.5.3.3 Aerodynamic forces 101
3.5.3.4 Tension forces 102
3.5.4 Static model 103
3.6 Concluding remarks 108
CHAPTER 4. EVOLUTIONARY ALGORITHMS 110
4.1 Introduction to Evolutionary Algorithms 111
4.2 Evolutionary Algorithms inside 115
4.2.1 Fitness evaluation 115
4.2.1.1 Multi-objective problems 117
4.2.2 Genome representation 118
4.2.3 Selection 123
4.2.3.1 Deterministic selection 127
4.2.3.2 Fitness proportional selection and fitness scaling 127
4.2.3.3 Ranking selection 130
4.2.3.4 Tournament selection 131
4.2.4 Recombination 132
4.2.4.1 Alphabet-based chromosome recombination 132
4.2.4.2 Real-value recombination 133
4.2.5 Mutation 136
4.2.5.1 Bit string mutation 136
4.2.5.2 Real-value mutation 139
4.2.6 Mutation vs. Recombination 139
4.2.7 Measuring performance 141
4.2.8 The problem of convergence and genetic drift 142
4.2.9 Schema theory 143
4.3 Genetic Algorithms 147
4.4 Evolutionary Strategies and Evolutionary Programming 149
4.4.1 Self-adaptation 150
4.4.1.1 Mutative strategy parameter control 151
4.4.1.2 Derandomised self-adaptation 153

4.5 Genetic Programming 158
4.5.1 Genome representation in GP and S-expressions 159
4.5.1.1 Function set and terminal set 163
4.5.2 Initial population 163
4.5.3 Genetic operators 164
4.5.3.1 Crossover 164
4.5.3.2 Other genetic operators 166
4.5.4 Convergence in GP 167
4.5.5 Bloat 168
4.6 Summary 169

CHAPTER 5. CONTROLLER DESIGN 171

5.1 Aircraft flight control 171
5.1.1 Types of feedback control 174
5.1.1.1 On-off control 174
5.1.1.2 PID control 175
5.1.1.3 Linear optimal control 176
5.1.1.4 Robust modern control 178
5.1.1.5 Nonlinear control 181
5.1.1.6 Intelligent control 182
5.1.2 Flight control for the UAV recovery task 185
5.1.2.1 Traditional landing 185
5.1.2.2 UAV control for shipboard recovery 190
5.2 UAV controller structure 194
5.2.1 Guidance controller 196
5.2.1.1 Outputs 196
5.2.1.2 Inputs 200
5.2.1.3 Terminal phase of approach 209
5.2.2 Flight controller 210
5.3 Evolutionary design 211
5.3.1 Discussion 212
5.3.2 Representation of the control laws 217
5.3.2.1 Representation of input signals 218
5.3.2.2 Representation of control equations and the structure mutation 219
5.3.3 Simulation environment 224
5.3.4 Fitness evaluation 227

5.3.5 Evolutionary Design algorithm outline 230
5.4 Concluding remarks 233
CHAPTER 6. CONTROLLER SYNTHESIS AND TESTING 234
6.1 Controller synthesis 234
6.1.1 Step 1: PID autothrottle 235
6.1.2 Step 2: Longitudinal control 239
6.1.3 Step 3: Lateral control 242
6.1.4 Step 4: Stall prevention mechanism 244
6.1.5 Step 5: Guidance 248
6.1.6 Possible hardware implementation issues 257
6.2 Controller testing 259
6.2.1 Simulation model 260
6.2.2 Robustness tests 261
6.2.2.1 Single perturbation tests 261
6.2.2.2 Simultaneous multiple perturbation tests 279
6.2.3 Performance tests 281
6.2.3.1 Default recovery boom position 283
6.2.3.2 Pivoting recovery boom 290
6.2.3.3 Side recovery boom 292
6.2.4 Operating envelope 294
6.3 Concluding remarks 299
CHAPTER 7. CONCLUSIONS 304
7.1 Discussion 305
7.1.1 UAV recovery 305
7.1.2 Evolutionary Design methodology 307
7.2 Limitations of this research 308
7.2.1 Limitations of the Evolutionary Design 308
7.2.2 Limitations of the developed recovery controller 309
7.3 Further work 311
7.3.1 Research opportunities on the Evolutionary Design 311
7.3.2 Application of the recovery controller design methodology to other UAVs 312
7.3.3 Physical implementation 312
7.3.4 Other issues arising from this research 313
7.4 Concluding remarks 313

APPENDIX A. NOMENCLATURE 314

APPENDIX B. BASIC ARIEL UAV CHARACTERISTICS 317

LIST OF PUBLICATIONS 318

BIBLIOGRAPHY 319

Glossary

ACLS	Automatic (*or* All-weather) Carrier Landing System
AMSL	Above Mean Sea Level
c.g.	Centre of gravity
c.m.	Centre of mass
DCM	Direction Cosine Matrix
DGPS	Differential GPS
EA	Evolutionary Algorithm
ED	Evolutionary Design
ES	Evolutionary Strategy
FAI	Fédération Aéronautique Internationale (International Air Sports Federation)
FLOLS	Fresnel Lens Optical Landing System
GA	Genetic Algorithm
GP	Genetic Programming
GPS	Global Positioning System
HMAS	Her Majesty Australian Ship
ICLS	Instrument Carrier Landing System
ILS	Instrument Landing System
INS	Inertial Navigation System
MIMO	Multi-Input Multi-Output
MSL	Mean Sea Level
MTOW	Maximum Take-Off Weight
NPC	Normalised Performance Cost (Chapter 6)
PC	Personal Computer
	or Performance Cost (Chapter 6)
PID	Proportional-Integral-Derivative
PN	Proportional Navigation
PPN	Pure Proportional Navigation
PSD	Power Spectral Density
RAST	Recovery Assist, Secure and Transit
RPV	Remotely Piloted Vehicle
SISO	Single Input Single Output
STOL	Short Take-Off and Landing

SWH	Significant Wave Height
TLS	Transponder Landing System
TPN	True Proportional Navigation
TUAV	Tactical Unmanned Aerial Vehicle
UAV	Unmanned Aerial Vehicle
UCARS	UAV Common Automatic Recovery System (Sierra Nevada Corporation)
USN	U.S. Navy
VTOL	Vertical Take-Off and Landing

Summary

The capability of autonomous operation of ship based Unmanned Aerial Vehicles (UAVs) in extreme sea conditions would greatly extend the usefulness of these aircraft for both military and civilian maritime purposes. The factors that hamper such operations are connected primarily with launch and recovery stages of flight. Due to very limited deck space available on ships, maritime operation generally requires the aircraft to either have Vertical Take-Off and Landing (VTOL) capability or some form of launch and recovery assistance. The advantages of fixed-wing aircraft over VTOL aircraft in terms of flight speed, range and endurance are well known and can be of great benefit to various missions. For this reason, operation of traditional fixed-wing UAVs with assisted launch and recovery is often preferable.

Shipboard UAV recovery has always been a challenging task. Even in benign conditions, deck space is usually insufficient to provide a landing run for a fixed-wing aircraft. A number of recovery techniques have been developed to solve this problem. They include parachute assisted recovery, net capture, deep stall and many others. However, very few of them achieved operational status. Any recovery method has serious disadvantages and as a rule is suitable only for a narrow class of UAVs. In this work, current methods of recovery are analysed and the problems associated with recovery in adverse weather conditions are identified. The major problem is presented by the random oscillatory motion of the ship. In a rough sea, both the angular and translational displacements of the ship body are so large that guidance to any specified point on the deck becomes extremely challenging. Other problems identified include large amount of turbulence and possibly very unfavourable wind direction which arises from undesirability for the ship to change its course.

Based on this analysis, a novel recovery method is proposed. This method, named *Cable Hook Recovery*, is intended to recover small to medium-size fixed-wing UAVs on frigate-size vessels. It is expected to have greater operational capabilities than the *Recovery Net* technique, which is currently the most widely employed method of recovery for similar class of UAVs, potentially providing safe recovery even in very rough sea and allowing the choice of approach directions.

The recovery method is supported by the development of a UAV controller that realises the most demanding stage of recovery, the final approach, which starts several hundred metres in the vicinity of the ship and continues until the moment of capture. The controller provides both flight control and guidance strategy that allow autonomous recovery of a fixed-wing UAV. It is found that the controller is capable of guiding the UAV to the recovery point with a specified accuracy even in very difficult conditions, in which the UAVs currently in

service are not usually operated.

The development process involves extensive use of specially tailored *Evolutionary Algorithms (EAs)* and represents the major contribution of this work. EAs are heuristic techniques being developed since the 1960s. Due to extremely rapid growth of the available computing power in the 90s, EAs gained wide popularity in the engineering field. However, the great majority of their applications is concerned with various types of nonlinear optimisation. The ability of EAs to *solve* problems is still underused. This research demonstrates how EAs can be used in a full cycle synthesis of a controller.

The *Evolutionary Design* algorithm developed in this work combines the power of *Evolutionary Strategies* and *Genetic Programming*, enabling automatic evolution of both the structure and parameters of the controller. The controller is evolved using the fully coupled nonlinear six-degree-of-freedom UAV model, making linearisation and trimming of the model unnecessary. The simulation environment also includes the atmosphere model, the ship model and the cable model. These models are continuously used within an integrated simulation environment during the evolution to produce a highly optimised controller. The developed algorithm is applied to both flight control and guidance problems with several variations, including optimisation of a routine PID controller.

It is demonstrated that Evolutionary Design is capable of generating human-competitive solutions, by the example of an automatically solved guidance task. The algorithm is faced with a problem definition and has no clues how to solve it. Nevertheless, it ends up with an optimised solution similar to those used in many today's guidance applications.

The design methodology is given with an example based around recovery of a small UAV on a naval frigate class vessel. The methodology produced a controller that has been tested comprehensively in a nonlinear simulation environment and has been found to allow the aircraft to be recovered in the presence of both large external disturbances and uncertainty in the simulation models.

The comprehensive testing procedure used identifies both the performance and robustness properties of the generated controller. The controller is found to have both good performance and robustness in nearly all reasonable situations. Several parameters of the UAV and other systems involved have been identified as having a direct effect on the aircraft recovery performance, while others are shown to have little effect. Finally, an operating envelope for successful recovery is produced, indicating the chances of successful recovery in various operating conditions.

The research is concluded with a discussion of the various techniques used. The main limitations of the research are highlighted and a range of methods of overcoming these limitations is suggested.

Chapter 1. Introduction

1.1 Unmanned Aerial Vehicles and shipboard recovery

Over the past two decades, the interest in Unmanned Aerial Vehicles (UAVs) from both the aerospace community and the general public has grown immensely. The use of UAVs is becoming a well accepted technique not only for the military applications but also in the civilian arena. The advent of modern intelligent control systems and navigational aids, such as the Global Positioning System (GPS), makes the usage of UAVs increasingly efficient and affordable to a wide range of customers.

Typical applications of UAVs range from such traditional military missions as battlefield surveillance, reconnaissance and target acquisition to atmospheric research, weather observation, coastal and maritime surveillance, agricultural and geological surveying, telecommunication signals retranslation, and search and rescue missions. The fact that these vehicles are unmanned make them particularly suitable for use in inhospitable or hazardous environments. In general, UAVs have the potential to complement traditional manned surveillance and monitoring operations or, for some missions, can be a cost effective alternative.

The critical parts of a UAV mission are the launch and recovery phases. Although some UAVs can be conventionally operated from runways, the ability of UAVs to be operated from confined areas, such as remote land sites, ships and oil rigs, greatly increase their practical applications. Such operations generally require the aircraft to either have Vertical Take-Off and Landing (VTOL) capability or some form of launch and recovery assistance. A VTOL capability is beneficial for aircraft in terms of launch and recovery procedures. However, the inclusion of this capability within the aircraft configuration can reduce the capability of the aircraft to perform certain missions. In particular, VTOL aircraft invariably have a reduced speed, range and endurance as compared to their non-VTOL counterparts of the same class. For this reason, operation of traditional fixed-wing UAVs with assisted launch and recovery is preferable in many cases.

External launchers can be used to assist the aircraft to gain a necessary speed on launch by rapid acceleration of the aircraft. These launchers may take two common forms: catapult launchers and rocket boosters. Catapult launch involves rapid acceleration of the UAV as it moves along a fixed launch ramp. Rocket assisted launch employs the attachment of an external rocket engine (or several engines) to the aircraft to provide initial acceleration and lift, which may later be jettisoned. The aspects of launch and control for maritime UAV operations are comprehensively covered in [47].

Unlike launch, the ways of UAV recovery are numerous. Probably the most widely used method, apart from runway landing, is the parachute assisted recovery. It employs an onboard parachute which is deployed over the intended area of landing and the aircraft descends with a safe rate. This technique is also used on many conventionally landing UAVs as an emergency method. Its main advantage is simplicity. Unfortunately, parachute recovery can hardly be used when the landing area is extremely limited (for example, a ship's deck) and in the presence of high winds and strong gusts. More elaborated versions employ various forms of controllable and gliding parachutes. They allow a limited degree of flight control; however, they lose the advantage of simplicity and are still sensitive to large wind disturbances.

The first practicable and widely used solution for shipboard recovery of a fixed-wing UAV was capturing by an elastic net. This method has been employed for the USN *RQ-2 Pioneer* UAV, first deployed in 1986 aboard the battleship USS *Iowa*. The recovery net is usually stretched above the stern of the ship and the aircraft is flown directly into the net. After successful capture of the UAV, the net is lowered onto the deck and the UAV is removed from the net manually. A serious disadvantage of this method is that it is quite stressful for the aircraft, with the recovery net often ripping off the propeller, antennas and even wing upon landing. Nevertheless, due to simplicity of the recovery gear and reasonably simple guidance and flight control during the approach, this technique is still very popular for maritime operations.

Other methods are extensively being developed and tested. They include such techniques as deep stall and perched recovery and various forms of convertible airframes. However, these methods often imply very specific requirements to the UAV design and high complexity of control. Therefore, a need is identified to develop an effective recovery method which enables the UAV to be recovered on a confined area in the presence of large disturbances and which does not have deficiencies of the current methods.

Solving this problem is an enormous engineering work. It can be separated into two distinct areas: design of the recovery technique itself and development of a UAV controller that provides flight control and guidance of the vehicle in accordance with the requirements of this technique.

The controller should provide autonomous guidance and control during the whole recovery process (or its airborne stage). Fully autonomous operation of UAVs plays an important role in expanding the area of their applications and increasing the cost efficiency of their missions. Many of the UAVs in current or past service are partially or fully Remotely Piloted Vehicles (RPV), with a human pilot required in some or all stages of flight. This not only entails high costs associated with the presence of a trained ground-based pilot, but also limits

reliability of the system and increases the attrition rate. For example, the primary source of operational losses of the aforementioned *Pioneer* UAVs during the tests and field operations were associated with improper recovery conducted by a remote human pilot [7, 152].

There exists a number of control design techniques applicable to the area of guidance and flight control. They all have different features and limitations, producing the controllers with different characteristics. It is expected that linear control techniques will not be sufficient for control of the aircraft through the whole recovery stage due to large atmospheric disturbances, ship motion and aircraft constraints. Landing and recovery usually imply a flight with a relatively low airspeed and high angles of attack, where effective control is complicated and the dynamic behaviour of the aircraft is sufficiently nonlinear. At the same time, ship motion may require the UAV to manoeuvre actively in these conditions. Another characteristics of the recovery process is large amount of uncertainties within the system. They include not only the uncertainties present in the environment and in the aircraft, but also those which arise from the absence of exact specifications of the parameters of the recovery process and the systems involved. Under these conditions when so many factors remain uncertain during the process of development, even the very approach to the control problem is unclear. It is desirable that the controller design methodology allow to produce an optimally suitable controller even when faced with such uncertainties.

The main aim of this thesis can be therefore summarised as:

To propose an effective recovery method suitable for recovery of a small to medium size fixed-wing UAV onboard a ship in a rough sea and in the presence of large atmospheric disturbances, and

To develop a controller design methodology that produces a controller capable of autonomous guidance and control of the UAV during the recovery process, considering the requirements set by the recovery method.

An extremely flexible and capable tool that can be used to develop such a methodology is presented by the Evolutionary Algorithms.

1.2 Evolutionary Algorithms

The Evolutionary Algorithms (EAs) is a group of stochastic search methods which combine such important characteristics as robustness, versatility and simplicity. They are inspired by the success of natural evolution and, indeed, proved the success in many applications, from nonlinear numeric optimisation to architectural design and automatic software creation. EAs are often considered as an example of *artificial intelligence* and a *soft computing* approach. Their unique ability to search for complete and global solutions to a given problem makes EAs a powerful problem solving tool.

Historically, there exist several branches of EAs, namely Genetic Algorithms, Genetic Programming, Evolutionary Programming and Evolutionary Strategies. Their development started independently in the 1960s and 70s. Nevertheless, all of them are based on the same fundamental principle. EAs employ computational models of evolutionary processes as the key elements in their design and implementation. 'Evolution' is used here in its Darwinian sense, the advance through 'survival of the fittest'. All major elements found in natural evolution are present in EAs: a population of individuals, the environment, genome, selection, reproduction and various genetic operations such as mutation and recombination. The evolution progresses generation by generation.

As a rule, the search starts from a randomly sampled initial population. During the course of evolution, better solutions are selected from each population, the offspring is produced from these solutions, a new population is formed and the process continues. As can be seen, the algorithm does not need any initial guess solution and thus can be used when no a priori knowledge is available about the outcome and the environment. The only required function is to distinguish between better solutions and poorer ones. This function, known as *fitness function*, defines the problem to be solved by the EA.

Despite such remarkable simplicity, EAs have proven to be capable of solving many practical tasks. The first and obvious application is numerical optimisation (minimisation or maximisation) of a given function. There are virtually no limitations on the type of function as no derivatives are required. EAs showed outstanding results for many of the difficult, ridged or ill-defined functions, and even for dynamic functions which change with time. Since the algorithm always maintains a population of solutions, it is particularly robust in terms of finding the global optimum and avoiding local optima. Moreover, EAs naturally allow a multi-criteria search for a Pareto-optimal set of solutions.

Each individual may represent a set of parameters for one or another system to be optimised. Given a fitness function that estimates the optimality of the system with a given set of parameters, this system can be optimised by an EA. This approach is widely used in the engineering field, including the area of control and trajectory optimisation (see, for example, [32, 43]).

However, EAs are capable of much more than function optimisation or estimation of a series of unknown parameters within a given model of a physical system. Due to, in a large part, their stochastic nature, EAs can *create* such complex structures as computer programs, architectural designs and neural networks. Several applications of EAs have been known to produce a patentable invention [120, 121, 123]. Genetic Programming (GP), conceived in the early 90s by J. Koza [119], is specifically designed to evolve computer programs.

Unfortunately, such a truly intelligent application of EAs is rarely used for practical purposes. A major deterrent of its usage is very high computational cost required. All EAs are quite expensive in terms of computational requirements. This may be considered as the price for their superb robustness. Thanks to rapid increase in the computing power available during the past two decades, practical applications of the traditional EAs become feasible. However, GP and similar algorithms often require a supercomputing power to produce an optimal solution for a practical task. This may be overcome, at least partially, by narrowing the search space, e.g. limiting possible structures emerging as a result of genetic operators.

A creative approach of EAs to solve problems would be of a great benefit for the task stated in the previous section. The newly introduced UAV shipboard recovery method is not well studied and thoroughly specified at the initial stage of design. Before developing a controller, a question must be answered as to *what* to control. It is quite possibly that none of the known guidance and control methods could be readily applied and would satisfy the requirements for this particular recovery technique. Even though in the end the optimal guidance strategy and the control laws may appear to be very simple and will not require a sophisticated controller, the choice of the controller structure and thus of its design process remains uncertain at the initial stage.

A general engineering practice in such circumstances is to propose a new design based on existing knowledge of various techniques (not uncommonly even from other fields) and no less important, intuition. Following this, the proposal is analysed, tried on a real system or its mathematical model, findings and errors are analysed again, the design is modified (or rejected) and the process continues until a satisfactory solution is found.

EAs work basically on the same principle, although, obviously, using less analytical analysis but more trial-and-error approach. It was found, however, that the process of selecting the most suitable solutions at each stage and producing the next iteration variants is, overall, largely intelligent and heuristic. EAs are capable to answer not only the question *how to do* something (how to control, in particular), but also the question *what to do* in order to meet the objective. It is therefore appealing to apply such a promising automated technique to a problem with no general solution at hand.

On the other hand, the guidance and flight control is not a totally unstudied area where no convincing guesses can be made and where no parallels with the existing solutions are possible. This fact allows to watch, understand and guide, to a certain extent, the process of evolution. It also enables to optimise the EA for the purposes of control design.

The latter is especially useful because there are still very little research done on artificial evolution of structures and controllers in particular. An overwhelming majority of EA applications is concerned with numeric optimisation. A few proponents of a more advanced

use (e.g. Koza [124], De Jong [53]) are keen to show the real scope of possible applications, including controller design. However, many of their examples are rather theoretical and illustrative. It is therefore the second motive under the use of EAs in this study: to show the usefulness and capability of this still relatively new technique as a tool to solve a practical real-world task. This contribution is particularly important for the EA community as the EAs are essentially stochastic methods with relatively little analytical support, and the subsequent works more often than not are based on the successful examples of the preceding applications.

1.3 The contribution of this thesis

The contribution of this thesis is twofold. It contains a contribution in the UAV field and a contribution in the evolutionary computation field. However, the major outcome of this thesis is creation of a methodology which integrates both the evolutionary techniques and the areas of flight control and guidance. This methodology allows to synthesise highly optimal control and guidance laws given a defined control task and the system models.

Conceptual analysis of the Evolutionary Algorithms (EAs) theory included in this thesis enabled to develop an approach to the adaptation of EA techniques for control design problems. A strong motivation behind this research is the necessity to minimise the search space in order to reduce computational requirements and allow the solutions to evolve within a sensible time. The *Evolutionary Design* algorithm developed as a result of this research combines the power of *Evolutionary Strategies* and *Genetic Programming*, enabling automatic evolution of both the structure and parameters of the control laws.

Evolutionary Design (ED) extensively uses simulation models which are developed within this body of work to simulate the UAV recovery scenario. The simulation environment includes a nonlinear six-degree-of-freedom UAV model, a ship motion model, a model of the atmosphere and a model of a flexible cable. The atmospheric model, in turn, includes the models of disturbances, particularly the turbulence model, the gust model and the ship airwake model. The latter is a novel model developed specifically for the UAV recovery scenario.

ED is applied to produce a recovery controller for a small fixed-wing UAV as a design example of the methodology. A comprehensive testing procedure is used to verify the obtained controllers in terms of both robustness and performance. Simulation results demonstrate excellent robustness and high performance of the controller, which proves validity of the developed methodology.

This ED algorithm is a contribution to both the EA and control communities. It is believed that with proper modification of the control laws representation it can be employed to

solve a wide range of control problems. At the same time, it demonstrates an approach which can be used to solve creative problems with limited computational resources.

The Cable Hook recovery method proposed in this work contributes to the UAV community. Simulation of the recovery scenario with the controller developed by the ED methodology shows a great potential of this technique. Although this method needs further development, together with the ED methodology it offers a comprehensive solution to the autonomous shipboard UAV recovery problem.

1.4 Thesis outline

The thesis is divided into 7 main chapters, including the Introduction and Conclusion. The content of these chapters will be briefly outlined here; however, the reader is referred to the opening sections of each chapter for comprehensive details.

In Chapters 2 to 5 the background of the UAV recovery, simulation, Evolutionary Algorithms and control design is presented. This prepares grounds for the development of the Evolutionary Design algorithm and for the recovery controller synthesis, which take place in Chapters 5 and 6.

Chapter 2: UAV Recovery Problem introduces the UAV recovery problem. It states objectives of recovery and analyses the factors that influence the recovery process. A review of the existing recovery techniques is then given, with the analysis of their applicability, advantages and shortcomings. The Cable Hook recovery method is introduced. This is followed by preliminary engineering analysis and specification of the method, including the approach procedure and the equipment used. The problem of local positioning is discussed and a solution is suggested.

Chapter 3: Simulation Models provides details on both the theoretical development and specific features of the simulation models used within this research. Presented in this chapter are dynamic models of a UAV, a ship, the atmosphere and a flexible cable. The atmospheric model includes the models of steady wind, turbulence, gusts and ship airwake. The cable model is supplemented by a simplified static model for efficient computation. In addition, the shipboard recovery gear is specified and the amplitudes of its motion are analysed using the ship model.

Chapter 4: Evolutionary Algorithms presents the background on Evolutionary Algorithms which is used later in the development of the control design methodology that is the result of this research. This includes the introduction to EAs, discussions of their key elements such as genome representation, selection and genetic operators, and the discussion on the underlying theoretical background. Four major branches of EAs relevant to this work are de-

scribed and their specific features are discussed. A particular attention is given to Evolutionary Strategies and Genetic Programming.

Chapter 5: Controller Design includes two major topics. The first is the analysis of control required for the shipboard UAV recovery method proposed in Chapter 2. Existing control methods and control design techniques are outlined. Current methods of guidance and control in the relevant tasks, such as conventional landing and homing guidance, are considered. Based on this, a general framework of the UAV control system is designed. The second topic is compilation of the Evolutionary Design methodology which can be used to synthesise the control laws for this control system. The core of this technique is a specially tailored evolutionary algorithm, which simultaneously evolves both the structure and the parameters of a set of control laws. The algorithm specification is accompanied by the descriptions of the internal control laws representation and fitness evaluation procedure. Simulation environment used for fitness evaluation of the UAV controllers is also presented.

Chapter 6: Controller Synthesis and Testing demonstrates the application of the Evolutionary Design methodology. The chapter starts from five-step controller synthesis, with discussions of intermediate results. The synthesis ends with a discussion of several issues that may arise when implementing the obtained controller within a physical system. The controller is then tested comprehensively for both robustness and performance. Sensitivity of the controller to perturbations in various modelling variables is analysed. Finally, the controller is tested across the whole range of operational conditions and the operational envelope is obtained.

Chapter 7: Conclusions discusses some of the key results of the research. This discussion includes a brief critique of the design methodology and the produced solution, with some limitations revealed. In the end, the areas requiring further work and possible areas of future research are identified.

Chapter 2. UAV Recovery Problem

The first stage of the design process involves identification of the issues associated with the recovery of a small fixed-wing UAV on a ship deck. Once these issues have been identified, decisions can be made regarding a suitable recovery technique, appropriate control methods and key design parameters. Since shipboard UAV operation have a considerable history by this time, relevant current experience has to be studied.

The chapter begins with stating the objectives of UAV recovery. The operational environment, tactical requirements and limitations are analysed and the task is formulated. A review and analysis of the existing recovery techniques is then made. Based on this study, a novel method of recovery is proposed and design parameters are identified.

2.1 Objectives of UAV shipboard recovery

Recovery is the final stage of the flight. Throughout all aviation history, landing and especially shipboard landing was regarded as one of the most difficult and dangerous flight phases. For a pilotless aircraft however, the danger associated with landing is usually less intense; the necessity of recovery is predominantly cost-driven. In some circumstances, even expendable design may be preferable. Nevertheless, the primary goal of recovery is to be able to reuse the aircraft in subsequent missions and in some cases, to collect the data gained during the current flight which could not be received on-line. Hence, the prime objective of UAV shipboard recovery can be formulated as *to provide an effective, reliable and safe transition from normal flight to stationery position onboard the ship whilst maintaining the integrity and operability of the UAV.*

Solving this problem is an enormous engineering work on its own. However, no matter which recovery technique is chosen, efficiency of the whole system is the first thing to consider. In a broad sense, it can be expressed as a cost minimisation procedure, which includes not only direct and indirect expenses, but also risks, time and labour requirements, and potential limitations. Some of the components of this cost are the cost of necessary equipment, such as deck installations and onboard devices, reduction of the UAV and ship capabilities (for example, reduction of maximum payload due to additional onboard equipment), and the cost of engineering works, including development, installation and testing. Operational costs are no less important; they arise not only directly from running the operation and maintenance but also indirectly from the constraints imposed on the ship during the operation. The latter may be of the utmost importance, especially for military use. For example, restricting the operation to calm weather may render the whole mission unsuccessful, should weather become worse

than required. Furthermore, risks for personnel and equipment also greatly affects the efficiency of the recovery system. The following list reviews the aspects which should be taken into account when analysing one or another recovery method.

- *Amount of required onboard equipment and ship installations.* Clearly, some sort of landing gear is a necessity. Even such a simple method as skid landing requires a valuable space to land on. Other methods may require very specific devices, which must be integrated into the ship (or UAV) structure. This includes not only the tools directly engaged in recovery, such as various arresting gears, but also additional sensors, controls and software.
- *Limitations to ship and UAV configuration.* In the same manner as current piloted flight operations shape the aircraft carriers, UAV integration aboard a ship will have a great impact on the design of both aircraft and vessels (see, for example, [65] for detailed analysis with relation to future aircraft carriers). More specifically, certain recovery methods may require the ship to have a large open platform on the stern (such as the helideck found on some types of naval vessels), which can hardly be 'added' as an upgrade but has to be integrated into the design from the beginning. From the UAV side, the aircraft may be restricted to pusher-prop configuration (e.g. for net recovery) or, the other way round, mission-specific sensors layout (particularly nose and underside mounted) may limit the options for recovery.
- *Amount of manual operations required.* Many UAVs, still in service, require manual remote control during launch and recovery phases, especially in relatively difficult environmental conditions. This requires specifically trained (and thus expensive) operators. However, even fully autonomous recovery may involve a significant amount of manual operations, such as disengaging the UAV from arresting gear, checking and reloading recovery devices (e.g. parachute) etc. These operations not only entail additional costs, but also increase turn-around time.
- *Turn-around time.* In the context of recovery, this is the time in which the recovery facilities become available for operation after a successful landing. The time may be different for the shipboard equipment and for the UAV itself. The former becomes more important as multiple UAVs are going to be operated simultaneously.
- *Potential weather limitations.* Open sea weather conditions are probably the most tremendous challenge for any recovery procedure. However, some recovery methods are inherently more susceptible to weather than others. For example, a requirement to place the recovery facilities on the stern (or the bow) of the ship invariably increases the amount of oscillations to deal with in wavy conditions. Certain methods may depend on some factors far greater than on the others, which

sometimes may give tactical advantage (or disadvantage). For instance, controllable ditching is less dependent on wind (as free choice of landing direction is possible) and does not depend on ship motion at all, but may be highly dependent on height of waves and even their shape.

- *Impact on ship transit and operational performance.* During recovery, the ship may be required to keep a certain position, speed and/or heading. Keeping a headwind direction is a common requirement to aircraft carriers during flight operations. This reduces the impact speed and helps to maintain consistent parameters from flight to flight. Meanwhile, studies (e.g. [177]) show that flight operations where the ship needs alignment with prevailing wind conditions drastically reduce the transit performance of the ship. Optimal planning of the flight operations becomes a serious problem on its own. In the case of a small UAV, the issue becomes even more pronounced: the cost of losing the transit performance may override the cost of UAV itself. Moreover, while flight operations on a carrier usually involve many aircraft and are carefully planned, UAVs are often operated sporadically (or this may be a desired capability). Aligning a 4,000 ton ship with the wind to recover a single UAV and then returning to the previous course may be considered as too high a price. Therefore, reliance on the ship speed and position should be minimised. However, this may be a weather-dependent attribute: in harsh weather conditions, tougher limits may apply.
- *Possible hazards for ship structures and personnel and consequences in the case of failure.* The possibility of a mishap during any stages of recovery must be envisaged. Even though the UAV itself may be considered expendable to some extent, it may cause severe damage to the ship and its personnel. Not all methods allow go-around at all stages of final approach. Moreover, even with normal procedure, a significant amount of manual operations may endanger the crew while in battle mission or in poor weather.

For the UAVs in current or past service, the above factors could be analysed on the basis of actual service records and experience. However, not many suitable recovery techniques have seen wide application; most of them have not proceeded beyond trial phase. Like with any new concepts, theoretical engineering analysis is the only option in such cases.

In the next two sections, the current state of the problem is discussed. First, the environmental factors affecting recovery are described. Then a survey of the existing known techniques is given. Based on this review, a new recovery method will be proposed in further sections.

2.2 The environment

This section is intended only to describe the important environmental factors and their potential influence on the recovery process. Their specific properties will be analysed in greater detail in the next chapter, where the models of the respective natural phenomena are designed. However, understanding the effects of the environment is crucial in analysis of existing and future recovery methods.

2.2.1 Atmosphere

Atmosphere is usually considered in the contexts of air properties and its motion. The air properties that have considerable effect on (primarily) UAV operations are temperature, air density, air pressure and humidity. Climatic factors such as precipitation also affect flight operations. All these factors change with aircraft position and altitude, as well as with time. However, recovery is more often than not a relatively quick procedure carried out at low altitudes (moreover, at sea it is an absolute, or 'above mean sea level' (MSL), altitude). Therefore, the dynamic changes of any of the above parameters are small and usually may be neglected.

The influence of the above air properties is generally aircraft specific (or is more or less common for all aircraft, e.g. air density) and not recovery method specific. Nevertheless, they may have indirect yet strong consequences on recovery. Some examples are:

- High temperatures and humidity, having adverse effect on power, disadvantage the methods that highly rely on power plant (in particular, VTOL aircraft).
- Rain, though being difficult for most small UAVs, may particularly hamper the recovery relying on friction, for example, skid or conventional runway landing.
- Icing conditions, which contribute to ice build-up on the wing, may be especially harmful if the UAV is recovered at near-stall or post-stall angles of attack.

To facilitate comparison and to maintain consistency, the reference data are usually obtained in (or re-calculated in accordance with) the reference atmospheric conditions. The standard atmosphere model [1] prescribes the values of air density, temperature, pressure and other parameters at different altitudes. This model is used in this study. If a particular recovery method being designed is believed to have potential difficulties in certain conditions and the appropriate data are available (e.g. engine characteristics at high air temperatures), the tests should be performed for these conditions as well.

2.2.1.1 Wind

Wind and, in general, atmosphere motion have an immense effect on UAV behaviour and hence on the recovery. Ships are affected by wind too; however, direct influence is con-

sidered to be negligible (unless it is a sailing ship) and the primary source of ship oscillations is assumed to be the waves (of which wind is the major cause).

Atmospheric motion is usually divided into constant flow (for the time scale considered), or steady wind, and random motion, such as turbulence and gusts. These motions are closely related (as a rule, the greater the steady wind, the more gusts and turbulence). However, significant turbulence may exist even without steady wind, particularly where non-uniform heat transfer is present—for example, during morning hours and in clouds.

Steady wind may be used to the benefit of recovery the same way as nearly any aircraft use it, taking off and landing upwind. This reduces the impact speed because it allows to maintain the required safe airspeed while reducing the ground speed. The ship's own drive can further reduce the impact speed. However, as said above, the requirement to align the vessel with the wind represents a significant performance downgrade for the ship and should be avoided. Thus, any wind direction should be expected during recovery. In these circumstances, those recovery techniques will have an advantage which are either resistant or robust to wind direction, or which offer the possibility to approach the ship from different directions.

Strictly speaking, wind direction is a 3D vector. However, significant vertical flows such as thermals may be neglected for shipboard recovery, which happens at very low altitudes, above a relatively flat area (this does not include vertical components of turbulence and gusts, which may be present). By convention, wind direction is often denoted as the heading to the 'origin' of the wind, which is opposite to the average velocity vector of air particles.

Apart from direction, steady wind has only one complementary parameter: magnitude (or speed). Wind speed is closely related to other environmental factors, particularly sea wave (see Section 2.2.2 below). Therefore, wind magnitude can be expressed in terms of Sea State. For example, sea state 6 corresponds to 'standard' (or design) wind speed of 30 knots (15 m/s) (according to [2], see also Section 3.3.1 for details). It should be noted that both wind direction and speed change with altitude, although altitude changes during recovery are usually fairly small so these variations do not pose a noticeable problem.

2.2.1.2 Turbulence and gusts

Turbulence and gusts are one of the most adversely contributing factors for recovery. Primarily, they affect the UAV and result in constantly changing random displacement of the flight path and aircraft attitude, as well as of the velocity and acceleration vectors. Operation at a high level of turbulence not only requires an increased tolerance to UAV position at the moment of landing, but also greatly increases the chances of exceeding operational aerial angles (angle of attack and sideslip), thus leading to necessity of higher safety margins for these angles.

There are two general ways to reduce the impact of turbulence on recovery. The first one is associated with UAV design, which includes both airframe and control system improvements. However, this often imposes undesirable and conflicting requirements to the UAV. For example, susceptibility to the turbulence may be reduced by increasing wing loading—with all associated problems such as increase of airspeed and downgrade of take-off and manoeuvring performance. Control system improvement is limited and usually already exhausted because flight path stabilisation is an actual problem for all stages of flight. Nevertheless, some innovative UAV designs are aimed particularly at turbulence rejection. They will be described later during the methods review.

Another way is employing a recovery method which allows greater tolerance to the flight path. Unfortunately, this almost unavoidably leads to use of larger capture facilities (nets, cables, etc.) Controllable ditching (or a hydroplane UAV configuration) somewhat circumvents this problem but requires further recovery from water.

On the positive side is that the turbulence over such a fairly flat and smooth surface as sea (compared to typical terrain) is more or less predictable and regular. It is well studied and easily simulated (see Section 3.2.2). However, the ship (and any nearby objects) may deliver a significant amount of additional turbulence which needs to be taken into account.

2.2.1.3 Ship airwake

One of the problems associated with ship airwake is that it is not just an added amount of turbulence. The air flow about a ship is of a low speed nature and is inherently unsteady [136]; it is also significantly affected by the ship's periodic motion [211]. Unfortunately, there is little published data suitable to build a generic model of ship airwake. The majority of research on this topic is performed with helicopter-on-deck scenarios in mind and are valid in a very limited area or at a single point (e.g. [91, 144]). This is insufficient for recovery where a large region of the airwake may be traversed.

The military standard MIL-F-8785C [5] includes a statistical model of airwake that describes its properties along a large portion of the approach flight path. However, it is designed for standard operations of piloted aircraft on aircraft carriers. Nevertheless, with the help of the studies of wind flow around buildings (such as [135, 160]) it can be generalised to some extent that allows simulation of the effects of airwake when approaching the ship from different directions. See Section 3.2.4 for further details.

From the point of view of recovery, the adverse effects of ship airwake can only be avoided if approach is made from the upwind direction (with respect to the ship) or from the side or when the UAV is captured far from the ship, as in the cases of ditching and kite parasail recovery (Section 2.3.3.3). This is an unfortunate condition, because it results in higher

impact speed and/or strong side wind. Nevertheless, recovery methods that are able to avoid the airwake can be more reliable due to lesser exposure to random and unsteady conditions. It is also important that they can be more realistically simulated, considering the current state of airwake models.

2.2.2 Sea and ship motion

Unlike atmospheric influences, the periodic ship motion due to the effect of sea waves is a specific problem of maritime operations. Wind plays the major role in producing waves, although they can also be generated from ocean currents, gravitational and tidal forces and geological phenomena such as earthquakes. The typical environmental conditions, including wave amplitudes and wind speed, are grouped into several Sea States (see Table 3.2). The general field that studies the dynamics of ocean (sea) vessels is known as *seakeeping*.

The problems with recovery arise from the fact that ship represents a constantly moving platform which is difficult to aim at. Moreover, even at the moment of successful capture, added linear velocity of the deck (or recovery devices) due to ship angular motion may represent a significant threat for the aircraft.

Although wave motion is highly periodic (see Section 3.3.1), real-time prediction of the actual position of the ship is a difficult problem. To date, only few seconds prediction (up to 3–4 s) is possible with adequate precision [167, 192, 229]. If a better prediction was possible, it could be used for aiming at a specified point on the ship instead of constant tracking of that point, and the problem of ship motion could largely be avoided. However, as ship motion prediction is far out of the scope of this research, it is assumed that the random component of ship position is unpredictable for the UAV.

Ship motion can be simulated using recorded statistical data of the motion of a real ship. Such statistical methods are commonly employed when generating simulated ship motions, because an analytic description of ship motion as a function of wave motion is still an open problem and is of little use [99]. See Section 3.3 for details.

2.3 Recovery methods review

There is a vast number of methods proposed or being developed for recovery of fixed-wing UAVs. Being not constrained by a human air crew and usually possessing a stronger airframe (with respect to the size, especially for small aircraft) as compared to their piloted counterparts, UAVs can adopt more ‘extreme’ ways of ‘landing’. This significantly increases the number of available options.

2.3.1 Conventional runway landing

This method is an obvious option, taking into account huge experience in land-based aircraft (both unmanned and piloted types) and aircraft carriers. Yet an obvious drawback is a requirement of a relatively large flat open deck used as a runway. Some sort of arresting gear, like a tailhook and arresting wires utilised on aircraft carriers, is often a necessity as well. Even though the approach speed of a small UAV can be sufficiently lower than that of a typical piloted aircraft and thus the arresting gear may be unnecessary to stop the aircraft, it may still need to be captured in order to prevent it from being blown out from the deck in windy and wavy conditions.

Tough tolerance to the landing path is another limitation. For example, with a 5 degree glideslope Θ_{gs}, an altitude allowance of 1 m requires $1/\tan(\Theta_{gs}) \approx 11.5$ m of deck length. Meanwhile, ship oscillations alone may exceed this figure (1 m) even in relatively mild conditions. Furthermore, the glideslope must be greater than the allowed angular amplitude of deck oscillations (in the direction of approach), otherwise touchdown will be impossible at the moments when the slope of the deck is greater than the glideslope (Fig. 2.1b). A too low approach is fatal because the UAV can hit the ramp (Fig. 2.1a). For this reason, the approach path should be planned so that there is an appropriate altitude margin at the moment of crossing the ramp even in the worst case, when the ramp is at its highest position (Fig. 2.1c). This means that a lengthy deck should be allocated only to absorb the variance in touchdown positions due to ship oscillations, let alone landing run distance. A steep glidepath angle may be a problem to the UAV as well, firstly because of a higher vertical speed and thus greater stress on the airframe (landing gear) on touchdown, and secondly because of potential difficulties to maintain a low approach speed at high dive angles without special air braking devices. These problems could be largely eliminated if the deck position at the moment of touchdown was known in advance. However, as noted in Section 2.2.2, this is considered to be unfeasible in this study.

Another disadvantage of this method is tight constraints to approach directions. Considering that a significant deck length is required, approach can be made only along the deck. Even if the deck width were adequate, landing from the side would be more complicated due

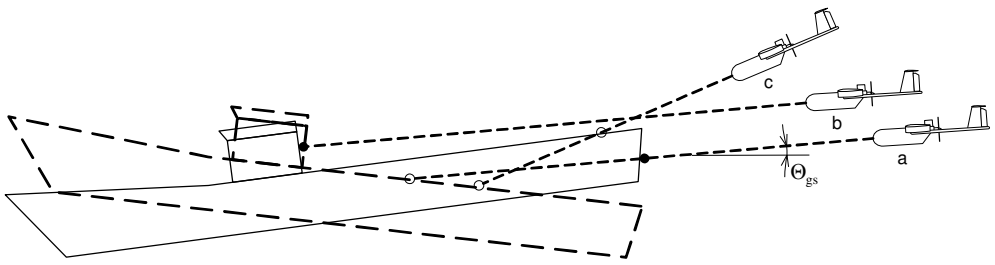


Fig. 2.1 The relationship between glideslope and ship pitch amplitude (not to scale)

to much greater roll amplitude of the ship as compared to pitch amplitude, see Table 3.3 in Chapter 3. Typically, the approach is made from the stern side, because it allows to use the ship's own speed to reduce the impact velocity. However, this requires the UAV to traverse ship airwake, except in a strong side or tailwind. In addition, enabling approach from the opposite direction requires an open through deck or two landing decks, which is an exigent demand to the ship design.

On the positive side of this method is a great deal of experience, gained during almost a hundred years of piloted flights from the ships. Many hidden problems, which are inevitable in any new design or method, have already been discovered; design schools are established. A technical advantage is the relative simplicity of arresting gears, both on the aircraft (conventional, though reinforced, landing gear and a simple arresting hook) and on the ship (arresting wires). Nevertheless, a precise electronic positioning system such as the Automatic Carrier Landing System (ACLS) used for piloted aircraft on air carriers is needed. However, it is required for nearly all methods.

Considering the above, conventional landing may be used primarily on large ships, which can allocate a large deck for flight operation and which are less sensitive to waves due to high mass and size. Several large UAVs such as *Predator* have been successfully operated from aircraft carriers. A few attempts to produce a fixed-wing UAV capable to land conventionally on a helicopter-deck-size area were generally unsuccessful. One of these attempts was the Alliant Techsystems' *Outrider* which, among other flaws, was unable to achieve the required landing capabilities despite a special design of the tandem wings [8].

2.3.2 Recovery net

Historically, the first practicable and widely used solution of shipboard recovery of a fixed-wing UAV was capturing by an elastic net. This method has been employed for the USN *RQ-2 Pioneer* UAV, first deployed in 1986 aboard the battleship USS *Iowa*. The recovery net is usually stretched above the stern of the ship and the aircraft is flown directly into the net. After successful capture of the UAV, the net is lowered to the deck and the UAV is

Fig. 2.2 Pioneer UAV net assisted recovery

removed from the net manually (Fig. 2.2).

This technique is one of the simplest one can imagine. Indeed, the deck installations are simple and despite the large size of the net itself, require little deck space and can be deployed and uninstalled on demand. The UAV does not need landing gear at all and is not required to perform special manoeuvres, such as flare-up or any other transition stages. It does not even need to hold the decent rate very precisely as required for conventional runway landing, it only has to keep the approach speed as low as possible. Therefore, from the engineer's point of view, the main task in implementing this technique is to build a controller that would be capable to fly the UAV into a relatively large central area of the recovery net with the lowest possible speed.

However, the largest operational (non-combat) losses of the *Pioneer* UAVs were associated with improper recovery. During flight tests in 1986, four Pioneers were lost, three of them during recovery [152]. About the same loss rate persisted during the actual operations in Desert Storm. The source [7] contains comprehensive statistics of various UAV losses, which may provide some insight in actual operational problems. On the other hand, the *Pioneer* was a Remotely Piloted Vehicle (RPV), not a fully autonomous UAV. Landing was particularly difficult because operators were attempting to fly the *Pioneer* into a target at a precise airspeed, altitude and angle of attack, without benefit of the visual, tactile, aural cues available to conventional pilots. After introducing the Unmanned Common Automatic Recovery System (UCARS) in 1996 (see Section 2.4.3.4.2), Pioneer has successfully completed the autonomous recovery flight tests in various environmental conditions [171]. Furthermore, many problems during shipboard deployments were experienced due to the fact that *Pioneer* was not designed for this operation, but instead adapted to shipboard use after design.[1]

Nevertheless, even a perfect approach does not guarantee a faultless recovery. Such an extreme method of capture is quite stressful for the aircraft. Approach speeds remain fairly high, and the recovery net often rips off the propeller, antennas and even wing upon landing. This fact not only incurs the cost of repairs, but also affects turn-around time and operational readiness of the unit. Such a high risk of damage to the UAV seems to be inherent to the net recovery systems, unless the approach speed of the UAV is reduced dramatically.

Net assisted recovery as such does not limit the approach direction as strictly as conventional landing does (Section 2.3.1). However, the net can hardly be reallocated in real time due to its size. For this reason, only one preferred direction remains (typically from the stern). Moreover, intuitively, chances of damage to the UAV rise if the aircraft is flown into the net with a sufficient sideways velocity component or non-zero sideslip. This jeopardizes recovery

[1] *Pioneer* UAV has been developed by Israeli manufacturer IAI Malat and acquired by the U.S. in light of the success of *Pioneer*'s predecessor *Scout* in Lebanon operation in 1982.

in strong side winds and thus deters from positioning the net along the deck, which could allow approach from both sides. In any case, two fixed directions instead of one probably do not justify such rearrangement, bearing in mind that it eliminates the benefit of using the ship speed.

There are several other limitations of this method which should be taken into account. One of them is difficulty to go around if the net is missed. If the deck is not open in the direction of flight (and in the case of a helicopter deck it is usually not), there are high chances that the UAV will crash into the ship superstructures, especially if the UAV overflies the net (too high approach). Too low approach is fatal for the same reasons as for conventional landing (Fig. 2.1a). After a certain point on the flight path (the decision point) either crash or successful recovery becomes unavoidable, without an opportunity for another try.

Another important consideration is limitations to UAV aerodynamic configuration. This method restricts using a conventional tractor propeller, because either it will cut the net or the net will break the blades. It can still be used if the engine is shut before the impact (a sufficient time in advance to stop the propeller) and the propeller is equipped with folding blades—which may be too much of a compromise. Fortunately, most of the current UAVs are designed in pusher-prop configuration to allocate the payload (such as optical sensors) in the nose.

Finally, the amount of manual operations is high and they can hardly be mechanised.

As a conclusion, net assisted recovery is simple and can be effective if damage to the UAV can be avoided. Thus, it is more suitable for light and ultra-light UAVs, which are typically stronger relative to their mass. One of the examples of such UAVs is *Sea-ALL* (Sea Airborne Lead Line) UAV [210], a 2.5 kg hand launched short range reconnaissance UAV first deployed in 2001. It is a naval derivative of AeroVironment's *Dragon Eye* UAV with net assisted recovery aboard a ship.

2.3.3 Parachute systems

An inherent problem of the fixed-wing UAVs is a relatively high minimum airspeed required to maintain controllable flight. One of the contributing factors here is limited wing lift capacity at low speed. It can be aided or even replaced with various parachute systems. These systems are widely used for ground-based UAV recovery, hence significant experience in parachute recovery has been gained. Many ground-based UAVs with conventional runway landing have parachute recovery as an emergency recovery method.

A common advantage of parachutes is reasonably low weight and small volume when packed. Their cost is also relatively small, so that, in some cases, the canopies can be expendable after a single use. Amongst general drawbacks is that although parachute can be easily

deployed, it cannot be automatically packed back in flight if needed (and even on the ground it remains a manual job). Therefore, after deployment, the flight must rely on the parachute; not all errors may be corrected at this stage. Another shortcoming is generally high susceptibility to wind.

2.3.3.1 Uncontrollable parachute systems

Uncontrollable systems usually feature a non-guided drag-only round or cruciform parachute. They are designed for steady vertical or nearly vertical descent (in no wind conditions). Apart from simplicity, their advantages are more stable and straightforward deployment and lower opening force as compared to parafoils (see Section 2.3.3.2). On the other hand, the UAV is completely uncontrollable during descent, being subject to wind turbulence and other uncertainties.

Among the UAVs with this type of recovery are British *Phoenix* and *Observer* [225, 226], Italian *Mirach 100*, Greek *3-Sigma* and U.S. *Exdrone*. Israeli *Ranger, Hunter*, the U.S. *Predator* and several other UAVs use parachutes as an emergency recovery method.

Current experience with ground-based UAVs with uncontrollable parachute recovery shows that landing deviation of the order of 10–20 m can be achieved [226]. The parachute deploys at a level of 50–200 m above the ground, with every effort being made to keep this figure as low as possible to improve landing accuracy. The main reason for this is the amount of uncertainties that have an effect on the UAV during descent. There are several factors that affect the accuracy of landing.

An apparent reason of deviation is the wind. Set aside unpredictable turbulence and gusts, even steady wind becomes a substantial problem. A wind estimation error of only 1 m/s will result in a position difference of 15–20 m on the ground after a descent from 70–100 m (assuming a typical descent rate of 4–5 m/s). Therefore, a vitally important component of any parachute recovery system is the wind estimation subsystem, which estimates the prevailing trend in wind speed and direction. It determines the point of parachute deployment, after which there is no chance (for an uncontrollable parachute) to correct any error.

Fig. 2.3 DRS Unmanned Technologies' Neptune UAV with parachute recovery

Wind estimation is based on two major elements: wind measurements on the current flight level and the model that estimates the actual wind at the lower levels, which the UAV will traverse during descent.

Wind may be measured onboard the vehicle in a variety of ways. Generally, the wind speed vector can be

obtained by subtracting airspeed from ground speed. Assuming co-ordinated level flight, this may be done using onboard airspeed measurements, body yaw angle and GPS-derived ground track information. The data is often averaged and filtered to obtain a more reliable estimate [226].

The environment model takes into account that surface friction has the dominant effect on wind speed at low altitudes. For this reason, wind speed falls near the ground, and this fall highly depends on the type of surface: plain, sea, forest, etc. Sea operations have a benefit here as the sea surface is fairly even, thus the wind speed distribution with altitude is more predictable. However, wind speed itself is usually greater over the open sea, which requires better relative accuracy of estimation. Secondly, ship airwake problems arise when the UAV gets closer to the ship, dramatically changing wind speed, direction and turbulence. Although these changes affect the descending UAV for only the last 3–5 seconds, it is enough to produce an unacceptable position error of some 5–20 m. Therefore, the model must take the airwake into account, which is, as discussed in Section 2.2.1.3, very problematic and requires additional investigation.

It should be noted that in the maritime context, recovery must be provided on a moving vessel. Therefore, the UAV, in addition, must estimate the future position of the ship at the moment of landing. While the requirement to the ship to maintain its course and speed during recovery is deemed to be reasonable and thus such estimation does not represent any difficulties (ignoring the periodic component), there is a potential problem of collision of the descending and uncontrollable UAV with the ship superstructures. For example, in no wind conditions (and therefore with vertical descent), with a considerable ship drive and the target landing point on the helideck at the stern, the UAV should deploy its parachute right ahead of the ship so that at the moment of touchdown the helideck would be right beneath the aircraft. However, during descent, the UAV will collide with the ship superstructures that are higher than the deck as the ship moves forward. Unfortunately, this problem cannot be solved without requiring the ship to stop or to change its path. A possible solution could be to provide the deck for recovery at the highest point of the ship or use a deck which is open from nearly all directions (such as the flight deck of the aircraft carriers), which seems to be unfeasible and unreasonable.

Another important subsystem that should be mentioned is the parachute discard mechanism. It is necessary to prevent the UAV from being dragged or overturned on the ground (or on the deck) after the vehicle has landed. Simple passive mechanisms acting upon cord tension sometimes fail to operate in high wind conditions. Meanwhile, more reliable active systems require an actuator and a sensor, which makes the whole system more complex.

A typical descent rate provided by a parachute is about 4 to 5 m/s [226]. This rate is determined by the canopy shape and loading (total weight to area ratio). The parachute can be designed for any required descent rate (for a given weight); however, the chosen rate cannot be changed during the flight without a special control system (such systems are discussed in the following sections). Therefore, choosing a descent rate is a compromise. On the one hand, a higher rate is desirable because it provides less exposure and sensitivity to wind and turbulence, allowing better landing accuracy. On the other hand, the kinetic energy of the descending UAV must be absorbed at the touchdown safely for the airframe, which becomes more difficult for higher rates. For this reason, most parachute recovered UAVs are equipped with some sort of impact attenuation system in order to cushion the vehicle as it hits the ground. Impact attenuators commonly used on UAVs are inflatable airbags and crushable foam or paper devices. Apart from impact deceleration within given limits, impact attenuators should cope with different directions of impact, as the horizontal velocity component may be even greater than the vertical one.

For possible shipboard operations, the onboard impact attenuator can be replaced with a shipboard capturing device such as a horizontal elastic net, if adequate landing accuracy is provided. This could give sufficient weight and internal volume benefits to the UAV.

In view of all these problems, the only feasible application of uncontrollable parachute systems for maritime use is ditching the UAV near the ship. This eliminates the issues of accuracy and impact attenuation, making the system truly simple. However, the problems of UAV buoyancy and water protection must be solved. Another important question that arises is recovery of the UAV from water. No matter how it is done—using a boat or a helicopter—it is a labour-intensive manual operation with high turn-around time. Currently, this method of recovery is employed primarily for target drones and similar military UAVs such as the *Kalkara* UAV used by the Australian Navy and Air Force for crew training and weapons system performance evaluations. Some early and expensive UAVs, such as Teledyne-Ryan *BQM-34 Firebee* target drone, allowed a more complicated final recovery technique. After deploying a round parachute, they could be grabbed in-flight by a helicopter. Obviously, this is not a perfect solution for small UAVs.

2.3.3.2 Gliding parachutes

Although drag-only parachutes can have a limited degree of control through the special slats, this is not enough to cope with wind gusts and to provide necessary landing accuracy. The vehicle's own aerodynamic surfaces are inefficient due to low airspeed. The only practicable solution is a gliding parachute.

Fig. 2.4 Korean Night Intruder 300 UAV, featuring parafoil recovery

A parafoil canopy is the most commonly used form of gliding canopies. Parafoils are usually rectangular or elliptical in planform, two-skinned and form an airfoil cross-section when inflated with ram air. Glide ratios in excess of 3:1 can be achieved. Parafoils can be controlled by deforming the trailing edge, using pull-down brake lines. Application of asymmetric deformation results in turn rates, while symmetric control produces a dynamic flare manoeuvre, significantly decreasing forward speed and sink rate for a limited time [134, 226].

This flare manoeuvre is particularly useful as it can provide a soft landing without an impact attenuator. However, shipboard operations are more demanding in this aspect, because the motion of the deck due to the sea waves can add up to 2–3 m/s to the impact speed. Therefore, some sort of capturing device or even an impact attenuator may be necessary for the same reasons as discussed in the previous section.

A parafoil-assisted flight can be either powered or unpowered. In unpowered implementation, the UAV shuts down the engine before deployment of the parafoil, similar to the parachute systems described before. Because a parafoil has a limited ability to guide the vehicle, the point of canopy deployment should be calculated carefully. Such a system may help to correct only a limited range of errors caused by gusts, inaccurate wind estimation and other uncertainties. Therefore, a fairly accurate wind estimation is important.

Powered flight on parafoil gives a clear advantage of full flight control and even the ability to go around if the approach is performed unsatisfactory [34]. However, parafoil has very low loading (as compared to wing) and generates much drag. Low area loading and low airspeed, being the purpose of using the parafoil and an advantage for landing, creates several problems. One of them, susceptibility to wind and gusts, is inherent to all parachute systems. In general, an aerial vehicle can cope with a wind speed not greater than the vehicle's own airspeed. Otherwise the vehicle will not be able to reach any point in upwind direction. Meanwhile, even moderate wind over the sea is comparable in magnitude and can possibly exceed the parafoil glide airspeed. At the same time, increased drag from the parafoil severely limits the maximum airspeed. A technical difficulty of powered gliding is to provide enough thrust for the increased drag at such low airspeeds, whereas commonly used fixed-pitch propellers are designed for optimal cruise speed. Once again, careful calculation of the parafoil deployment point becomes necessary in high winds.

Parafoil deployment itself is another problem that requires particular attention. The parafoil canopies are made of low or non-porous fabric, because they must remain inflated to

operate as intended. For this reason, parafoil systems experience higher opening shock loads than many other canopy types. Furthermore, canopy cells inflation must be controlled and the canopy must be aligned more precisely. Because of this, parafoil deployment mechanisms tend to be more complicated, often involving some sort of reefing, multi-stage release and multi-point attachment. Deployment is further complicated by the need to protect the control-line servos from the opening loads [226]. These measures result in a time and therefore altitude consuming procedure. Another problem is to provide a seamless deployment without endangering the parafoil to be damaged by the rotating propeller, particularly for pusher-prop design.

Apparently because of such complexity, gliding parachutes have not seen such widespread use for UAV recovery as uncontrollable round or cruciform parachutes. Among those UAVs which employ parafoils for recovery are *Skyeye* (BAE), *Eyeview* (IAI Malat), *Sentry* (S-TEC), *Poisk-1/2 (KhAI)* and a few others, although for many of them parafoil recovery is optional or for emergency landing only.

2.3.3.3 Parasail recovery

A parasail is an application of a parafoil used as a tethered glider, very much like a kite. It may be employed in two configurations: with a shipboard or airborne parafoil.

A shipboard parasail may be used to lift various recovery devices, for example, a recovery net (see Section 2.3.2). This allows use of a large net without building massive constructions, which dramatically simplifies the approach for the UAV. A practical example is *Skyhook* recovery system (Section 2.3.6.1), in which parasail can be used, as an option, to lift a long recovery line.

An interesting implementation of an onboard parasail has been suggested by R. D. Greenhalgh et al (BAC Ltd.) as far back as in 1974 [85] and has been flight-tested in 1991 on the *Skyeye* UAV [34]. In this technique, the UAV carries a haul line and a controllable parafoil on board. The aircraft makes a powered approach on the parafoil with the haul line extended. Passing above the ship, the line captures a special pole, connected to a shipboard constant tension winch. After that, the UAV continues its flight but remains connected to the winch and behaves much like a kite. When an appropriate attitude of the UAV with respect to the ship and the wind is achieved, the UAV is gradually winched down to the deck.

Flight tests have proven good performance of this method for a wide range of wind speeds and sea states. Requirements to the size of the landing spot were quite low, comparing to direct (runway) landing on a parafoil even in low wind conditions. Moreover, the technique was partly compatible with Recovery Assist, Secure and Transit (RAST) system, widely used for helicopter recovery. Nevertheless, it remains a relatively complex, multi-stage operation

with a high turn-around time. The weight and complexity of the onboard equipment (a parafoil, its control system and a haul line with a hook) is considerable. Moreover, this method inherits all the drawbacks of parafoils discussed in the previous section.

2.3.3.4 Dynamic parachute deployment

This method combines a conventional approach, as for runway landing or net assisted recovery, and landing on parachute, deployed a second or two before touchdown. The aim of the method is to extend the 'normal' approach and to reduce the parachute-assisted flight as much as possible. This can improve landing accuracy and diminish wind susceptibility. To the author's knowledge, such a method has not yet been used or proposed for UAV recovery. The idea comes from the ability of some aircraft to engage a brake parachute or reverse thrust at landing before touchdown and also from the deep stall landing (see Section 2.3.4) used, in particular, by free-flight aeroplane models (FAI F-1 class).

The UAV approaches the deck as usual for a fixed-wing aircraft (Fig. 2.5a, see also Section 2.3.1). The glidepath ends some 0.5–3 m above and several metres before the designated landing point, depending on the UAV mass and parachute size. Then, the aircraft stops the engine and deploys a simple unguided parachute. Initially, it acts like a brake parachute. As the forward speed decreases and the UAV starts to fall down, the parachute gradually turns upward and acts like a conventional landing parachute (Fig. 2.5b, c).

Because this stage is unsteady, it is not necessary to have a big canopy to achieve a safe sink rate. If the manoeuvre is performed precisely enough, the height to fall from will be of the order of a metre or two, which is not enough to develop a steady descent speed. Therefore, the parachute may be smaller than that required for conventional parachute recovery. However, some sort of impact attenuator will be necessary. As discussed in Section 2.3.3.1, it can be either onboard or shipboard, for example, a horizontal or inclined net, as illustrated on Fig. 2.5.

The main challenge with this technique is the accuracy requirements. While positioning requirements may be generous enough, the moment of parachute deployment must be calculated with pinpoint accuracy. A fraction of a second delay will result in

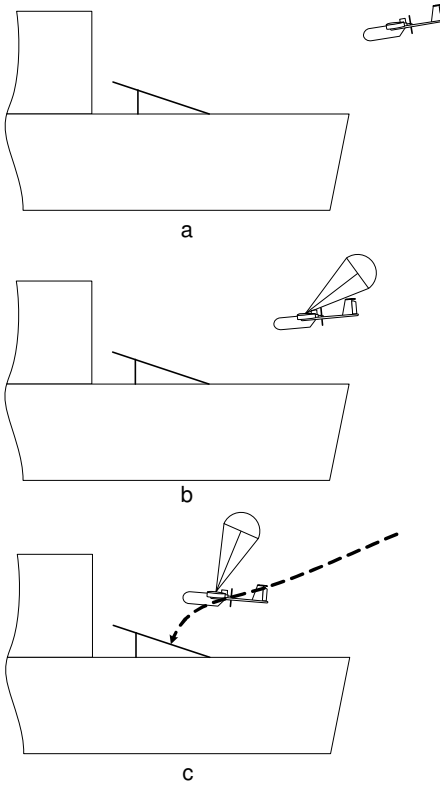


Fig. 2.5 Recovery using dynamic parachute deployment

an overshoot of several metres. This raises two major problems. First, the distance between the landing point and the UAV, as well as the relative speed between them, must be measured with extreme precision and a very high rate to determine the moment of deployment. GPS systems seem to be unsuitable here, so a dedicated tracking system must be installed. Second, the process of parachute deployment must be very quick and reliable in terms of time and stability. This undoubtedly leads to a complex pyrotechnic-assisted deployment system.

Although wind susceptibility of this method should be much lower than that of conventional parachute or parafoil recovery (due to both a smaller parachute size and shorter exposure to wind), it cannot be eradicated completely. Therefore, the wind over the landing point must be taken into account in advance, similar to conventional parachute systems. However, a principal advantage is that there is no need to *predict* the wind. The local wind may be constantly monitored and the data sent to the UAV via a radio uplink. This real-time data may then be used to determine the actual point of parachute deployment. Unlike the conventional descent on parachute, the region to be traversed on parachute is small and thus the last measured wind should not change significantly during descent.

From an engineering point of view, this technique is difficult for simulation and development, as often happens with unsteady transition phases. A precise dynamic model of parachute deployment must be designed; the model of the UAV at the post-stall angles of attack and sideslip may be required as well. All these models are highly specific and deserve a dedicated investigation.

2.3.4 Deep stall landing

A parachute is not the only option for slowing down an airborne aircraft beyond the normal flight envelope. Many aeroplanes can reach an extremely low and even zero airspeed in an unsteady dynamic manoeuvre. Such manoeuvres are often demonstrated by agile aerobatic and fighter aircraft and include Tailslide, Cobra, Hammerhead, Humpty-Bump and some others. As a rule, the aircraft cannot retain such a low airspeed and have to recover quickly from these regimes, often with significant altitude loss. However, some aircraft are able to maintain steady low-speed flight, using excess of thrust for both lift and control.

In nature, many gliding animals land in an unsteady manoeuvre which may involve stall aerodynamic angles. Most birds use wing flapping to produce additional braking force; however, sometimes they choose ‘gliding’ landing. Flying squirrels, which are unable to flap, and woodpeckers when landing on a tree trunk, use dynamic flare to lose speed.

The idea of using a similar dynamic manoeuvre for landing is being investigated for both manned and unmanned aircraft. In fact, free-flight aeroplane models (in particular, FAI F-1-A,B&C classes) have used deep stall for forced landing for many years. A similar-sized

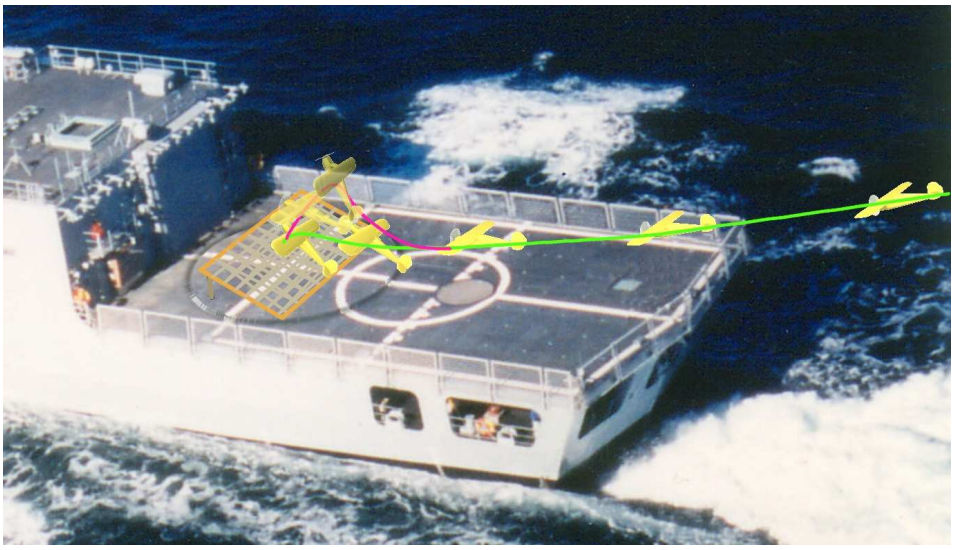


Fig. 2.6 Deep stall recovery

FQM-151 Pointer mini-UAV from AeroVironment employs a deep stall manoeuvre for automatic landing on a flat terrain [10]. However, these systems are unable to perform a precise 'spot' landing, required for shipboard recovery.

There are several possible techniques that allow achievement of a minimal speed at the moment of touchdown. Deep stall is only one of them, though investigated rather more than others. An example of recovery using this technique is shown in Fig. 2.6. At the final stage of approach, the UAV increases its angle of attack to very high, post-stall values via abrupt 'pull-up' elevator input. The wing partially loses its lift and increases drag, which causes the UAV to lose its speed rapidly. For a few moments, the aircraft will maintain a descent rate low enough for safe recovery. A horizontal or inclined shipboard elastic net may be used as an impact attenuator, similar to the parachute-assisted methods described before.

The actual form of the manoeuvre may vary (green and magenta paths on Fig. 2.6 are two examples) depending on aircraft characteristics, particularly wing loading, available thrust and elevator capacity. In a sense, this technique resembles dynamic parachute deployment (Section 2.3.3.4) performed without parachute.

Another method is 'perched landing', proposed in [46]. In this method, the UAV loses its speed in a steep ascending flare manoeuvre at normal (pre-stall) angles of attack. At the highest point of trajectory, the velocity is minimal, the UAV stalls and falls down to a prepared spot (e.g. the same elastic net, see Fig. 2.7). Theoretically, touchdown at a zero speed can be achieved. However, an

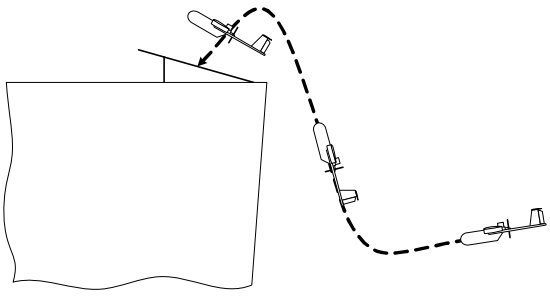


Fig. 2.7 Perched landing

elevated and unobstructed landing spot must be provided.

One of the most problematic challenges with all these forms of manoeuvre is stability and consistency of behaviour of the UAV during this phase. At such angles of attack and speeds, the aircraft can have very limited or no controllability. Moreover, its behaviour in these conditions is generally much more complicated and less predictable than that of a parachute. The post-stall behaviour is very aircraft-specific and should be carefully investigated for each particular UAV. For example, a faint horizontal asymmetry (either aerodynamic or mass, e.g. due to displaced payload) may result in quick roll development as the stall angles are reached [40]. Therefore, such manoeuvres put strong yet not clearly defined (in terms of actual configuration) constraints on the UAV design. Another drawback is that the optimal trajectory becomes highly UAV-specific as well, which may require different approaches to the controller design. Whilst for some aircraft a simple step elevator input produces satisfactory results [46], other UAVs may require a complex continuous control—only due to their aerodynamic configuration.

Therefore, considering the lack of controllability and complexity of dynamics, the manoeuvre must be carefully pre-planned, similar to dynamic parachute deployment described in Section 2.3.3.4. This involves the vital necessity of, firstly, a comprehensive mathematical UAV model, which is valid in the post-stall region of angles of attack—for the controller design; and secondly, a real-time data of the wind over deck to make necessary adjustments before entering into the regime with limited controllability.

The situation can be improved by the means of active attitude control. They enable the UAV to have partial or even full control on the trajectory during the manoeuvre, allowing, in some cases, a steady approach at post-stall angles of attack and extremely low airspeed. However, because the UAV's own aerodynamic control surfaces are ineffective at these speeds, either non-aerodynamic forces or some sort of jet control may be used.

Non-aerodynamic forces are limited to gyroscopic and gravitational forces. The latter can be manipulated by changing the position of centre of mass and by jettisoning spare weights. As a rule, such control is impractical for a fixed-wing aircraft. A more practical jet control can be of two general forms. The first one employs dedicated steering jet nozzles, as one can see on VTOL jets and spacecraft. Another option is propulsion thrust vectoring. This includes utilisation of the conventional control surfaces blown over with the slipstream of the propeller, as widely used by aerobatic aeroplanes. Thrust vectoring for flight control becomes more popular these years due to advances in engine technology and onboard control systems. It proved the ability to control the aircraft at the speeds below stall speed limit, as been spectacularly demonstrated by the modern super-manoeuvrable aircraft such as Su-30MKI and X-31.

Furthermore, the UAV can take advantage of the additional lift due to the vertical component of the propulsive force, which may be significant at such high pitch angles. This may further reduce the landing speed.

Being a promising technique, this recovery method has an inherent engineering problem. Although, in the simplest implementation, the UAV may carry no dedicated onboard recovery equipment (such as parachute or tailhook) other than the controller and perhaps the associated sensors, specific requirements to its flight dynamics demand that the UAV must be *initially* designed with such a recovery method in mind. It is rather unlikely that an existing fixed-wing UAV can seamlessly *adopt* this method of recovery. Even if it potentially can, a vast amount of additional investigation into post-stall dynamics of that particular UAV must be done. It is even more so if thrust vectoring or other special control methods are to be used. This makes it particularly difficult to employ the technique in this study, as it diverts the topic to the UAV design problem.

2.3.5 Convertible airframe

The recovery methods survey would be incomplete without mentioning the convertible aircraft. These aircraft are able to change their aerodynamic configuration in flight to provide adequate cruise and takeoff/landing capabilities. In line with the research, hovering convertible designs (i.e. essentially VTOL aircraft) will not be considered. Nevertheless, there is scope for these ideas.

The primary objective of the landing configuration is reducing the landing speed. Generally, there are three ways to achieve this objective: enhancing the wing lifting capabilities, increasing the wing area and providing an active lift by the means of dedicated or main propulsive engines.

One of the most widespread solutions for enhancing landing characteristics of the aeroplanes is wing reconfiguration. Along with the usual flaps, slats and other high-lift devices, sweep angle variation and active boundary layer control is used on some piloted aircraft. However, small UAVs, which are the focus of this research, are relatively low-speed vehicles with a not particularly wide speed range, thus their aerodynamic configuration is inherently of a low-speed nature. Boundary layer blow-off or suction requires, as a rule, a jet engine, which is rarely seen on small UAVs. There is not much room for such measures. Only simple flaps and less often, leading edge slats are commonly used on such UAVs to increase wing lift at low airspeeds. Although this helps to reduce the approach speed by some 20–50%, strictly confined areas of the shipboard landing decks usually require a more dramatic improvement.

Fig. 2.8 Eagle Eye tilt-rotor VTOL UAV

As a possible solution to reduce the approach speed significantly, the wing area may be increased accordingly, i.e. in the order of twice the size at least. It is believed that only a soft wing design can be utilised to achieve that without severe weight penalty. It can be implemented in several different forms: foldable delta (Rogallo) wing, web-like wingtip extensions similar to the wings of a bat, etc. In a sense, additional soft wing is similar to a gliding parachute (see Section 2.3.3.2) but probably offering better controllability, greater speed range, lower wind susceptibility as well as more stable and less stressful deployment. On the other hand, the framework of the soft wings (or wing extensions), as well as the stretched skin, may be more fragile than the airframe of the UAV, thus the requirements to impact attenuation may be higher. In addition, integration of the soft wings into the UAV is more problematic than that of a parachute or parafoil.

Another branch of convertible designs represents tilt-rotor, tilt-body and free-wing concepts. They take advantage of a flexible adjustment of the direction of propulsive force with respect to wing lift so that the configuration is maximally optimal for both low and high speeds. In other words, these aircraft are able to turn an unused at low speed portion of thrust into lift while maintaining the wing at an optimal angle of attack. It is possible to use a dedicated lifting engine, however it implies a severe weight penalty.

So far, the tilt-rotor design was used entirely for S/VTOL aircraft, both piloted and unmanned: Bell's *XV-15, V-22, Eagle Eye*. Nevertheless, it is believed that a simplified design with a limited degree of conversion may be used for short landing ability. On the other hand, tilt-rotor design requires a sophisticated mechanism for controlled rotation of the rotor axle (or of the whole engine set). Essentially, tilt rotor is a special case of propulsive thrust vectoring (see Section 2.3.4) with all its advantages and disadvantages: a high degree of control but complicated dynamics of the transition phase, high cost of development and strong implications on the overall UAV design.

In the context of non-VTOL aircraft, a more promising solution appears to be a mix of tilt-rotor and free-wing concepts.

Initially the free-wing concept has been developed for better gust response and stall characteristics [174]. The idea was to allow the wing to rotate freely about the spanwise hinges, which enables it to operate at a constant angle of attack with variable incidence, exactly opposite to conventional aeroplanes. A sudden increase of angle of attack, for example

Fig. 2.9 Scorpion free-wing tilt-body UAV

due to vertical gust, causes the wing to rotate and decrease its local angle of attack back to the specified value. However, although this theoretically allows to set a narrower safety margin and use higher wing angles of attack, lift control becomes more complicated. Indeed, extending the trailing edge flaps, for instance, will inadvertently divert the whole wing to a smaller angle of attack. In fact, there is no pitch control and no way to *put* the classic free wing to a desired angle of attack. Only speed (via thrust) becomes the source of lift (and therefore altitude) control—the wing automatically 'adapts' its angle of attack to the current speed. At any rate, lift of the clean wing limits the minimum airspeed and thus the landing characteristics.

In the late 1980s, the *Freewing Aerial Robotics Corp.* [76] made a significant improvement to the free-wing idea. Its design mates the free-wing concept with the tilt-body, which creates vectored thrust to obtain short takeoff and landing operation. The free-wing tilt-body *Manta* UAV, developed jointly by *Freewing* and *Veda Inc.* (now Veridian), and the *Marvel* UAV, developed together with Matra BAe (now BAE Systems), are designed especially for maritime purposes [146].

The manufacturer of the *Scorpion/Marvel* claims that this 260-kg (MTOW) UAV is able to land on a standard helicopter deck assuming a 20 knots (10 m/s) wind over deck [77]. Therefore, ship manoeuvring may be necessary, although this is often acceptable for recovery of a larger UAV.

2.3.6 In-flight arresting devices

So far, all discussed recovery techniques, except for the net assisted recovery (Section 2.3.2), were concerned about reducing the approach speed so that it would be possible to land the UAV on a confined deck area. However, this is not the only solution for shipboard recovery. Another tactics focuses on the methods of capture and safe deceleration of the UAV from the normal approach speed. Of course, approach speed should always be minimised. Nevertheless, only conventional for a fixed-wing UAV means of reducing the speed such as trailing edge flaps are considered in this context.

The original idea of the in-flight capture comes from the arresting wires used on air carriers. These wires, caught by the landing aircraft, provide enough braking force to stop the aircraft within a specified distance. However, even forced deceleration requires much of the deck space. In addition, such landing requires extreme touchdown accuracy, which is difficult to achieve on a constantly moving deck (see Section 2.3.1). On the other hand, it is not neces-

sary for a pilotless vehicle to run down the deck in order to reduce speed. A UAV, especially a small UAV, can adopt more extreme methods such as in-flight capture.

One of the techniques of this type is the aforementioned net assisted recovery. However, as noted in Section 2.3.2, this capture method leaves much to be desired, particularly from the point of view of stress on the airframe. A potentially more secure way can be derived using the experience of the air carriers' operations. To provide in-flight capture, the arresting wire should be extended outside the deck. This can be done with the help of a boom or a similar device. Unlike carriers' systems, the external arresting wire can be either horizontal or vertical.

2.3.6.1 Skyhook™

The small *SeaScan* and *ScanEagle* UAVs, currently under development by The Insitu Group [212], use the company's *Skyhook* technique, which employs a vertical suspended wire. The wire is freely suspended on a boom or is raised by a kite or a paraglider (see also Section 2.3.3.3). A self-locking hook is fixed on a wingtip of the UAV. On approach, the UAV flies directly into the wire so that the wire hits the leading edge of the one of the half-wings, slides towards the wingtip and locks itself into the hook (Fig. 2.10).

This method has been successfully flight-tested and seems to be effective for small or ultra-small UAVs. One of its advantages is simplicity. However, its performance in strong wind remains doubtful, because a freely suspended wire will do complex oscillations under the influence of wind and sea waves. Moreover, the amplitude of the tip of the high boom will be significant even in relatively mild sea. Fixing the loose end will solve the problem only partially. This is further complicated by the accuracy requirements. Apparently, the horizontal allowance is limited by the wingspan of the UAV. At the same time, the oscillations of the wire will be predominantly horizontal.

Another issue is strict requirements to the UAV design. The *Skyhook* technique requires the UAV to have a swept-back wing with a reinforced leading edge and with as long wingspan as possible. This requirement favours a tailless flying-wing design with high aspect ratio, which may not be optimal for certain applications.

Fig. 2.10 Skyhook™ recovery

2.3.6.2 Cable hook

An in-flight capture can be done with a more traditional arrangement: a horizontal arresting wire and a lowered onboard tail hook. The principal difference with the aircraft carrier systems is that the wire is stretched not on the deck, but high above or to the side of the deck using a boom, raised poles or similar facilities. Furthermore, the length of the hook is not limited by the height of the landing gear. Therefore, the hook can be implemented as a line of any appropriate length with a lock on its loose end. A longer hook gives a greater tolerance to the UAV position: the allowed vertical error is largely determined by the length of the cable-hook and the allowed horizontal error is determined by the length of the arresting wire.

An example of the recovery of this type is illustrated in Fig. 2.11. The recovery procedure may be implemented as follows. The approaching UAV extends a long cable with a self-locking hook attached. Passing above the boom, the cable slides over it and the hook catches the arresting wire. The UAV cuts off the engine. Deceleration of the aircraft can be provided in a number of ways: by an automatic onboard constant-tension winch, by a rotating damped boom itself, or by using an elastic or damped arresting wire, stretched along the boom, like that illustrated in Fig. 2.11. After successful capture, the UAV is lifted up with either onboard or shipboard winch.

To the author's knowledge, this technique has not yet been used for UAV recovery. The closest equivalent is parasail recovery [85] (see Section 2.3.3.3), which involves capture of the shipboard winch by the line suspended from the UAV. A credit must be given to A/Prof. Cees Bil for the original idea.

This method has several advantages over all others mentioned before. Firstly, this is simplicity of approach, which arises from relatively low demands on its accuracy. Also, there

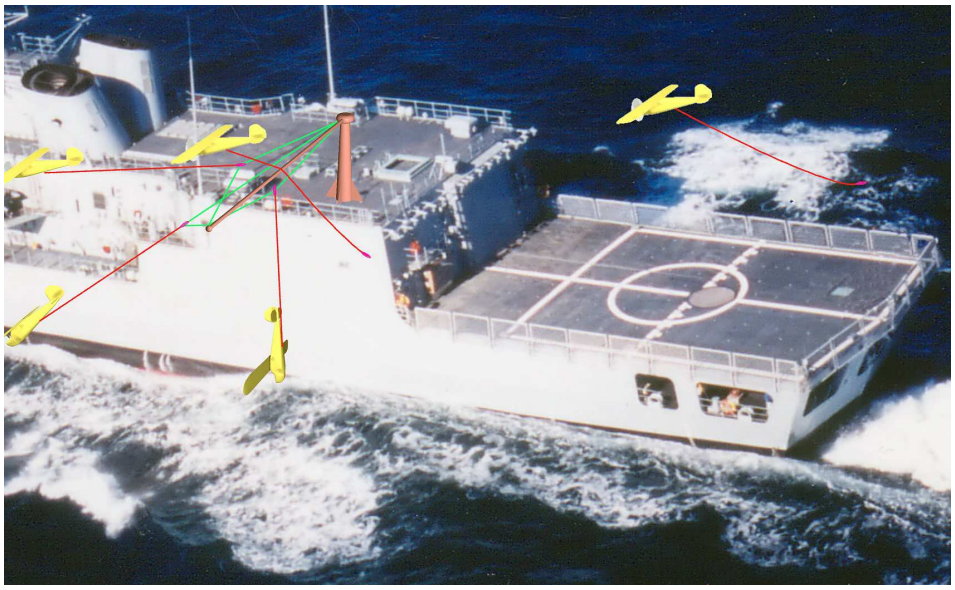


Fig. 2.11 Cable hook in-flight arresting sequence

are no unsteady transition flight stages with all associated complexity of dynamics and simulation. Moreover, the high location of the recovery boom may help to avoid a significant portion of ship airwake.

Secondly, it entails no specific requirements to the UAV design. Fitting the cable into the UAV, obviously, requires some engineering effort, especially considering the stress passed through the cable to the airframe; however, it is deemed to be not more complicated than fitting a parachute. A trailing cable seems to discourage the pusher-prop configuration because of the danger of damaging the cable by the propeller; nevertheless, it can be aided by installing a safety ring around the propeller, which can be also aerodynamically efficient.

The third advantage is safety for the ship and its personnel in the case of failure or a missed approach. If the UAV passes too high or too low, it can perform a go-around and try again.

The possibility to choose the approach direction deserves a more thorough investigation. Normally the UAV approaches the arresting wire perpendicular to it. Therefore, to attain arbitrary landing direction without making the ship change its heading, the arresting wire orientation must be adjustable. This can perhaps be most easily implemented using a cantilever boom to stretch the arresting wire, as shown in Fig. 2.11. Such a boom can be rotated to any direction (deck space permitting). At any rate, the ability to rotate will be required at least to stow the boom when it is not in use.

Nevertheless, orientating boom is not the only requirement for safe in-flight capture. A good clearance to any objects and water (ground) is required not only beneath the boom but also forward and backward (to the direction of flight), because the UAV will unavoidably swing on its cable hook during and after slowdown. It is difficult, if possible, to provide an adequate damping in this case. If the boom cannot be raised high above the deck so as to provide enough clearance (which is likely even for small UAVs, see calculations below), the only option remains is to put the boom off the board to take the benefit of the deck height above the waterline (as illustrated in Fig. 2.11). In this case, boom orientation becomes very constrained. Indeed, the boom position in Fig. 2.11 (perpendicular to the ship board) almost cannot be changed, otherwise the UAV risks hitting the ship, swinging either forward or backward after capture. As a result, with the boom arrangement shown in Fig. 2.11, only two approach directions remain feasible, which gives no advantage over many other recovery methods in this aspect.

From this point of view, a better location of the recovery boom can be found at the bow of the ship (Fig. 2.12). Potentially, it allows the choice of an approach direction up to 320 degrees, which enables the UAV to avoid undesirable wind directions (in particular, strong side and tailwind) virtually for any ship heading. In addition, a good ground (waterline)

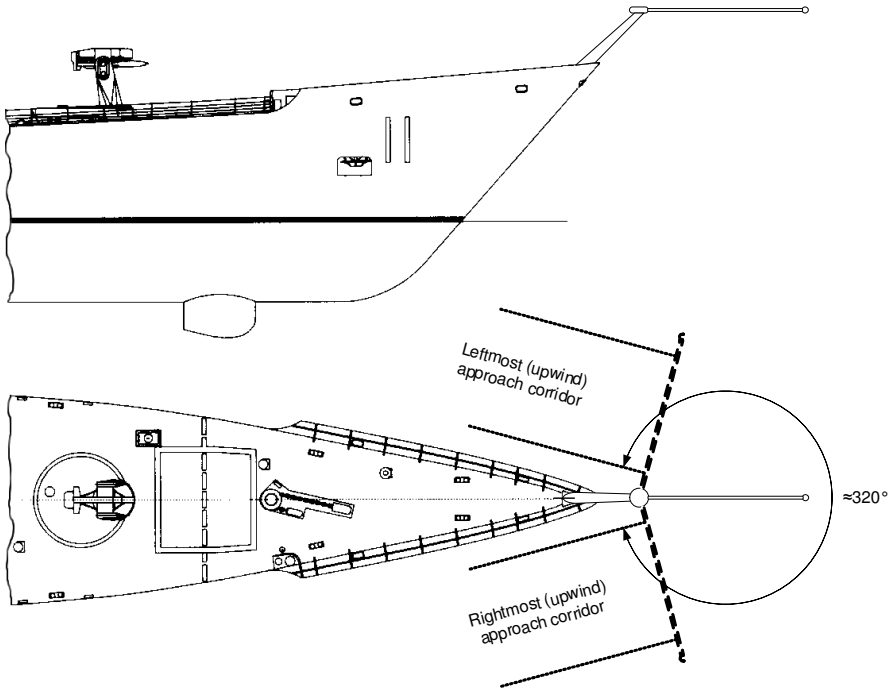


Fig. 2.12 Bow location of the recovery boom

clearance is easily obtained without a very high boom tower, because the nose tip of a ship is typically several metres above the deck level. However, this location, being far from the ship's centre of mass, tends to oscillate significantly in high waves. On the other hand, the angular amplitude of pitch and yaw motions of a ship is usually much smaller than that of the roll motion; therefore, the random displacement of the forward pointed recovery boom located at the bow may in fact be comparable to the side location (Fig. 2.11) at the same sea state. The actual amplitudes of the boom are investigated in Section 3.4.1.

It should be pointed out however, that the front location of the recovery device has an operational disadvantage for the current frigate vessels, because the whole aviation infrastructure (the helideck, hangars, etc.) is typically located at the stern, and the UAV may need to be delivered there after recovery.

Considering all the above, the cable hook recovery method has been selected for this research. It is more thoroughly investigated in the next sections.

2.4 Cable Hook recovery model

In this section, a general reference model of the recovery procedure accepted for this study is described. The design requirements for the recovery system are briefly investigated as well; however, more suggestions will be given after actual testings of the model are performed, see Chapter 6. The mathematical model of the key element of the system, the onboard

cable, is developed in Chapter 3, Section 3.5. Additional analysis of the influence of sea waves is given in Section 3.4.

It should be noted that engineering design of the recovery system as such is not a part of this work, although the actual properties of the materials involved are taken into account when necessary. It is believed that the actual implementation of the system (in particular, the hook, the winch and the shipboard equipment) is feasible. Also, it should be kept in mind that the following is rather a qualitative analysis in order to build a suitable model; the exact parameters will vary with the ship and aircraft dimensions and performance and can be specified when the actual project requirements are known.

2.4.1 Recovery procedure

The recovery procedure consists of several distinctive stages. Not all stages are modelled in this research; for example, the pre-landing descent and manoeuvres are not considered. It is supposed that at least a slow line-of-sight data link between the UAV and the ship exists so as to feed the UAV control system with the current ship location and velocity, as well as wind speed and direction, sea state and other parameters if necessary. This is not for real-time landing control but rather to guide the UAV home after the mission completion and to communicate about the best landing strategy in current conditions (for example, approach direction and speed). A real-time data link, however, may be helpful in some circumstances.

The stages of the recovery are as follows.

1. When the UAV is in the vicinity of the ship (a few kilometres), the recovery strategy is determined on the basis of current wind over deck readings and other local conditions. The system chooses the approach corridor and determines the entry point (where the glidepath starts) taking into account ship speed and UAV position.
2. The ship arms its shipboard recovery devices (extends the boom to an appropriate position and makes any other necessary adjustments). From this moment, the ship should maintain its speed and course.
3. The UAV is commanded to prepare for recovery and to descend to the specified point. Until this command is issued, the UAV should not descend below the highest possible obstruction (ship antennas, etc., about 50 m) and should circle around the designated point if it reaches the point before the command is issued. If the conditions change significantly and the current strategy becomes inappropriate or dangerous, the UAV may be commanded to go around at any stage of flight. In this case, the UAV climbs to a safe altitude and circles near the ship until new commands are received.

The ‘ready to recover’ command may be issued, if appropriate, in advance, when the UAV is on its way home. In this case, the UAV takes the quickest practicable way directly to the designated point.

4. Upon receiving the command, the UAV must reach the approach corridor entrance already in the landing configuration: flaps set to an appropriate position, the cable hook is extended to the full length, airspeed is reduced for landing, and velocity vector aligned with the glidepath. There are specified allowances for position and velocity errors at the beginning of the final approach.
5. The UAV performs the final approach. During this stage, the UAV configuration does not change. The UAV aims at a point which is optimal for the capture of the arresting wire, i.e. approximately half the vertical distance from the aircraft to the trailing hook (cable sag h_S) above the middle point of the arresting wire (Fig. 2.13).
6. If the final approach is performed well, the cable hook contacts the recovery boom, slides over it and locks on the arresting wire. As soon as the successful capture is detected (this can be done in a number of ways, for example, detecting a strong negative longitudinal acceleration or an increased cable tension), the UAV shuts down the engine. Further stages are provided by the shipboard equipment and crew. If the boom is missed, the UAV automatically performs a go-around as described earlier.
7. After the UAV comes to rest (to a practical degree), it is lifted up, the boom is swung so that the UAV can be reached from the deck, the cable lock is released and the UAV is removed by the crew.

From the above sequence, only stage 5 (final approach) and the choice of approach strategy (stage 1) is the area of interest of the following study. There are several reasons why the research is limited to these stages.

First of all, the final approach is the key element of the whole recovery process. If performed well, successful recovery is guaranteed (providing that the equipment does not fail). What happens before is a preparation for final approach and is made for the sake of conven-

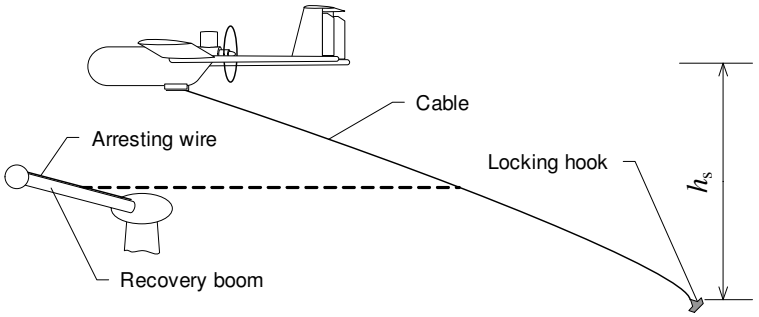


Fig. 2.13 Cable hook recovery configuration (not to scale)

ience for the final approach. Moreover, from the point of view of system control, this preparation (stages 3–4) is mostly a navigation planning task, which is quite different to the direct flight control required for the final approach. Meanwhile, developing flight control for the approach is by itself a difficult task, considering all the disturbances and uncertainties which the aircraft may encounter during this stage. The derivation of the flight control laws for the final approach is the core of this study and is presented in Chapter 6.

On the other hand, the exact procedure which takes place after the moment of capture highly depends on the actual layout and dimensions of the shipboard equipment. However, this study is not a project to develop a complete recovery technology for a particular UAV and ship; this is rather a development of a general methodology, which can be applied to produce a concrete recovery procedure for any UAV of the given class. Therefore, stages 6–7 are only concisely described in this chapter, and the mathematical models found later in this research consider only free flight before capture.

2.4.2 Approach corridor

The choice of the approach corridor depends generally on the UAV performance. If there are only two approach directions available (e.g. with the side boom location), the natural choice would be that direction which offers a lower impact speed (the UAV speed with respect to the ship), i.e. upwind direction with respect to wind over deck. This reduces the stress on the airframe and on the arresting wire. When the ship is underway, the UAV usually approaches from the stern, unless a strong tailwind is present.

There may be circumstances when the upwind approach is not possible. This may happen if wind over deck is greater than the UAV approach airspeed (for example, full ship drive with strong headwind). On the other hand, approach from the opposite side is equally impossible in such circumstances because it will result in an enormous impact speed. The only solution is to slow down the ship for a short time. This is considered to be a better compromise than to change course.

If a multiple choice of approach directions exists (for the bow location in particular, Fig. 2.12), the best strategy is determined by the combination of the UAV's capability to withstand side wind and to handle apparent oscillations of the recovery boom (which may be different from different directions). Detailed analysis of the recovery boom oscillations is presented in the following chapter, Section 3.4.1.

When the direction of approach is chosen, the entry point can be determined. The entry point is the location at which the final approach starts. The UAV is supposed to pass through the entry point already in the landing configuration. The entry point is calculated to be on the perpendicular from the centre of the arresting wire, taking into account the ship's future loca-

tion at the moment when the UAV is expected to reach the point. After passing the entry point, the UAV constantly tracks the recovery boom position and compensates the ship drive and wind by changing attitude. The distance of the entry point from the recovery boom is accepted in this study to be approximately 300 m from the arresting wire (or about 12–14 s of flight with no wind).

The initial altitude at the entry point is chosen so as to provide a moderate descent during the final approach. Since the reference UAV (*Ariel*, see Section 3.1.2.1) is initially designed as a ground-based aircraft, the glideslope is chosen to be rather small: 2°40', which implies about 14 m elevation (with respect to the arresting wire) at the distance 300 m. A steeper descent often causes saturation on the throttle (considering necessary manoeuvres) and the UAV accelerates even on the idle setting. However, since the vertical speed at the moment of recovery is not as important as for runway landing, and because the recovery boom is elevated, a generous allowance to the initial altitude can be given.

The UAV is assumed to be navigated to the entry point by a common navigation system, most likely the GPS and/or INS (see also Section 2.4.3.4.1) or using the data from the ship's radar, plus barometric altitude readings. From this assumption, the allowance for position error at the entry point is given in accordance with the estimated available accuracy of these systems. The errors are assumed to be normally distributed with a 10 m allowance for altitude and 20 m for sideways displacement to either side (with 95% (2σ) confidence). The distance error, being less important and more affected by the ship and UAV speeds, is assumed to have a 40 m error allowance (2σ).

The tolerance to the final position (at the moment of capture) is determined by the recovery equipment which is discussed below. Obviously, the sideways error depends on the length of the arresting wire and the altitude allowance is determined by the cable hook sag h_S (Fig. 2.13). A 0.5 m safety margin is allocated from both ends of the arresting wire and around the aircraft, i.e. if the UAV passes closer than 0.5 m from the recovery boom or catches the arresting wire less than 0.5 m from the end, the recovery is considered unsuccessful.

During the approach, the area is considered to be without obstacles, except for the ground (water). The UAV crashes if the altitude falls below h_S (cable sag) plus 1 m.

Table 2.1 summarises the parameters of the approach accepted in this study ($N(m, \sigma)$ denotes normally distributed value with a specified mean m and variance σ^2).

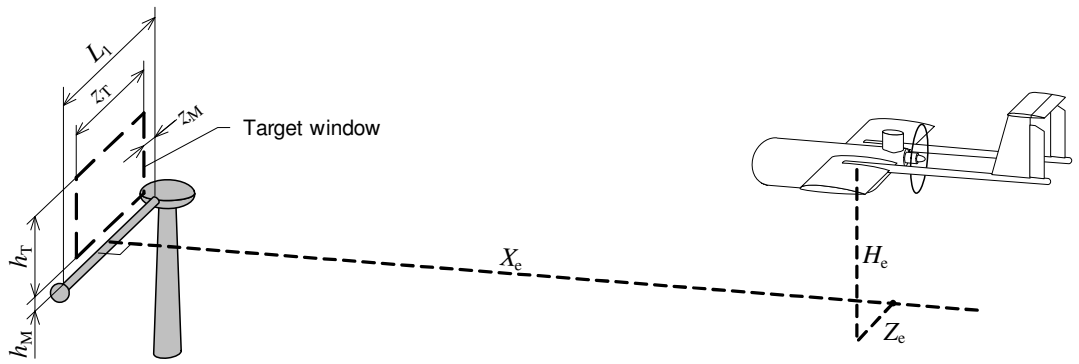


Fig. 2.14 Final approach entry point and the target window (not to scale)

Horizontal distance (X_e)	N(300, 20) m
Elevation (H_e)	N(14, 5) m
Lateral displacement (Z_e)	N(0, 10) m
Target window width (z_T)	$L_1 - 2z_M$
Target window height (h_T)	$h_S - h_M$
Horizontal margin (z_M)	0.5 m
Vertical margin (h_M)	0.5 m

Table 2.1 Approach corridor and entry point parameters (refer to Fig. 2.14 and Fig. 2.13)

2.4.3 Recovery equipment

As noted before, engineering design of the equipment is not the point of this research; however, a brief analysis of the possible location, dimensions and other properties of both the shipboard and airborne recovery equipment is made in this section. This should be regarded as a 'starting point' for the recovery model. Further research and testing may require to rectify or to change some of the parameters.

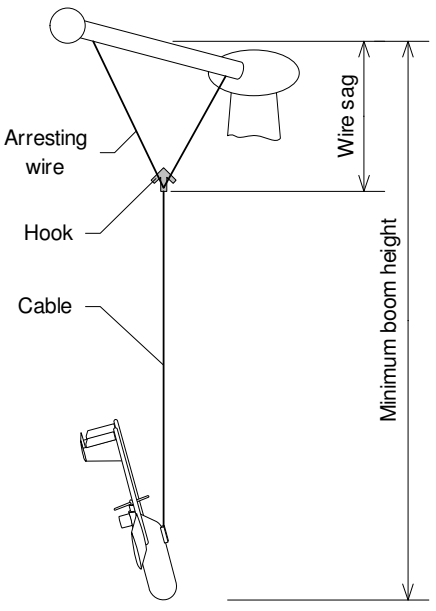


Fig. 2.15 Hang-down position after successful recovery

2.4.3.1 Length of the cable and height of the arresting wire installation

The recovery system stops the UAV virtually in midair. After coming to rest, the aircraft hangs down on the hook cable and the arresting wire (Fig. 2.15). This limits the combined length of the cable and the maximal sag of the arresting wire by the height of the recovery boom installation above the water level (or any other obstruction below).

Clearly, there must be a compromise between the sag and the cable length on the one hand and the height of the boom on the other hand. The

wire sag determines the stopping distance, at which the UAV slows down from its approach speed to zero (with respect to the ship). The longer the stopping distance, the softer the deceleration and the less stress on the airframe is passed. It should be noted, however, that the braking mechanism of the arresting wire may be designed so that it winches the arresting wire back as soon as the tension falls. In this case, the wire sag may be partly or even completely eliminated by the moment when the UAV reaches its lowest point, allowing softer braking and leaving more clearance to the water, thus enabling lower boom installation and/or a longer cable. On the other hand, such a construction may be more complicated and expensive.

Unneeded to say that a longer cable gives a greater tolerance to the vertical error of the flight path (see Fig. 2.13). This is especially valuable in rough sea, when the oscillations of the boom may be significant. As a lightweight cable poses little problem for the UAV (see Section 2.4.3.2), the cable length is limited mostly by the boom height. A higher installation is desirable. However, there are several limitations to this.

First, a high raised boom will have greater amplitude of oscillations, particularly due to ship roll motion. Since the periodic motion of the ship represents one of the toughest problems for recovery, the influence of the boom location on its random motion should be carefully investigated.

Second, a very high pole, tower or mast, which may be needed to raise the boom, is obviously an unwelcome construction on the deck. Although it may occupy little deck space, it must be a strong and most probably permanent structure, because it absorbs a several thousand Newtons dynamic load (see calculations below) at the very tip, not to say about its own weight and the boom weight and the wind loads. Given that at least a 6–8 m cable is required for reliable recovery (very possibly much more in rough sea), a 10 m height is needed as the very minimum. Meanwhile, even a 10 m high mast capable to withstand such a load is a bulky and expensive construction.

A natural solution to this is to put the boom off the board and to take benefit of the height of the deck level above the waterline, as illustrated in Fig. 2.11 and Fig. 2.12. However, not all locations are available for such installation, taking into account the requirements outlined in Section 2.3.6.2 above, particularly the clearance all around the boom.

The highest possible location is at the top platform of the central mast of the ship, which is at the 26–27 m level (the measurements are taken from the scale drawing of the Perry Clark frigate, Fig. 3.8). However, it is unclear if this area can be used for this purpose. Not only a strength analysis of the mast is needed, but also such things as radio interference with the radar and numerous neighbouring antennas may be necessary, especially taking into account the landing positioning radio system (Section 2.4.3.4).

Two of the most practical locations for the recovery boom can be found on the upper deck of the ship (Fig. 2.11) and on the bow (Fig. 2.12). The height of both the upper deck and the nose tip of a 4000-ton frigate (Fig. 3.8, Fig. 3.9) is about 9 to 11 m above the waterline. Thus, with a reasonably sized 3–6 m tall pole, a practical 12–17 m clearance may be obtained. The advantages of these locations were discussed in Section 2.3.6.2 and the model locations will be thoroughly examined in conjunction with the ship model in the following chapter (Section 3.4).

In order to determine available length for the cable, stopping distance should be estimated. As noted above, it should be taken into account when the wire brakes do not recoil after stopping the UAV, leaving a sag in the arresting wire.

The aircraft is slowed down from its actual landing speed. For the *Ariel* UAV, employed as a reference aircraft in this study (see Section 3.1.2.1), the recommended approach airspeed with flaps at 40 degrees is approximately 26 m/s [153]. The wind and ship's own speed will, obviously, have an effect on the impact velocity. However, even the most constrained choice of two approach directions (with the side boom location, Fig. 2.11) allows to use the wind and the ship drive for the benefit of the UAV, i.e. to reduce the closing (and impact) speed, irrespective of the ship heading. Nevertheless, for safety reasons, a slight allowance for tailwind should be given. A figure of 30 m/s will be used as the maximum impact speed $V_{\text{imp.max}}$ for reference. For a small aircraft such as *Ariel*, the deceleration a of the order up to 10g (≈100 m/s^2) is deemed to be appropriate. This yields the stopping distance

$$l_s = \frac{V^2}{2a} = 4.5\,\text{m} \tag{2.1}$$

and twice as much for a more generous deceleration of 5g, given the deceleration is uniform. Comparing these figures with the stopping distance available using the much tested Net recovery (Section 2.3.2) with significantly larger UAVs (such as *Pioneer*), it can be reasoned that even the first figure is rather conservative.

Having a maximum 17 m of water clearance and leaving a 2–3 m safety margin (this should be enough for calm weather, but more may be required for high waves), it is possible to allocate about 10 m for the cable (refer to Fig. 2.15). This value will be used as a starting point. If testing of the landing controller (Chapter 6) shows that this length is not enough for a good success rate, a higher location for the boom may be requested, or a quick arresting wire recoil may be implemented.

2.4.3.2 Cable strength

Such a rapid slowdown (up to 10g, as given above) sets rigorous requirements for the cable strength. For 10g deceleration, the inertial force from the 35 kg (maximum mass) *Ariel*

UAV will be $F_s = m \cdot 10\text{g} \approx 3500$ N ≈ 350 kgf. It is assumed that the UAV shuts down the engine immediately after capture of the arresting wire, thus no additional thrust is taken into account. However, even if it does not, the actual landing thrust of a few kgf is negligible for this qualitative estimation. Taking a reasonable dynamic load safety factor 2,[1] a figure of 700 kgf for ultimate tensile strength is obtained.

This strength can be achieved by using a standard 1/4" (6.35 mm) twisted nylon rope. According to various online sources (e.g. [6]), its breaking force is approximately 750 kgf. At the same time, its linear density is 22.3 g/m, thus a 10 m cable will have the weight of only 0.223 kgf. In addition, nylon has excellent capabilities to withstand dynamic loads, resistance to wear and to the wet and chemicals. If greater strength is desired and/or weight requirements are tough, stronger and more expensive materials such as Kevlar/composite fibre may be used.

Apart from tensile properties, aerodynamic drag of the line is important. It has an effect on both the aerodynamics of the UAV and the shape of the cable while in flight. The method of drag calculation follows that in [213] and is described in Section 3.5.3.3.

2.4.3.3 The hook

The purpose of the hook is to catch and hold the arresting wire. It is attached at the end of the recovery cable. Tail hooks fitted on the piloted aircraft are extensively used for the same purpose on air carriers. However, there are additional requirements for the in-flight capture of a UAV.

First, the hook should lock securely on the arresting wire, because the UAV remains on the wire for a much longer time than a conventional aircraft landing on the deck. After slowing down, the UAV, attached to the arresting wire by the recovery cable, becomes a pendulum with a significant amount of energy. Moreover, due to spring forces in the nylon cable and the recoil of the arresting brakes, the cable may get slack. The hook must not detach in these conditions. At the same time, it must slide easily along the arresting wire in order to maintain a symmetrical load at both ends of the wire when the wire is captured off centre.

Second, such a long flexible line virtually does not pass torque; moreover, twisted rope can rotate under load. Therefore, the hook attached at the end of the cable should be ‘omnidirectional,’ i.e. able to catch the wire at any (lateral) angle.

[1] For static loads in typical use, the maximum load on a rope does not usually exceed 15 to 20% of the breaking force. However, taking into account strict requirements to the minimum weight and typically better stress response to short dynamic loads, a lower safety factor is deemed to be more reasonable. Nevertheless, a more thorough analysis will be needed for actual design.

A possible conceptual design of the hook is presented for illustrative purposes in Fig. 2.16. However, as the actual design may vary greatly and is unknown at this stage, the parameters for the model are chosen arbitrarily: hook mass of 0.1 kg and for the purposes of aerodynamic drag calculation, the shape of a ball 5 cm in diameter. Although real parameters may be significantly different, this is not very important for the research, because the model, as well as the whole work, provides rather a proof-of-concept methodology of design. When the exact parameters are known, they can be easily adapted by the model.

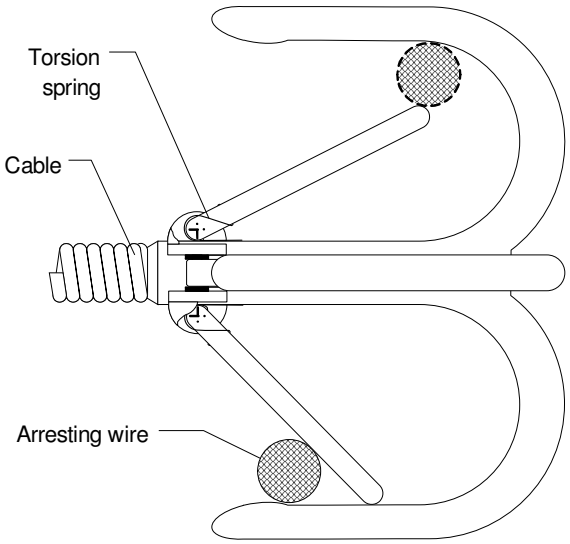


Fig. 2.16 Arresting hook (design concept). The device can have 4 (as shown) to 6 individual hooks, which may be collapsible to reduce dimensions when inside the aircraft. The upper and lower parts on the figure demonstrate different stages of capture.

A full cable hook model is presented in Chapter 3, Section 3.5.

2.4.3.4 Local positioning system

One of the most difficult engineering problems standing before the designer of the recovery system is to provide real-time information about the current position of the UAV with respect to the arresting wire during final approach. Taking into account that in rough sea the periodic displacement of the arresting wire position may exceed its length (an analysis of the amplitude is given in Section 3.4), aiming at an 'average' central position is not possible. The UAV must track the actual position of the arresting wire and change the flight path accordingly.

It is also important to note that the UAV relies on the positioning system for altitude measurement during final approach. The barometric altitude sensor has far too insufficient accuracy for recovery; at the same time, it is unlikely that a small UAV will be equipped with a radio altimeter.

2.4.3.4.1 Positioning with GPS and INS

A Global Positioning System (GPS), whether NAVSTAR, GLONASS or forthcoming GALILEO, is a convenient system for navigation and is likely to be a standard part of the onboard equipment. It uses radio signals from several satellites orbiting the Earth on precisely calibrated orbits. However, it has several shortcomings which arise from its original design and purpose [87, 165].

First of all, with the size of the landing window of about 3–5 m (horizontal and vertical allowance at the moment of capture of the UAV), a sub-metre positioning precision (probably of the order of a few centimetres) is needed. Such accuracy (although with respect to the world coordinate system) is achieved by GPS only in the differential mode, which uses the additional signal of a ground-based station (an example of successful application of the DGPS to full flight control can be found in [150]). Needless to say, such stations will not be available at open sea. Moreover, the altitude measurements, which are perhaps the most important for the UAV, are significantly less accurate than horizontal coordinates for GPS systems. However, this can be circumvented by synchronising GPS-derived positions of both the UAV and the ship. After all, only their relative position (and velocity) is required. In this case, the ship constantly sends its GPS coordinates to the UAV via a real-time data link, the UAV controller compares them with its own GPS coordinates and calculates the relative position.

Another difficulty lies in the time domain. The nature of accurate GPS measurements relies on Kalman filtering [87], which requires substantial computational power and time to reach high accuracy. In most commercially available GPS receivers, the position update rate is about 1 Hz, and up to 5 Hz is available for more expensive units.[1] In general, the better precision and confidence is desired, the longer observations are needed. Moreover, a lengthy start-up procedure (many seconds or even minutes) may be needed for some of the most precise modes (in particular, the so-called *Real-Time Kinematic (RTK)* mode [106]), which can be performed only when the UAV is already in the vicinity of the ship (or the base receiver).

The situation can be improved by mating the GPS and inertial navigation. Since the inertial navigation system (INS) is supposed to be a part of the standard UAV onboard equipment, this combination may be a wise choice not only for recovery but also for other regimes. Indeed, the INS can give a very accurate estimation of the current position, but only in a short-term period: its precision degrades over time due to accumulating errors. In contrast, GPS errors are not accumulating and thus the measurements are independent of time (neglecting variations in satellite constellation) but they are not as precise and may not be available at any instant. Therefore, by judicious combination of these systems (for example, simultaneously filtering both the measurements with a Kalman filter [87]), a very accurate real-time positioning can be achieved.

However, this system, although feasible, is fairly complex. It requires two sets of GPS receivers and INS (on the UAV and on the ship) with a real-time radio data link between them. Modelling and simulation of such a system is out of the scope of this study. In addition, it highly depends on a third-party system (GPS), on which there is no control and which can

[1] Higher update rates (up to 25 Hz) are specified for some industrial GPS receivers, such as NavCom SF-2050M, which can hardly fit a small UAV.

be jammed relatively easily. Nevertheless, the potential feasibility of this system allows us to assume that local positioning of the UAV will be available.

2.4.3.4.2 UAV Common Automatic Recovery System (UCARS)

The UCARS AN/UPN-51(V), together with its derivatives UCARS-V2 and TALS (Tactical Automatic Landing System), is developed by Sierra Nevada Corporation [193] exactly to provide automatic recovery and take-off for both fixed and rotary-wing UAVs, either shipboard or ground-based. It employs the same principle as used in the Transponder Landing System (TLS) for conventional piloted aircraft, which provides standard glidepath information similar to that of the localizer and glideslope systems (ILS). The UCARS consists of two separate pieces: an airborne transponder (weighing about 1.5 kg) and a ground (or shipboard) track subsystem. The latter has a millimetre-wave (35 GHz) radar which automatically locates and tracks the transponder. The radar platform uses its own sensors for ship motion compensation which do not require a GPS signal.

The UCARS has been initially developed for the *Pioneer* UAV, which uses a net for shipboard recovery (see Section 2.3.2). The system has been successfully tested in 1997 and is currently used or tested with the *Shadow 200* tactical UAV (TALS variant), *Fire Scout* rotary-wing TUAV (V2 variant), *Hunter* UAV and some others.

The system has a great advantage of being a standard piece of equipment which can fit many UAV types (or at least it is supposed to be). However, its specifications (especially concerning tracking ability and ship motion compensation) are unavailable for the general public. In addition, the transponder may be considered too heavy for the selected UAV (Section 3.1.2.1) and consuming too much power, although in general the UCARS can be used with small UAVs (for which the recovery system is intended). Nevertheless, similar to GPS, the potential availability of the UCARS option ensures that the UAV position will be known with acceptable accuracy.

2.4.3.4.3 Distance metering and triangulation

A more basic way of determining the position is to measure the distances to three (or more) points with known locations and then to resolve them into the position via a classic triangulation task (Fig. 2.17). If these points are located on a moving ship, the UAV will always determine its relative position, taking into account ship's drive and periodic motion.

There are several methods of measuring the distance, including ultrasonic echoing, laser metering, opti-

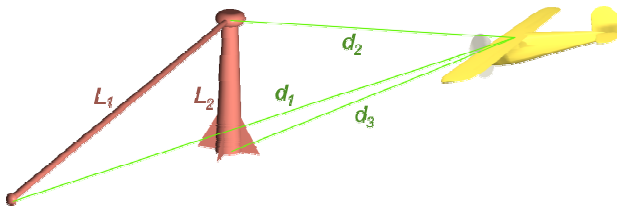


Fig. 2.17 Determining position via three distances

cal image recognition and various radio methods. Bearing in mind weather conditions, the amount of motion and other limitations, it may be reasoned that the radio metering is the most suitable.

One of the most practical implementations of radio distance metering is comparing the phase difference between the original signal emitted by the 'master' transmitter and its echo, reflected or retranslated back by the other side. Knowing the phase shift $\varphi_0 - \varphi_1$ and the frequency f of the signal (or the wavelength $\lambda = c / f$, where c is the speed of light), the double distance which the radio waves had to travel to and back can be obtained with high accuracy. A phase difference of 2π denotes the double distance of one full wavelength:

$$
\begin{aligned}
&\text{Transmitted signal:} \quad s_0(t) = A_0 \sin(2\pi f t + \varphi_0) \\
&\text{Received signal:} \quad s_1(t) = A_1 \sin(2\pi f t + \varphi_1) \\
&\forall t = t_1, \text{ the phase of the received signal } (2\pi f t_1 + \varphi_1) \text{ is the reflection} \\
&\text{of the phase of the original signal at } t = t_1 - \Delta t = t_0\text{: } (2\pi f t_0 + \varphi_0) \\
&\Rightarrow \text{ phase shift } \quad (2\pi f t_1 + \varphi_0) - (2\pi f t_1 + \varphi_1) = (\varphi_0 - \varphi_1) \Leftrightarrow \\
&\qquad\qquad\qquad\quad (2\pi f t_1 + \varphi_0) - (2\pi f t_0 + \varphi_0) = 2\pi f (t_1 - t_0) \Rightarrow \\
&\text{Distance:} \quad d = \frac{c}{2}(t_1 - t_0) = \frac{c}{4\pi f}(\varphi_0 - \varphi_1) = \frac{\lambda}{4\pi}(\varphi_0 - \varphi_1)
\end{aligned}
\tag{2.2}
$$

There are two important notes, however. First, the frequency of the received signal will not be the same as the original frequency for moving objects due to the Doppler effect. This must be taken into account in calculations. In addition, this enables to measure speed with respect to the transmitter, which may be useful.

Second, the phase difference repeats with the period 2π, which cannot be detected unless the signal is specially modulated. This means that the wavelength used for detection should not be shorter than the maximum double distance to be measured, otherwise the reading will be ambiguous. However, increasing the wavelength adversely affects the accuracy of measurements. This problem can be avoided by using two or more signals with slightly different frequencies. In this case, the maximum double distance will be determined by the smallest difference between the frequencies.

Note that the frequency (or multiple frequencies) used for measurements is not limited by the available radio bands. It can be carried by a higher frequency radio wave using one of the numerous modulation methods. In fact, this is required for the current application, firstly because the frequencies needed to measure the distances of the order of several hundred metres (i.e. about 0.5 to 5 MHz, wavelength 600 to 60 m) are far below the convenient radio bands; and secondly because it allows to take several measurements simultaneously using several different carriers and a multi-channel receiver (transponder) without mixing them.

As a result, the local positioning system may be implemented as follows. Three transmitters are placed at three different locations on the ship. They emit three radio signals on different channels f_{c1}, f_{c2} and f_{c3}, modulated with a relatively low frequency measurement signal s. This signal may be the same for all channels, either single- or polyharmonic, depending on the accuracy requirements. The UAV is equipped with a three-channel transponder which receives each of the three signals, extracts the measurement signal, modulates with it the responding carriers f_{r1}, f_{r2} and f_{r3} (separately for each signal) and radiates them back. Different channels for reception and retranslation (f_{ci} and f_{ri}) are needed to avoid interference of transmitted and received signals on the ship side. The ship is also equipped with three receivers located at the same points as the corresponding transmitters. These receivers receive the signals f_{r1}, f_{r2} and f_{r3}, extract the measurement signals s_{r1}, s_{r2} and s_{r3} and estimate the phase difference between the emitted signal s and the corresponding received signal s_{ri}. The distances d_1, d_2 and d_3 are then calculated according to (2.2). These distances (or a ready-calculated position) are immediately transmitted back to the UAV using either a common real-time data link (if available), or the same carriers f_{ci} (providing that this data signal does not interfere with the measurement signal s, which can be achieved, for example, using a sub-carrier with an intermediate frequency). The position is calculated geometrically from the distances either by the UAV controller or on the ship. At the same time, speed with respect to each transmitter is determined via Doppler shift.

This scheme can be implemented the opposite way, with the shipboard transponders. This will eliminate the need of the data uplink to the UAV (for the positioning purpose at least). However, having the transmitters on the ship offers an advantage when only the offset of the UAV position from the flight path is needed. A measure of this offset can be obtained by evaluating the difference between the distances themselves. For example, if two transmitters are located symmetrically around the ideal glidepath, e.g. on the ends of the recovery boom (transmitters 1 and 2 in Fig. 2.17), a zero difference between the distances ($d_1 - d_2 = 0$) would mean that the UAV is on the ideal glidepath within the respective plane (that contains both transmitters and the UAV); while a non-zero difference would indicate that the UAV had diverged from the ideal path. Assuming neutral (horizontal) boom position, the difference ($d_1 - d_2$) controls lateral displacement of the UAV, while difference ($d_3 - d_2$) determines vertical position, somewhat similarly to localizer and glideslope indicators of the common piloted aircraft.

Note that the differences are nearly proportional to the angular displacement ε of the UAV (as seen from the ideal touchdown position between the transmitters) rather than to the linear offset from the ideal path (see Fig. 2.18). This is convenient because the offset is more

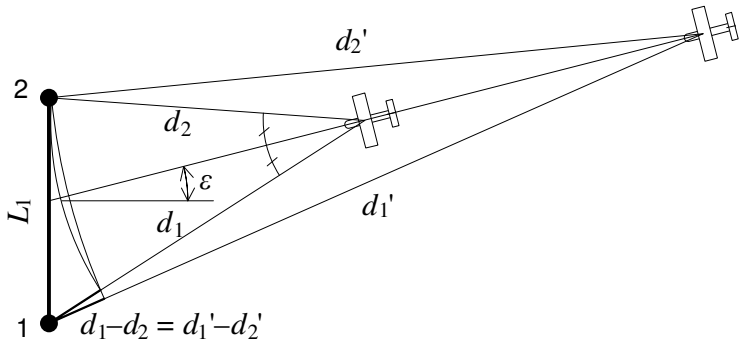


Fig. 2.18 The essence of the distance differences

accurately measured near the ship while a greater allowance is given at the beginning of approach (see also discussion of the landing procedure in Chapter 5).

The differences can be measured without knowing the distances themselves by comparing the phases of the measurement signals s_i that come from different transmitters, providing that the signal is synchronised among the transmitters. This can be implemented onboard the UAV, which eliminates the need of the transponder and makes the onboard positioning equipment fully passive, with all associated weight and power consumption savings. The lack of knowledge of the distance from the ship can be compensated, if required, using GPS. The accuracy requirements to this distance are usually several order of magnitude lower (i.e. several metres) because there is no need to track the distance precisely on landing. Even if the exact position is required and the distances are measured, such a scheme allows to measure the divergence of the flight path with greater accuracy.

It can be noted that the accuracy of the calculated position depends on the separation of the transmitters in space. If the transmitters are located too close to each other, the distance from the UAV to any of them will be nearly similar and the calculation of the position will be inaccurate. This effect is known as *dilution of precision* and is also characteristic for satellite-based GPS systems. However, it is deemed that for the current task this effect will represent less of a problem, or can even be beneficial to a certain degree, because the accuracy requirements are different at different distances from the ship. Indeed, high accuracy is required when the UAV is close to the ship and needs very good alignment with the recovery boom. At close distances, such accuracy can be achieved because separation of the transmitters (L_1 and L_2 in Fig. 2.17) becomes comparable to the distances themselves (d_i). In contrast, a much greater allowance may be given at far distances, otherwise the UAV controller may be excessively disturbed by the changes in measurements caused by periodic ship motion. In other words, as mentioned above, the angular position of the UAV should be more important for successful recovery than the actual coordinates.

Apart from position measurement, velocities of the UAV with respect to the recovery boom may be required. As stated above, translational velocities with respect to each transmitter can be measured using the Doppler shift of the same signal. In addition, an approximate measure of angular velocities can be obtained at no extra cost using the difference derivatives $d(d_i - d_2)/dt = \dot{d}_i - \dot{d}_2$, where $i = 1$ or 3 is the number of transmitter (similarly to the angles, $i = 1$ for horizontal angular velocity and $i = 3$ for vertical one). Once again, these derivatives can be measured without unreliable numerical differentiation of either the distances or the differences. This can be done by comparison of the frequencies of the signals coming from the respective transmitters. If the closing velocities relative to each transmitter are different, the Doppler shift will also be different. Even if this difference is too small to be detected reliably by independent measurement of the Doppler shifts, the comparison of the frequencies of the two signals can be done very accurately. From this, the distance difference derivatives can be obtained. In Chapter 5, calculation of the actual position and velocities from the raw measurements will be discussed in greater detail.

Overall, this is a simple and effective mechanism for local positioning. For this reason, it is adopted for this study. However, design of such a system is out of the scope of this research. On the other hand, without at least a draft of this system, the specifications (such as precision, latency, etc.) are unknown and can only be speculated about. Therefore, the positioning system is considered to be ideal in this study. It obtains exact distances, differences between them and their derivatives from the simulation data. Nevertheless, to provide a measure of robustness of the UAV controller to the uncertain characteristics of the system, noise can be added to the measurements.

For simplicity and because it is indeed believed to be optimal (unless better positioning accuracy and thus transmitters separation is needed), the transmitters are located as shown in Fig. 2.17: two at the ends of the recovery boom and one at the base of the supporting tower, right below the second transmitter, so that the base legs are $L_1 = 6$ m and $L_2 = 5$ m (see also Section 3.4). The transmitters are numbered according to the distances d_i shown: #1 at the tip of the recovery boom, #2 at the other end and #3 at the base of the tower.

It is assumed that the location of the transmitters on the ship, whether fixed or changing/switching, is known to the UAV at any time. However, even for a fixed layout, the geometry of the transmitters from the point of view of the UAV may be different depending on which side the UAV is approaching from. In Fig. 2.17, the supporting tower (and the whole ship) is located to the right of the approaching UAV; however, an approach from the other side can be performed so that the ship is to the left of the UAV. Moreover, for given measurements and known transmitters layout (where only three transmitters are present), an ambiguity exists as to which of the two possible UAV positions is effective. It is supposed, how-

ever, that this ambiguity and the variance in apparent geometry is worked out by knowing the initial formation from the navigation system. For this reason, only one-sided configuration (as shown in Fig. 2.17) will be considered in the study. The changes in configuration are expected to be transparent for the landing controller. One of the ways to facilitate managing 'mirrored' transmitter layouts for the UAV controller is switching the numbers (codes) of the transmitters according to anticipated direction of approach, so that #1 is always the leftmost transmitter and #2 is the rightmost one with respect to the UAV. The difference $(d_3 - d_2)$ is accordingly replaced with $(d_3 - d_1)$ for the opposite direction.

Altogether, the positioning system provides ten measurements to the UAV controller:

- d_1, d_2 and d_3: distances from the UAV to each of the three transmitters. It is supposed that the UAV's antenna is located close to the centre of mass, or the distances are adjusted as if they are measured with respect to c.g.
- $\dot{d}_1$, $\dot{d}_2$ and $\dot{d}_3$: rates of change of the distances, i.e. closing speeds with respect to each transmitter (positive when approaching for convenience).
- Distance differences $(d_1 - d_2)$ and $(d_3 - d_2)$. (The third difference $d_3 - d_1$ is disregarded as redundant).
- Rates of change of the differences: $\left(\dot{d}_1 - \dot{d}_2\right)$ and $\left(\dot{d}_3 - \dot{d}_2\right)$.

Although for an ideal system this set of measurements is redundant, the rates and differences are introduced as independent values to simplify possible integration of the complete model of the positioning system, where these parameters may be measured separately as described above. In addition, for robustness tests the noise can be added independently to each type of measurement.

2.5 Concluding remarks

Shipboard UAV recovery is one of the most difficult tasks of the entire flight. Unlike conventional runway landing, which is suitable for aircraft from air models to large transports, shipboard operation imposes extremely tight constraints which make the engineers search for the most suitable ways for each aircraft class. A number of very different methods of recovery have been developed over the last decades and more are still being envisaged, developed and tested. Some of them are analysed in this chapter, the analysis leading to development of a new recovery technique. The technique, termed *Cable Hook recovery*, is aimed to recover small fixed-wing UAVs up to the size of *Pioneer* UAV with better operational qualities than the most widely used *Pioneer*'s net recovery. The goal is to provide automatic recovery under all conditions that the UAV may encounter at open sea, including high waves (and thus periodic ship motion), winds, turbulence and undesirability to change the ship's heading while underway.

The Cable Hook recovery offers the choice of at least two directions of approach, allowing to benefit from the wind irrespective of the ship and wind heading. The tolerance to wind and ship motion is reasonably expected to be greater than that of most other methods, particularly the net assisted recovery, considering all the imposed requirements.

The recovery system consists of the shipboard components: a recovery boom with the arresting wire over it, a boom supporting tower or mast, a local positioning system and an airborne cable with a self-locking hook. This equipment is described only to the extent needed to build a general simulation environment of the recovery procedure. Before any hardware implementation is possible, a comprehensive design of the recovery gear must be made. Although there may be significant discrepancies with the current model, the controller design methodology employed in this research can handle the changes in the models without difficulties (see Chapter 5 and Chapter 6).

The following study is concentrated on the final approach of the UAV. The parameters of the approach as well as the dimensions of the recovery gear necessary to build a simulation environment have been described in this chapter. However, these parameters should be regarded as a starting point; further investigations may require some of them to be reviewed.

Chapter 3. Simulation Models

A critical part in the control system design is development of the mathematical models of the systems involved. Very often, for the purpose of control system analysis and design, simplified models (e.g. linear models) are used; the resulting controller is then tested on the original comprehensive mathematical models and/or real equipment. This two-stage approach comes from the fact that although many of the numerous contributing real world factors can be measured and taken into account, this leads to very complex, highly coupled non-linear models. Whilst such models can be built without principal difficulties, they generally require appropriately complex control methods, which are hard to design.

Nevertheless, the design method employed in this study (described in the following chapters) allows to synthesise the control laws on the basis of the full non-linear models. On the other hand, it has high demands to computational efficiency of the models, because hundreds of thousands simulation runs may be required. Therefore, building comprehensive yet effective models of the elements involved in the recovery process is a critical part of the entire design. In some cases, full models may be simplified for faster computation during the design stage. However, the full variants can be used for validation and verification of the designed controller at the later stages.

There are number of elements involved in the case of ship-based UAV recovery. They include the aircraft, the ship, the recovery devices (the airborne recovery cable and the ship-board boom), plus the environment, including various disturbances from the wind and sea. All of these elements, as well as interactions between them, must be carefully modelled.

This chapter will present the mathematical models used for simulation of the cable hook UAV recovery. These models include the aircraft model, the ship model (which incorporates sea disturbances), the recovery cable model and the atmospheric model. Most of these models are presented in the usual manner, so specific details are given only where the models differ significantly from a conventional approach.

3.1 Aircraft model

This section presents the model of an aircraft, used for simulation of the recovery stage of the flight. As some sort of computational analysis is implied, a computer implementation of the model should be kept in mind during design. For example, using tabulated data may be preferred over the analytical functions in some cases (or vice versa).

For the sake of compatibility, the same reference UAV and mathematical model is used for both launch and recovery. The launch model has been designed by M. Crump in his

Ph.D. thesis [47]. The description in this section generally follows Crump's model; however, a few modifications were made to make the aircraft model more suitable for landing conditions. Also, it should be noted that different axes system and nomenclature (Appendix A) are used in this research. The computer model of the aircraft has been completely redesigned as well to improve performance and readability.

It is a common practice to separate the mathematical model of an aircraft into two principal components: the dynamic model and the force model. The dynamic model generates the motion of the aircraft under the influence of certain forces. The force model generates the forces acting on the aircraft under the influence of its motion and position. It is clearly seen that these two models are mutually dependant and thus closely linked together.

Given an aircraft velocity and position, the forces acting on the aircraft can be generated. These forces then interact with the aircraft through the equations of motion, which produce a new aircraft velocity and position. The process is repeated many times for a specified length of time or until certain conditions are met. The following sections will detail the equations of motion and the aircraft force models used in this research.

3.1.1 The equations of motion

The derivation of equations of motion for a rigid body aircraft is rather trivial and can be found in many sources [19, 40, 66, 148, 178]. It is important, however, to understand which motion is to be modelled. All structures have some inherent flexibility, and the vibration caused by it can be quite significant. However, the interaction of this vibration and the aerodynamic behaviour of the aircraft is extremely complicated. The internal flexible motion is usually dispersed into a set of harmonic oscillations, called modes, which adds many degrees of freedom to the model. On the other hand, a dynamic model of a rigid body has considerably less degrees of freedom (only six) and is an important tool in aircraft dynamics and control. For basic design, the motion of the aircraft can be assumed as a rigid body motion. Moreover, in case of a conventional small fixed-wing UAV, the rigid body approximation is usually more reliable than in the case of a large aeroplane. Indeed, due to the fact that for a given object the mass diminishes faster with the size (as a cube of linear dimensions) than the area, which is linked with most aerodynamic parameters (e.g. lift and drag) and material capacities (e.g. breaking strength), smaller constructions usually have greater relative cross sections of the internal structures, which results in greater relative strength and rigidity.

A rigid body model assumes the body is made up of a completely rigid material that exhibits no flexibility under any applied loads. Such a body has six degrees of freedom: three translational degrees and three rotational degrees. The state variables which fully describe the

state of the body will correspond to these degrees, with two variables for each degree of freedom: one for velocity and one for position, giving twelve state variables altogether.

Therefore, the UAV is modelled in this research as a rigid body construction, with 6 degrees of freedom and 12 state variables.

3.1.1.1 Axes systems

A number of different *axes systems* (or *reference frames*) is used in flight dynamics. The choice of any of the systems is based on the convenience of analysis and ease of expressing the equations. Very often, several axes systems are used in one model simultaneously. It is always possible to switch from one system to another. It is good engineering practice to choose so called right orthogonal axis systems in order to simplify such conversions, i.e. the systems in which the direction of the third axis is set so that the shortest rotation from the first axis to the second one, as seen from the 'top' of the third axis, is anti-clockwise; or, in other words, the direction of the third axis should coincide with the cross product vector of the vectors, aligned with the first and second axis. This preserves the sign of rotations when switching from one axis frame to another.

The basis behind the equations of motion is Newton's second law, which relates applied force to change in linear momentum with respect to an inertial reference frame:

$$\boldsymbol{F} = \frac{d}{dt}\left(m\boldsymbol{v}_{\text{abs}}\right) \tag{3.1}$$

where $\boldsymbol{F}$ is the vector sum of all forces acting on the body and $\boldsymbol{v}_{\text{abs}}$ is absolute velocity of the body.

Therefore, the first step is selecting an inertial reference frame. Ideally, this reference frame should be non-rotating and non-accelerating. However, although inertial frames do exist, all convenient frames rotate and/or accelerate (the earth, sun and stars all rotate and move through space). Nevertheless, for simplicity, certain motions may be assumed to be negligible in comparison to the motions of interest. If around the globe navigation is necessary, a convenient reference frame usually used is an orthogonal earth centred frame. However, if navigation is only required over a small area of the earth, as it takes place in this study, a 'flat earth' approximation may be used.

In this case, the origin of the orthogonal system (O_{g}) is assigned to a point on the surface of this 'flat earth', the axis y_g is pointed vertically upwards, x_{g} is directed north and z_{g} is directed east (see Fig. 3.1). This inertial (or nearly inertial) frame is common for the whole scene and thus for all its objects (the aircraft, the ship etc.) Their absolute positions are expressed in the inertial frame $O_{\text{g}}x_{\text{g}}y_{\text{g}}z_{\text{g}}$.

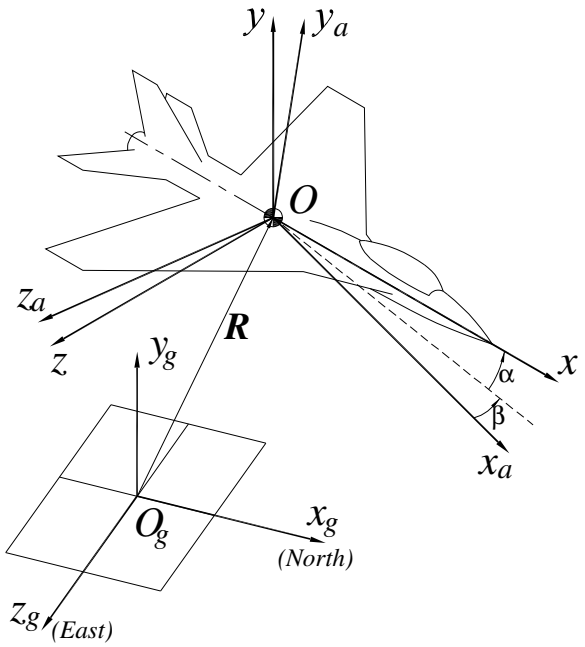


Fig. 3.1: Axis systems

There can be other right axis layouts, for example, the commonly used in western-school flight dynamics North-East-Down frame. However, to have a positive direction upwards is more intuitive for flight applications, and in practice using such frames (forward-up-right) appear more consistent with traditional sign agreement.[1]

Apart from gravitational forces, which act negative to y_g axis, the directions of all other significant forces acting on an aircraft (i.e. aerodynamic and propulsive forces) depend upon the orientation of the aircraft. Therefore, it is convenient to express these forces with respect to an axis system which is fixed to the aircraft. This system will not be inertial, because it translates and rotates with the aircraft.

The origin O of such a system, called *aircraft body frame*, is linked to the aircraft centre of gravity (c.g.)[2] The Oxy plane is usually aligned with the plane of symmetry (if any) of the aircraft, with the axis x pointing forward, y pointing up and z pointing to the right (Fig. 3.1).

The aerodynamic forces and to some extent, propulsive forces depend not only on the orientation of the aircraft, but also on the motion of surrounding air masses. Thus, the so called *aircraft wind axes* are introduced. The axis x_a of this axes frame corresponds to the direction of the aircraft aerial velocity vector $\boldsymbol{V}_a$, which is related to the aircraft body axis system by the angle of attack (α) and the sideslip angle (β) (see Fig. 3.1). The y_a axis always lies in the Oxy plane.

[1] With the exception of yaw/heading angle, which is calculated anti-clockwise.

[2] While 'centre of mass' and 'centre of gravity' may not coincide precisely in some cases, the difference is assumed to be negligible for this study and these terms may be used interchangeably.

The equations for V_a, α and β are as follows:

$$\begin{aligned} V_a &= |\boldsymbol{V}_a| = \sqrt{V_{ax}^2 + V_{ay}^2 + V_{az}^2} \\ \alpha &= -\arctan\left(\frac{V_y}{V_x}\right) \\ \beta &= \arctan\left(\frac{V_z}{\sqrt{V_x^2 + V_y^2}}\right) \end{aligned} \tag{3.2}$$

Here V_{ax}, V_{ay} and V_{az} are the projections of the airspeed vector onto the respective axes of the aircraft body frame. Also note that for high angles, the four quadrant arctangent is required. The aerial velocity differs from the inertial velocity in body frame $\boldsymbol{V}$ (the total velocity) by the instantaneous wind velocity $\boldsymbol{W}$ (also expressed in body frame):

$$\boldsymbol{V} = \boldsymbol{V}_a - \boldsymbol{W} \tag{3.3}$$

While translating one coordinate to another is trivial, the rotational relationship requires some calculations. The relationship between the aircraft body coordinate system and the inertial axes is called the *attitude* of the aircraft and is described with three Euler angles, known as *pitch (θ), bank (γ)* and *yaw (ψ)*. Note that when working with consecutive rotations, the order of the rotations is very important, because each new rotation takes place about the new, previously rotated axis. In the aeronautical field, it is a standard practice to apply the rotations in the following order when transforming from the inertial $O_g x_g y_g z_g$ frame to the body $Oxyz$ frame (the opposite order applies for reverse transformation):

1. Rotation about y_g-axis (yaw ψ);
2. Rotation about new z_g-axis (pitch θ);
3. Rotation about new x_g-axis (bank γ).

The signs of the rotations are determined as usual: a positive rotation is always anticlockwise as seen from the 'top' of the respective axis. In the above case, positive rotations are nose left, nose up, right wing down.

When transforming from wind to body coordinate system, the order applies as follows:

1. Rotation about y_a-axis by the sideslip angle β;
2. Rotation about new z_a-axis by the angle of attack α.

Coordinate transformations caused by rotation can be expressed within a matrix. The individual matrices for each rotation can be multiplied together in the correct order (right to left) to obtain a transformation matrix relating one coordinate frame to another. Such a matrix is called *Direction Cosine Matrix (DCM)*. For the transformation from the inertial to the body frame ($O_g x_g y_g z_g \rightarrow Oxyz$) the DCM will be [40]:

$$A_{\mathrm{gb}} = \begin{bmatrix} \cos\psi\cos\theta & \sin\theta & -\sin\psi\cos\theta \\ \sin\psi\sin\gamma - \cos\psi\sin\theta\cos\gamma & \cos\theta\cos\gamma & \cos\psi\sin\gamma + \sin\psi\sin\theta\cos\gamma \\ \sin\psi\cos\gamma + \cos\psi\sin\theta\sin\gamma & -\cos\theta\sin\gamma & \cos\psi\cos\gamma - \sin\psi\sin\theta\sin\gamma \end{bmatrix} \tag{3.4}$$

The coordinates in the target frame are obtained by multiplication of the DCM and the column vector of coordinates in the source frame:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = A_{\mathrm{gb}} \begin{bmatrix} x_{\mathrm{g}} \\ y_{\mathrm{g}} \\ z_{\mathrm{g}} \end{bmatrix} \tag{3.5}$$

The equivalent DCM for aircraft wind frame to body frame transformation is

$$A_{\mathrm{ab}} = \begin{bmatrix} \cos\alpha\cos\beta & \sin\alpha & -\cos\alpha\sin\beta \\ -\sin\alpha\cos\beta & \cos\alpha & \sin\alpha\sin\beta \\ \sin\beta & 0 & \cos\beta \end{bmatrix} \tag{3.6}$$

Reverse transformation (from the body to the wind frame and from the body to the inertial frame in this case) requires the inverse of the respective DCM. However, as the rotations are orthogonal, the inversion is simply the transpose, A^{T}. Thus, for example, $A_{\mathrm{bg}} = A_{\mathrm{gb}}^{\mathrm{T}}$.

3.1.1.2 Translational motion

The kinematic equations relating the position of the c.g. of the aircraft to its velocity in the inertial frame can be written as

$$\frac{d\boldsymbol{R}}{dt} = \boldsymbol{V}_{\mathrm{g}} = A_{\mathrm{gb}}^{\mathrm{T}}\boldsymbol{V} \tag{3.7}$$

Here $\boldsymbol{R}$ is the position of the aircraft in the inertial frame (see Fig. 3.1), $\boldsymbol{V}_{\mathrm{g}} = \left(V_{x_{\mathrm{g}}}, V_{y_{\mathrm{g}}}, V_{z_{\mathrm{g}}}\right)^{\mathrm{T}}$ is the aircraft velocity in the inertial frame and $\boldsymbol{V}$ is total velocity with respect to the body frame, which corresponds to $\boldsymbol{v}_{abs}$ in (3.1).

Expanding out Newton's second law (3.1) and taking note of the above results in

$$\boldsymbol{F} = \boldsymbol{V}\frac{dm}{dt} + m\frac{d\boldsymbol{V}}{dt} \tag{3.8}$$

In the case of aircraft, the first component of this equation, containing the derivative (i.e. rate of change) of mass, is normally quite low. This term may become significant for rockets or some aircraft with exceptionally high fuel burn; however, in other cases it can be safely neglected. Nevertheless, this term will be important in the case of in-flight jettisoning of any parts of a significant mass.

The equation (3.8) contains the differentiation of a vector in one coordinate system ($\boldsymbol{V}$ in body system) with respect to another (inertial frame). To move to the body frame, a cross

product of angular velocity of the frame ($\boldsymbol{\omega}$) and the velocity of the aircraft in this frame should be added [40]:

$$\boldsymbol{F} = m\left(\dot{\boldsymbol{V}} + \boldsymbol{\omega} \times \boldsymbol{V}\right) \tag{3.9}$$

Note that the angular velocity of the aircraft body frame corresponds to the angular velocity of the aircraft, since the body frame is fixed to the aircraft. Simple rearrangement of this equation leads to the velocity state equation:

$$\dot{\boldsymbol{V}} = \frac{1}{m}\boldsymbol{F} - \boldsymbol{\omega} \times \boldsymbol{V} \tag{3.10}$$

3.1.1.3 Angular motion

The Newton's second law equivalent to angular motion can be written as

$$\boldsymbol{M} = \frac{d\boldsymbol{K}}{dt}, \tag{3.11}$$

where $\boldsymbol{M}$ is the moment (torque) acting on the aircraft at its c.g., and $\boldsymbol{K}$ is the angular momentum of the aircraft. Similarly to (3.1), this law is valid only in an inertial axis frame. Likewise, it can be expressed with respect to the body frame as

$$\boldsymbol{M} = \dot{\boldsymbol{K}} + \boldsymbol{\omega} \times \boldsymbol{K} \tag{3.12}$$

To determine the angular momentum $\boldsymbol{K}$, which is an 'angular equivalent' to linear momentum $m\boldsymbol{v}$, the whole body can be divided into infinite small mass elements. Considering one of these particles δm with the position $\boldsymbol{r} = (x, y, z)^{\mathrm{T}}$, its angular momentum can be given by:

$$\delta\boldsymbol{K} = \left[\boldsymbol{r} \times \left(\boldsymbol{\omega} \times \boldsymbol{r}\right)\right]\delta m$$

The angular momentum is then obtained by integrating the angular momentum components over the entire aircraft mass. This gives, after expanding out the vector triple cross products:

$$\boldsymbol{K} = \begin{bmatrix} \omega_x \int \left(y^2 + z^2\right) dm - \omega_y \int xy\, dm - \omega_z \int xz\, dm \\ -\omega_x \int xy\, dm + \omega_y \int \left(x^2 + z^2\right) dm - \omega_z \int yz\, dm \\ -\omega_x \int xz\, dm - \omega_y \int yz\, dm + \omega_z \int \left(x^2 + y^2\right) dm \end{bmatrix}$$

The individual integrals in this vector are known as the moments and cross products of inertia. For example,

$I_x \equiv I_{xx} = \int \left(y^2 + z^2\right) dm$ is the moment of inertia about x axis;

$I_{yz} \equiv I_{zy} = \int yz\, dm$ is the cross product of inertia of y and z axis.

The angular momentum can be expressed in a convenient form as the product

$$\boldsymbol{K} = \begin{bmatrix} I_x & -I_{xy} & -I_{xz} \\ -I_{xy} & I_y & -I_{yz} \\ -I_{xz} & -I_{yz} & I_z \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \equiv I\boldsymbol{\omega} \tag{3.13}$$

where I is known as the *inertia tensor* and can be considered as an 'angular equivalent' to mass for linear motion.

Now taking into account (3.13), the vector equation of angular motion (3.12) in the body frame can be written as

$$I\dot{\boldsymbol{\omega}} + \dot{I}\boldsymbol{\omega} = \boldsymbol{M} - \boldsymbol{\omega} \times (I\boldsymbol{\omega})$$

and assuming no inertia tensor time dependence (using the same argument as for mass time dependence), this can be rearranged giving the angular velocity state equation:

$$\dot{\boldsymbol{\omega}} = I^{-1}(\boldsymbol{M} - \boldsymbol{\omega} \times (I\boldsymbol{\omega})) \tag{3.14}$$

It is usual to simplify this equation by taking advantage of aircraft symmetry along the *Oxy* plane. It means that the only non-zero cross product is the I_{xy} term. However, for a computational approach, this will not give any noticeable benefit. Moreover, despite usual geometrical symmetry, the weight distribution may not be symmetrical due to equipment, fuel and payload positioning.

The final set of state equations are kinematic equations which relate the rate of change of the Euler angles to the angular velocity. They can be derived by considering three separate consecutive rotations by the Euler angles in the correct order, described above in Section 3.1.1.1. Using this approach, four coordinate systems will be generated: the initial system #1 (which corresponds to the inertial system); the system #2, produced by rotating #1 by the yaw angle ψ; that produced by rotating #2 by the pitch angle θ (#3); and finally that produced by rotating #3 by the bank angle γ (body system, #4). If each axis system has associated direction indicated by the unit vector $(i, j, k)_n$, the following vector equation is obtained:

$$\boldsymbol{\omega} = \boldsymbol{i}_4\dot{\gamma} + \boldsymbol{j}_2\dot{\psi} + \boldsymbol{k}_3\dot{\theta} \tag{3.15}$$

The unit vectors in this equation with respect to the body axes can be obtained from the respective direction cosine matrices, taking out the first, second or third column of the DCMs for the unit vectors $\boldsymbol{i}, \boldsymbol{j}$ and $\boldsymbol{k}$ respectively. The DCM for the rate of change of bank angle is trivial: $\dot{\gamma}$ directly corresponds to ω_x. The DCM for the rate of change of pitch (i.e. of the axis system #3) is that of the body system before the bank angle is applied, thus it corresponds to the DCM of the individual rotation by this angle (A_γ). Further, the rate of change of yaw is calculated in the axis system #2, which DCM is obtained through two rotations, by the bank and pitch angles, and equals to the product $A_\gamma A_\theta$. As a result, the directions of the unit vectors in reference to the body axes will be

$$\boldsymbol{i}_4 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \boldsymbol{j}_2 = \begin{bmatrix} \sin\theta \\ \cos\theta\cos\gamma \\ -\cos\theta\sin\gamma \end{bmatrix} \quad \boldsymbol{k}_3 = \begin{bmatrix} 0 \\ \sin\gamma \\ \cos\gamma \end{bmatrix} \tag{3.16}$$

Now substituting this into (3.15) gives:

$$\boldsymbol{\omega} = \begin{bmatrix} 1 & \sin\theta & 0 \\ 0 & \cos\theta\cos\gamma & \sin\gamma \\ 0 & -\cos\theta\sin\gamma & \cos\gamma \end{bmatrix} \begin{bmatrix} \dot{\gamma} \\ \dot{\psi} \\ \dot{\theta} \end{bmatrix} \tag{3.17}$$

This can be inverted to yield the state equations:

$$\dot{\boldsymbol{\Phi}} = \begin{bmatrix} 1 & -\cos\gamma\tan\theta & \sin\gamma\tan\theta \\ 0 & \dfrac{\cos\gamma}{\cos\theta} & -\dfrac{\sin\gamma}{\cos\theta} \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} \boldsymbol{\omega}, \quad \boldsymbol{\Phi} = \begin{bmatrix} \gamma \\ \psi \\ \theta \end{bmatrix} \tag{3.18}$$

Note that these equations have a singularity at $\theta = \pm 90°$, which should be taken care of.

As a result of all these efforts, a full set of state equations is obtained. It is represented by those (3.7), (3.10), (3.14) and (3.18). They can be expanded into twelve scalar equations; however, the vector form is often more convenient for both analytical analysis and computer simulation. Nevertheless, for further reference, a full state vector is defined as

$$\boldsymbol{x}^{\mathrm{T}} = \left[V_x, V_y, V_z, \omega_x, \omega_y, \omega_z, \gamma, \psi, \theta, x_{\mathrm{g}}, y_{\mathrm{g}}, z_{\mathrm{g}} \right] \tag{3.19}$$

Note that in this research y_g simply corresponds to altitude H, because the origin of the inertial system is fixed to the earth surface.

In the state vector, the first three terms (translational velocity) are obtained through the equations (3.10), the next three terms (angular velocity) are calculated by (3.14), the next three (Euler angles) are defined via kinematic equations (3.18) and the last three (position of the aircraft in the inertial frame) are obtained through (3.7).

3.1.2 The force model

The equations of motion presented in the previous section are general by nature and have been derived with few simplifications relevant to the aircraft structural model. Thus, these equations of motion are applicable to nearly all aircraft. In contrast, force models are aircraft specific. The only thing they have in common with the equations of motion is that the force models generate forces from the state vector (3.19). Therefore, a specific aircraft must be chosen for a force model. The next sections describe the model used in this study.

The force model is based upon the principle of component build-up. The components represent the basic airframe (wing and body), the tailplane, the propulsion unit (engine and propeller) and all control surfaces (ailerons, elevator, rudder, flaps).

Each component has a contribution to each of the three forces and three moments acting on the aircraft. For example, the lift force is generated by the basic aerodynamic force on the wing and fuselage, plus the change in lift due to elevator deflection, plus change in lift due to rate of change of angle of attack, etc. Likewise, the drag force and any other force and moment are affected by all these variables, although some of the dependencies may be rather small and negligible. This type of model usually implies that the individual components are independent of each other, for example, assuming that the change in lift due to elevator deflection is not affected by the deflection of any other control surface. Although this may not be the case (for instance, the elevator efficiency is often affected by the turbulent airwake behind the extended flaps), these interferences are assumed to be negligible. However, in some cases they may be taken into account by introducing additional couplings between the components.

It is usual practice that all the forces and moments acting on the aircraft are expressed through non-dimensional coefficients. These coefficients are calculated with respect to dynamic pressure ($q = \rho V^2/2$), wing area (S) and (for moments), the mean aerodynamic chord (b_a) for longitudinal motion or wingspan (l) for lateral and directional motions. This makes the coefficients very demonstrative and comparable for virtually any aircraft of a specified type (e.g. aeroplanes), although these aircraft may vary in size significantly. For example, the lift coefficient, which represents the lift force Y, is defined as

$$C_y = \frac{Y}{qS} \tag{3.20}$$

Moreover, for convenience, even the parameters of motion are often non-dimensionalised and/or referred to more general parameters, such as velocity or dynamic pressure. For example, a non-dimensional angular velocity about x-axis (non-dimensional roll rate) is

$$\overline{\omega}_x = \omega_x \frac{l}{2V} \tag{3.21}$$

3.1.2.1 The Ariel UAV

The aircraft chosen as a prototype for the research is the UAV *Ariel* (Fig. 2.17), which is an unmanned aircraft developed by the UAV group in the Department of Aeronautical Engineering at the University of Syd-

Fig. 3.2 The Ariel UAV

ney. As noted earlier, the same aircraft has been used as a prototype for the research on launch [47]. The majority of data on this aircraft is available in [153]. The force model is based largely upon this information.

The *Ariel* is a relatively small aircraft (35 kg maximum takeoff weight and 3 m wingspan), manufactured predominantly from fibreglass and foam components. The key design parameters are listed in Appendix B. The aircraft has three main aerodynamic controls (elevator, ailerons and rudder), plus flaps and throttle control.

3.1.2.2 The aerodynamic model

The aerodynamic forces are calculated in the aircraft wind axes $Ox_a y_a z_a$ (Fig. 3.1). The three forces generated by the aerodynamic model are referred to as *Aerodynamic Drag, Lift* and *Sideforce* and correspond to the forces acting opposite to x_a and positive to y_a and z_a axis respectively. There are also three moments acting about each of the axes, called *Rolling Moment, Yawing Moment* and *Pitching Moment*.

The forces are resolved from the wind axes into the aircraft body axes as required by the equations of motion (see section 3.1.1.1). In addition, the aerodynamic forces are distributed by nature over the whole surface of the aircraft, and the resultant aerodynamic force may be applied to different locations in different conditions. To meet the assumptions of the equations of motion, that the forces are applied to the aircraft c.g., the forces obtained from theoretical calculations or wind tunnel testing (given in [153]) must be moved to the aircraft c.g. taking into account resulting moments.

The force model is based upon wind tunnel testing of a ⅓ scale model of the *Ariel*. This wind tunnel model had its own axes system due to the mounting of the model within the wind tunnel at a point different to c.g. Therefore, the wind tunnel data were obtained relative to this mounting point. Apart from the additional moments mentioned above, the effect of rotations about the mounting hinge upon the relative wind vector used for wind tunnel data must be taken into account. This is done simply by taking the cross product of the location vector of the wind tunnel axis with respect to the c.g. and the angular rate vector, and the result is added to the aircraft and wind velocities:

$$\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix}_{w.t.} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{c.g.\rightarrow w.t.} \times \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} + \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} + \begin{bmatrix} W_x \\ W_y \\ W_z \end{bmatrix} \tag{3.22}$$

The original force model [153] was based upon limited wind tunnel data, from which approximate polynomial functions were generated as functions of angle of attack, sideslip angle and various control surface deflections. These polynomials are of a high order and have a

limited range of applicability; in particular, low reliability at and beyond the extremes of the data set. On the other hand, this approach has relatively low computational requirements.

When full range of data is required, especially if the data represent a complex dependence on input variable(s), look-up tables provide greater flexibility and reliability. Thus, look-up tables of aerodynamic data were generated. Within the valid wind tunnel testing region, they are based on actual measurements of the aerodynamic forces and moments. However, in the case of landing in adverse wind conditions, it is likely that high angles of attack and/or sideslip angles may be present. Thus, further aerodynamic data for these regions is necessary. Whilst wind tunnel testing at these high angles is desirable, for the purpose of this research it was unfeasible. Therefore, a theoretical approximation is used, employing several techniques, described below (see also [47]).

Despite significant progress in the area of high angles of attack studies in the last 20 years, there is still little publicly available aerodynamic data for aircraft at these angles, due to the fact that most of the aircraft operating in these regimes are military. However, known data ([40] for example) show that the behaviour of an aircraft at high angles of attack is hardly predictable without experimental testing, and that it depends on many factors, both geometrical and aerodynamic, which are often not taken into account. They include, for example, any asymmetry of the aircraft and the sign of rate of change of angle of attack.

Due to these difficulties, the approximate models used in this research are based upon general aircraft behaviour and may not approach the behaviour of the actual aircraft. In particular, post-stall angles of attack are simply associated with drop of lift and rise of drag and appropriate change of moments with typical profile for the *Ariel*'s aerodynamic configuration, without attempt to simulate asymmetrical stochastic flow which may be necessary, for example, for simulation of spin. Also, tail blanketing is not modelled as well. However, the aircraft model is used simply for demonstration and testing of the controller development, therefore the lack of correlation to the real aircraft should pose little or no problem. Actually, the choice of the *Ariel* itself is fairly arbitrary—any UAV of its class could be used. For a complete control design to physical implementation for a specific UAV, high accuracy models of the actual

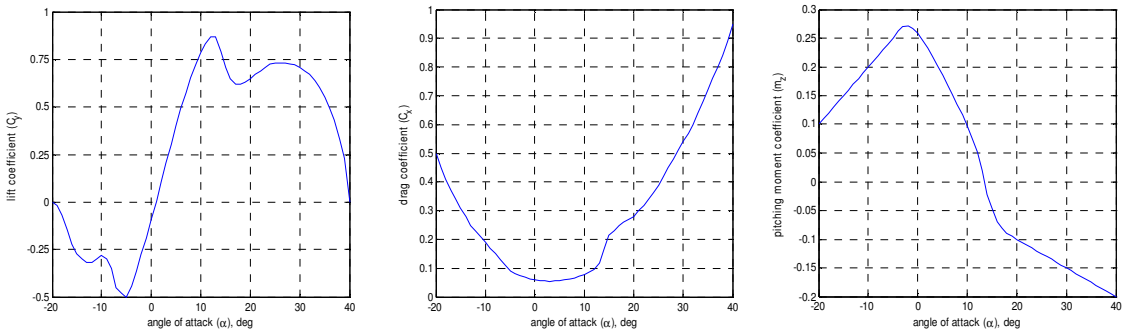


Fig. 3.3: Extended basic aerodynamic characteristics (wing body only, no flaps)

aircraft will be required. The control design will incorporate some robustness to model mismatch; however, it is desirable for the simulation model to match reality as closely as possible.

The first modification is extending the angle of attack range into both negative and positive regimes. The measured in wind tunnel range includes post-stall angles of attack (see Table III in [153]); however, the testing did not investigate the post-stall behaviour, although the stall angles of attack and peak lift values were detected for each flap setting. The aerodynamic data beyond the range of –4.9…15.5 degrees (no flaps) is extrapolated according to general behaviour of a straight untapered wing with relatively high aspect ratio and known aerofoil [40, 47, 175]. The plots of the extended lift, drag and pitching moment curves in the most important range of α = –20…40 deg. are shown in Fig. 3.3.

In a similar process, the effects of higher sideslip angles, as well as the effects of high angles of attack on the control surface data, are incorporated into the relevant aerodynamic data. The full models can be found in Appendix A.2 of [47].

3.1.2.3 The propulsion model

The *Ariel* uses pusher-prop configuration with three blade constant pitch propeller. The propeller is powered by an internal combustion engine with a maximum power output at sea level of P_{max0} = 5.6 kW (according to the manufacturer's specifications). The engine and propeller model closely follows that outlined in [153] and is briefly discussed here.

The maximum engine power varies with atmospheric conditions. The allowance is given to the most contributing factor—ambient air density ratio $\bar{\rho} = \frac{\rho}{\rho_0}$, $\rho_0 = 1.225 kg/m^3$:

$$P_{max} = P_{max\,0}\left(1.1324\bar{\rho} - 0.1324\right) \tag{3.23}$$

The engine control is defined in terms of throttle lever angle δ_t, expressed as a fraction in the range from 0.01 (idle) to 1.0 (maximum available power). Thus, the engine power output is given linearly as

$$P = \delta_t P_{max} \tag{3.24}$$

The propeller is modelled using momentum blade element theory, based upon a Clark Y aerofoil section, and also includes induced flow effects. In [153], high order polynomial functions of engine advance ratio J were generated for each of the four coefficients:

$$C_{\mathrm{T}} = \frac{T}{\rho n^2 D^4}$$

$$C_{\mathrm{P}} = \frac{P}{\rho n^2 D^5}$$

$$C_{Z} = \frac{F_z}{\rho n^2 D^4}$$

$$C_{\mathrm{G}} = \frac{M_{\mathrm{G}}}{\rho n^2 D^5}$$

, where

T is propeller thrust

P is propeller power

F_z is propeller normal and side force

M_G is propeller gyroscopic moment (yawing and pitching moments)

(3.25)

Like with aerodynamic data, in the current model the polynomials are converted to 2-D look-up tables for simulation speed and stability issues.

The engine advance ratio is calculated by:

$$J = \frac{V_{\mathrm{Tp}}}{nD}$$, where

V_{Tp} is the propeller tip speed

n is the engine rotational speed

D is the propeller diameter

(3.26)

For a given engine power output (3.24), a corresponding advance ratio can be found through the second equation of (3.25), numerically solving the corresponding polynomial equation $C_{\mathrm{P}} = f(J)$. However, this tends to be computationally expensive and inefficient to perform at every time step. To circumvent this problem, one more look-up table containing a pre-calculated range of advance ratio values corresponding to different power values is generated.

As the final step, the engine forces and moments are transformed from the engine axes into the aircraft body axes.

3.1.2.4 Sensor and actuator models

The aircraft simulation includes simple dynamic models of the aircraft control actuators and the aircraft sensor bank. Their characteristics are similar to those used in [47].

The control surface actuators are modelled as second order linear systems having the general form

$$\ddot{x} + 2\zeta\omega\dot{x} + \omega^2 x$$

All control surface actuators are assigned with a natural frequency of 10 Hz (62.8 rad/s) and a damping coefficient of 0.6. This gives the transfer function in the Laplace form as

$$G_{\delta_{actual}}^{\delta_{demand}} = \frac{\omega^2}{s^2 + 2\zeta\omega s + \omega^2} = \frac{3947.8}{s^2 + 75.4s + 3947.8} \tag{3.27}$$

There are also non-linear parameters applied, such as positional and rate saturations. The rate limit for all actuators is taken as ±250 degrees per second, while positional limits differ for different control surfaces: ±16 degrees for rudder and ailerons, and –32 to +16 degrees for elevator (negative values for nose-up control).

The engine throttle control dynamics is a simple aperiodic function with a relatively large time constant of 1 second. This gives the throttle transfer function as

$$G_{\delta_{\text{t.actual}}}^{\delta_{\text{t.demand}}} = \frac{1}{s+1} \tag{3.28}$$

This dynamics is assumed for both power up and power down. As mentioned in the previous section, the throttle has positional limits of 0.01 to 1.0, plus the rate limit of ±2 per second.

The UAV is assumed to have the following onboard sensors:

- rate gyros (measuring body angular rates ω_x, ω_y, ω_z);
- strapdown INS (uses rate gyros to calculate Euler angles γ, ψ, θ);
- accelerometers (measuring body accelerations a_x, a_y, a_z);
- velocity and angle vanes (measuring airspeed V_a and wind angles α, β);
- altitude sensor;
- control surface sensors (measuring actual deflection of control surfaces δ_a, δ_r, δ_e);
- engine sensor (measuring engine speed).

Each measurement sensor is assumed to have simple aperiodic dynamics in the form

$$G_{x_{\text{measured}}}^{x_{\text{actual}}} = \frac{1}{\tau s+1} \tag{3.29}$$

The time constant for each sensor is taken as 0.04 seconds. The positional and rate limits are individual and listed below:

Sensor	*Positional Limit*	*Rate Limit*
Accelerometer	100 ms^{-2}	400 ms^{-3}
Gyro	2π rad	8.7 rad s^{-1}
Rate Gyro	8.7 rad s^{-1}	8.7 rad s^{-2}
Velocity Vane	100 ms^{-1}	1000 ms^{-2}
Angle Vane	0.5 rad	5 rad s^{-1}

Table 3.1 Positional and rate limits of the sensors

Noise characteristics are included as well, with a moderate amount of noise placed upon the accelerator and gyro readings and a large amount of noise included in the vane readings. Noise is simulated by adding a white noise signal with appropriate power. Since the exact amount of noise for each sensor is not specified, the maximum possible noise power will be determined in the robustness tests in Chapter 6.

3.2 Atmospheric model

A critical part of any aircraft simulation is the model of the atmosphere. As long as large scale nature (such as climate) is not considered, the model can be divided into two basic parts: an altitude profile and a model of the movement of the air (wind model).

Altitude profile. Although expected altitude changes are relatively small for the ship landing scenario, a model of standard altitude profile can be incorporated because it is very simple and widely used. A thorough description of the standard altitude profile can be found in many sources, for example, MIL-STD-210C [1].

In the current implementation, analytical calculation of the atmospheric parameters has been used instead of look-up tables. The key parameter is the ambient *air temperature*. At zero altitude (mean sea level, MSL) it is defined as 288.15K (15°C), and it falls by 6.5K every thousand metres of altitude until 11000 m. At higher altitude the temperature becomes constant and then starts to rise, but it was not implemented in the model, as these altitudes lie beyond the operational ceiling of the aircraft, not to say about landing conditions.

Air pressure is calculated from the temperature as

$$P(T) = P_0 \cdot \left(\frac{T}{T_0} \right)^{5.25588} \text{ [Pa]}, \tag{3.30}$$

where $P_0=101325\ Pa$ is the reference air pressure at MSL, and $T_0=288.15K$ is the reference temperature at MSL.

Air density is defined as a function of both air pressure and temperature, given above:

$$\rho(T,P) = \frac{P}{287.0529 \cdot T} \text{ [kg/m}^3\text{]}. \tag{3.31}$$

And *speed of sound*, if needed, can be easily calculated as

$$a_s(T) = 20.0468 \cdot \sqrt{T} \text{ [m/s]}. \tag{3.32}$$

Wind model is the critical part of the simulation model. It consists of four main components:

- Steady wind;
- Atmospheric turbulence;
- Gusts;
- Ship airwake.

These components are described in the following sections. It should be noted that the wind model applies through the aircraft force equations. Essentially, it represents an additional airspeed vector on the aircraft: it changes airspeed, angle of attack and sideslip. The wind can be assumed as a disturbance from the environment and investigated from the point

of view of the robust characteristics of the designed control system. In addition, knowledge of the current environment conditions (e.g. wind strength and direction), either measured on-board or supplied remotely, can be used by the controller to improve its performance.

3.2.1 Steady wind model

The steady wind model is extremely simple. It consists of only two parameters: wind magnitude and direction, which are usually selected before simulation. In reality, both of them depend on altitude and surrounding terrain; moreover, they can change with time. However, as this research is focused on short term low altitude oceanic flights, only the most significant variation is taken into account—the variation of wind speed with altitude.

According to the wind model suggested in the military standard [4], the reference wind conditions (W_0) are assumed to exist at 6 m (20 ft) above ground level or MSL. Magnitude of the wind velocity varies with altitude according to

$$W(H) = W_0 \frac{\ln(H/H_0)}{\ln(6/H_0)} \tag{3.33}$$

where H_0 is the altitude of zero wind.

H_0 = 0.05 m for Category C flight phases, and

H_0 = 0.6 m for other flight phases.

According to [4], Category C are 'terminal flight phases that are normally accomplished using gradual manoeuvres and usually requiring accurate flight path control.' As this research focuses on the landing task, the model is fixed to H_0 = 0.05 m and the altitude is restricted to about 300 m.

3.2.2 Turbulence model

Turbulence is usually simulated by filtering wide band white noise, thus producing a stochastic output of a given spectrum. The filters are designed so that the output simulates the changing wind velocity at the location of the aircraft. There is no need to simulate the whole field of ambient turbulence, unless the scale of turbulence is comparable to the size of the aircraft itself. Therefore, it is more convenient to think in terms of spatial frequency Ω, with an aircraft flying through a 'frozen' turbulence field with a given speed V. The spatial frequency is related to circular frequency through the aircraft's true airspeed:

$$\Omega = \frac{\omega}{V_a} \text{ [rad/m].} \tag{3.34}$$

An obvious limitation of this approach is that it cannot simulate the influence of turbulence at zero airspeed. This is not important for this research as hovering or other techniques involving near-zero airspeed are not considered.

There are two most commonly used turbulence models: the von Kármán model and the Dryden model [96]. They are very closely related as the Dryden model is a simplified version of the von Kármán model. The advantage of the Dryden model is that its power spectral densities are rational functions and thus can be used in their exact form for building white noise filters.

However, of the two spectra, the von Kármán gives the better fit to observed data and also is supported by theory. It is the standard for design use [96], p. 46. It is allowed, nevertheless, to use the Dryden model, 'when no comparable structural analysis is performed or when it is not feasible to use the von Kármán form' [5]. The difference between the Dryden and von Kármán form is basically a small variation in the high frequency content [79, 145].

Being a small aircraft, the UAV similar to *Ariel* in size may be susceptible to high frequencies, although its airspeed may not be high enough to produce much of them. In any case, present-day computational limitations allow for use of high order filters without significant time penalty. Therefore, the von Kármán model has been chosen for this application.

The von Kármán power spectral densities (PSDs) for each translational velocity component are specified in [5] as follows:

$$\Phi_x(\Omega) = \sigma_x^2 \frac{2L_x}{\pi} \frac{1}{\left(1 + (1.339 L_x \Omega)^2\right)^{5/6}}$$

$$\Phi_i(\Omega) = \sigma_i^2 \frac{L_i}{\pi} \frac{1 + \frac{8}{3}(1.339 L_i \Omega)^2}{\left(1 + (1.339 L_i \Omega)^2\right)^{11/6}} \tag{3.35}$$

where $i \equiv y$ or z, vertical and lateral component respectively (their PSDs are similar).

In these formulas σ_i is the turbulence intensity in the direction i and L_i is a scale length for the respective direction. It is interesting to note that these parameters are defined differently in various sources, for example, [5] and [3], despite the PSDs being the same. The fact is that the gust response depends hardly at all on L—any value, as long as it is not too small, can be selected as long as it is used consistently ([96, p.47]). This research follows the guidelines specified in [5] and [4] for low-altitude flight (with the conversion to the metric system).

The turbulence intensity σ depends on steady wind. Using section 3.7 of MIL-F-8785C [5], turbulence intensities are

$$\sigma_y = 0.1 W_0$$

$$\sigma_x = \sigma_z = \sigma_y \frac{1}{(0.177 + 0.000823H)^{0.4}} \tag{3.36}$$

and the turbulence scale lengths

$$
\begin{aligned}
L_y &= 300 \text{ m} && \text{for } H > 300 \text{ m} \\
&= H && \text{for } 300 \text{ m} > H > 3 \text{ m} \\
&= 3 \text{ m} && \text{for } 3 \text{ m} > H
\end{aligned}
$$

$$
\begin{aligned}
L_x &= 300 \text{ m} && \text{for } H > 300 \text{ m} \\
&= \frac{H}{(0.177 + 0.0027H)^{1.2}} && \text{for } H < 300 \text{ m;} \\
L_x &\equiv 23 \text{ m} && \text{if } L_x < 23 \text{ m.} \\
L_z &= L_x
\end{aligned}
\tag{3.37}
$$

The model can be extended to include rotational velocity components. However, their effect is hardly worth double complication of the turbulence model. The MIL-HDBK-1797 [4][1] states the conditions when rotational turbulence may be considered important. For example, the rolling disturbance (which is usually the most significant component) is important only if

$$
\sqrt{\frac{b_a}{2L_y}} \left| m_x^{\bar{\omega}_x} \right| > \left| m_x^{\beta} \right| \tag{3.38}
$$

Here b_a is the aircraft's mean aerodynamic chord. For the *Ariel* UAV, the left part of this inequation ranges from approximately 0.14 to 0.015 [1/rad] with altitude (0 to 300 m respectively), while the right part varies from 0 to 0.014 [1/rad] across the whole operational envelope (0.026 being the maximum value, extrapolated to extreme sideslip angles). Therefore, the effect of rotational turbulence components has been neglected.

One more remark should be made about mutual influence of the translational velocities. 'Whilst it is known that significant correlation exists between the W_x, W_y and W_z components of turbulence, there is at present insufficient information to allow realistic representation of such correlation. Consequently, the three components may be assumed to be uncorrelated' [3].

The simulation requires time domain modelling of the velocities based on the above spectra. In order to do that, filter transfer functions for each velocity component must be developed. These transfer functions are related to the respective PSDs as functions of temporal frequency ω:

$$
\left| T(i\omega) \right|^2 = \Phi(\omega) \tag{3.39}
$$

As the von Kármán PSDs are not rational functions, a rational polynomial approximation must be generated for each transfer function. Ready for use third-order transfer functions

[1] At the time of publication, the MIL-HDBK-1797 handbook has been reverted to the MIL-STD-1797 military standard.

can be found in [4] (Boeing linear filter forms) and MATLAB's reference to the Aerospace Blockset [142]. In this research, the latter ones are used, although both forms have good agreement (note that the Boeing functions in [4] lack the $\sigma\sqrt{V/L}$ coefficient). According to [4], a third-order representation provides correct approximation of the spectral content up to 100(*V/L*) rad/s; however, a second-order filter was chosen for the longitudinal component, because it delivers approximately the same precision for the simpler Φ_x function.

For making the following formulas more readable, let us define

$$q_i \equiv \frac{V}{L_i}$$

where *i* can be *x, y* or *z* for the respective velocity component. Then the linear filter transfer functions in Laplace form will be

$$\begin{aligned} T_x(s) &= \sigma_x\sqrt{q_x}\,\frac{1.258s + 5.033q_x}{s^2 + 6.829q_x s + 5.033q_x^2} \\ T_j(s) &= \sigma_j\sqrt{q_j}\,\frac{2.208s^2 + 17.855q_j s + 6.498q_j^2}{s^3 + 12.836q_j s^2 + 19.466q_j^2 s + 6.498q_j^3} \end{aligned} \tag{3.40}$$

where $j = y$ for normal velocity transfer function and *z* for lateral velocity transfer function. Note that in contrast to [142], the transfer functions are given in the 'physical form,' with the highest order coefficient in denominator of 1—this makes it easier to build a state-space representation of the filters.

3.2.3 Gust model

The gust model uses the '1 – cosine' model, specified in [4] and [96] as a standard tool for design and evaluation purposes:

$$\begin{aligned} W &= 0 && \text{, for } x < 0; \\ W &= \frac{W_{\max}}{2}\left(1 - \cos\left(\frac{\pi x}{l_m}\right)\right) && \text{, for } 0 \le x \le l_m; \\ W &= W_{\max} && \text{, for } x > l_m. \end{aligned} \tag{3.41}$$

Here *W* is gust velocity, W_{max} is maximum gust velocity, *x* is the distance travelled and l_m is the ramp length.

Assuming the aircraft is travelling at near constant velocity during the gust rise, the distance variable *x* can be replaced with a time variable.

Several gusts may be superimposed, all having different directions and magnitudes. For a pilotless aircraft, gusts are usually preset at specified time or space positions to check the aircraft's response in a worst case scenario. However, they may be randomly allocated to closely simulate real world environment.

3.2.4 The airwake model

Unlike the launch case scenario, ship airwake is very important for landing an aircraft on board a ship. In most cases, the aircraft approaches the ship from the downwind side, which allows to reduce the relative approach speed while maintaining the airspeed high enough to fly. On the other hand, a ship generates a significant amount of turbulence behind which actually can be far greater than the ambient free-air turbulence. This effect cannot be neglected.

However, no appropriate airwake model has been found. Despite that there is a lot of preliminary research on ship airwake models (see also discussion in Section 2.2.1.3), these models are either insufficient or military based and as such unavailable. In the case of launch [47], the effect of the airwake has been neglected, which was rather an appropriate simplification for the launch, but is inappropriate in this research. Therefore, an attempt to build a suitable airwake model has been made in this work.

3.2.4.1 Discussion

It is possible to build an approximation to ship airwake using the motion of the ship and the wind as inputs. However, even a simple custom model which could provide adequate and reliable results is worth separate research. As this is not the major component of the current study, this approach has been rejected. Nevertheless, even a simplified airwake model requires significant research effort. Only the model which assures adequate and reliable results can be used, thus it requires appropriate verification and validation. The research in this area is out of the scope of this thesis; however, it could be considered for future investigation.

One of the possible ways to overcome the problem is employment of a common turbulence profile (von Kármán, for example, which is used for generating the free-air turbulence, see Section 3.2.2), applying greater intensity in the area affected by the airwake. After all, the purpose of the model is to test the landing controller in various conditions to ensure that the controller design meets its objective. Added ‘severity’ of turbulence could roughly simulate harder conditions within the airwake. This is the easiest way; however, such simplification has some significant disadvantages.

First, the amount of added intensity is unknown and can only be roughly estimated. This is not a major concern, considering the stated objective. Nevertheless, it is obvious that the turbulence changes within the airwake (at least, the closer to ship, the more intense the turbulence is), which makes the flight unsteady. This is a qualitative change and may affect the controller. However, the profile which could realistically reflect such a variation of the turbulence parameters is hardly possible to estimate and almost impossible to validate, considering the lack of actual data.

Second, it is evident that there are components in the airwake that cannot be simulated as a filtered noise similarly to the free-air turbulence models. For example, a significant amount of up and downwash should be expected. These components may affect the aircraft as well.

The way out of this situation has been found in using a specialised airwake model as a base for a more universal model suitable for this research. This specialised model is described in the US military standard MIL-F-8785C [5] and its further development MIL-HDBK-1797 [4]. Actually, this is the only publicly available airwake model found which is reliable, proven, comprehensive enough and, on the other hand, allows real-time (or faster) simulation.

However, this model has some restrictions, most of them coming from its specific purpose:

- The MIL-F-8785C airwake model has been developed for the case of landing of a conventional (piloted) aircraft on a carrier-sized ship (length of approx. 300 m).
- The model describes the properties of air disturbances only along the standard glidepath of the landing aircraft: that is, with ≈4 degrees glideslope; approaching from the downwind side when the ship is moving along the wind; and within a narrow angle of a few degrees from the longitudinal axis of the carrier.
- It is valid, apparently, for the conditions which are normal or nearly normal for landing of a piloted aircraft. This implies, for example, the pitch motion of the carrier within 2–3 deg. An UAV is expected to cope with severer conditions.
- The purpose of the model is not simulation of the airwake as such, but the use of it as a component of a simulation environment aimed to estimate flying qualities of a piloted (including remotely piloted) aircraft.
- Taking a human pilot into account has its influence, especially in randomisation of the processes involved. For example, a simple harmonic approximation of the ship angular motion may be sufficient for automatic control, while a human pilot starts to predict such a simple motion very quickly. In this sense, an airwake model for a UAV may be simpler.

Despite all these limitations, it is believed that the model can be redesigned to include a broader spectrum of conditions. As this work is fairly insightful and creative, it is described more thoroughly in the next section.

3.2.4.2 The airwake model design

First of all, the objective should be kept in mind: to build a model which provides an adequate effect on the UAV during landing on a ship. It is not the point to simulate the whole three-dimensional airwake. Secondly, despite its importance, the airwake has only indirect

effect on the aircraft and is used as a measure to estimate the aircraft's qualities and the robustness of the controller, similar to the general ambient turbulence (see Section 3.2.2). Therefore, the airwake properties are considered only to the point of how they influence the aircraft and the landing.

The MIL-F-8785C airwake model does not replace the common free-air disturbance model; instead, it supplements it. The airwake model consists of four parts:

- Free-air turbulence (ambient turbulence in the airwake, independent of the aircraft relative position);
- Steady component (up/downwash, tail/headwind);
- Periodic component, caused by the ship angular motion;
- Random component (ship-related random turbulence).

All components are expressed as velocities of the air at the position of the aircraft in the wind-over-deck axes $Ox_{\mathrm{w}}y_{\mathrm{w}}z_{\mathrm{w}}$. This axis frame corresponds to the inertial frame $Ox_gy_gz_g$, rotated about the y_g axis so that the x_g axis is aligned with the mean *wind over deck vector (*$\boldsymbol{V}_{\mathrm{wd}}$*)*, and translated to the aircraft c.g. position. The $\boldsymbol{V}_{\mathrm{wd}}$ vector consists of combined wind vector and inverse ship velocity, thus representing the wind direction and intensity as felt on the ship deck.

Basically, the free-air and random components are obtained by filtering white noise through the specific filters, much like the von Kármán and Dryden models described above. The steady component is given as an approximation of the wind function of the distance from the ship. Finally, the periodic component is simulated as a harmonic oscillation of the wind vector and depends on periodic ship motion and the distance from the ship. Needless to say all components depend on the wind over deck velocity.

The MIL-F-8785C model is limited to the distance of 800 m (1/2 nm) aft of the ship. This is less than three lengths of the ship itself, but it is deemed to be adequate. For a common fixed-wing aircraft it represents the last 10–15 seconds of approach.

The main challenges arising in generalisation of the model are:

- to scale down the airwake appropriately with the ship size;
- to cope with any relative wind direction, not only aligned with the ship;
- to provide smooth transitions when moving in and out of the airwake, because the landing strategy may choose to cross the airwake instead of approaching entirely from the ship's downwind side.

The first problem, scaling down, is not crucial in obtaining the whole picture of the airwake. Indeed, it can be assumed that the shape of the airwake (at a relatively far distance) is approximately the same for both bigger carrier and smaller frigate, because the difference in Reynolds numbers is fairly small for the given conditions (2–4 times) and both the numbers

are of the same scale of tens of millions (with respect to the ship length). Therefore, the overall properties of the turbulent wake (mainly intensity and spectrum) remain approximately the same for a frigate-sized ship.

Somewhat less apparent is the 'physical size' of the airwake. Generally, the smaller the ship, the smaller the length of the airwake. Theoretically, the length is infinite, but practically it can be defined as the distance where the turbulence intensity falls by a given percent. Nevertheless, considering the large scale of the turbulence, the wake fades out slowly and so the length decreases 'slower' than the ship dimensions. However, the degree of this reliance is barely predictable for such a large Reynolds number. Therefore, a simple assumption can be taken as a good compromise between complexity and use. After all, the only point here is to scale a coefficient to the turbulence intensity; it does not have much effect on the image of forces acting on the UAV, but rather affects how fast it changes during approach. As a human being is not involved, even an error of 2–3 times should not affect the results. For this research, a direct linear dependency is taken: the length of the airwake changes exactly the times of the ship size. The characteristic width of the airwake is assumed to be the size of the ship itself and be constant along the wake trail.

The direction of the airwake supposed to be aligned with the mean wind over deck vector $\boldsymbol{V}_{\text{wd}}$. The original MIL-F-8785C model is designed for the case when $\boldsymbol{V}_{\text{wd}}$ is directed approximately towards the stern of the carrier; that is, for the case of headwind (more precisely, for the $\boldsymbol{V}_{\text{wd}}$ aligned with the landing runway of the carrier, which is tilted 14 degrees from the longitudinal axis of the ship for the Nimitz-class carrier). This is a common situation for piloted aircraft; however, as discussed in Section 2.1, it may be impractical to change ship heading only for the landing of a small UAV. Therefore, all $\boldsymbol{V}_{\text{wd}}$ directions should be considered.

The $\boldsymbol{V}_{\text{wd}}$ direction affects primarily two airwake components: steady and periodic ones. First, let us consider how side wind could change the steady airwake component.

The figure from [4] which illustrates steady wind functions is shown in Fig. 3.5. A scaled ship silhouette below the X axis demonstrates the relative positions of key points of the functions. It can be seen that the main origin of the airwake (from the point of view of the approaching aircraft) corresponds to the isle position. In addition, strong upwash exists at the upwind edge of the ship deck. These facts are fairly obvious to be taken as assumptions for the extended model.

Although the typical deck layout of a frigate is more complex than that of a carrier, only one point on the deck can be taken realistically as the 'airwake generator' for a simple model. Therefore, the deck is assumed as a flat surface with one island. In addition, given that frigate-class ships are more symmetrical, the model can be simplified further by assuming that

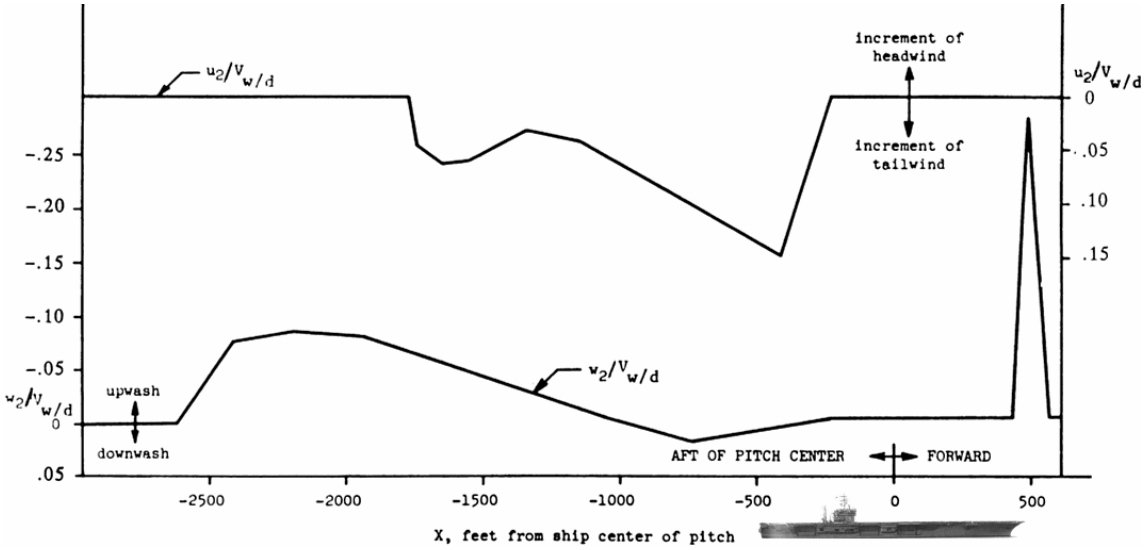


Fig. 3.5 Standard steady airwake component

the island is positioned on the centreline of the deck (Fig. 3.4). This assumption is useful for calculation of the position of 'airwake origin' relative to the ship pitch centre, which is taken as the origin of the ship axis frame. It should be added that the steady component model shown above is valid, apparently, only when the aircraft passes *near* the island, what actually happens with carrier landings. For an aircraft passing *over* the island, the picture of up/downwash will be different; however, this will not be simulated due to lack of actual data. In addition, it is unlikely that the UAV will travel over the ship in the proposed cable hook recovery procedure.

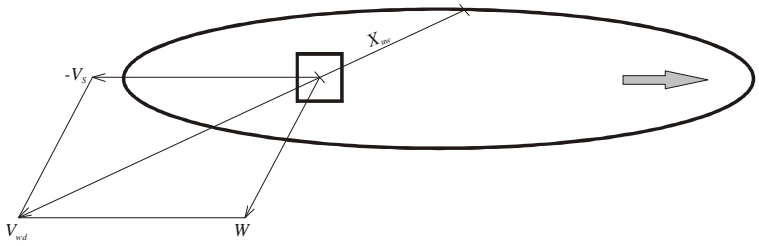


Fig. 3.4

As a result, the difference from the standard military model is the length of the 'flat' upwind portion X_{uw} (see Fig. 3.4) between the 'origin point' (approx. –67 m (–220 ft) in Fig. 3.5) and the 'end point' (150 m (500 ft) in Fig. 3.5). Plus, the distance (X axis) will be scaled down according to the ship size (see above), and the axes origin will be shifted from the pitch centre to the island position to simplify calculations.

Periodic component of the airwake is rather more difficult to generalise. The most important limitations are:

- The MIL-F-8785C model takes into account pitch motion only. This is enough if the glidepath is close to the longitudinal axis of the carrier; however, approaching, for example, from the side of the ship will require taking roll motion into consideration as well. All the same, the influence of pitch and roll motions on the airwake

is fairly different. Unfortunately, there are no specific data available for such a complex motion. It can only be speculated that pitch motion has greater influence on the vertical wind component (and this is seen from [4]), while roll motion influences mostly the longitudinal (tail/headwind) component of the wind, or at least has comparable effect on both of the components due to the geometry of the vessel.

- Normally, operational pitch amplitude of a carrier lies within 2.5–3 degrees even in rough sea conditions [16]. The same amplitude for a smaller ship will be far greater, not to say about roll oscillations which may reach 30° (see Table 3.3 below). Although the amplitude is included into the periodic components linearly ([4], p. 688), this linearity can be invalid beyond the range of motion of a carrier. However, for Sea State 6 (the worst case considered), pitch amplitude of a frigate is about 4.8 degrees (Table 3.3), which is still within linear range. Only roll motion (up to 29 degrees amplitude) falls far beyond.
- No sway or other linear motions are considered. In addition, ship motions are assumed to be simple harmonic oscillations (see Section 3.3). However, these assumptions may be taken as is to keep the model as simple as possible.

Considering the above, it can be concluded that only pitch influence can be simulated realistically enough on the basis of available data. Therefore, the periodic component is limited to about ±20 deg. around the longitudinal axis of the ship. If the UAV approaches from the side of the ship, the component should be inhibited. In addition, in order to estimate the importance of these periodic wind vector oscillations, landings from the stern of the ship will be carried out both with and without the periodic component. If they happen to have great influence, simulation of landings from the side should be considered less valid and suggested for further research.

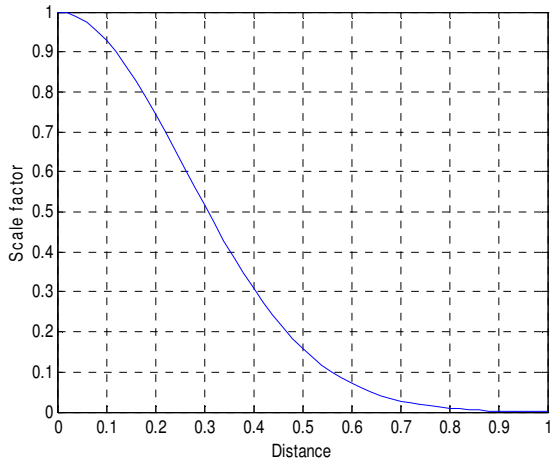


Fig. 3.6 Airwake turbulence intensity as a function of the distance from the airwake trail in ship lengths

The last noted problem, providing the smooth transition across the airwake, is rather qualitative and can be implemented simply enough by applying any appropriate bell-shaped function to the airwake wind vector. The fade-out function chosen is

$$k_s(d) = e^{-(e \cdot d)^2} \tag{3.42}$$

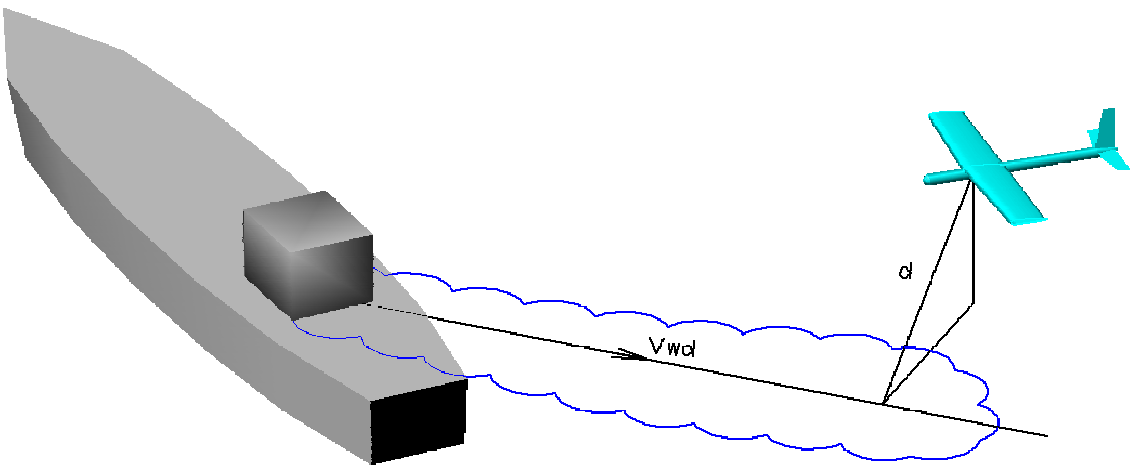


Fig. 3.7 Distance from the airwake

(see Fig. 3.6), where *d* is the distance between the aircraft position and the 'axle' of airwake, expressed in ship lengths (Fig. 3.7).

3.3 Ship model

Ship motion is very important for the simulation environment. It not only directly affects the chances of successful landing, but also influences other parts of the environment, for example, ship airwake properties (see Section 3.2.4 above). The ship model is used to approximate the effects of ship motion in various oceanic conditions.

The model generally follows the one described in [47], which was used in the launch case scenario for the same UAV. Several minor adjustments has been made to allow it to handle specific landing conditions.

The following sections briefly describe both the dynamic model of the motion and the physical model used to estimate the actual parameters of the motion.

3.3.1 Dynamic model

Usually ship motion is dynamically modelled like the motion of any rigid body, using a six-degree-of-freedom model of the ship with equations of motion very similar to those of an aircraft. The forces acting on the ship arise from gravitational, buoyancy, hydrodynamic and propulsive forces.

The motion of a ship is described in terms of three Euler angles *roll, yaw* and *pitch,* relating to rotations about the axis *x, y* and *z* respectively, and three translational terms, called *surge, heave* and *sway,* corresponding to motions in the *x, y* and *z* directions respectively.

However, an accurate simulation of ship motion is not the major issue of this research. In addition, there is no need to simulate any particular existing ship, but rather the motion of some 'abstract' ship of the frigate class. This allows to turn from physical modelling, based on the Newton's second law, to the much simpler statistical modelling, which describes the

motion itself directly. Instead of simulating the forces acting on the body and the applicable laws of nature, this approach simulates the resulting motion directly. The simulated motion is adjusted so that it resembles the motion of a real ship.

The equations for such a motion are derived from the observed motion of a real ship or the comprehensive physical model. Despite the simplicity of the dynamic model, it can take into account a full range of external variables, such as Sea State, ship mass etc.

The most important component affecting the UAV recovery is pseudo-periodical motion caused by oceanic waves. This motion is superimposed with the base forward (or, broadly speaking, commanded) motion, which is usually predefined before simulation and forms one of the landing conditions.

In general, wave motion is highly periodic with slightly varying frequencies and amplitudes. The severity of the wave amplitude is measured by a scale known as the *Sea State.* This scale associates an average wave height with a qualitative description of the waves and assigns it a number. The scale can be found in [2], p. 450 (see also [47]), and is shown in Table 3.2.

Sea State	*World Meteorological Organisation*		*Design Wave Slope*	*Design Wind Speed (kn)*
	Description	*Significant Wave Height (m)*		
0	Calm (glassy)	0	–	–
1	Calm (ripples)	0–0.1	–	–
2	Smooth (wavelets)	0.1–0.5	–	–
3	Slight	0.5–1.25	1:20	15
4	Moderate	1.25–2.5	1:16	20
5	Rough	2.5–4.0	1:12	25
6	Very Rough	4.0–6.0	1:10	30
7	High	6.0–9.0	–	–
8	Very High	9.0–14.0	–	–
9	Phenomenal	Over 14	–	–

Table 3.2 Sea State parameters

The Significant Wave Height (SWH) is defined as the average value of the height (vertical distance between trough and crest) of the largest 1/3 of waves present.

In this research, the ship motion model is based upon the premise of simple harmonic motion:

$$y = A\sin\left(\omega t + \varphi\right) \tag{3.43}$$

where A represents the amplitude of the motion, ω the frequency and φ the initial phase. This assumption is a commonly used approximation for ship motion when the absolute details of the ship motion itself are not the focus of the study. This form of motion should provide a good enough approximation to the ship conditions. However, a few additional observations should be taken into account.

Any simple determined form of motion, like simple harmonic oscillation, can be used only if the landing controller is incapable of predicting the motion on the basis of observable data. (Such prediction can be used for on-the-fly adjustments.) While determined motion can be predicted for virtually any time in advance, in reality ship motion is a stochastic process and the prediction of the ship position for more than few seconds with adequate precision is an extremely challenging task [192, 229]. Therefore, for some types of adaptive controllers (including, of course, a human being as a special case), a more realistic form of ship motion should be sought. In this research, prediction of the ship position is not used in the controller development, therefore simple harmonic oscillations are adopted.

Another consideration indirectly follows from the above one. A determined motion will result in exactly the same ship position for any number of runs with the same timespan. This can bias the results. Therefore, some sort of randomisation should take place, especially between multiple runs with similar conditions. As the amplitude and frequency of the harmonic approximation are predetermined by many variables (the most important of them being the Sea State, the wave direction and the ship speed), a natural way of randomisation is selecting a random initial phase. This should be done separately for each degree of freedom.

As the result of the above, the current model uses a six-degree-of-freedom decoupled harmonic motion approximation to simulate the three translational and three rotational components of ship motion. The initial phase for each degree of freedom is randomised, and the amplitude and frequency for each degree of freedom is determined by the physical model.

3.3.2 Physical model

The data used for the physical model in this research are based on a frigate size naval vessel such as that shown in Fig. 3.8. The Australian navy operates two classes of vessel that fall into this type of ship: the Oliver Hazard Perry Class FFG frigate and the ANZAC Class frigate [47]. These two classes are somewhat similar in size and geometry, with the displacement ranging from about 3600 to 4100 tons (see Fig. 3.9).

Generic data on the parameters of motions were obtained from [99], which describes the motion of the FFG Sydney, and also from the standard statistics used to describe wave motion [173], applied to an ANZAC frigate. The standard statistical form uses a Root Mean Square (RMS) value, defined as the most frequent wave amplitude A_{RMS}. Other important values are derived from the RMS value. These include: the average amplitude, which is defined as $1.25A_{RMS}$; the Significant Wave Height (SWH), defined as the average amplitude of the largest 1/3 waves present, which is $2A_{RMS}$; and the average amplitude of the top 1/10 waves, defined as $2.55A_{RMS}$.

The relationship between the SWH and the Sea State is shown in Table 3.2. The frequencies of motion are independent of the Sea State; however, the wave heading angle does have an effect (likewise, it has an effect on the amplitude). The wave heading angle is defined as the relative heading of the wave to the ship, where 0° corresponds to a wave travelling in the same direction as the ship.

Some of the sample values of the amplitude and frequencies are shown in Table 3.3 and Table 3.4. Obviously, Sea State 8 is an extreme condition, with over 60° roll amplitude. Like with the launch case [47], Sea State 6 has been chosen preliminary as the worst case scenario for landing.

Sea State	2		6		8	
Wave Direction, deg	0	90	0	90	0	90
Surge Amplitude (X), m	0.235	0.149	2.830	1.790	6.604	4.176
Heave Amplitude (Y), m	0.211	0.277	2.532	3.327	5.908	7.764
Sway Amplitude (Z), m	0.137	0.232	1.652	2.792	3.855	6.515
Roll Amplitude (γ), deg	2.216	2.385	26.59	28.62	62.05	66.80
Yaw Amplitude (ψ), deg	0.165	0.175	1.991	2.106	4.647	4.914
Pitch Amplitude (θ), deg	0.399	0.308	4.791	3.697	11.18	8.627

Table 3.3 Sample Ship Amplitudes (top 1/10 waves)

Wave Direction, deg	0	30	60	90	120	150	180
Surge Frequency, Hz	0.073	0.074	0.080	0.089	0.096	0.096	0.096
Heave Frequency, Hz	0.076	0.082	0.090	0.098	0.099	0.098	0.097
Sway Frequency, Hz	0.082	0.086	0.089	0.095	0.095	0.097	0.095
Roll Frequency, Hz	0.114	0.113	0.114	0.115	0.116	0.116	0.116
Yaw Frequency, Hz	0.102	0.099	0.106	0.104	0.108	0.106	0.108
Pitch Frequency, Hz	0.086	0.087	0.102	0.116	0.128	0.125	0.124

Table 3.4 Sample Ship Frequencies

Another important thing that should be taken into consideration is the location of the landing facilities. The ship motion described above applies to the centre of mass of the ship. However, the recovery boom is likely to be located away from the c.g. Apart from significant periodic displacement, this induces additional velocity to the boom due to angular motion of the ship. This is analysed in the following section.

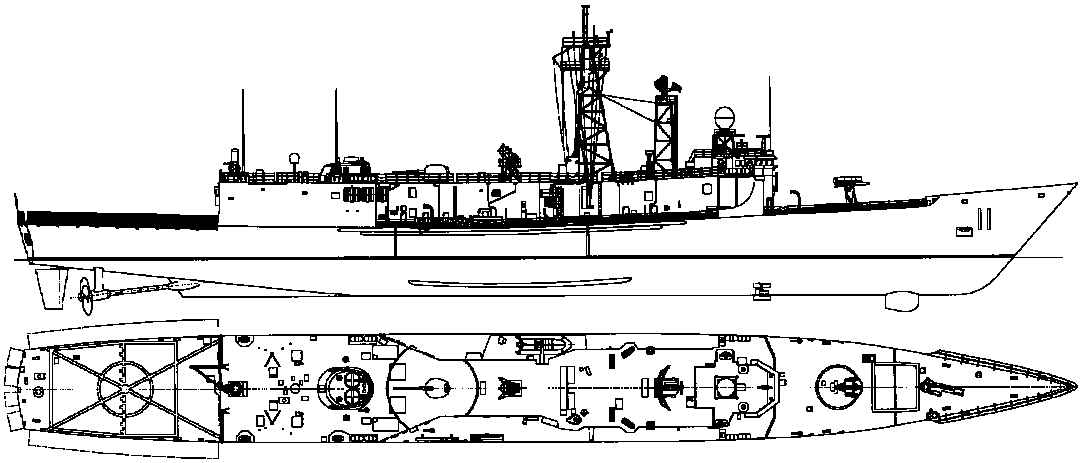


Fig. 3.8 Perry Clark FFG11 frigate

a) ANZAC FFH-150 frigate

b) Oliver Hazard frigate

c) Helideck of the ANZAC frigate

Fig. 3.9 Frigate class naval vessel

3.4 Shipboard recovery gear

In Chapter 2, Section 2.3.6.2, two of the most practical sites on the ship for the recovery boom have been determined. Later in Section 2.4.3.1, the requirements to the height of the boom has been analysed. In this section, these locations are modelled and the amount of periodic motion at these locations is analysed.

For the model, the dimensions of an ANZAC frigate [179] are used for reference. These vessels have the length of 118 m and the beam (width) 14.8 m. For simplicity, the centre of mass is taken at the geometrical centre of the circumscribed rectangle, i.e. 59 m from both the bow and the stern, on the centreline and on the water level. The upper deck level is located approximately 11 m above the waterline. The bow (nose tip) of these frigates is about 2 m lower (9 m above the waterline); however, on some other similar-sized ships such as Perry Clark frigate (Fig. 3.9) it is as high as the upper deck. For simplicity and in order to obtain more comparable results between the boom locations, both points are taken at an 11 m level, and the geometry of the recovery gear (the boom, the arresting wire with brakes/winches and the supporting tower or mast) is assumed to be similar in both cases for this research. In reality, an inclined mast at the bow, as illustrated in Fig. 2.12, might be more appropriate due to both easier integration into the ship and better clearance to the ship structure. An outwards inclined pole or a longer boom may also be required for the alternative side location to improve the side clearance.

The boom length is taken as 6 m, which is approximately two wingspans of the UAV for which the recovery system is being designed. It is believed that the guidance system will be able to fly the aircraft through a 5 m wide gate (leaving 0.5 m safety margins on both sides). However, the length may be adjusted in view of the simulation results.

The side location of the recovery boom is illustrated in Fig. 3.10. The port board is

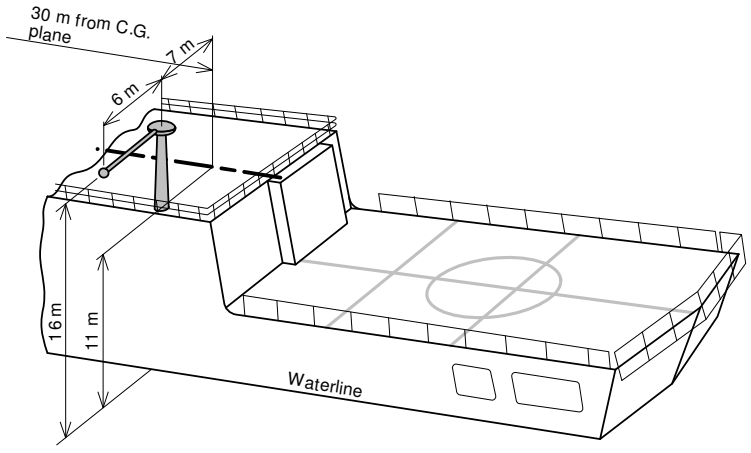


Fig. 3.10 The port board location of the recovery boom

chosen arbitrarily, as the ship is considered symmetrical for the purposes of this study. The longitudinal location (–30 m) is selected so that the UAV could be delivered directly onto the lower deck after recovery by swinging the boom backwards. The working boom position is fixed at 90 degrees to the centreline of the ship, allowing only two approach directions: from the stern ('chasing') and from the bow ('heading').

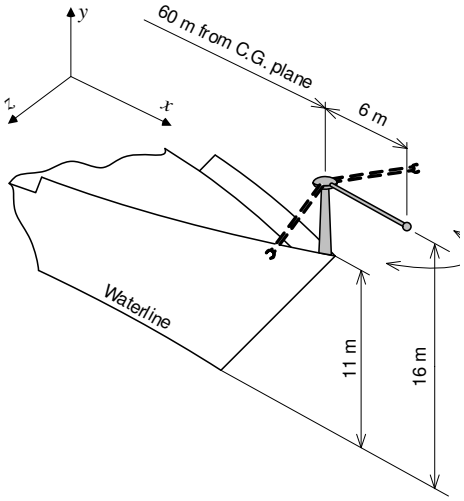


Fig. 3.11 The bow location of the recovery boom

The bow location is shown in Fig. 3.11. It has an advantage of allowing to swing the boom 70 degrees to either side without compromising safety clearances (see Fig. 2.12), which enables the choice of any approach direction within the 320° sector (taking into account two directions for every boom position). On the other hand, it is located approximately twice as far away from the ship's centre of mass (note that the distance is slightly rounded with respect to ANZAC FFH length), which potentially causes greater sensitivity to ship motion.

3.4.1 Recovery boom oscillations

The amount of oscillations of the boom due to sea waves can be estimated in advance so as to choose the main location for the study. Until the analysis of the simulations is done, it is unclear if a better choice of wind would outweigh potentially greater boom oscillations and the infrastructural inconvenience of the bow location. However, the preliminary analysis may give some insight.

The periodic motion of a ship is usually separated into three translational (*surge, heave, sway*) and three angular (*roll, yaw, pitch*) motions (refer to Section 3.3.1 for details). The motion of a given point of the ship then consists of the same translational motion plus the circular motion about the centre of gravity (c.g.):

$$\Delta \boldsymbol{x} = \Delta \boldsymbol{x}_{\text{c.g.}} + \Delta \boldsymbol{\theta}_{\text{s}} \times \boldsymbol{r} \tag{3.44}$$

where $\Delta \boldsymbol{x}$ is the displacement of the point x located at the radius $\boldsymbol{r}$ from the c.g. due to ship's translation $\Delta \boldsymbol{x}_{\text{c.g.}}$ and rotation $\Delta \boldsymbol{\theta}_{\text{s}}$. Therefore, the radius of the motion $\boldsymbol{r}$ (i.e. the distance from the c.g. to the point) is needed to determine the amplitude of a specified location on the ship. Note that for small angular amplitudes (in most operational cases the amplitude of ship angular motion lies within 5 degrees, see Section 3.3.2), the trajectory of a given point will be

nearly linear, thus the translational amplitude of the point (due to angular motion of the ship) will be simply proportional to both the angular amplitude and the radius ***r***.

In Table 3.5, the coordinates of the recovery gear locations are summarised. The ship coordinate frame follows the axis layout of an aircraft, assumed for this research (see Section 3.1.1.1): *x* forward, *y* up and *z* right (see Fig. 3.11), with the origin at the centre of mass of the ship.

Boom location / *Point*	Side			Bow (boom 0°)			Bow (boom ±70°)		
	x	*y*	*z*	*x*	*y*	*z*	*x*	*y*	*z*
Pole root	–30	11	–7	60	11	0	60	11	0
Pole top / Boom root	–30	16	–7	60	16	0	60	16	0
Boom centre	–30	16	–10	63	16	0	61.0	16	±2.82
Boom tip	–30	16	–13	66	16	0	62.1	16	±5.64

Table 3.5 Recovery boom coordinates (metres)

As the full motion of the ship is considered separately along and about each axis (de-coupled model), three distances in each of the three orthogonal planes *xy* (pitch motion), *xz* (yaw motion) and *yz* (roll motion) are calculated (see Table 3.6). This is done in a usual manner for Euclidian distance, e.g. $d_{xy} = \sqrt{x^2 + y^2}$ for the pitch motion. Only the centre of the boom is taken as the characteristic point.

Location / *Point*	Side			Bow (boom 0°)			Bow (boom 70°)		
	Roll	*Yaw*	*Pitch*	*Roll*	*Yaw*	*Pitch*	*Roll*	*Yaw*	*Pitch*
Boom centre	18.9	31.6	34.0	16.0	63.0	65.0	16.2	61.1	63.1

Table 3.6 Effective distances from the c.g. for each angular motion (metres)

Analysing Table 3.3 from Section 3.3.2 (Sample ship amplitudes), it can be concluded that the motion pattern remains nearly the same across the Sea States. In particular, yaw amplitude is always about 7.4% of roll amplitude, and pitch amplitude is 12.9% to 18% of roll amplitude, depending on wave direction (90 to 0 degrees respectively). A similar picture is obtained for translational components. The amplitudes of ship motion at Sea State 6 ('very rough sea') are 12 times those at Sea State 2 ('smooth sea'). Therefore, the influence of the angular motion can be analysed for any one Sea State. In Table 3.7, the amplitudes of the centre of the boom at Sea State 6 (top 1/10 waves) are given. The worst case is taken for estimation: the maximum amplitudes are selected from either 0 or 90 degrees wave direction.

	Side location			Bow location			Translation (both locations)		
	Roll	*Yaw*	*Pitch*	*Roll*	*Yaw*	*Pitch*	*Surge*	*Heave*	*Sway*
Boom centre (neutral position)	9.44	1.16	2.84	7.99	2.32	5.44	2.83	3.33	2.79
Boom centre (70° position)	—	—	—	8.09	2.25	5.28	"	"	"

Table 3.7 Amplitudes of oscillation of the centre of recovery boom due to ship's angular and translational periodic motion (Sea State 6, conservative estimation), metres

It can be seen that, naturally, roll accounts for the most severe oscillations. Even at the bow, which is located nearly four times as far forward from the c.g. as it is up from the c.g., roll delivers greater displacement than pitch, which is due to the far greater roll amplitude of the ship. However, it can be expected that the oscillations in different planes have a significantly different effect on the recovery process. Indeed, the movement along the flight path (forward/backward) may have very little effect as compared to vertical or sideways displacement. In particular, quite strong oscillations due to roll at the bow location (neutral boom position) will be seen from the point of view of approaching UAV as a periodic forward/backward displacement of the boom, which likely does not need any compensation at all. In contrast, the same ship roll will account for vertical and transverse displacement of the boom located on the side of the ship. In addition, the angular motion of the boom, particularly in the vertical plane, which is seen by the UAV as rocking of the boom from side to side, may well have substantial impact on the approach. Therefore, an analysis of the periodic motion from the point of view of the UAV will be more illustrative than that given for the ship (Table 3.7).

The vertical (Δh) and transverse (Δw) amplitudes can be calculated using the data from Table 3.7 taking into account the relative position of the boom to the c.g. in the respective planes (Fig. 3.12 for the transverse plane). For simplicity, the motion is linearised in view of typically small angular amplitudes. Bearing in mind that the UAV normally approaches perpendicularly to the recovery boom with small descent angle, the motion is considered in the vertical plane containing the boom. For the side boom location, this is the ship's cross-section plane, and the roll amplitude is the most critical. Additionally, ship pitch and yaw may cause a substantial amount of vertical and lateral translation to the boom. In the case of the bow location, the plane and motion of interest depend on the orientation of the boom. In the neutral position, when the UAV approaches with ±90 degree course to the ship, pitch motion of the ship will cause the most troublesome oscillations to the boom. However, with the extreme 70 degrees boom position, all three components of the angular motion will have a significant effect. It can be determined by taking a projection of each amplitude onto the vertical boom plane.

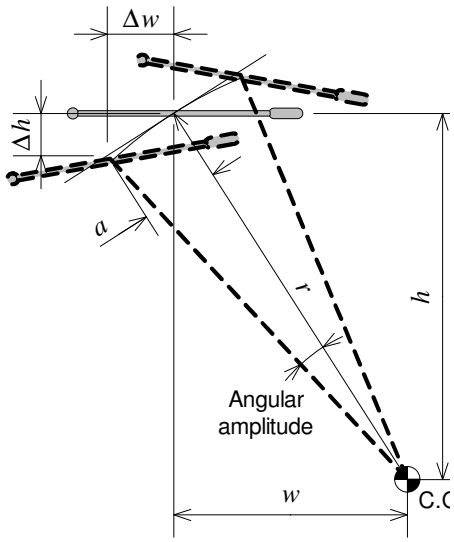


Fig. 3.12 Vertical and transverse amplitudes of the recovery boom due to transverse (with respect to the flight path) ship oscillations

The amplitudes in the boom plane are calculated as follows (refer to Fig. 3.12):

$$\Delta h = \frac{aw}{r}, \quad \Delta w = \frac{ah}{r} \tag{3.45}$$

where $r = \sqrt{h^2 + w^2}$ is the distance from c.g. to the centre of the boom in the respective orthogonal plane (Table 3.6), coordinates *w* and *h* are taken from Table 3.5, and *a* is the linear amplitude at the boom centre due to rotation (Table 3.7). The results are summarised in Table 3.8.

Type of motion	*Boom location* / *due to ship's…*	Side	Bow (0°)	Bow (70°)
Vertical travel, m (from neutral position)	Lateral rotation	5.0 (roll)	5.3 (pitch)	1.4 (mostly roll)
	Longitudinal rotation	2.5 (pitch)	1.0[1] (roll)	5.1 (mostly pitch)
	Vertical translation	3.3 (heave)	3.3 (heave)	3.3 (heave)
	Total	**10.8**	**9.6**	**9.8**
Lateral travel, m (from neutral position)	Lateral rotation	8.0 (roll)	1.3 (pitch)	8.0 (mostly roll)
	Directional rotation	1.1 (yaw)	~0 (yaw)	2.1 (yaw)
	Lateral translation	2.8 (sway)	2.8 (surge)	3.6 (mostly sway)
	Total	**11.9**	**4.1**	**13.7**
Lateral rocking, °	**Lateral rotation**	28.6 **(roll)**	4.8 **(pitch)**	28.7 **(mostly roll)**

Table 3.8 Amplitudes of periodic motion at the centre of the recovery boom with respect to the UAV approach path (Sea State 6, conservative estimation)

Two notes should be kept in mind when analysing Table 3.8. First, similarly to all calculations above, this is a conservative estimation of the worst case: the maximum amplitudes of the ship motion in all possible circumstances are taken. They never realise at once. For example, when the maximum roll amplitude (28.6° at Sea State 6) is possible, i.e. when the waves travel at 90° course to the ship, the maximum pitch amplitude is about 1° less than the absolute maximum (which is achieved at 0° wave course (see Table 3.3)), and vice versa.

Second, all six components of the periodic motion of the ship are considered independent in this study. While a certain coupling between these components may exist (in a given set of conditions), the exact details of ship motion is not the focus of this research (see Section 3.3). Therefore, summing up the amplitudes of the independent components (rows 'Total' in Table 3.8) does not provide an adequate estimation of amplitude of the resulting superimposed motion. The boom travel may reach these values only in a very unlikely case when the phases of each motion (which all have different frequencies, see Table 3.4) meet at the peak levels. However, on average, the greater the 'total' figure, the larger the amplitude should be expected.

To give a better visualisation of the actual position of the boom, a 120-second sample of motion has been obtained via simulation of the ship model. The conditions were set the

[1] This figure is obtained as half the height of a 2×28.6=57.2° arc with the radius 16 m (the double roll amplitude (full travel) and the distance from the roll axis to the boom). This is the case when nonlinearity of motion plays a small role, producing some vertical movement even for centred position ($w = 0$ in Fig. 3.12). In a slightly calmer sea, however, this motion will be virtually negligible.

same as used above, i.e. Sea State 6, top 1/10 waves, except for the wave direction, which was fixed at 90° (maximum roll; slightly in favour of the pitch dominated bow location with 0° boom angle). The measurements were taken simultaneously at all locations, i.e. all the graphs represent one realisation of the random ship motion. For convenience, positions are shown relative to the respective neutral position of the boom, e.g. '0' on the graphs of vertical position corresponds to 16 m (refer to Table 3.5). The graphs are presented in Fig. 3.13.

It should be noted that the obtained form of motion does not represent the 'shape' of motion of a real ship that could be measured even at a perfectly 'calibrated' Sea State 6. At real sea, wave heights occupy a fairly wide range of values (see Section 3.3.2), while the simulated motion reflects only a homogeneous sea. However, current estimations use a very conservative case of a sea consisting of the highest 1/10 waves normally occurring at Sea State 6. The average wave amplitude is about two times less. As a result, the real motion is expected to be less regular than simulated and with smaller average amplitude at the same conditions.

It can be seen from the graphs that the estimated maximum travel (double amplitude) satisfactorily reflect the observed boom travel, although being up to 45% conservative (which is partly due to a relatively short period of observation and about 25% smaller pitch amplitude):

Location / *Full travel*	Side	Bow (0°)	Bow (70°)
Estimated maximum vertical travel	*21.6*	*19.2*	*19.6*
Observed vertical travel	16.7	15.1	13.5
Estimated maximum lateral travel	*23.8*	*8.2*	*27.4*
Observed lateral travel	21.2	5.6	23.9

Table 3.9 Amount of periodic motion of the recovery boom, conservative estimation and observed values (metres)

It is worth noting that the roll dominated motions (especially at the side location) are slightly biased to either positive or negative values. This is due to non-linearity of such a violent roll motion at Sea State 6 (almost 30 degrees amplitude). In a calmer sea, this bias will be less noticeable. This may have a certain impact on the recovery process, because the 'middle point' to aim at will be slightly displaced (down and inwards to the ship) at high Sea States.

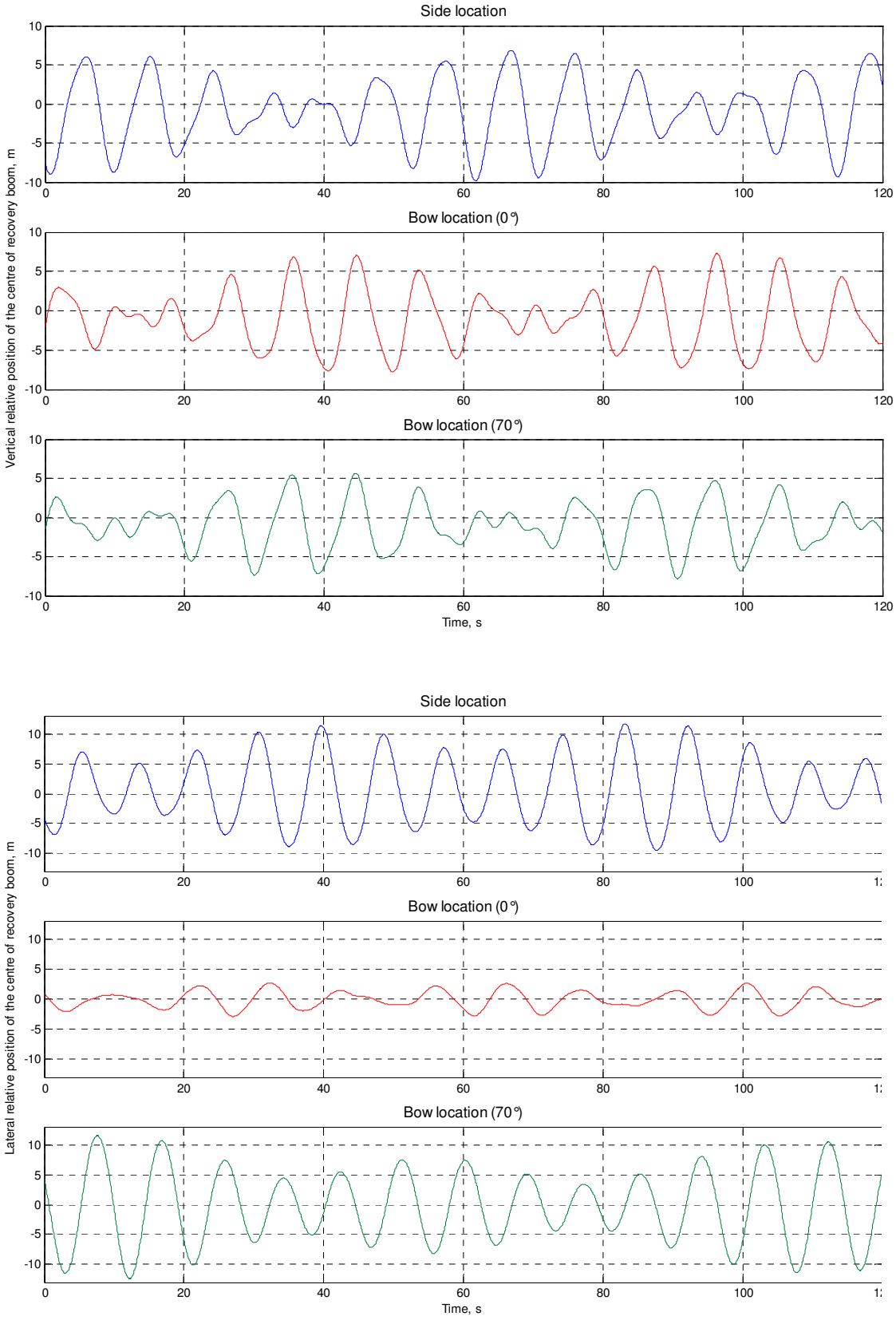


Fig. 3.13 A sample of the periodic ship motion at different locations as seen from the approaching UAV

It is also useful to keep in mind that the final approach takes about 8 to 20 seconds, which covers about 1 to 2 full periods of typical motion (the frequencies are independent of Sea State, see Section 3.3.2).

Overall, it may be reasonable to expect that such intense oscillations (up to four times the length of the boom) will appear too difficult for the UAV to cope with. In order not to overtighten the weather limits, it may be more sensible not to use such overconservative parameters of ship motion. In particular, using the Significant Wave Height (top 1/3 waves) instead of the top 1/10 waves (i.e. 28% smaller amplitude) provides a more realistic estimation in most of the cases, while being still fairly conservative (1.6 times the average wave height).

From the point of view of the disturbances which the UAV must compensate, the bow location offers a significant advantage in terms of lateral amplitude (both translational and angular) when the boom is positioned neutral. However, at the extreme 70° boom position, the level of disturbances is comparable to that observed at the side location which is twice as close to the centre of mass. Nevertheless, the flexibility of choosing the approach direction which the swinging boom offers allows to suggest the bow location as the preferable one, despite its potential infrastructural inconveniences (see also Section 2.3.6.2). Even if simulations show that the UAV can handle side wind so well that adjusting the approach direction is not necessary, the bow location with a fixed boom can still provide much smaller lateral amplitude of the arresting wire and better clearance to the ship structure. This may improve the landing envelope in extreme conditions.

3.5 Cable model

As discussed in the previous chapter (Sections 2.3.6.2 and 2.4), the recovery method being developed involves capture of a damped shipboard arresting wire by an onboard hook. This hook is attached to a relatively long yet lightweight flexible line (also referred as cable), trailing behind the aircraft. The line is extended from the aircraft by the means of a motorised or freely spun winch just before entering the landing glidepath.

The trailing line parameters greatly influence recovery capabilities and aircraft performance. The length and geometrical shape of the line determines recovery conditions, while the additional drag it delivers affects the UAV's characteristics. Both shape and drag change dynamically during the flight, therefore a dynamic model of the line is required for correct simulation and analysis.

Basic parameters of the line were determined in Section 2.4.3. Table 3.10 summarises the cable hook data, used in this work.

Item	*Properties*
Cable material	Nylon twisted rope 6.35 mm diameter
Cable length	10 to 15 m
Cable linear density	0.0223 kg/m
Cable drag coefficient	1.2 (with respect to cross-section)
Hook mass	0.1 kg
Hook shape equivalent	0.05 m diameter ball

Table 3.10 Recovery line parameters

3.5.1 Model design

There is a number of cable models in literature designed for various applications, from underwater towing [9] to vibration dynamic response simulations [116]. Several more generic models are proposed as well [38, 100]. In 2002, the *RMIT Towed Body Systems Research Group* produced a model and the software for an undersea towing cable under contract to DSTO Australia. This model has been used as a theoretical basis for the current development. The report [213] contains the outline of the both historic and design aspects of cable modelling, and the reader should refer to this report for such details. This section will only describe the actual model implementation used in this work.

During preliminary investigations, the model [213] has been re-implemented in MATLAB for simulation with the *Ariel* model (Section 3.1). Unfortunately, it revealed some disadvantages which made it difficult to use the model for the required task. The most significant problem was that due to the low mass of the cable and relatively high tensional stiffness, the model was prone to spurious high-frequency longitudinal oscillations. Carefully chosen internal damping could not eliminate the problem completely, because it is attributed not only to accumulating round-off errors, but also to the high natural frequency of a lightweight cable. This means that a very small time step was necessary to simulate the cable accurately.

Whilst there exist models which specifically address this problem (e.g. [116]), it appeared to be reasonable to eliminate the longitudinal (tensional) degree of freedom. Indeed, during the flight, the stretching forces on the line come only from its own weight, aerodynamic drag and inertia. They are all relatively small due to the low mass and diameter of the cable and are several orders of magnitude below the breaking strength of the rope. (The approach is simulated only to the point of the arresting wire catch-up). Therefore, the line can be considered inextensible. Slackness (negative tension) is not expected as well. Moreover, for the application considered in this work high-frequency modes, either tensional or transverse, are not important, because the dynamics of both the UAV and the ship is of several orders slower.

In addition, from the point of view of aerial use, the RMIT's model [213] is overcomplicated in other aspects and may be reduced. Nevertheless, it can be used as the basis for the

current model, particularly its force equations, and as a sample for validation purposes. The following list outlines the features of the model being developed:

- Finite element method with straight and rigid links. For simplicity of formulation, lumped mass approach is used.
- The links are connected by zero-friction 2-degree-of-freedom (ball) joints. This approximation is reasonable due to very low bending stiffness of the thin rope (with respect to its length).
- Cable twist and torsional stiffness, as well as slack conditions, are not considered.
- The cable is constrained by the UAV at one end and has an unconstrained end-body on the other end.
- Handling of cable deployment and retrieval is not required.
- The model is intended to be used mostly in conjunction with the UAV model. Therefore, both velocity and acceleration at the UAV point are available.
- Buoyancy and the added mass effect are negligible.

With the above requirements, the model can be formulated relatively simply. For simulation, the model has been implemented as a Simulink S-Function in MATLAB language.[1]

3.5.2 Kinematic equations

The cable is divided into *N* rigid links with the masses lumped at the far (from the UAV) end of each link (Fig. 3.14). With adequate discretisation, the lumped mass representation provides good accuracy while simplifying handling of inter-element coupling [213].

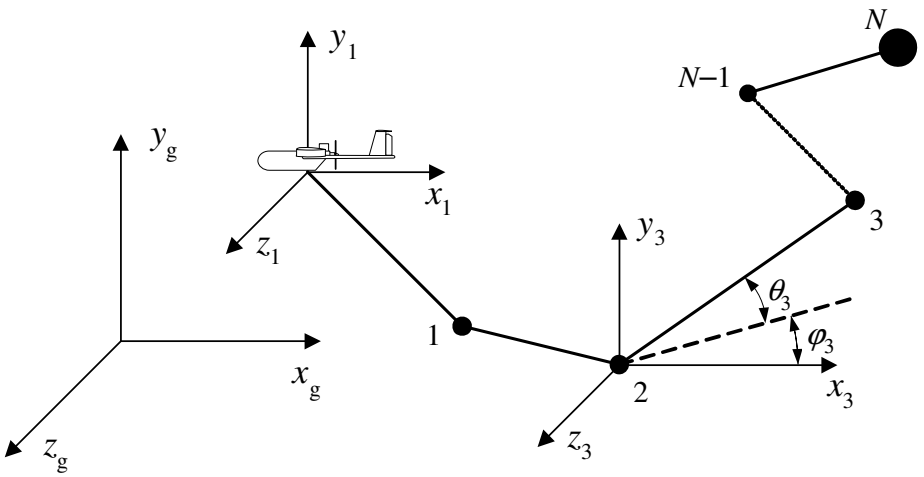


Fig. 3.14 Cable model kinematics

[1] MATLAB has a convenient and powerful toolbox for simulation of mechanical systems, called SimMechanics. However, it could not be used due to licensing costs.

The links are numbered 1 to N, starting from the UAV. The mass of the end body (the hook) is added to the last segment. The inertial (world, or ground) system $x_g y_g z_g$ is the same as used for the UAV model (see Fig. 3.1).

The motion of each link is considered in its own local axis frame $x_i y_i z_i$, where $i = 1 \ldots N$ is the link number. The axes direction coincides with that of the ground system; each local system can translate, but does not rotate. The origin is located at the front (closer to the UAV) end of the respective link, or which is the same, at the lumped mass of the previous link. This local frame is not inertial, thus inertial effects must be included when calculating local forces.

In the local frame, each link has two rotational degrees of freedom. The rotations are described by the azimuth angle φ_i about y_i axis and the latitude angle θ_i about new z_i' axis. As the length of each link l_i is known, the geometry of the whole cable is described by $2N$ parameters, plus three coordinates of the origin (attachment) point in the ground system to locate the cable position.

Since the joints are considered ideal due to cable flexibility, the coupling between the adjoining segments consists of tension only. Only the longitudinal component of the whole force spectrum of each segment is passed to its neighbour. Therefore, to make the force calculation more convenient, *local body frame* $x_{bi} y_{bi} z_{bi}$ (or normal-tangential frame, subscript b) is also used. It has the same origin as the respective $x_i y_i z_i$ frame, but is rotated by the kinematic angles φ_i and θ_i (in that order) so that direction of x_{bi} axis follows the link and points towards the lumped mass. The rotation from the local (or ground) to body frame can be expressed by the following direction cosine matrix (DCM) (cp. with (3.6)):

$$C_{gbi} = \begin{bmatrix} \cos\theta_i \cos\varphi_i & \sin\theta_i & -\cos\theta_i \sin\varphi_i \\ -\sin\theta_i \cos\varphi_i & \cos\theta_i & \sin\theta_i \sin\varphi_i \\ \sin\varphi_i & 0 & \cos\varphi_i \end{bmatrix} \tag{3.46}$$

The motion of each segment is formulated in terms of angular momentum and angular velocity (see Section 3.1.1.3 and equation (3.14)). In the case of a lumped mass m_i located at the distance l_i from the centre of rotation, the moment of inertia I_i becomes simply $m_i l_i^2$, and the equation for angular acceleration yields

$$\dot{\boldsymbol{\omega}}_i = I_i^{-1} \boldsymbol{M}_i = \frac{\boldsymbol{M}_i}{m_i l_i^2} \tag{3.47}$$

In the lumped mass model, the forces are grouped at each mass. Therefore, the moment $\boldsymbol{M}_i$ can be expressed as $\boldsymbol{F}_i l_i$, where $\boldsymbol{F}_i$ is the total force acting on the ith mass which has the arm l_i relative to the origin. As a result, the angular acceleration is

$$\dot{\boldsymbol{\omega}}_i = \frac{\boldsymbol{F}_i}{m_i l_i} \tag{3.48}$$

The transformation from the angular velocity $\boldsymbol{\omega}_i$ to velocities of the kinematic angles φ_i and θ_i represents some difficulties. The transformation has singularities at $\theta_i = \pm 90°$ (see (3.18), the equation is similar ignoring the roll angle γ), leading to the phenomenon known as *gimbal lock*. While these angles are unusual for an aircraft in normal flight conditions (unless it is an aerobatic or other agile aircraft), a cable segment can be placed in any position. In particular, $\theta = -90°$ represents a hanging down cable, which is a very common condition and should be expected in simulation. Therefore, a different representation is used which is not prone to gimbal lock.

The method involves the concept of *quaternions* as a representation of a rotation in 3D space. Their exact formulation can be found in various mathematical and engineering literature (e.g. [40, 125]), and only a brief description with application to spatial rotation will be given here.

3.5.2.1 Quaternions and 3D rotation

Mathematically, quaternion is a four-element extension of complex numbers with three independent imaginary units *i, j* and *k* satisfying the following relations:

$$\begin{aligned} i^2 = j^2 = k^2 = ijk = -1 \\ ij = k \quad ji = -k \\ jk = i \quad kj = -i \\ ki = j \quad ik = -j \end{aligned} \tag{3.49}$$

Every quaternion is a linear combination of the *basis quaternions* 1, *i, j* and *k*, i.e. every quaternion is uniquely expressible in the form

$$z = a + bi + cj + dk \tag{3.50}$$

where *a, b, c* and *d* are real numbers. In a manner somewhat similar to complex numbers, quaternion can be viewed as a sum $a + \boldsymbol{u}$ of a real part *a* and an imaginary 3D vector $\boldsymbol{u} = (b, c, d) = b\boldsymbol{i} + c\boldsymbol{j} + d\boldsymbol{k}$, where $\boldsymbol{i}, \boldsymbol{j}$ and $\boldsymbol{k}$ correspond to the orthogonal unit vectors in a Cartesian axes system.

Being a representation of a real parameter and a vector, quaternion can be thought of as a rotation characterised by the real part *a* about the vector $\boldsymbol{u}$. Henceforth, for consistency and convenience of use, only normalised quaternions and vectors will be considered, that is, those having modulus 1:

$$\hat{\boldsymbol{u}} = \frac{\boldsymbol{u}}{\|\boldsymbol{u}\|}, \quad \hat{z} = \frac{z}{\|z\|} \tag{3.51}$$

One of the computational advantages of quaternions before direction cosine matrices is simplicity of normalisation, which becomes necessary due to round-off errors. The quaternion's unit vectors are orthogonal by definition, while DCM can lose its orthogonality.

It turns out that, for a normalised vector, an anti-clockwise rotation by the angle α can be represented as

$$z = \cos\frac{\alpha}{2} + \sin\frac{\alpha}{2}\hat{\boldsymbol{u}} \tag{3.52}$$

which, as can be easily shown, produces a normalised quaternion.

Composition of arbitrary rotations corresponds to left-to-right quaternion multiplication. As is natural with rotations, the sequence of rotations is important, and the multiplication is non-commutative (see (3.49)). It can be also noted that z and $-z$ represent the same rotation.

The integration of angular motion with angular velocity $\boldsymbol{\omega}_\mathrm{b} = (\omega_{x\mathrm{b}}, \omega_{y\mathrm{b}}, \omega_{z\mathrm{b}})$ in local body frame can be done directly to quaternion [40, 141][1]:

$$\begin{bmatrix} \dot{a} \\ \dot{b} \\ \dot{c} \\ \dot{d} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} -b & -c & -d \\ a & -d & c \\ d & a & -b \\ -c & b & a \end{bmatrix}\begin{bmatrix} \omega_{x\mathrm{b}} \\ \omega_{y\mathrm{b}} \\ \omega_{z\mathrm{b}} \end{bmatrix} \tag{3.53}$$

For the cable model, twisting $\omega_{x\mathrm{b}}$ is neglected and thus (3.53) can be reduced accordingly. In addition, for numerical integration, quaternion normalisation can be implemented,[2] thus producing:

$$\begin{bmatrix} \dot{a} \\ \dot{b} \\ \dot{c} \\ \dot{d} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} -c & -d \\ -d & c \\ a & -b \\ b & a \end{bmatrix}\begin{bmatrix} \omega_{y\mathrm{b}} \\ \omega_{z\mathrm{b}} \end{bmatrix} + K\varepsilon\begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \tag{3.54}$$
$$\varepsilon = 1 - \left(a^2 + b^2 + c^2 + d^2\right)$$

where K is the normalisation gain ($K = 1$ has been used in this work).

For convenient kinematic calculations, a corresponding DCM and Euler angles representation can be produced as follows:

$$C_{\mathrm{gb}} = \begin{bmatrix} a^2 + b^2 - c^2 - d^2 & 2(bc + ad) & 2(bd - ac) \\ 2(bc - ad) & a^2 - b^2 + c^2 - d^2 & 2(cd + ab) \\ 2(bd + ac) & 2(cd - ab) & a^2 - b^2 - c^2 + d^2 \end{bmatrix} \tag{3.55}$$

[1] The MATLAB 6.5 documentation [141] is inconsistent with the actual block implementation. The actual block interconnection (which conforms to other sources) has been used here for reference.

[2] The actual simulations showed that for 8 to 20 seconds of integration, as used in this study, significant round-off errors do not yet accumulate and normalisation is not required.

Comparing it with (3.46), the following expressions for the two angles of rotation can be obtained:

$$\varphi = \arctan\frac{-C_{\mathrm{gb}(1,3)}}{C_{\mathrm{gb}(1,1)}} = \arctan\frac{-2\left(bd-ac\right)}{a^2+b^2-c^2-d^2}$$
$$\theta = \arcsin C_{\mathrm{gb}(1,2)} = \arcsin\left(2\left(bc+ad\right)\right) \tag{3.56}$$

It should be noted that ignoring possible round-off errors, this conversion always produces a correct result without singularities, provided the quaternion is normalised (and thus every element of the matrix (3.55) is within the range [–1, 1]) and four-quadrant arctangent calculation (atan2) is performed.

As a result, the integration of the cable model is done through the equations (3.48) and (3.54) for each segment. If traditional (instead of quaternion) representation is needed for geometry calculations, it can be obtained using (3.55) and (3.56).

3.5.3 Force equations

The force model is intended to accurately simulate the environmental conditions of the cable. Since the cable is assumed to be inextensible and transversely flexible, the set of forces acting on each body consists of (see Fig. 3.15) gravity $\boldsymbol{F}_{\mathrm{G}}$, inertia $\boldsymbol{F}_{\mathrm{I}}$, aerodynamic drag $\boldsymbol{F}_{\mathrm{D}}$ and tension from the neighbouring segment $\boldsymbol{F}_{\mathrm{T}}$. The forces are expressed in the non-inertial local frame $x_i y_i z_i$ (see Fig. 3.14); all these frames have the same orientation, which facilitates inter-link coupling. Some forces are naturally expressed in the body frame $x_{\mathrm{b}i} y_{\mathrm{b}i} z_{\mathrm{b}i}$ and can be converted to the local frame via the respective DCM (transposed (3.55)).

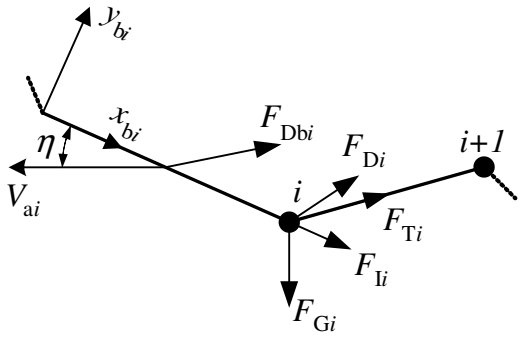


Fig. 3.15 Forces acting on a cable segment (shown in 2D for simplicity)

$$\boldsymbol{F}_i = \boldsymbol{F}_{\mathrm{G}i} + \boldsymbol{F}_{\mathrm{I}i} + \boldsymbol{F}_{\mathrm{D}i} + \boldsymbol{F}_{\mathrm{T}i} \tag{3.57}$$

3.5.3.1 Gravity forces

The gravitational acceleration is assumed to be constant and in $-y_g$ direction. Therefore, the gravitational force in ground frame can be written simply as

$$\boldsymbol{F}_{\mathrm{G}i} = \begin{bmatrix} 0 \\ -m_i g \\ 0 \end{bmatrix} \tag{3.58}$$

where m_i is the mass of the ith segment.

3.5.3.2 Inertial forces

Inertial forces arise from non-inertiality of the body frame in which the forces are considered. It should be noted that inertial forces are fictitious and reflect the phenomenon of inertia, which can be viewed as a force in a non-inertial frame for convenience of force balance representation.

As the mass is assumed to be constant (no added mass effect), the inertial forces can be written using the simplest form of the Newton's Second Law, taking note of the opposite direction:

$$\boldsymbol{F}_{\mathrm{Ib}i} = -m_i \boldsymbol{a}_{\mathrm{Ib}i} \tag{3.59}$$

It is important to include only the accelerations $a_{\mathrm{Ib}i}$ of the body frame itself (at the point of the lumped mass), not the accelerations arising from the current forces. The frame accelerations come from both translational acceleration of the origin and rotation of the frame.

For the mass m_i attached at the distance l_i along the $x_{\mathrm{b}i}$ axis (see Fig. 3.15), the centripetal acceleration due to frame rotation will be

$$\boldsymbol{a}_{\mathrm{Irb}i} = \begin{bmatrix} \left|\boldsymbol{\omega}_{\mathrm{b}i}\right|^2 l_i \\ 0 \\ 0 \end{bmatrix} \tag{3.60}$$

where $\boldsymbol{\omega}_{\mathrm{b}i}$ is the known (through integration (3.48)) angular velocity.

Translational acceleration of the origin corresponds to the acceleration of the end of the previous segment. However, due to the presence of the imaginary ball joint between the links, only the tangential component (with respect to the current segment) will be passed to the current lumped mass. Therefore, the acceleration due to translation at the ith mass can be obtained by iterative procedure

$$\boldsymbol{a}_{\mathrm{Itb}i} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \left(C_{\mathrm{gb}i} \boldsymbol{a}_{\mathrm{I}(i-1)} \right) \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^{\mathrm{T}} \tag{3.61}$$

where $\boldsymbol{a}_{\mathrm{I}(i-1)}$ is the acceleration of the previous (i–1)th mass in ground axes, and the local DCM $C_{\mathrm{gb}i}$ is obtained through (3.55). The acceleration of the origin of the very first link $\boldsymbol{a}_{\mathrm{I}0}$ is provided by the UAV model.

The total acceleration in body frame at the end point of the ith link for (3.59) is then

$$\boldsymbol{a}_{\mathrm{Ib}i} = \boldsymbol{a}_{\mathrm{Irb}i} + \boldsymbol{a}_{\mathrm{Itb}i} \tag{3.62}$$

Both the inertial force and acceleration can be converted to the universal ground frame:

$$\boldsymbol{a}_{\mathrm{I}i} = C_{\mathrm{gb}i}^{\mathrm{T}} \boldsymbol{a}_{\mathrm{Ib}i}, \quad \boldsymbol{F}_{\mathrm{I}i} = C_{\mathrm{gb}i}^{\mathrm{T}} \boldsymbol{F}_{\mathrm{Ib}i} = -m_i \boldsymbol{a}_{\mathrm{I}i} \tag{3.63}$$

3.5.3.3 Aerodynamic forces

Aerodynamic forces depend on the airspeed $V_{\mathrm{a}i}$ of a particular segment i. This airspeed varies not only between the segments, but also along each segment due to its rotation. For this reason, velocity is calculated at the middle point of each segment and the force is applied at this point as well. After calculation, the aerodynamic forces are grouped at each lumped mass by averaging the neighbouring forces (see (3.70)).

The airspeed of each segment is calculated iteratively in a manner similar to acceleration (3.61), starting from the known velocity of the UAV ($\boldsymbol{V}_0$):

$$\begin{aligned} \boldsymbol{V}_i &= \boldsymbol{V}_{i-1} + C_{\mathrm{gb}i}^{\mathrm{T}} \boldsymbol{\omega}_{\mathrm{b}i} l_i \\ \boldsymbol{V}_{\mathrm{a}i} &= \boldsymbol{V}_{i-1} + \frac{C_{\mathrm{gb}i}^{\mathrm{T}} \boldsymbol{\omega}_{\mathrm{b}i} l_i}{2} - \boldsymbol{W} \end{aligned} \tag{3.64}$$

Here $\boldsymbol{W}$ is the wind vector. In the presence of turbulence, $\boldsymbol{W}$ may be different at each individual segment. However, since the turbulence model (Section 3.2.2) calculates the instantaneous wind vector only at a single point, the variations of wind along the cable have been neglected.

Additionally, for the purposes of the end body drag calculation, its airspeed is obtained:

$$\boldsymbol{V}_{\mathrm{ae}} = \boldsymbol{V}_N - \boldsymbol{W} \tag{3.65}$$

Calculation of aerodynamic forces is performed using Driscoll and Nahon's approach, employed in [213]. For the purposes of drag calculation, airspeed V_{a} is converted to the local body frame and is split into tangential V_{ab}^{x} and normal $\boldsymbol{V}_{\mathrm{ab}}^{\mathrm{n}} = \left(V_{\mathrm{ab}}^{y}, V_{\mathrm{ab}}^{z}\right)$ components:

$$\boldsymbol{V}_{\mathrm{ab}i} = C_{\mathrm{gb}i} \boldsymbol{V}_{\mathrm{a}i} \equiv \begin{bmatrix} V_{\mathrm{ab}i}^{x} \\ V_{\mathrm{ab}i}^{y} \\ V_{\mathrm{ab}i}^{z} \end{bmatrix} \tag{3.66}$$

The aerodynamic force components in the body frame are calculated as follows ([213, pp. 7 and 20]):

$$\boldsymbol{F}_{\mathrm{Db}i} = -\tfrac{1}{2} C_{\mathrm{D}} \rho d l_i \left|\boldsymbol{V}_{\mathrm{ab}i}\right|^2 \begin{bmatrix} f_{\mathrm{t}} \operatorname{sign} V_{\mathrm{ab}i}^{x} \\ f_{\mathrm{n}} V_{\mathrm{ab}i}^{y} \Big/ \left|\boldsymbol{V}_{\mathrm{ab}}^{\mathrm{n}}\right| \\ f_{\mathrm{n}} V_{\mathrm{ab}i}^{z} \Big/ \left|\boldsymbol{V}_{\mathrm{ab}}^{\mathrm{n}}\right| \end{bmatrix} \tag{3.67}$$

where f_{n} and f_{t} are the normal and tangential loading functions of angle of attack $\eta \in [0; \pi/2]$ (see Fig. 3.15), given by

$$\begin{aligned} f_{\mathrm{n}} &= 0.5 - 0.1\cos\eta + 0.1\sin\eta - 0.4\cos 2\eta - 0.011\sin 2\eta \\ f_{\mathrm{t}} &= 0.01\left(2.008 - 0.3858\eta + 1.9159\eta^2 - 4.16147\eta^3 + 3.5064\eta^4 - 1.187299\eta^5\right) \end{aligned} \tag{3.68}$$

d is cable diameter;

ρ is air density;

C_{D} is the drag coefficient, dependent on Reynolds number $Re = \dfrac{\rho d \left|\boldsymbol{V}_{\mathrm{ab}}^{\mathrm{n}}\right|^2}{\mu}$. For the operational conditions ($\rho \approx 1.2$ kg/m^3, $\mu \approx 1.8 \cdot 10^{-5}$ Pa·s, $d = 0.00635$ m), Re lies within the range $(400; 10^5)$ if the normal component of the airspeed is 1 to 15 m/s. In this range, the experimentally determined value of the drag coefficient remains constant and is 1.27 ([213, eq. (1.2)]). For greater airspeed ($Re > 10^5$), C_{D} drops to 0.3. However, taking into account that the airspeed of the UAV during approach is less than 30 m/s, the normal components of the airspeed of the segments of the trailing cable usually do not exceed 15 m/s. For this reason, the drag coefficient is assumed to be constant across the entire operational range: $C_{\mathrm{D}} = 1.2$.

The end body (the hook) drag is calculated as

$$\boldsymbol{F}_{\mathrm{De}} = \frac{1}{2} C_{\mathrm{De}} \rho d_{\mathrm{e}}^2 \left|\boldsymbol{V}_{\mathrm{ae}}\right| \boldsymbol{V}_{\mathrm{ae}} \tag{3.69}$$

where d_{e} is the end body diameter equivalent (see Table 3.10) and $C_{\mathrm{De}} = 1.0$ is the end body drag coefficient.

Having calculated the aerodynamic forces for each segment according to (3.67), the forces can be converted to the universal ground frame and grouped at the respective masses (see also [213], eq. (1.51):

$$\begin{aligned} \boldsymbol{F}_{\mathrm{D}N} &= \tfrac{1}{2} C_{\mathrm{gb}i}^{\mathrm{T}} \boldsymbol{F}_{\mathrm{Db}i} + \boldsymbol{F}_{\mathrm{De}} \\ \boldsymbol{F}_{\mathrm{D}j} &= \frac{1}{2}\left(C_{\mathrm{gb}j}^{\mathrm{T}} \boldsymbol{F}_{\mathrm{Db}j} + C_{\mathrm{gb}(j+1)}^{\mathrm{T}} \boldsymbol{F}_{\mathrm{Db}(j+1)}\right), \quad j = 1, \mathrm{K}\,, N-1 \end{aligned} \tag{3.70}$$

3.5.3.4 Tension forces

Tension is passed to the current mass i from the neighbouring $(i+1)$th segment, which is, in turn, influenced by the following segment. Tension is internal to the cable, it represents

the coupling between the segments and does not come from environmental factors. As such, directly passed tension is a replacement of elastic forces which exist in extensible cables.

Therefore, tension can be calculated simply by extracting the longitudinal (tangential) component of the (i+1)th segment's forces and applying it to the current (ith) segment. By doing that, the 'original' tangential force on the (i+1)th segment is cancelled, according to the Third Newton's Law, through internal tension in the segment. The force has an effect on the motion of the ith segment and therefore on the motion of the (i+1)th local body frame and thus on its inertial forces. The remaining normal components of the forces of the (i+1)th segment effect in rotation of the segment in its local frame.

As the cable has a free end, the calculation of tension (as well as of total force balance) is better done in a backward iterative procedure:

$$\begin{aligned} &\boldsymbol{F}_{\mathrm{T}N} = \mathbf{0} \\ &\boldsymbol{F}_{\mathrm{T}j} = C^{\mathrm{T}}_{\mathrm{gb}(j+1)} \left(\begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \left(C_{\mathrm{gb}(j+1)} \boldsymbol{F}_{j+1} \right) \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^{\mathrm{T}} \right), \quad j = N-1, N-2, \mathrm{K}\ , 0 \end{aligned} \tag{3.71}$$

where the total force $\boldsymbol{F}_{j+1}$ is calculated according to (3.57) at each iteration, taking into account previously calculated tension $\boldsymbol{F}_{\mathrm{T}(j+1)}$. At the end, the tension $\boldsymbol{F}_{\mathrm{T}0}$ will represent the force passed to the UAV. It forms the output vector of the cable model at each time step, along with 2N kinematic angles obtained using (3.56).

3.5.4 Static model

The dynamic cable model described above, although being extremely simple, involves integration of motion of N bodies with 6N scalar states (2 for angular velocities and 4 for quaternion components for each segment). This makes the simulation quite time consuming. If the model is used in genetic search (see next chapter), where hundreds of thousands of simulations may be required, the time penalty associated with cable simulation may become prohibitively high.

For this reason, a simple static model has been developed. It is based on the precalculated data obtained from simulation of the dynamic model in the conditions expected during approach. The full dynamic model is used only for testing and verifying of the obtained solutions.

For the estimation if a given recovery attempt is successful, knowing the exact shape of the trailing cable is not necessary. The shape is required to determine the exact point when the line touches the boom with the arresting wire; however, an estimation of the recovery possibility can be done at the point when the UAV passes over the boom if the current *sag* of the line is known (see Fig. 3.16). The sag determines the maximum allowed elevation of the UAV relative to the recovery boom. The estimation is made on the premise that the remaining path

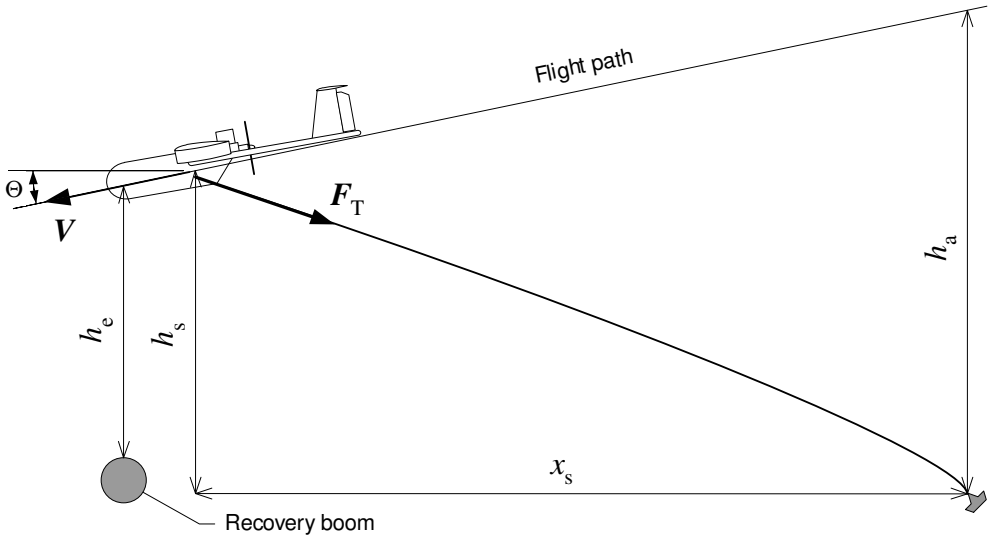


Fig. 3.16 Arresting line sag and tension

(which is shorter than the length of the cable, i.e. less than 10–15 m, which is less than 0.5 seconds of flight) will not change significantly. Even if it does, the line will not reflect these changes immediately due to its own dynamics.

The *effective sag* h_a is measured as the distance from the arresting hook position to the current flight path, which is assumed to be a straight line. The angle between the current flight path (the velocity vector) and the horizontal plane is known as *trajectory angle* or *flight path slope* Θ (negative in Fig. 2.1). In contrast, the real ('physical') sag h_s is measured relative to the attachment point of the line. Obviously, $h_s = h_a$ in level flight. The recovery is considered successful if at the moment when the UAV passes over the recovery boom its elevation h_e is less than the effective sag h_a (but is greater than 0.5 m for safety reasons).

To obtain the effective sag, the following expression can be used:

$$h_a = h_s - x_s \tan\Theta \tag{3.72}$$

where x_s is the *trail* of the line.

In addition to geometry of the line, the tension force $\boldsymbol{F}_T$ should be determined for the given conditions. This force is passed to the UAV during the flight and is taken into account by the UAV model. Of course, this force reflects only static conditions, which is not entirely correct for the whole flight. Nevertheless, it is the best available approximation given that running the dynamic model is not feasible.

With the above assumptions, the shape and thus the sag h_s and trail x_s of a given cable is fully determined by the aerial velocity of the UAV (which is in static conditions the same for all cable segments as well). The aerial velocity depends on both the aircraft's velocity and the wind, which can have a vertical component. As the aerial velocity modulus (the airspeed) is a key parameter for an aircraft, it is more convenient to express the aerial velocity as the

airspeed V_a and the aerial trajectory angle Θ_a, instead of two velocity components. Therefore, the static model should calculate h_s and x_s (as well as the components of $\boldsymbol{F}_T$) as a function of these two parameters, V_a and Θ_a, the latter being measured as the angle between the aerial velocity vector and the horizontal plane. Note that in no wind conditions $\Theta_a = \Theta$.

To obtain the necessary data, the dynamic cable model has been simulated for the airspeeds 20 to 50 m/s (with the grid size 2 m/s) and the angles Θ_a between –30 and 30 degrees through every 5 degrees. During the simulation, enough time has been allocated to stabilise the cable at every data point in order to obtain steady-state data. At every point, the sag, trail and tension have been measured and stored in a look-up table. To obtain these parameters at any intermediate point, the static model simply performs 2D linear interpolation within the table.

The following examples in Fig. 3.18 show the actual simulated shape of a 10 m cable in various conditions. The parameters are according to Table 3.10. The cable has been simulated with 20 links, whose lengths are gradually shortening towards the end of the cable because greater curvature usually takes place closer to the free end.

Fig. 3.19 illustrates the shape of the cable with different masses of the end body attached. It should be noted that the size of the end body does not change. If it did accordingly to the body mass, the difference in line shapes would be slightly less significant due to greater aerodynamic drag of the heavier bodies.

Fig. 3.20 shows physical and effective sag of a 10 m cable with a 0.1 kg end body (Table 3.10) for varying airspeed V_a. The effective sag is calculated for no wind conditions, when $\Theta_a = \Theta$. It can be seen that although real sag can be very small and even negative for high negative trajectory angles (descending path), these angles have rather minor influence on the effective sag unless the airspeed is greater than about 33 m/s. At a higher airspeed and a positive Θ, the angles of attack of the cable segments become small, the cable generates enough aerodynamic lift to support its own weight and the weight of the end body, this causes the cable to bend up, which reduces the effective sag. This tendency is apparent even at a lower airspeed, see Fig. 3.18. However, high airspeeds (>30 m/s) and positive flight slope angles (i.e. ascending trajectory) are not normal for UAV recovery, hence it can be concluded that the effective sag depends primarily on the airspeed (and obviously on the length of the cable).

Fig. 3.21 presents the static tension force of the same cable. It is better illustrated against the path slope Θ_a. Tension comes from both aerodynamic drag and cable weight. It is interesting to note that as the cable weight is 0.223 kgf + 0.1 kgf = 0.323 kgf = 3.17 N, the cable generates enough lift to support much of its weight in a great range of conditions: the

tension is lower than it would be from a suspended cable at zero airspeed. Only at a high airspeed, particularly with moderate positive angles Θ_a, does drag cause higher tension.

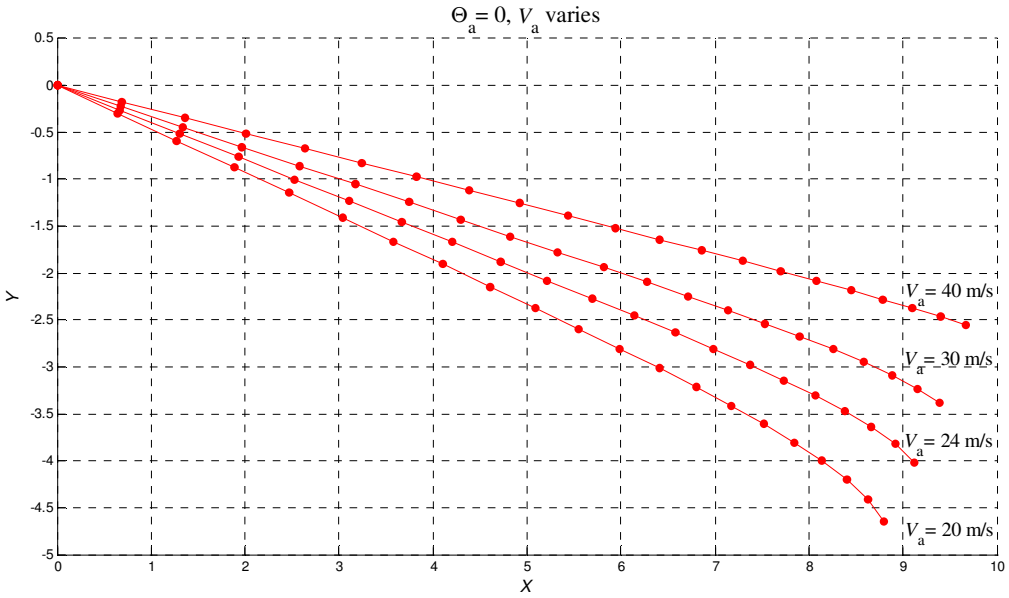


Fig. 3.17 Simulated arresting cable shape for varying airspeed V_a

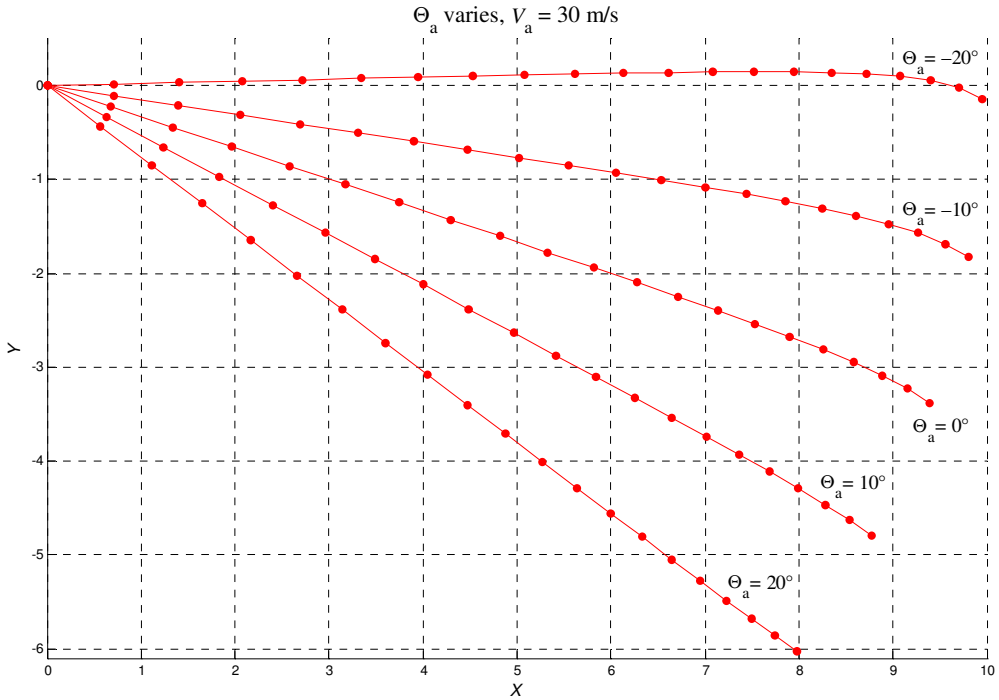


Fig. 3.18 Simulated arresting cable shape for varying aerial trajectory angle Θ_a

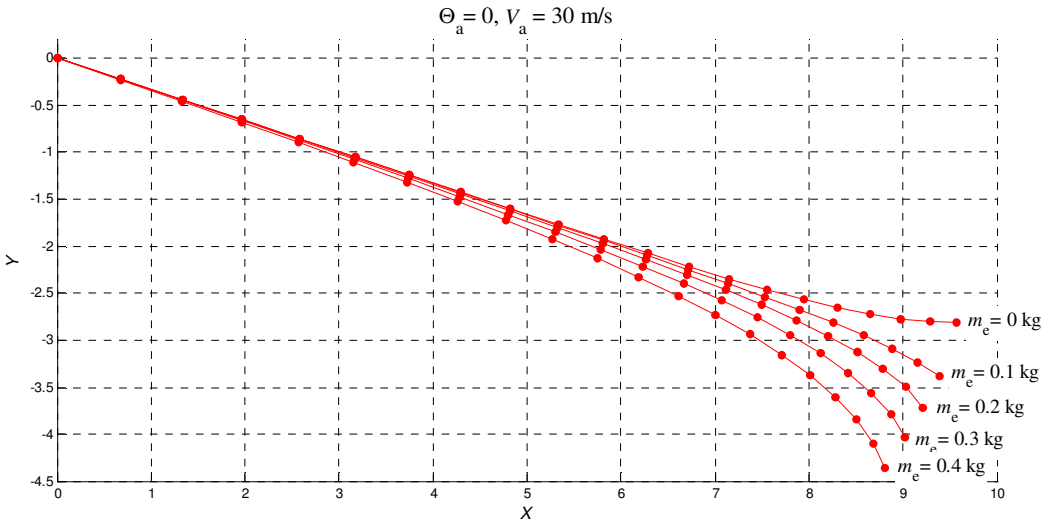


Fig. 3.19 Simulated cable shape for different end body masses

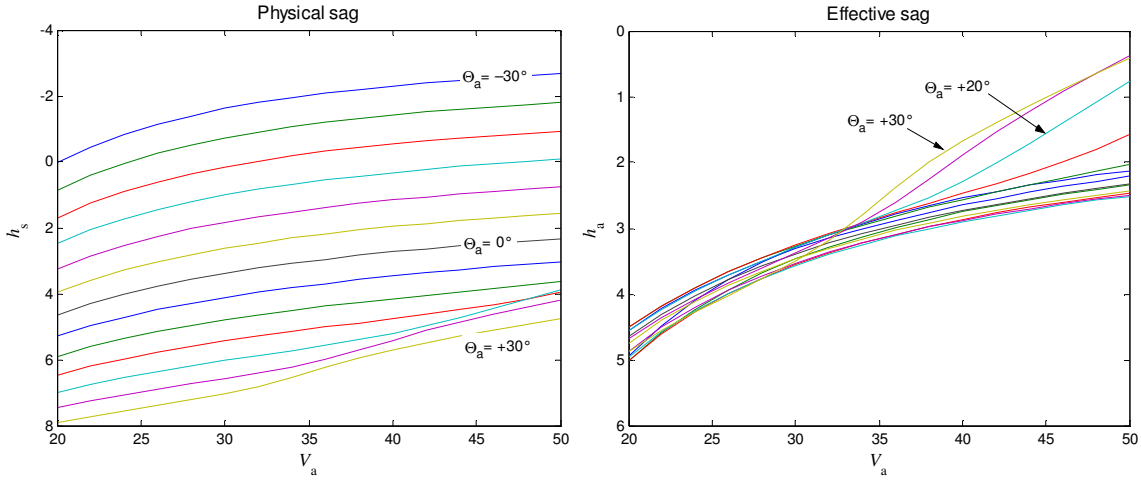


Fig. 3.20 Line physical sag h_s and effective (no wind) sag h_a in metres for different airspeeds and flight path slopes. Y direction is downwards for illustrative purposes

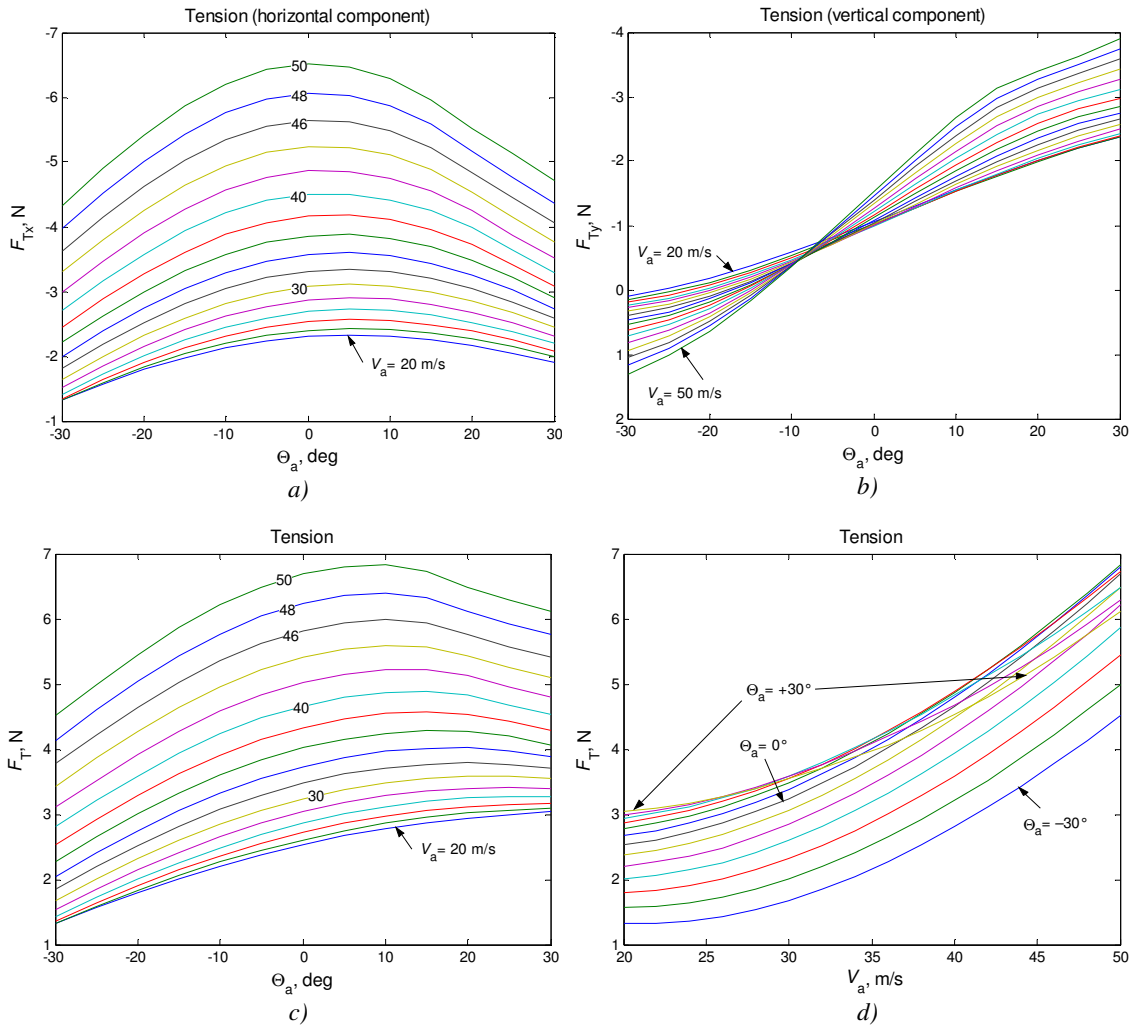


Fig. 3.21 Static tension of the trailing line (length 10 m). a) and b) Horizontal and vertical components; negative sign indicates downward and backward force direction. c) and d) Total tension force for varying flight path slope and airspeed respectively

3.6 Concluding remarks

The models presented here are integrated in a simulation environment, further details of which can be found in Chapter 5. The models have been made as accurate as realistically possible; however, it is accepted that large discrepancies, particularly in the atmospheric model, could be possible. The lack of comprehensive validation data for the airwake model is unfortunate but unavoidable. Nevertheless, it is believed that the testing procedures used in Chapter 6 (Controller Synthesis and Testing) cover a sufficient range of conditions which should hopefully encompass the disturbances found in nature.

Most of the models are built upon the reference data or models which have been already successfully used and validated. In particular, the *Ariel* UAV model has been used to design a launch controller [47]; the model presented in Section 3.1 is rather a minor improvement of that model for better integration into the simulation environment. The models of

gusts and turbulence are the implementations of the standard models, widely used in aviation engineering. The ship airwake model is an extension of the standard model, used to evaluate flying qualities of piloted aircraft. The model of ship motion is a behavioural model, it directly simulates the motion observed on a real frigate ship. Finally, the cable model is a creative extension (and, at the same time, reduction) of the model employed for simulation of underwater towed vehicles.

All models are designed with the implementation side of the problem in mind. The performance characteristics are of the utmost importance for the stochastic controller design method employed in this work. The next chapter outlines the basic concepts of Evolutionary Algorithms, the core of the design methodology.

Chapter 4. Evolutionary Algorithms

Over the hundred years of aviation history, various linear control methods have been successfully used in the aerospace area due to their simplicity and analytical justifiability. Despite their natural limitations, linear control techniques still remain as one of the most accepted design practices. However, growing demands to the performance of aircraft, and on the other hand, a remarkable increase in available computation power over the past years have led to significant growth if the popularity of nonlinear control techniques.

A principal difficulty of many nonlinear control techniques, which potentially could deliver better performance, is the impossibility or extreme difficulty to predict theoretically the behaviour of a system under all possible circumstances. In fact, even the design envelope of the controller often remains largely uncertain. Therefore, it becomes a challenging task to verify and validate the designed controller under all real flight conditions. A practical solution to this problem is extensive testing of the system (or rather of its mathematical model), which is a computationally and time demanding engineering task. The design process itself is often built around the continuous testing of the controller in the loop with a real system or its mathematical model, rather than relies on theoretical analysis only. It becomes clear that there is a need to develop a consistent nonlinear control design methodology that enables to produce a required controller for an arbitrary nonlinear system while assuring its robustness and performance across the whole operational envelope. In addition, a positive engineering experience on application of a nonlinear design methodology is crucially important because such methodology is not as strongly supported by theory as classic linear methods.

The potential applications of possible nonlinear designs are numerous and diverse, and the response of such a system to change of its parameters may be virtually unpredictable. This makes many common search and optimising techniques such as gradient search unusable or unreliable for the design process, as all of these methods lack the ability to comprehend the global design space. To overcome this limitation, stochastic methods are widely employed. They may be very effective for complex multi-dimensional problems, however they cannot *guarantee* to find an optimal solution even under trivial conditions. Nevertheless, the practice shows that the chances of finding a near-optimal solution are quite high.

The Evolutionary Algorithms (EAs) is a group of such stochastic methods which combine such important characteristics as robustness, versatility and simplicity. They are inspired by the success of natural evolution and, indeed, proved the success in many applications, such as neural network optimisation [187], finance and time series analysis [140], aerodynamics and aerodynamic shape optimisation [145, 159], automatic evolution of computer software

[118] and, of course, control [43]. EAs are often considered as an example of 'artificial intelligence' and 'soft computing' approach. This makes sense because EAs appear to have the ability to gain 'experience' during the evolution without (or with little) a priori knowledge about the system and the environment. This ability is intrinsically incorporated into the process of evolution. Moreover, EAs have shown the ability to evolve complex systems from the simplest ones, which makes them not only an optimisation tool, but also a versatile design instrument.

In this work, a set of different EAs is used to develop and optimise the control laws of the UAV recovery controller. To give a clear picture why and how particular methods are used, an introduction to evolutionary algorithms is given in this chapter. The next chapter describes the actual implementation of the EA-driven controller design.

4.1 Introduction to Evolutionary Algorithms

Evolutionary algorithm is an umbrella term used to describe computer-based problem solving systems which employ computational models of evolutionary processes as the key elements in their design and implementation. 'Evolution' is used in its Darwinian sense, the advance through 'survival of the fittest'. As such, all major elements found in natural evolution are present in EAs. They are:

- *Population*, which is a set of *individuals* (or *members*) being evolved;
- *Genome*, which is all the information about an individual encoded in some way;
- *Environment*, which is a set of problem-specific criteria according to which the fitness of each individual is judged;
- *Selection*, which is a process of selecting of the fittest individuals from the current population in the environment;
- *Reproduction*, which is a method of producing the offspring from the selected individuals;
- *Genetic operators* (also referred to as *search operators*), such as *mutation* and *recombination (crossover)*, which provide and control variability of the population.

The process of evolution takes a significant number of steps, or *generations*, until a desired level of fitness is reached.

It should be noted that generally not the simulation of the natural world evolution but the outcome of the whole process (the evolved individual or population) is the goal of EAs. Although many of the principles are inspired directly from nature, they are not necessarily followed in the implementation. For example, the amazingly complex process of genome reparation found in cells nuclei is not usually necessary as the optimal level of mutation can be controlled directly in the algorithm. Another example—the lifespan of all individuals can

be restricted to one generation, or conversely, the fittest (*elite*) individual(s) can be propagated to the further generations intact as long as they keep the best fitness in the population.

The 'outcome' as given above should not be interpreted as if some particular species are expected to evolve. The evolution is not a purposive or directed process. It is expected that highly fit individuals will arise, however the concrete form of these individuals may be very different and even surprising in many real engineering tasks. Indeed, several applications of EAs have produced a patentable invention [120, 121, 123]. None of the size, shape, complexity and other aspects of the solution are required to be specified in advance, and this is in fact one of the great advantages of the evolutionary approach. Moreover, compared to conventional search and optimisation techniques, EAs are much less sensitive to initial conditions, because not a single guess value but the entire population of candidate solutions is processed in parallel, giving the best solution much less chances to get stuck at local optima. In fact, the problem of initial guess value rarely exists in EA applications, and the initial population is sampled at random.

The idea of mimicking one or more attributes of life in computer science has a long history. It can be traced back to the 1940's when John von Neumann began exploring the idea of a universal constructor—a program that has the ability to process data and automatically replicate itself. Later in the 50's and 60's the idea of using a population of solutions to solve both theoretical and practical problems has been considered several times, firstly by biologists to perform simulations of genetic systems [25, 75]. A demonstrative example of population evolution, extremely popular amongst mathematicians (at least in 1970s and 80s), is the John Conway's *Game of Life* [78] (1970)—a cell game on the plane in which a predefined population evolves according to simple selection rules. The first dissertation to apply *genetic algorithms* to a pure problem of mathematical optimisation was Hollstien's work [98] (1971). However, it was not until 1975, when John Holland in his pioneering book [97] established a general framework for application of evolutionary approach to artificial systems, that practical EAs gained wide popularity. Until present time this work remains as the foundation of genetic algorithms and EAs in general.

Now let us consider the very basics of EAs. A typical pseudo code of an EA is as follows:

```
1. Create a {usually random} population of individuals;
2. Evaluate fitness of each individual of the population;
3. until not done {certain fitness, number of generations etc.}, do
4.     Select the fittest individuals as 'parents' for new generation;
5.     Recombine the 'genes' of the selected parents;
6.     Mutate the mated population;
7.     Evaluate fitness of the new population;
8. end loop.
```

It may be surprising how such a simple algorithm can produce a practical solution in many different applications, but it does. Although there is some theoretical basis under EAs, in particular the schema theory (Section 4.2.9) first given in the aforementioned Holland's book [97] and this basis is constantly growing, maybe the most important support for EAs' applicability is their practical success. As with any stochastic method, nothing is guaranteed; however, many thousands of successful implementations is good empirical evidence that EA is a robust and reliable method of solving difficult practical problems.

A major part of applications of EAs is focused on function optimisation, that is, finding a minimum (or maximum) value of a given function. EAs showed outstanding results for many of the difficult, ridged or ill-defined functions, and even for dynamic functions which change with time. Moreover, EAs naturally allow a multi-criteria search for a Pareto-optimal set of solutions. However, EAs are capable of much more than function optimisation or estimation of a series of unknown parameters within a given model of a physical system, and some authors are keen to remind us of that ([53] for example). EAs can *create* such complex structures as computer programs, architectural designs and neural networks. This ability is used in this work for evolving control laws for the flight controller.

Almost every operation of an EA is stochastic. Some operations can be either stochastic or deterministic; for example, *selection* may simply take the best half of the current population for reproduction, or select the individuals at random with some bias to the fittest members. Although the latter variant can sometimes select the individuals with very poorly fitness and thus may even lead to temporary deterioration of the population's fitness, it often gives better overall results. In addition, even fitness evaluation, which is deemed to be an 'objective' process, can suffer from noise and uncertainty in a dynamic environment. Therefore, EAs are inherently probabilistic and may give sufficiently different results every time they run.

The probabilistic nature is not the only unusual feature of EAs (for an engineering mind) which should be clearly realised and *accepted* to avoid misunderstanding and misinter-

pretation of the results. Many of these features are explained by John Koza in [118] and they will be briefly outlined here due to their importance.

It is noted that a common (if not the only accepted) engineering and scientific approach is to seek the solutions which are *correct, consistent, justifiable, certain, orderly, parsimonious* and *decisive*. However, not all (if any) of these principles should be used in solving every problem. Only occasionally evolutionary approach can give a solution which complies with some of these principles. Some of them may not be applicable to simple function optimisations; however, for complex design problems an understanding of all of them is important.

Correctness. By correctness it is usually assumed that the solution, though being affected by errors, inaccuracies, simplifications, etc., is centred on *the* correct solution, or at least the degree of incorrectness (bias) is known. For example,

$$x = \ln a + a \cdot 10^{-10}$$

is not considered as the correct solution to the equation $e^x = a$, even though the additional incorrect term $a \cdot 10^{-10}$ introduces an error that is considerably smaller than everyday precision requirements. However, EAs works only with admittedly incorrect solutions, and more often than not the incorrectness is not as clearly visible as in the above example and cannot be simply eliminated.

Consistency. EAs actively encourage and use a diverse set of clearly inconsistent and contradictory approaches in attempting to solve a problem. For example, the landing flight control generated by an EA can produce totally different trajectories for faintly different conditions (and sometimes even for similar conditions), as long as they satisfy the predefined goal. In fact, great diversity of the population, however inconsistent it may be, helps EA to arrive at the solution faster.

Justifiability. The solutions generated, in essence, at random, have no logical justification. There is no sound logical sequence of reasoning which leads to any particular solution, like there is no deduction which lead to inclusion of the extra term $a \cdot 10^{-10}$ in the example above.

Certainty. With great advances in stochastic methods (from Monte Carlo to fractal and dynamic chaos theory), the potential and properties of probabilistic approach is rather better understood. However, even for these entirely random methods people tend to seek a deterministic explanation. It should be made clear that, as already noted, every step of EAs is probabilistic, thus anything can happen and nothing is guaranteed.

Orderliness. Most of the algorithms used for problem solving are not only deterministic but also have inherent order: they proceed in a controlled and synchronised way. In contrast, EAs consist of a number of uncoordinated, independent, distributed processes operating

asynchronously without central supervision. Untidiness and disorderliness are fundamental features of biological processes in nature as well as of EAs.

Parsimony. Scientists and engineers strongly prefer the simplest possible solutions and explanations, which is aptly expressed in the Occam's Razor principle. Conversely, fitness, not parsimony, is the dominant factor of evolution, whether natural or algorithmic. Once a good solution is found, it is usually enshrined and propagated. Thus, many examples of seemingly indirect and complex (but successful) ways of solving problems can be observed in nature as well as in the solutions generated by EAs. Parsimony seems to play a role only when it interferes with fitness (for example, when the price is paid for excessively complex solutions).

Decisiveness. Decisive algorithms are those which have a well-defined termination point at which they converge to a result which is a solution to the problem at hand. However, in EAs not always such a termination point can be defined. Biological evolution has no termination point at all. Even if the process is interrupted, it offers numerous inconsistent and contradictory answers (although the external observer is free to focus on the best current answer).

All the considerations stated above will be clearly visible later on when the controller is designed.

4.2 Evolutionary Algorithms inside

There are several branches of EAs which focus on different aspects of evolution and have slightly different approaches to ongoing parameters control and genome representation. Many of the preferences are due to historical reasons and years of independent development. They do not contradict but rather supplement each other, although theoretical results are not always compatible. In this work, a mix of different evolutionary methods is used, combining their advantages. These methods are described in the following sections. They have the common names: *Genetic Algorithms (GA), Evolutionary Programming (EP), Evolutionary Strategies (ES)* and *Genetic Programming (GP)*. There are several evolutionary machine learning methods as well, such as *Classifier Systems*, which deal with behaviour of finite state automata. They have little relevance to this work and are not presented here.

As noted above, all the evolutionary methods share many properties and methodological approaches. Therefore, a description of all key elements of EAs is first given, with only brief references to particular evolutionary methods when necessary. Then, in the section 4.3 and further, the methods themselves are outlined.

4.2.1 Fitness evaluation

Fitness, or *objective value*, of a population member is a degree of 'goodness' of that member in the problem space. As such, fitness evaluation is highly problem dependent. It is

implemented in a function called *fitness function*, or more traditionally for optimisation methods, *objective function*. It accepts the phenotypic (decoded) value of the member's genome (see 4.2.2), interprets it and calculates the fitness according to the problem at hand. The plot of fitness function in the problem space is known as *fitness landscape*.

The word 'fitness' implies that greater values ('higher fitness') represent better solutions. However, mathematically this does not need to be so, and by optimisation both maximisation and minimisation are understood. (In the case of minimisation, objective function is sometimes referred to as *cost function* or *penalty function*). It is up to the selection procedure (Section 4.2.3) how to handle fitness. Some selection schemes, e.g. *fitness proportional selection* (4.2.3.2), offer only maximisation of the fitness. In such cases, a minimising fitness function can be easily turned into the maximising one by subtracting the original fitness value from an arbitrary value a: $f' = a - f$. However, some selection methods may be sensitive to the value a, and it should be chosen, generally, as a minimum value to avoid negative fitness (to which these methods are also usually sensitive).

For a wide area of optimisation applications, the fitness function is the problem to be solved by itself. For example, if the problem is to minimise the function $f(x) = x^2$, this function itself will serve as the fitness function. However, in many tasks only a qualitative description of what is 'good' and 'bad' is given. In these cases, a quantitative expression of that description must be designed. This may be a challenge when little is known about the ways of achieving the optimal solution. Although EAs are proved themselves as robust methods, their performance depends on the shape of the fitness landscape. If possible, fitness function should be designed so that it exhibits a gradual increase towards the maximum value (or decrease towards the minimum). In the case of GAs (Section 4.3), this allows the algorithm to make use of highly fit *building blocks* (see 4.2.9), and in the case of ESs (Section 4.4)—to develop an effective search strategy quickly.

In solving real world problems, the fitness function may be affected by noise that comes from disturbances of a different nature. Moreover, it may be unsteady, that is, changing over time. This means that the fitness function may return different values when evaluated several times with the same input. Generally, EAs can cope very well with such types of problem, although their performance may be affected. However, it is better if the algorithms are tuned deliberately for noisy and/or unsteady fitness function (see e.g. [149]). The first and obvious observation is that the fitness of a member should always be re-evaluated in the next generation even if the member is propagated there from the previous generation intact. Other adjustments usually include allowing higher variability and diversity of the population through higher mutation and recombination rates and larger population sizes.

It is accepted that the performance of an optimisation algorithm (and in calculus in general) is measured in terms of objective function evaluations, because in most practical tasks objective evaluation takes considerably more computational resources than the algorithm framework itself. For example, in optimisation of the aerodynamic shape of a body, each fitness evaluation may involve a computer fluid flow simulation which can last for hours. Nevertheless, even for simple mathematical objective functions EAs are always computationally intensive (which may be considered as the price for robustness), and the performance issue of the fitness function evaluation should receive due attention. For further discussion on performance measurement, see Section 4.2.7.

The computation performance may be greatly improved by parallelisation of the fitness evaluation. EAs process multiple solutions in parallel, therefore they are extremely easily adopted to parallel computing.

4.2.1.1 Multi-objective problems

Another useful consequence of maintaining a population of solutions is the natural ability of EAs to handle multi-objective problems. A common approach to reduce a multi-objective tasks to a single-objective one, by summing up all the criteria with appropriate weighting coefficients, is not always possible. Unlike single point optimisation techniques, EAs can evolve a set of *Pareto optimal* solutions simultaneously. A Pareto optimal solution is the solution that cannot be improved by any criterion without impairing it by at least one other criterion. This is best illustrated with a simple example.

Suppose the problem is to minimise both x and y, constrained by $(x-6)^2 + (y-6)^2 < 25$ (the area inside the circle of radius 5 with the centre at (6; 6), see Fig. 4.1). Consider the solutions, denoted by o's on the figure. No one of them may be improved within the given constraints by either x or y without adverse increase of another parameter. These solutions represent the *Pareto optimal set*, or *Pareto front*.

From the standpoint of the fitness function, implementing the multi-criteria optimisation is straightforward: the fitness function should return a vector of criteria values instead of a scalar. A slightly trickier

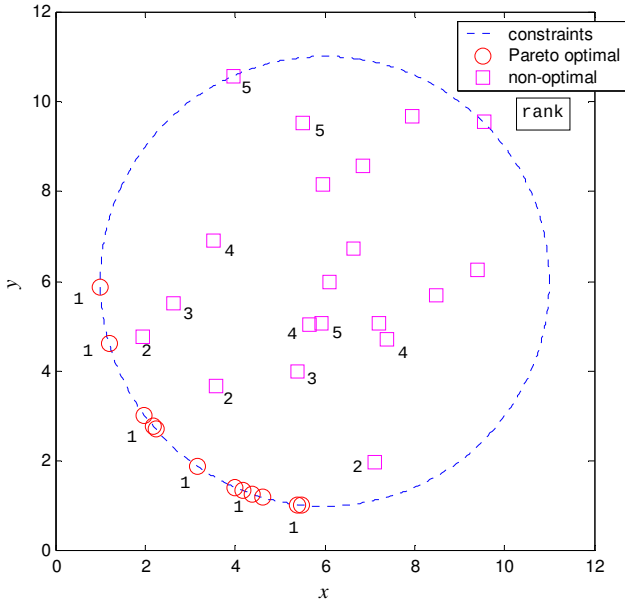


Fig. 4.1 Pareto optimality

task is to adapt the selection procedure. Although the following sections outline selection mechanisms in detail, the approach to handle multi-criteria selection is given here, because it is applied 'on top' of the common selection methods and can be used with nearly any of them.

The multi-criteria selection utilises the concept of solution dominance. A solution is said to be dominated if there exists another solution in the population that is unconditionally (i.e. by all parameters) better. Otherwise, the solution is non-dominated. Clearly, the set of the non-dominated solutions in a population represents currently best solutions. As not one of these solutions may be preferred by any other one, they should be selected either with equal probability or all at once.

However, identifying the current winners is not enough for a steady advance. The distinction between 'not very good' and 'bad' solutions should be made as well. This is usually achieved through a *non-dominated sorting procedure*. The procedure is somewhat similar to the principle of single-criterion selection (described later in 4.2.3.3), but the population is ranked on the basis of non-domination. All non-dominated members receive the highest rank. Then they are removed from the population and the next set of non-dominated solutions is identified. This set receives a lower rank (see Fig. 4.1). The process continues until the entire population is ranked. After ranking, common selection techniques are applied such as niching and proportionate selection. Niching and speciation may be especially useful in the case of multi-criteria optimisation, because they help to prevent excessive competition within the Pareto optimal front, which may arise due to significant distance between its members. See also [183, 232] and [80] for detail.

4.2.2 Genome representation

In EA theory, much as well as in natural genetics, *genome* is the entire set of specifically encoded information that fully defines a particular individual. The term *genotype* is sometimes used to describe the pure information content of the genome. The actual shape of the individual, or, broadly speaking, the representation of the genome in the original problem space, is called *phenotype*.

For some pure mathematical optimisation problems, the transformation from genome to phenotype and back may be direct—that is, the genome may be the same as phenotype (for example, a real number). However, in most engineering applications a more sophisticated mapping is required, because many parameters of a different nature are optimised simultaneously. Although some theoretical analysis of choosing the best (or at least highly effective) genome representation is available for some types of EAs [80], in practice building a suitable representation is, in a sense, an art. A general guideline to building a genome is that the genome should be as compact as possible yet allow a natural expression of the problem and ap-

plication of genetic operators (Sections 4.2.4, 4.2.5). A deeper insight in how the genome is processed and how the encoding may influence the performance of the algorithm is given in Section 4.2.9.

This section focuses only on numeric genome representation. However, EAs are not limited to numeric optimisations, and for more 'creative' design tasks a more sophisticated, possibly hierarchical, genome encoding is often required. In fact, virtually any imaginable data structure may be used as a genome. This type of problem is addressed in Section 4.5 *Genetic Programming*.

One of the most widely used genome encodings is a string-like representation. It is particularly popular in Genetic Algorithms. In GAs, the genome is usually represented as an alphabet-based string, called *chromosome*. Mimicking of biological genome representation is obvious and is in fact inspired by nature.

In nature, chromosomes are character strings in base-4 alphabet. The four nucleotide bases which encode the information along the length of the DNA molecule are adenine (A), guanine (G), cytosine (C) and thymine (T). The sequence of these bases constitutes the chromosome string or the genome of a biological individual (except for some viruses that store heritable information in a simpler RNA molecule in which thymine is replaced with uracil). The encoding is not a simple 'bit mapping' in which each single character in the string has a definite meaning. Instead, the nucleotide bases form three-letter 'words' called *codons*, which determine the position of a specific amino acid in a protein molecule during protein synthesis. Moreover, a large sequence of codons occupying a fixed position on a chromosome construct 'sentences,' better known as *genes*, which actually determine characteristics of an individual (or regulate the operation of other genes).

A computer implemented encoding may be much simpler than the biological one or may be just as sophisticated. Binary digital computers, dominating the world, favour using binary encoded chromosomes with base-2 alphabet. These computers are particularly fast at manipulating with binary strings, which probably was an especially important consideration in the 70's when GAs were quickly developing and computer resources were too limited. However, many other alphabets, particularly problem-oriented, have been successfully used.

A general framework of a typical GA is exactly as shown in the EA pseudo code above (see Section 4.1). A commonly used genome representation in GAs is a fixed length binary string with real numbers mapped into integers. There are several reasons for that apart from simplicity of encoding and storage, and most of them are linked with convenience of applying genetic operators, recombination (crossover) and mutation, which are described in the following sections.

As an example, let us consider a so called 'sphere problem', which in its simplest form and in two dimensional case is expressed as

$$R = x^2 + y^2, \quad R \to \min \tag{4.1}$$

and has an obvious answer $x = y = 0$. For genetic optimisation, the two unknowns x and y can be encoded as binary numbers of length n, and the genome can be formed by concatenating these numbers into a one bit string of the length $L = 2n$.

The choice of any particular n is determined by the trade between the required precision and available computation resources. The search space for binary encoding has the size of 2^L, and thus each additional bit in the chromosome doubles the search space, which greatly affects performance. On the other hand, a length n binary number has 2^n discrete states, giving accuracy of $1/2^n$ for linear mapping. Linear mapping is expressed as follows:

$$x = \frac{x_{max} - x_{min}}{2^n - 1} z + x_{min} \tag{4.2}$$

and the reverse

$$z = \mathrm{int}\left(\left(2^n - 1\right) \frac{x - x_{min}}{x_{max} - x_{min}} \right) \tag{4.3}$$

where x is the problem specified variable, z is its integer representation and $[x_{min}; x_{max}]$ is the specified range. 'int' denotes the integer part of the result.

For example, if 8 bit numbers are used and the problem determines the range of [–1.0; 1.0], this range will be mapped into integers [00000000; 11111111] ([0; 255] decimal)[1]. As 255 corresponds to 1.0, the next possible integer, 254, will correspond to 0.99215686, and there is no means of specifying any number between 0.99215686 and 1.0.

This is a common drawback of digital (discrete) machines, and care should be taken when choosing appropriate representation. The required accuracy is often set relative to the value, not the range. In this case, linear mapping may give too low accuracy near zero values and too high accuracy towards the end of the range. This was the reason for inventing the so called floating point number representation, which encodes the number in two parts: mantissa, which has a fixed range, and an integer exponent, which contain the order of the number—the same principle as used for scientific decimal representation like '$0.85 \cdot 10^4$', where '0.85' is mantissa and '4' is exponent with base 10 (which may be implicit in some notations, like '0.85E+4'). For binary floating point encoding, naturally, base 2 for exponent is usually im-

[1] For zero-mean range [–1.0; 1.0], a more convenient signed binary representation [10000000; 01111111] ([–128; 127] decimal) could be used, however this would involve explanation regarding inverse encoding of negative numbers, commonly used in digital computers to provide smooth transition through zero. In most practical GA implementations where linear mapping is applied, unsigned integers are used.

plied, and about three quarters of the allocated bits is used for mantissa. This representation is implemented in nearly all software and many hardware platforms which work with real numbers.

However, dealing with binary encoded floating point numbers in a way common for GAs has several disadvantages due to significantly different roles of the bits. Modifying the exponent by splitting the chromosome or by swapping bits will mostly have a 'lethal' effect, producing values far out of the domain of the objective function. Therefore, if the floating point precision is required, either nonlinear mapping into integers or specific algorithms tailored for real numbers should be used. Some of these algorithms are described in the sections dedicated to *Evolutionary Strategies*.

Let us return back to the Sphere problem. Suppose that the range (–1;1) and 8-bit fixed point precision (0.4%) is specified. The chromosome can then be formed as a 16-bit string:

xxxxxxxxyyyyyyyy

An individual 0100101010110011, for example, can be decoded in two steps:

- splitting the chromosome into two 8 bit parts, the left one (01001010, 74 decimal) for x and the right one (10110011, 179 decimal) for y.
- applying linear scaling (3.1), yielding $x = -0.4196$ and $y = 0.4039$.

Upon obtaining x and y, fitness can be calculated (4.1): $R = (-0.4196)^2 + 0.4039^2 = 0.3392$.

Binary representation, although fast at computation and convenient, is not the only option even for simple problems. Alternatively, fixed point decimal representation may be used, for example. For the range (–1.0; 1.0) and accuracy 1%, only two fractional digits is enough. However, handling negative values should be provided in some form. For instance, a field for minus and plus sign can be incorporated directly into the chromosome. This makes the chromosome very demonstrative at a glance (e.g. '–36+68', which could mean $x = -0.36$, $y = 0.68$) and straight-forward for encoding/decoding, but requires non-uniform, more sophisticated application of genetic operators as the chromosome alphabet is different for different fields.

Many engineering tasks require not the optimisation of real value parameters but optimal selection of specifically listed items. For example, water pipes have standard sizes, and optimal planning of the pipelines may involve selection of particular sizes for each section. This can be easily encoded by enumerating all available sizes and assigning an integer code for each size. Furthermore, other (independent) parameters may be added, such as, for example, material. Suppose there are 16 pipe sizes available in steel and plastic (which may have an impact on the cost and durability). Then one bit in the chromosome can be allocated for

material (so called ‘flag bit’, e.g. 0 means ‘metal’ and 1 means ‘plastic) and four bits for the code of the size (this gives exactly $2^4 = 16$ size combinations). Therefore, each section occupies 5 bits on the chromosome, while the total length of the chromosome will depend on the size of the network being optimised.

There may arise a question: what to do if we have, say, 14 standard sizes? It is impossible to allocate a fraction of a bit (although a bit can be *shared* between two parameters in some cases). Therefore, four bits are required for any number of items between $2^3 + 1 = 9$ and $2^4 = 16$. This means that some of possible binary combinations will have no assigned meaning. There are three general solutions for this problem.

First, the ‘spare’ codes may be assigned with the same meaning as some of the existing codes, creating redundant codes. This is the way exploited by nature: the $4^3 = 64$ possible codons in a human DNA chain encode only 20 amino acids (plus a termination command). This results in up to six different codes for one amino acid. Considering that mutation and other genetic operations are uniform, that is, evenly applied along the chromosome, such redundant encoding obviously results in a bias towards the values represented by multiple codes. This may be both beneficial and unfavourable, depending on the actual problem. In nature, such bias is observable: the frequency of the codons and the frequency of their amino acids is correlated (with only one exception). For the pipeline problem with 14 sizes, if there is a reason to believe that, on average, two of the sizes are more suitable for the network than others, these sizes may be assigned with additional codes. It should be mentioned that there are some more complicated consequences of redundant encoding which are out of the scope of this text.

Another way is preventing the genetic operations (including initial random sampling) from creating ‘prohibited’ codes. Though relatively simple, this approach complicates even simpler mutation and crossover operations and often involves partial decoding of the chromosome. More importantly, it rejects the schemata (see Section 4.2.9 below) which may be of good value. That is, if for example, the codes 1110 and 1111 (14 and 15 decimal) are always rejected, this apart from not processing the schema 111* reduces the number of processed schemata 1*** and 11**, which may contain the optimal value. This may affect the overall performance (especially if crossover is extensively used).

The third and commonly used way is a general one dealing with domain constrains. This method allows emerging the prohibited (out-of-scope) values, but assigns them very low fitness during fitness evaluation. However, even such ‘penalty’ should not be, usually, so prohibitively high that the affected individuals are unavoidably purged from the population in the current generation. Otherwise the same problem as in the previous case may arise. Generally,

even poorly performing individuals should have some chance to participate in producing the offspring. This issue is addressed in greater detail in Section 4.2.8.

Another type of genome encoding which should be mentioned is the permutation encoding. It is used mostly in ordering problems, such as the classic Travelling Salesman Problem, where the optimal order of the given tokens is sought after. In this case, a chromosome can represent one of the possible permutations of the tokens (or rather their codes), for example, '0123456789' or '2687493105' for ten items. When this type of encoding is used, special care should be taken when implementing genetic operators, because 'traditional' crossover and mutation will produce mostly incorrect solutions (with some items included twice and some items missing).

Finally, a vector of floating point real values can be used as a chromosome to represent the problem that deals with real values. However, domain constraints handling should be implemented in most cases, as the floating point numbers, unlike integers, have virtually no highest and lowest values (in a practical sense). Moreover, special recombination and mutation operations should be used in this case. As real-valued encoding is not alphabet-based, these operations work on essentially different principles than 'classic' genetic operators, and their properties should be carefully examined before including in the algorithm. Some of the real-valued recombination and mutation operators are described below in the appropriate sections.

There are many other methods of genome representation. Performance, simplicity, convenience and other factors should be taken into account when choosing one or another method.

4.2.3 Selection

Selection is the key element of all evolutionary algorithms. During selection, a generally fittest part of the population is chosen and this part (referred as *mating pool*) is then used to produce the offspring.

A few words should be said here about *reproduction*. In EAs, reproduction is a general term that describes the whole process of producing the offspring after selection (i.e. from the mating pool). It may range from simple copying of the selected members into the next generation population to a sophisticated sexual mating of the selected individuals that produce 'children'. If recombination is used in the EA, it usually forms a part of the reproduction process. Therefore, some aspects of reproduction are outlined in this section as well, although the actual implementation of some of the steps is described later in the following sections.

The requirements for the *selection* process are somewhat controversial. On the one hand, selection should choose 'the best of the best' to increase convergence speed. On the

other hand, there should be some level of diversity in the population in order to allow the population to develop and to avoid premature convergence to a local optimum. This means that even not very well performing individuals should be included in the mating pool.

This question is known as the conflict between *exploitation* and *exploration*. Indeed, fast convergence is achieved through massive exploitation of the good-so-far genes, concentrating the search around an already discovered optimum. However, intensive ('greedy') exploitation of the currently best individuals comes at the cost of neglecting the exploration of the search space. Moreover, this quickly leads the population to domination by one, or at most a few, 'super-individuals'. With very little genetic diversity in such a population, new areas in the search space become unreachable and the process stagnates. Although exploration takes valuable computation resources and may give negative results (in terms of locating other optima), it is the only way of gaining some confidence that the global optimum is found. For further discussion about this keystone question the reader is referred to [80] or Holland's [97].

It should be noted that the optimal balance between exploitation and exploration is problem dependent—and not only in terms of the structure of the search space. For example, real-time systems may want a quick convergence to an acceptable sub-optimal solution, thus employing strong exploitation; while engineering design which uses EAs as a tool is often interested in locating various solutions across the search space, or may want to locate exactly the global optimum. For the latter tasks, greater exploration and thus slower convergence is preferred. Moreover, for better overall performance of the algorithm, different convergence rates may be induced at the initial and final stages of search, allowing greater exploration in the beginning and thorough refining near the end. Nevertheless, GAs are able to identify optimal or near-optimal solutions under a wide range of selection pressures, demonstrating remarkable intrinsic robustness [83] (it will be shown later that shifting towards exploitation over time is an inherent feature of most GA implementations).

Balance between exploitation and exploration can be controlled in different ways. For example, intuitively, stronger mutation favours greater exploration. However, it is selection that controls the balance directly. This is done by managing the 'strength' of selection. Very strong selection realises exploitation strategy and thus fast convergence, while weak selection allows better exploration.

A general characteristic that describes the balance between 'perfectionism' and 'carelessness' of selection is known as *selection pressure*. This term is well known in biology: the greater competition between the species for the limited resources, the greater the selection pressure is. In other words, selection pressure is the degree to which the better individuals are favoured. The higher the selection pressure, the more the better individuals are favoured. Selection pressure control in a GA can be implemented in different ways; a very demonstrative

(but not necessarily comprehensive) parameter is the size of the mating pool (i.e. the number of the selected individuals) relative to the size of the population. As a rule, the smaller the mating pool, the higher the selection pressure. If for example, only one best member is selected, this means extreme selection pressure, while if the mating pool is exactly the size of the population, it may mean no selection at all.

A quantitative estimation of the selection pressure may be given by the *take-over time* Γ [82]. With no mutation and recombination, this is essentially the number of generations taken for the best member in the initial generation to completely dominate the population. The value Γ depends not only on the selection scheme, but also on the fitness function. For example, for optimisation of exponential functions with fitness proportional selection, Γ has the general order $N \ln N$, where N is the population size [44, 82].

There is a number of various selection methods commonly used in EAs. The efficiency of one or another method largely depends on population properties and characteristics of the whole algorithm. In practice, the choice of selection strategy is based mostly on empirical data, following an established approach in a particular area. However, a theoretical comparison of the selection schemes may be found in [82].

Before a brief description of the selection methods, some common properties of all methods should be discussed.

First, all selection methods can be divided into two groups: *stochastic* and *deterministic*. Deterministic methods use the fitness value of a member directly for selection. For example, the best half of the population may be selected or all individuals with the fitness better than a given value may be included in the mating pool. In contrast, stochastic selection methods use fitness only as a guide, giving the members with better fitness more chances to be selected. Therefore, even the best individual is not guaranteed to reproduce when stochastic selection is employed. As a rule, stochastic methods allow greater exploration (if other settings are equal), because even 'bad' individuals participate in production of the offspring and the best individuals are sometimes disregarded. However, deterministic methods can be tuned to allow greater exploration. For example, every second member in a sorted by fitness list can be taken for reproduction instead of the best half of the population.

A general observation (applicable not only to selection) is that stochastic methods work better for large populations, while small populations prefer a deterministic approach (see [27] for example). This is due to *stochastic sampling errors* of the stochastic methods [44], which may be fatal for small populations. These errors happen when the selection significantly diverts from its 'average behaviour', flooding the mating pool with the worst members. There are always the chances that 'bad' members receive unusually high number of hits (and

'good' members lose), and for small populations even a single sampling error will affect the whole evolution.

One of the simple ways to reduce the possible impact of stochastic sampling errors is to guarantee that the best, or *elite*, member(s) is always selected for reproduction. This approach is known as *Elitism*. In effect, elitism is the introduction of a portion of deterministic selection into a stochastic selection procedure. However, elitism itself may be applied stochastically, that is, not in every generation but with some (usually high) probability.

In most cases, elitism assumes that the elite member is not only selected, but also propagated directly to the new population without being disrupted by recombination or mutation. This approach ensures that the best-so-far achievement is preserved and the evolution does not deteriorate, even temporarily. In the more general *elitist strategy*, a monotonic course of evolution is ensured by selection not only from the parent population, but also from its offspring, thus selecting from the union of parents and offspring.

Elitism and elitist strategies imply that a good member can theoretically have an unlimited lifespan. This may seem beneficial (and in many cases it is), however this is not always so, especially for some types of ridge functions and noisy and/or dynamic environments [164]. Here is a good place to point out again that EAs often produce counterintuitive results, therefore all intuitive 'improvements' to the 'messy' evolutionary algorithms should be made with due caution and analysis. In particular, temporary deterioration of the best fitness may be useful to help to leave the region of attraction of a local optimum and reach a better optimum.

Another feature which may be applied to any selection scheme is *population overlapping*. If a part (or the whole) of the parent population is included in the new population, then the populations are said to be overlapped. The fraction of the old population which is replaced with the fresh members is called a *generation gap* [52]. In the extreme form, only one new member is added to the parent population (and one, usually weakest, member is removed). This is how most of the *continuous selection schemes* (e.g. [182]) work. The fitness of the overlapping part usually does not have to be recalculated; however, this is not by itself an advantage as accordingly less new members are tried. Most of the practical GAs use non-overlapping populations, where new population fully replaces the old one. Nothing particularly advantageous is found in overlapping schemes, although they may be useful for some problems, in particular for steady and noiseless environments [82].

It should be also noted that some reproduction schemes allow *multiple selection* of one member, while others do not. The former case (also referred to as *replacement*) means that the selected member is returned back to the mating pool and thus may be selected again in the same generation. This feature is often used in stochastic selection methods such as fitness proportional selection and tournament selection, although both the ways can be applied to any

method, including the deterministic ones. However, multiple selection does not necessarily mean that the new population will be crowded with multiple instances of the fittest members. If any of the genetic operators is used, all (or at least the majority) of the new individuals will be different. In fact, multiple selection more closely follows natural selection, where the fittest members produce more 'children'.

4.2.3.1 Deterministic selection

This is the most straightforward selection method. All members are sorted according to their fitness value and some of the best members are picked up for reproduction. A certain number of members (or population percentage) is usually chosen, or the members may be selected one by one and reproduced until the next generation population is filled up.

As noted before, deterministic methods are more suitable for the algorithms with small populations (less than about 20–40 members). They are therefore used in the areas where small populations are desirable (e.g. when fitness evaluation is extremely costly) or where it is traditionally adopted (in particular in *Evolutionary Strategies,* see Section 4.4).

It is generally accepted that the performance of an EA is measured in terms of fitness evaluations, because in most practical tasks, fitness evaluation takes considerably more computational resources than the algorithm framework itself. However, the resources taken for the elements of the algorithm (*time complexity*) can and should be estimated as well. This may help to chose a more efficient operator (selection in particular) if other considerations are equal.

Sorting of the list takes on average $O(N \log N)$ steps (N is the population size hereafter), using standard techniques [82]. After that, selection of each member requires simply $O(1)$ steps. Thus, the overall time complexity of the deterministic selection is $O(N \log N)$.

4.2.3.2 Fitness proportional selection and fitness scaling

In fitness proportional selection, all individuals receive the chances to reproduce that are proportional to their objective value (fitness):

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{4.4}$$

where p_i is the probability that the ith individual is selected, and f_i is the fitness of the ith individual (the greater the value, the fitter the individual is).

This method more closely follows natural selection than any deterministic scheme. It is common in classic Genetic Algorithms. There are several implementations of this general scheme which vary in stochastic properties and time complexity: *roulette wheel (Monte*

Carlo) selection [52], *stochastic remainder selection* [33] and *stochastic universal selection* [24]. The first of these methods is described here in more detail.

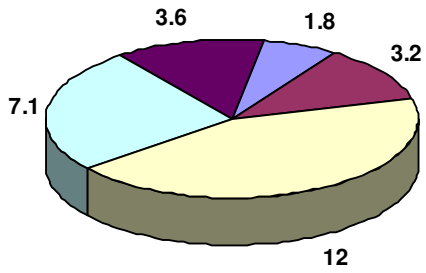


Fig. 4.2 Roulette wheel selection

The analogy with a roulette wheel arises because one can imagine the whole population forming a roulette wheel with the size of any individual's slot proportional to its fitness. The wheel is then spun and the 'ball' thrown in. The probability of the 'ball' coming to rest in any particular slot is proportional to the arc of the slot and thus to the fitness of the corresponding individual [44]. This approach is illustrated in Fig. 4.2 for a population of 5 individuals with the fitness 1.8, 3.2, 12, 7.1 and 3.6.

This may be implemented in the following algorithm:

1. Sum the fitness of all population members f_Σ
2. Chose a random number R between 0 and f_Σ
3. Let $s = 0$, $i = 0$
4. while $s < R$, do
5. $\quad i = i + 1$
6. $\quad s = s + f_i$
7. end loop
8. return f_i as the selected individual.

The algorithm is repeated as many times as necessary for each individual to be selected. Of course, the first step may be executed only once in a generation. The time complexity of the presented scheme is $O(N^2)$ steps, because each selection is done in $O(N)$ steps (on average, half the list is searched) and usually N 'spins' are required to fill the new population. This may be improved to $O(N \log N)$ steps if the list of cumulative slot totals is prepared beforehand and a binary search is used to locate the correct slot [82].

As already noted, the take-over time Γ of fitness proportional selection has the general order $N \ln N$ [82].

In the form given in the equation (4.1), there is no means to control the selection pressure and the convergence speed: they are determined entirely by the fitness of each individual. However, such a control is often necessary. If for example, early during a run one particularly

fit individual is produced, fitness proportional selection with replacement can allow a large number of copies of this individual (or its immediate derivatives) to flood the subsequent generations. Furthermore, during the later steps of the search when many of the individuals will be similar, this selection scheme will pick up an approximately equal number of each individual. One of the methods intended to overcome this problem and to maintain a steady selection pressure is *linear fitness scaling* [44].

Linear fitness scaling works by pivoting the fitness of each individual about the average population fitness. The scale is chosen so that an approximately constant proportion of copies of the best members is selected compared to the 'average member'. This proportion value c_m is typically in the range 1.0 to 2.0. If $c_m = 2$, then twice the number of best individuals will be selected (on average) than will average individuals. Values closer to 1.0 produce a weaker selection pressure. The condition is denoted as $\tau_{best} = c_m \tau_{avg}$, where τ is the number of times an individual is selected.

The linear transformation to achieve the required scaling is:

$$f_i^s = af_i + b \tag{4.5}$$

where f_i is the true fitness of the ith individual, f_i^s is its scaled fitness, a and b are scale coefficients which are chosen every generation according to the above principle as follows.

Having two requirements: the mean fitness f_{avg} should be unchanged:

$$f_{avg}^s = f_{avg}$$

and the scaled fitness of the best individual should be c_m times the average fitness:

$$f_{max}^s = c_m f_{avg}$$

the following solution for a and b is obtained:

$$\begin{aligned} a &= \frac{(c_m - 1) f_{avg}}{f_{max} - f_{avg}} \\ b &= (1 - a) f_{avg} \end{aligned} \tag{4.6}$$

Such a transformation can produce a negative scaled fitness. This can be avoided in a number of ways, the simplest one being assigning a zero fitness if a negative value occurs. However, such a crude work-around introduces some bias to the selection, and a more elegant solution would be uplifting the average fitness with appropriate recalculation of both a and b coefficients.

There are some more sophisticated scaling techniques, such as *sigma scaling* [44], in which the (expected) number of trials each member receives is adjusted according to the standard deviation of the population fitness σ_f.

$$\tau_i^{exp} = 1 + \frac{f_i - f_{avg}}{2\sigma_f} \tag{4.7}$$

4.2.3.3 Ranking selection

This is a development of the fitness proportional selection (see above), aimed to achieve greater adaptability and to reduce stochastic errors (i.e. the same problem addressed by fitness scaling). The idea represents the combination of fitness proportional and deterministic selection. The population is sorted according to the fitness, and a rank is assigned to each individual. The rank may be a simple 'order number', ranging from N (the best individual) to 1 (the worst one), or may represent any appropriate non-increasing function which is problem-dependent [23]. After assigning the rank, a proportionate selection is applied as described in the previous section, using rank values instead of fitness.

Ranking has two main advantages before fitness proportional selection (even that with fitness scaling). First, the required selection pressure can be controlled more flexibly by applying a specific rank assignment function. Second, it softens stochastic errors of the search, which can be especially destructive for the fitness functions affected by noise. The latter may be illustrated using the same example that has been used for fitness scaling explanation (see Section 4.2.3.2). Suppose that a particularly fit member is generated that stands well off the whole population (an 'outlier', see Fig. 4.3). Even if the proportionate selection is constrained by fitness scaling, this best member will be greatly favoured, whilst the rest of the population will receive very low selection pressure because the differences between their fitness values are insignificant as compared to the 'outlier'. In contrast, ranking will establish a predefined difference between the neighbouring members, ensuring an adequate selection pressure for the whole population.

Time complexity of this method can be calculated considering two separate stages. First, sorting requires $O(N \log N)$ steps, using standard techniques [82]. Then, a fitness proportional selection requires between $O(N \log N)$ and $O(N^2)$ (see Section 4.2.3.2), which gives

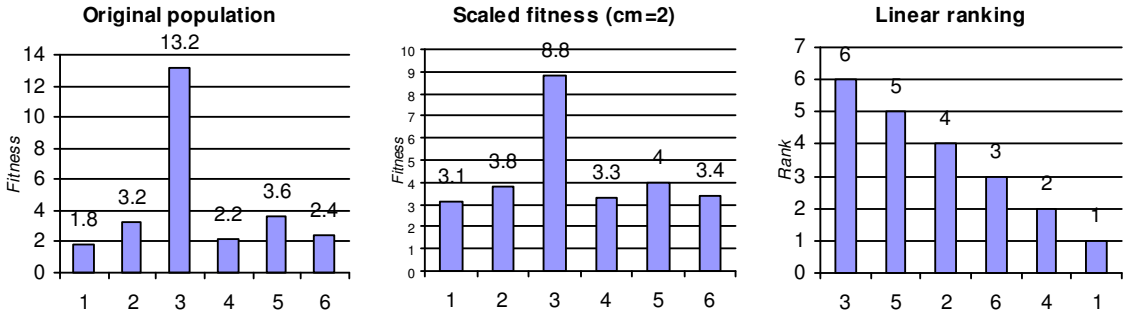


Fig. 4.3 Linear fitness scaling compared to linear ranking

the overall time complexity of the same order $O(N \log N)$ to $O(N^2)$.

The take-over time depends on the actual ranking scheme used. For linear ranking, it is of the order $\ln N$ [82], i.e. much lower than with fitness proportional selection.

4.2.3.4 Tournament selection

Tournament selection is a simple yet flexible stochastic selection scheme. The idea is as follows. Choose some number s of individuals randomly from a population and then select the best individual from this group. Repeat as many times as necessary to fill up the mating pool. This somewhat resembles the tournaments held between s competitors. As a result, the mating pool, being comprised of tournament winners, has a higher average fitness than the average population fitness.

The selection pressure can be controlled simply by choosing appropriate tournament size s. Obviously, the winner from a larger tournament will, on average, have a higher fitness than the winner of a smaller tournament. Moreover, as the absolute difference between the fitness of competing members is not taken into account, tournament selection has essentially the same self-adaptation properties as *ranking selection* (see Section 4.2.3.3 above)—that is, it maintains a certain selection pressure across the population and over the time, irrespective of the actual distribution of fitness.

In addition, selection pressure can be further adjusted via randomisation of the tournament. Instead of simply choosing the fittest individual from the group, the winner may be determined by the following stochastic algorithm:

```
choose the best individual from the tournament with probability p
or, choose the second best individual with probability p(1–p)
or, choose the third best individual with probability p(p(1–p))
...and so on until the last individual.
```

Here p is a predefined value in the range (0.5; 1]; the greater values apply stronger selection (when $p = 1$, this scheme turns into the usual deterministic tournament). As shown in [149], a binary (i.e. with $s = 2$) probabilistic tournament with properly selected p exactly matches the linear ranking selection in expectation. However, the most commonly used variant is deterministic binary tournament. If a stronger selection pressure is required, a greater tournament size is used; however, it rarely exceeds 5–6.

One of the advantages of the tournament selection is its algorithmic efficiency. The time complexity of this method is only $O(N)$. Indeed, each competition in the tournament requires the random selection of a constant number of individuals from the population. The comparison among those individuals can be performed in a constant time, and usually N such competitions are required to fill a generation [82]. Moreover, tournament selection is very

simple to code on both parallel and non-parallel computer architectures. Simplicity of parallelisation is especially remarkable because evolutionary algorithms are particularly suitable for parallel computing.

The take-over time (for deterministic tournament) is [82]:

$$\Gamma \approx \frac{1}{\ln s}\left(\ln N + \ln\left(\ln N\right)\right) \tag{4.8}$$

This implies that the take-over time rapidly decreases as the tournament size s increases.

4.2.4 Recombination

Recombination allows solutions to exchange the information in a way similar to that used by a biological organism undergoing sexual reproduction. This effective mechanism allows to combine parts of the solution (*building blocks*, see Section 4.2.9) successfully found by parents. Combined with selection, this scheme produces, on average, fitter offspring. Of course, being a stochastic operation, recombination can produce 'disadvantaged' individuals as well; however, they will be quickly perished under selection.

Note that sexual reproduction does not imply presence of different (two or more) sexes. The only required ability is exchange of genetic information, which may occur between the individuals of the same sex. In fact, the concept of different sexes (*sexual determination*) is not used in practical EAs and all individuals in the population can mate with each other. Another point is that the number of parents to supply the genetic information is not required to be two, as well as the number of 'children' is not strictly limited. However, typical GAs deal with two parents producing one or two children.

Recombination is usually applied probabilistically with a certain probability P_c. For GAs, the typical P_c value is between 0.6 and 0.8; however, the values up to 1.0 are common.

4.2.4.1 Alphabet-based chromosome recombination

In essence, recombination is 'blending' the information of two (or more) genomes in some way. For typical GAs, an approach from natural genetics is borrowed. It is known as *crossover*[1]. During crossover, chromosomes exchange equal parts of themselves. In its simplest form known as *single-point crossover*, two parents are taken from the mating pool. A random position on the chromosome (*locus*) is chosen. Then, the end pieces of the chromosomes, starting from the chosen locus, are swapped (Fig. 4.4).

Note that if the chromosome carries multiple parameters (refer to the example given in the Section 4.2.2), the crossover locus generally does not need to observe the parameters'

[1] Biologists also use the term *crossing over*. In artificial evolutionary approach, *crossover* is often used as the general term for any recombination method.

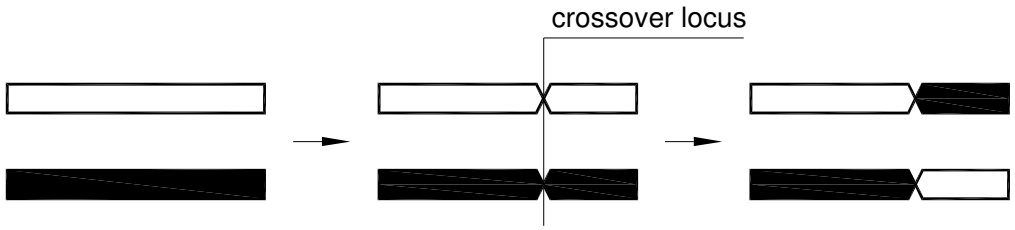


Fig. 4.4 Single point crossover

boundaries. Moreover, exactly swapping the *parts* of the parameters is the primary goal of crossover. However, in some cases, especially when the mating chromosomes are not homogenous, keeping the boundaries may be required. In such cases, another way of changing the parameters is necessary, for example, a special 'internal' crossover or mutation.

Single-point crossover can be generalised to *k*-point crossover, when *k* different loci are chosen and then every second piece is swapped. However, according to De Jong's studies [52], multi-point crossover degrades overall algorithm performance increasingly with an increased number of cross points. More recent studies, e.g. [207], confirm this conclusion.

There is another way of swapping pieces of chromosomes, known as *uniform crossover*. This method does not select crossover points. Instead, it considers each bit position of the two parents one by one and swaps the corresponding bits with a probability of 50%. Although the uniform crossover is, in a sense, an extreme case of multi-point crossover and can be considered as the most disruptive its variant, both theoretical and practical results [204, 206, 207] show that uniform crossover outperforms *k*-point crossover in most cases.

The above methods can be extended in a number of ways to add greater flexibility and controllability which may be needed for a specific task. For example, special control bits may be added to the chromosomes that control application of the crossover to the respective chromosome [204]. A particularly interesting approach is the adaptive multi-parent symbolic combination of partially-defined chromosomes, developed by R. Watson and J. Pollack [221]. Instead of recombining the parts of the chromosomes, it merges the whole individuals. Combined with other elegant adaptive methods, it forms what the author calls *The Symbiogenic Evolutionary Adaptation Model (SEAM)* [222].

4.2.4.2 Real-value recombination

Real-coded EAs require a special recombination operator. Unlike bit strings, real parameters are not deemed as strings that can be cut into pieces. Instead, they are processed as a whole in a common mathematical way.

Due to rather historical reasons, real-coded EAs were mostly developing under the influence of Evolutionary Strategies and Evolutionary Programming (see Section 4.4), which

put more emphasis on mutation than recombination. As a result, real-value recombination has not been properly considered until the fairly recent past ('90s). Nevertheless, a number of various recombination techniques have been developed. Detailed analysis of them is available in [27, 57, 92].

The simplest real-value recombination one can think of is the averaging of several parents, which is known as *arithmetic crossover*. This method produces one offspring from two or more parents. Despite the presence of more advanced techniques, arithmetic crossover is used in many ESs which employ their own adaptation technologies and thus do not require self-adaptation from the recombination [90]. However, averaging may be weighted according to parents' fitness or using random weighting coefficients.

Self-adaptive recombination create offspring statistically located in proportion to the difference of the parents in the search space. These recombination operators generate one or two children according to a probability distribution over two or more parent solutions. Simply stated, if the difference between the parent solutions is small, the difference between the child and parent solutions should also be small.

The most popular approach is to use a uniform probability distribution—the so called *'blend crossover'*, *BLX* [64]. The BLX operator randomly picks a solution in the range

$$\left[x_1 - \alpha\left(x_2 - x_1\right); x_2 + \alpha\left(x_2 - x_1\right)\right]$$

for two parents $x_1 < x_2$. α is the parameter which controls the spread of the offspring interval beyond the range of the parents' interval $[x_1; x_2]$. For $\alpha = 0$, BLX simply picks a solution within the parents' interval $[x_1; x_2]$. In a number of papers it has been reported that $\alpha = 0.5$ delivers the best performance (e.g. [58]). This value implies that the probabilities that the offspring will lie inside the parents' interval and outside it are equal. The α value is usually included in the designation of the operator in the way BLX-α (e.g. BLX-0.5).

Other approaches suggest non-uniform probability distribution. The *Simulated Binary Crossover (SBX)* uses a bimodal probability distribution with its mode at the parent solutions. It produces two children from two parents. This technique has been developed by K. Deb and his students in 1995 [55]. As the name suggests, the SBX operator simulates the working principle of the single-point crossover operator on binary strings. The procedure of computing the children solutions x_1^* and x_2^* from parents x_1 and x_2 is as follows. First, the distribution index η must be specified, which

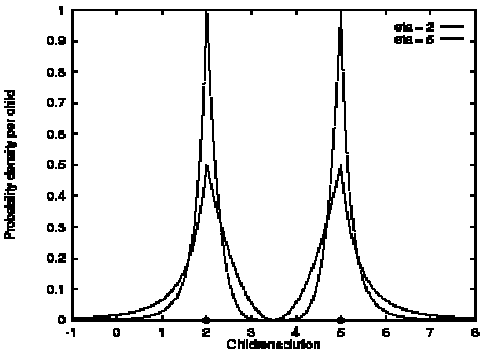


Fig. 4.5 Probability distribution for creating children solutions in SBX crossover

may be any non-negative real number. A large value of η gives a higher probability of creating near parent solutions, while a small value allows distant solutions to be produced. Fig. 4.5, taken from [58], shows an example of the distribution for the two values of η of 2 and 5 and the parents $x_1 = 2$ and $x_2 = 5$. This distribution is derived to have a similar search power as that in a single-point crossover in binary-coded GAs. η is usually selected once for the whole search. The other parameters are calculated for each recombination procedure.

A uniformly distributed random number u between 0 and 1 is then generated. Thereafter, a so called spread factor β is calculated as follows:

$$\beta = \begin{cases} (2u)^{\frac{1}{\eta+1}} & \text{if } u \le 0.5 \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases} \tag{4.9}$$

After that, the children solutions may be obtained:

$$\begin{aligned} x_1^* &= 0.5\left((1+\beta)x_1 + (1-\beta)x_2\right) \\ x_2^* &= 0.5\left((1-\beta)x_1 + (1+\beta)x_2\right) \end{aligned} \tag{4.10}$$

Note that two children solutions are symmetric about the parent solutions. This is deliberately used to avoid any bias towards any particular parent solution in a single crossover operation.

The SBX operator has been extended by its developers to a multi-parent case in the *Parent-Centric Crossover (PCX)* [57].

There is a closely related method which uses a triangular probability distribution instead of that shown in Fig. 4.5. The method is known as *Fuzzy Recombination (FR)* [218]. For FR, β is calculated as

$$\beta = 1 + 2\zeta d \tag{4.11}$$

where ζ is a random number with a triangle distribution in $[-1; 1)$, which can be easily obtained by summing two independent uniformly distributed in $[0; 1)$ random numbers u: $\zeta = u_1 + u_2 - 1$. d is a strategy parameter similar to α for BLX crossover and η for SBX. As a rule of thumb, d may be set to 0.5, although values from about 0.3 to 1.2 may be more effective for some problems [27]. After obtaining β, children solutions are calculated using (4.10).

A different approach is demonstrated by the *Unimodal Normal Distribution Crossover (UNDX)* [162]. It uses multiple (usually three) parents and create offspring solutions around the centre of mass of these parents. Techniques with such a behaviour are called *mean-centric recombination* (unlike *parent-centric* SBX and FR) [57]. UNDX uses a normal probability distribution, thus assigning a small probability to solutions away from the centre of mass. Another mean-centric recombination operator is the *Simplex Crossover (SPX)* [94]. It differs

from UNDX by assigning a uniform probability distribution for creating offspring in a restricted region marked by the parents. This makes the SPX related to the BLX-0 operator, generalised to a multi-parent case.

Although both mean-centric and parent-centric recombination methods were found to exhibit self-adaptive behaviour for real-coded GAs similar to that of ESs (see Section 4.4) [27, 56], in a number of reports parent-centric methods were found generally superior [57].

4.2.5 Mutation

Mutation is another genetic operator borrowed from nature. However, unlike recombination, which is aimed at producing better offspring, mutation is used to maintain genetic diversity in the population from one generation to the next. As such, mutation is a purely explorative operator. As noted above (Section 4.2.3), sufficient exploration allows the search to avoid local optima and thus increases robustness of the algorithm.

Not unlike recombination, mutation works differently for alphabet-based chromosomes and real-coded algorithms. However, in both cases it is merely a blind variation of a given individual. Nonetheless, parameters of this variation may be controlled and adjusted to obtain a required degree of exploration, similar to adaptive recombination operators discussed above.

4.2.5.1 Bit string mutation

In nearly all ordinary GAs, mutation is implemented as variation of a random bit in the chromosome. Each bit in the chromosome is considered one by one and changed with certain probability P_m. Bit change can be applied either as flipping (inverting) of the bit or replacing it with a random value. In the latter case, the actual mutation rate will be twice as low, because, on average, half of the bits are replaced with the same value. A more computationally effective way may be employed as well with essentially the same outcome. For example, an individual may be chosen for mutation with a probability $L \cdot P_m$ (L is the chromosome length), and then a random bit selected and changed. This way, not more than one bit in a chromosome is changed, but slow bit-by-bit scanning is avoided.

In GAs, mutation is usually controlled through mutation probability P_m. As a rule, GAs put more stress on recombination rather than on mutation, therefore typical mutation rates are very low, of the order of 0.03 and less (per bit). Rates close to 0.001 are common. For sphere problem (see example in the Section 4.2.2), $P_m = 1/L$ has been proven to be optimal [20]. Nevertheless, mutation is very important because it prevents the loss of building blocks which cannot be recovered by recombination alone.

This can be illustrated by a simple example. It is possible that a population may have all its members having the same value of one (or more) particular bit. For instance, all mem-

bers of the population 1010, 1110, 0000, 0010, 1000 have the last bit set to 0. This may happen due to poor initial sampling or greedy exploitation—at any rate, the stochastic nature of GAs can lead to such a result. In this case, not one of the crossover methods discussed in the Section 4.2.4.1 would lead to appear 1 at the last position. Nonetheless, the optimal solution may contain 1 there. Introducing a mutation operator helps to circumvent this problem. Even despite a very low mutation rate, there are significant chances that the lost bit will be mutated (they are of the order $1/L$ for each mutation applied) and this mutation will be reproduced.

Mutation probability can be controlled in a number of ways. For simple GAs, it may be worthwhile to reduce the mutation rate gradually as the strategy shifts towards exploitation. A number of such control functions $P_m(t)$ have been proposed [21]. However, this may easily have an adverse effect—mutation is known for the ability to 'push' the stagnated process at the later stages. This is particularly true if selection with replacement is used (see Section 4.2.3), as this approach tends to crowd the population with multiple copies of highly fit building blocks (even though all individuals may be different after undergoing crossover, see 4.2.9).

Another technique to avoid stagnation is so called 'hypermutation'. Hypermutation is a method in which mutation probability is significantly (10–100 times) increased for a small number of generations (usually one) during the run, or such a high rate is applied constantly to a certain part of the population (e.g. 20%). Hypermutation may be utilised periodically or when stagnation is detected. An extreme case of hypermutation is an influx of *random immigrants*, i.e. replacing a part of the population with randomly initialised individuals (or simply adding them to the population). Both hypermutation and random immigrants techniques are especially effective for dynamic environments, where the fitness landscape can change significantly over time [86].

The above mutation control methods have a common drawback: they are not selective. Mutation is controlled for the whole population or a random subset of the population. At the same time, an individual approach may be desirable. In particular, stronger mutation might be applied to the weakest members, while less disruptive mutation to the best individuals. Alternatively, some more sophisticated methods can be developed. Such fine tuning is a more common feature of Evolutionary Strategies (see 4.4) and real-coded GAs. However, similar methods are available for bit-string chromosomes as well.

The first aspect to be taken into account is mutation strength with respect to *phenotype*. The effect of mutation on the phenotype depends on both mutation properties and the genome encoding used (Section 4.2.2). For simple binary integer encoding, a single mutated bit can have an effect on the genotype as increasing/decreasing the genotype value by 1 (when the least significant bit is changed) to 2^{n-1} (the most significant bit is changed), where n is the

length of the string used to represent a particular parameter. If linear mapping is used, the corresponding effect on the phenotype will range from the minimal possible $S/(2^n-1)$ to about $S/2$, S being the defined range for the respective variable. Thus, the strongest possible mutation causes the 'jump' of the parameter value the size of a half of the given range in the search space. Therefore, if all bits in the chromosome are mutated with equal probability, like in the way described above, there will be a fairly high proportion of such strong mutations. Most of them will be lethal, especially at the latter stages of evolution.

To illustrate this, let us return to the sphere problem example from the Section 4.2.2. In that example, 8 bit encoding was used to represent the values from the range (–1; 1). Suppose the individual 10110110 (182 decimal) is being mutated. According to linear mapping (3.1), this number corresponds to the problem space (phenotypic) value 0.42745. Flipping the least significant bit yields 10110111 (183 decimal), which corresponds to 0.43529. This is rather a small mutation. However, if the most significant bit is inverted, the result will be 00110110 (54 decimal), that is –0.57647. It is extremely unlikely that such a strong mutation will be of any help. Therefore, an apparent idea would be to shift the mutation probability towards lower bits to increase the number of 'soft', useful mutations. Indeed, this simple solution proved to be beneficial for some problems [69].

Another solution is to use a genome encoding with adjacency property. In such an encoding, adjacent integers differ by a single bit, for example, 000 (0), 001 (1), 011 (2), 010 (3), 110 (4), 111 (5), 101 (6), 100 (7). There exist many different variations of this code, and the most commonly used variant is known as *Gray code* (shown above, see also [44, 80]). Application of Gray code to GAs has been studied in one of the very first publications about GAs [98]. Using Gray code in the genome does not eliminate strong mutations caused by swapping of a single bit; however, mutations are distributed more favourably.

There are plenty of more sophisticated mutation control methods. Some of them have already been mentioned when discussing genome representation (Section 4.2.2), in particular, enumerative redundancy. In general, redundancy introduces extra bits (or just unused genotype values) to the representation of a particular variable. These bits do not affect the phenotype, but they do attract some number of mutations. Thus, by varying the relative number of 'spare' bits of each individual parameter, different mutation rates can be achieved. In addition, some of the 'most used' phenotype values may be assigned to the newly added bits giving a favourable bias, like in the example with a pipeline network from Section 4.2.2.

Some more direct control methods have been proposed as well (see, for example, [86]). Not unlike the above method, they utilise additional bits in the genotype which do not affect the phenotype directly. However, these bits control the mutation itself. In the simplest case, a single flag bit can control the applicability of mutation to the respective parameter:

zero value disables mutation and unity enables it. Any imaginable information can be stored in the control bits: mutation probability, level of bias, etc. This approach closely resembles Evolutionary Strategies (see 4.4), but unlike them, is tuned for binary strings and generally plays relatively minor role for the algorithm.

4.2.5.2 Real-value mutation

Implementation of a real-value mutation is rather more straightforward than that of alphabet strings. Unlike the latter, it is not an operation directly inspired by nature; however, as the real-coded algorithms generally do not use tricky encoding schemes and have the same problem-space and genotype values of the parameters, real-value mutation can be considered as the operation working on a higher level, up to direct phenotype mutation for function optimisation problems.

Mutation to a real value is made simply by adding a random number to it. As mutation is considered as an explorative genetic operator, it is usually required to be undirected. Therefore, the distribution of the random numbers to be used for mutation should be zero mean. An overwhelming majority of real-coded algorithms use Gaussian (normal) distribution for this operation. Sometimes, a uniform distribution within defined limits is used. Gaussian distribution naturally allows a higher number of small mutations and small number of distant solutions to appear.

It is evident that the strength of real-value mutation can be controlled in a very convenient way, through the variance of the distribution (or the window size for the uniform distribution). Like with the common GAs that operate string-type chromosomes, mutation strength can be adapted using many different techniques, from simple predefined linear decrease to sophisticated adaptation strategies (see Section 4.4).

More often than not real-coded EAs apply mutation to all individuals in the population, leaving the mutation control to the mutation strength as described above. However, probabilistic mutation is possible as well.

4.2.6 Mutation vs. Recombination

Which genetic operator—recombination or mutation—is more important for performance and robustness of an EA? Are they both necessary? There is much controversy in the EA community about this question. The existence of many different forms of recombination further complicates the issue. Traditionally, genetic algorithms and genetic programming stress the role of recombination, while evolutionary programming and evolution strategies stress the role of mutation. There are many successfully applied algorithms which use only one of these operators.

One of the first simulations of evolutionary process has been conducted by Fogel [70, 73]. It used iterative mutation and selection to evolve a logic suitable to predict symbol strings generated from Markov processes—an approach developed later into evolutionary programming (EP, see 4.4). A process similar to mutation is known and widely used in a variety of optimisation techniques, for example, *simulated annealing* [216].

On the contrary, Holland [97] found that exactly crossover was the driving force for his genetic algorithms (along with selection, of course), and mutation played only a secondary role. Apparently, this situation has some connection with natural evolution, in which mutations are occasional and scattered—unlike crossover, which takes place during every (sexual) reproduction. However, as it has been shown above in 4.2.5.1, mutation is required for classical GA in order not to lose important building blocks.

Direct comparisons of the effect of the two operators are somewhat controversial too. Fogel in [71] clearly states that any recombination operators have no benefit for real-coded EAs in terms of both robustness and performance. Moreover, he negates that GAs with chromosome strings are different in this sense and states that recombination operators 'are merely a subset of all random mutations. As in all subsets, their applicability will be strongly problem dependent, if advantageous at all.'

Other researchers give less rigorous conclusions. Schaffer and Eshelman [184] empirically compared mutation and crossover and concluded that crossover can exploit epistasis (positional nonlinearity) that mutation alone cannot. Considerable effort has been made for theoretical comparison as well (e.g. [205]). However, the theories are not sufficiently general to predict when to use crossover or what form of crossover to use. In particular, they do not consider population size, yet population size can affect the relative utility of crossover and mutation operators. Mutation appears to be more useful than crossover when the population size is small, while there is evidence that crossover can be more useful than mutation when the population size is large [204, 207]. Intuitively, it makes sense because recombination works on information exchange between individuals and thus heavily depends on genetic diversity of the population. On the contrary, mutation is an independent operation.

However, this independence and simplicity of mutation has a downside. Pure mutation does not exhibit any self-adaptive behaviour and therefore requires, more often than not, external control (see 4.2.5 and 4.4). Unlike this, the perturbations introduced by crossover naturally diminish as the genetic diversity reduces near optimum.

The current state of the EA practice generally confirms the above observations. ESs and EP, which rely on mutation, use relatively small populations (up to 10–20 members in most cases), while recombination-based GAs and GP exploit larger populations.

4.2.7 Measuring performance

Performance of an EA is the measure how effective the algorithm is in search of the optimal solution. As evolutionary search is a stochastic and dynamic process, it can hardly be positively measured by a single figure. A better indicator of the performance is a *convergence graph*, that is, the graph 'fitness vs. computation steps'. This graph may take several forms which are described below.

Why the picture of the dynamic process is important is illustrated in Fig. 4.6. This figure presents two typical convergence graphs of two independent runs of the same GA with different sets of parameters. It can be seen that although the first run (blue line) converges faster, it stagnates too early and does not deliver the optimal solution. Moreover, as the optimal solution is usually unknown in advance, stagnation is not an indicator of reaching the optimum. Occasional mutations can push the search further even after long stagnation. Besides, a temporary deterioration of the fitness is common, especially in dynamic environments, and is not a sign of failure. Therefore, not the convergence speed nor time to reach a specified value, nor any other single parameter can be considered as a sole performance measure.

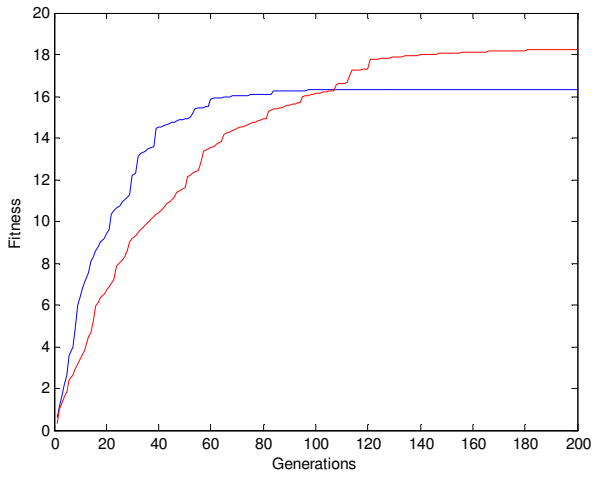


Fig. 4.6 Typical convergence graphs

De Jong in his milestone dissertation [52] used two measures of the progress of the GA: the off-line performance and the on-line performance. The off-line performance is represented by the running average of the fitness of the best individual, f_{max}, in each population:

$$f_{off}(g) = \frac{1}{g}\sum_{i=1}^{g} f_{max}(g) \tag{4.12}$$

where g is the generation number. In contrast, the on-line performance is the average of all fitness values calculated so far:

$$f_{on}(g) = \frac{1}{g}\sum_{i=1}^{g} f_{avg}(g) \tag{4.13}$$

The on-line performance includes both good and bad guesses and therefore reflects not only the progress towards the optimal solution, but also the resources taken for such progress.

Another useful measure is the convergence velocity [20]:

$$V(g) = \ln \sqrt{\frac{f_{max}(g)}{f_{max}(1)}} \tag{4.14}$$

In the case of a non-stationary dynamic environment, the value of previously found solutions is irrelevant at the later steps. Hence, a better measure of optimisation in this case is the current-best metric instead of running averages. For dynamic environments, the graphs $f_{max}(g)$ and $f_{avg}(g)$ are often used rather than off-line and on-line performance values.

It is important that such performance measures are averaged over a sufficiently large number of runs if sensible results are to be achieved. Evolutionary search is a stochastic process and different runs may result in different performance for the same initial settings.

When comparing the performance of different algorithms, it is better to use the number of fitness evaluations instead of the number of generations as the argument for performance characteristics. Different algorithms may use significantly different population sizes (which may change dynamically), and thus the comparison over the number of generations is inappropriate. Moreover, as noted above in Section 4.2.1, it is generally accepted in calculus that the performance of an optimisation algorithm is measured in terms of objective function evaluations, because in most practical tasks objective evaluation takes considerably more computational resources than the algorithm framework itself.

Finally, it should be noted that there is no best method for all classes of problems. This is mathematically expressed in the so called 'no-free-lunch' theorems of Wolpert and Macready [224]. Therefore, any comparison of competing algorithms must be qualified by a specification of the class of problems (i.e. fitness landscapes) under consideration.

4.2.8 The problem of convergence and genetic drift

From the example in the previous section it is seen that the progress of EAs is by no means linear. Usually, it is fast at first and then loses its speed and finally stagnates. This is a common scenario for optimisation techniques. However, it has been shown (see also Section 4.2.3) that progressing too quickly due to greedy exploitation of good solutions may result in convergence to a local optimum an thus in low robustness. Several methods that help to control the convergence have been described in the above sections, including selection pressure control and adaptation of recombination and mutation rates.

However, it is unclear to which degree and how in particular to manage the algorithm's parameters. What is 'too fast' convergence? Unfortunately, the current state of EA theory is inadequate to answer this question *before* the EA is initiated. Only for the simplest optimisation tasks such as sphere model and flat fitness landscape does the theory provide optimal settings. Existing more general models are computationally expensive and too complex for

evaluation (see, for example, [26, 154]). For the most of the real-world problems, it takes several trial runs to obtain an adequate set of parameters.

The picture of convergence process, as noted before, is not a good indicator of the algorithm's characteristics. Nor does it show what is happening inside the algorithm. In contrast, the plot of *genetic diversity* of the population against generation number (or fitness function evaluations) gives a picture which can explain some performance problems. If the population quickly loses the genetic diversity, this usually means a too high initial selection pressure. The further saturation may be attributed to reduction of selection pressure at the later stages. The loss of genetic diversity is known as *genetic drift*.

The amount of genetic diversity in a population can be measured in several ways. For bit string genomes, an easily calculable measure is $\eta_{\max}$, the bit-to-bit similarity between the fittest individual and the other members of the population [44]. To calculate $\eta_{\max}$, each bit in the fittest string is compared with the same bit in all other strings in turn. Any matching bits increment η_{max} by 1. The result is then normalised by dividing by the total number of bits that have been compared, i.e. $L(N–1)$, where L is the chromosome length (see also Section 4.2.2). For real values, genetic diversity can be measured as the variance of genome values of the population.

Fortunately, there exist many techniques which help an EA to acquire self-adaptation properties. Some of them have been discussed already in the previous sections, for example, fitness scaling (4.2.3.2) and adaptive recombination schemes (4.2.4.2). Such techniques implicitly take genetic diversity into account. A thorough study of self-adaptation features of EAs is presented in [27]. In addition, there are evolutionary methods purposely developed to be self-adaptive to a wide range of difficult problems (see Section 4.4).

4.2.9 Schema theory

Schema theory has been developed by John Holland [97] and popularised by David Goldberg [80] to explain the power of binary-coded genetic algorithms. More recently, it has been extended to real-value genome representations [64] and tree-like *S*-expression representations used in genetic programming [132, 158]. Due to the importance of the theory for understanding GA internals, it is briefly outlined here.

A schema is a similarity template describing a subset of strings with similarities at certain string positions [80]. Schemata are built from the same alphabet as the chromosome (for simplicity, let us constrain it here to the most popular binary alphabet {0;1}), plus a special *don't care* symbol *. Therefore, a schema is a string built of {0;1;*}. A schema matches a particular string if at every location a 1 in the schema matches a 1 in the string, a 0 matches a 0, or * matches either (note that * replaces only one digit, unlike the respective wildcard on

many computer systems). For example, the schema 100* matches two strings: 1000 and 1001. These strings are said to be the *instances* of this schema. The schema *00* matches four strings, two the above plus 0000 and 0001. Each string of the length L belongs to 2^L schemata (note that this number is independent of the size of the alphabet used) [118]. Therefore, a population of size N contains between 2^L and $N \cdot 2^L$ schemata, depending on the population diversity.

All schemata are not created equal. Some are more specific than others. For example, the schema 11*0 is a more definite statement than the schema 1***. The number of fixed positions (that is, of 1's and 0's) in a schema H is called *schema order* and is denoted by $o(H)$. The above schemata have the order 3 and 1 respectively. Furthermore, the distance between the leftmost and rightmost defined bits in a schema is also important and is referred as *defining length* $\delta(H)$ of the schema. For example, the schema 10**1* has the defining length 4, while the defining length of the schema ****1* is 0.

A demonstrative visual conception of schemata can be obtained by looking at them as on defining hyperplanes in the search space $\{0;1\}^L$, as shown in Fig. 4.7 [74]. Each schema defines a subspace (hyperplane) of the original search space. As a rule, the greater schema order, the smaller the subspace is. At the extremes, the schema **** (quite useless one) covers the whole search space, and a fully defined individual represents a point in the search space. Any of the four hyperplanes in Fig. 4.7 also belong to two lower-order hyperplanes.

Schemata can be viewed much the same way as fully defined genomes, as the individuals in the search space, with the exception that they define not the points but various regions. A GA is thought to work by directing the search towards schemata containing highly fit regions in the search space. Like any individuals, each schema may be associated with a fitness value. In particular, the *average fitness of a schema* is the average of the fitness values of each individual in the population that belongs to a given schema. In turn, the fitness of any individual in the population gives some information about the average fitness of the 2^L different schemata of which it is an instance, so an explicit evaluation of a population of N individuals is also an implicit evaluation of a much larger number of schemata.

This implicit operation of GA is crucial for understanding how it works. By sampling the search space explicitly with a relatively few number of points (individuals), GA implicitly gathers the information about

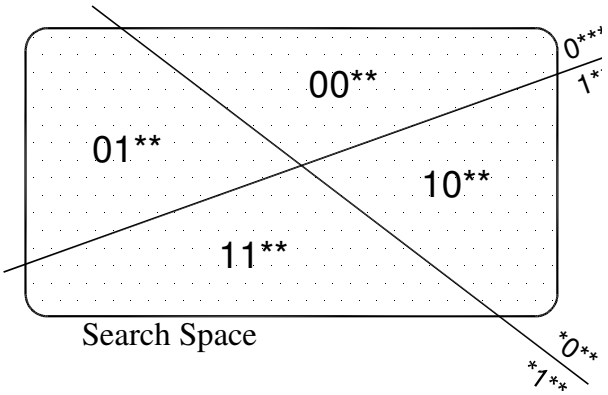


Fig. 4.7 Schemata and the search space

the large regions in the search space. This information is stored in the gene pool of the whole population. Koza [118] explains it by drawing a parallel with nature: 'In the genetic algorithm, as in nature, the individuals actually present in the population are of secondary importance to the evolutionary process. … The individuals in the population are merely the vehicles for collectively transmitting a genetic profile and the guinea pigs for testing fitness in the environment.'

In order to understand how GA processes schemata, the effect of reproduction, crossover and mutation on schemata should be considered. The effect of reproduction is straightforward: since more highly fit individuals have higher probability of selection, on average, the number of samples to the best observed schemata steadily increases. More precisely, schemata with fitness values above the population average receive an increasing number of samples in the next generation, while schemata with fitness values below the population average receive a decreasing number of samples. Therefore, the fitness proportional reproduction alone allocates an *exponentially* increasing number of trials to above-average schemata (or decreasing number for below-average schemata, refer to [80] or [97] for proof). Holland [97] and De Jong [52] also show, by connecting GA search with the so called *multi-armed bandit problem*, that such a strategy is optimal for a statistical search among given alternatives when no a priori information is given.

It should be noted that this behaviour is carried out with every schema in a population. That is, by sampling only N points, all 2^L to $N \cdot 2^L$ schemata contained in the population are processed. However, not all of them are processed usefully, because a significant part of the schemata is disrupted by crossover, which is explained below. Holland estimates the number of usefully processed schemata as N^3 (see also [80] for derivation). The ability of GAs to process about N^3 schemata in every generation by sampling only N points is known as *implicit parallelism*.

Unlike pure selection, genetic operators such as crossover and mutation allow to sample *new* points in the search space. However, both these operators can destroy schemata. To observe how it can happen, let us consider a simple example. Assume an individual A of length $L = 6$ is undergoing crossover with another individual, and the crossover locus is between 4th and 5th bits (counting from the left). Of all 2^6 schemata of which it is an instance, we consider two different ones:

A = 1 1 0 1 | 0 0
H_1 = 1 * * * | * 0
H_2 = * * 0 1 | * *

Unless the second mating individual is identical to A at the fixed positions (which is rather unlikely for practically used string lengths), the schema H_1 will be destroyed by crossover,

while the schema H_2 will survive because for H_2 the 'tail' part of the second individual is irrelevant. Therefore, a schema is likely to be destroyed by crossover if the crossover point (which is chosen at random) falls between any fixed positions of the schema. Clearly, the chances to be destroyed depend on the defining length of the schema: the greater the defining length $\delta(H)$, the more chances that the crossover point will lie between the defined positions. In the example, the schema H_1 has the maximum possible defining length $\delta(H_1) = 5$ and thus will be destroyed almost inevitably (leaving for survival the marginal chances of mating with an instance of the same schema). In contrast, $\delta(H_2) = 1$ and so H_2 has fairly low chances to be destroyed. This can be expressed quantitatively: if the crossover point is selected uniformly at random among the $L - 1$ possible sites, the probability of a schema H to survive will be $P_s = 1 - \delta(H)/(L - 1)$.

The effect of mutation is easy to determine. As discussed in Section 4.2.5.1, mutation is the random alteration of a single bit with probability P_m. In order of a schema H to survive, all of the specified bits must themselves survive. The number of such positions is schema order $o(H)$. Each position survives with the probability $1 - P_m$. Thus, for statistically independent mutations, a schemata survives with the probability $(1 - P_m)^{o(H)}$. For practically used in GAs $P_m << 1$, this expression can be approximated as $1 - o(H) \cdot P_m$.

Therefore, it can be concluded that short defining length, low order schemata usually survive from generation to generation and thus are of particular importance for GA. Instead of building high performance individuals by trying every conceivable combination, GA *constructs* better and better individuals from the best partial solutions of past samplings. Such low order, short defining length and highly fit schemata that deliver these solutions are called *building blocks*, and the claim that they are used by GA to construct high performance individuals is known as *building block hypothesis*.

Combining the effect of reproduction, crossover and mutation, the following expression for the expected number of copies m a particular schema H receives in the next generation can be obtained [80]:

$$m(H, g+1) \geq m(H, g) \frac{f(H)}{f_{avg}(g)} \left(1 - P_c \frac{\delta(H)}{L-1} - o(H) P_m \right) \tag{4.15}$$

Here P_c and P_m are crossover and mutation probabilities. It can be seen that the former conclusion about exponential growth of the above-average schemata number holds for the algorithm itself. This is what is known as the *Schemata Theorem*.

The following sections briefly describe several types of evolutionary algorithms commonly used in practice. All of them are extensions of the general scheme presented in Section

4.1, and the above discussion is highly relevant to them. Nevertheless, their properties, applications and, no less important, usage practice differ considerably. This is due to not only mathematical or algorithmic difference, but also traditional preferences and years of independent development. However, the actual development of the field is characterised by a progressing integration of the different evolutionary approaches, so that the utilisation of the common labels 'genetic algorithm,' 'evolution(ary) strategy' and 'evolutionary programming' sometimes might even be misleading [22]. Not all of the existing EAs are described, of course, but only those which are relevant to this study.

4.3 Genetic Algorithms

Genetic algorithms are one of the first evolutionary methods successfully used to solve practical problems, and until now they remain one of the most widely used EAs in the engineering field. Of all practical EAs, GAs most closely follow biological evolution and are more inspired by it than others. In the classical implementation, GAs are very suitable and even natural for digital computers and the application of genetic operators is simple and straightforward. Moreover, this is the most general evolutionary approach and therefore extremely versatile. John Holland in [97] provided a general framework for GAs and a basic theoretical background, much of which has been discussed in the former sections. There are more recent publications on the basics of GA, for example [44, 51, 80], and the reader is referred to them for greater detail.

The basic algorithm is exactly as in the Section 4.1; however, several variations to this scheme are known. For example, Koza [118] uses separate threads for asexual reproduction, crossover and mutation, chosen at random; therefore, only one of these genetic operators is applied to an individual in each loop, while classical GA applies mutation after crossover independently.

One of the particularities of typical GAs is genome representation. The vast majority of GAs use alphabet-based string-like chromosomes described in Section 4.2.2, although real-coded GAs are gaining wider popularity. Therefore, a suitable mapping from actual problem space parameters to such strings must be designed before a genetic search can be conducted. In many cases, this is not a trivial task: inappropriate encoding of the parameters may hinder the algorithm performance considerably. Apart from the loss of precision and excessive computation time associated with too lengthy representation as discussed in 4.2.2, problems with convergence may arise if the encoding is *deceptive* in terms of building block hypothesis. GA-deceptive (misleading) encoding usually contain isolated optima, i.e. the best points tend to be surrounded by the worst. See [80] for details.

Encoding is not the only source of potential problems. The objective function itself can have the same deceptive properties. However, in most practical cases little is known about the fitness landscape (otherwise more direct and explicit and thus less computationally intensive techniques could be employed), therefore this problem can be attributed to the inherent task difficulties. Nevertheless, if the fitness function is to be designed for a specific engineering task (for example, an estimate of the flight control quality, as will be used in this study later), attention should be paid to avoiding rugged and other GA-difficult fitness landscapes.

One of the common difficulties is associated with different scales along different dimensions for multi-dimensional tasks (each independent parameter is considered as a dimension). In classical GA, crossover and mutation are applied uniformly along the chromosome, and if sensitivity to these operators of one of the independent parameters differs significantly from that of the others, this may have a negative effect on the performance. Therefore, all parameters should be appropriately scaled before encoding, especially if real (floating point) encoding is used. Another solution is selective mutation and crossover for the parameters encoded in the chromosome, an approach common for Evolutionary Strategies (see next section).

Of the two genetic operators, recombination (crossover) plays the most important role in GAs (see 4.2.4). Typical probability of crossover is 0.6 to 0.8 and even up to 1.0 in some applications. On the contrary, mutation (Section 4.2.5) is considered as an auxiliary operator, only to ensure that the variability of the population is preserved. Mutation probabilities range from about 0.001 to 0.03 per bit.

Population size is highly problem dependent; however, typical GAs deal with fairly large or at least moderate population sizes, of the order of 50 to 300 members, although smaller and much larger sizes (up to several thousands) were used (see e.g. [204]). As noted in Section 4.2.6 above, large population size provides the basis for effective crossover operation. See also [57, 81, 82, 207] for discussion about the effect of population size on various GA implementations.

As a rule, GAs employ some form of stochastic selection scheme, in particular fitness proportional selection (4.2.3.2) and tournament selection (4.2.3.4). Ranking (4.2.3.3) is common as well.

Although by far the largest application of GAs is optimisations of different sorts, from simple function optimisations to multi-parameter aerodynamic shape optimisation [145] and optimal control [43], GAs are suitable for many more tasks where great adaptation ability is required, for example, neural networks learning [187] and time series prediction [140]. The potential of GA application is limited virtually only by the ability to develop a suitable encoding.

4.4 Evolutionary Strategies and Evolutionary Programming

Evolutionary Programming was one of the very first evolutionary methods. It was introduced by Lawrence J. Fogel in the early 1960s [70], and the publication [73] by Fogel, Owens and Walsh became a landmark for EP applications. Originally, EP was offered as an attempt to create artificial intelligence. It was accepted that prediction is a keystone to intelligent behaviour, and in [73] EP was used to evolve finite state automata that predict symbol strings generated from Markov processes and non-stationary time series. This was done as follows. The original finite state machines were measured in their ability to predict each next event in their experience with respect to whatever payoff function had been prescribed. Five different kinds of mutation were then applied to the machines to create offspring population: change of an output symbol, change of a state transition, addition of a state, deletion of a state and change of the initial state. The mutations were performed with uniform probability, and the number of mutations for a single offspring was either fixed or also chosen according to a probability distribution. Those finite state machines from both parent and offspring populations that judged to be superior survived to become the new parents [22, 71].

In contrast, Evolutionary Strategies appeared on the scene in an attempt to solve a practical engineering task. In 1963, Ingo Rechenberg and Hans-Paul Schwefel were conducting a series of wind tunnel experiments in Technical University of Berlin trying to optimise aerodynamic shape of a body. This was a laborious intuitive task and the students tried to work strategically. However, simple gradient and coordinate strategies have proven to be unsuccessful, and Rechenberg suggested to try random changes in the parameters defining the shape, following the example of natural mutations and selection [42]. Originally, ES used only one solution as a parent (i.e. population size $N = 1$), and the selection was made between the parent and its offspring (a so called (1+1)-ES strategy). Mutations were performed by adding a normally distributed random value to the parameters. Later this has been extended [186] to $N > 1$ with both mutation and recombination to create more than N offspring. Selection is then used to return the population to N individuals.

As it can be seen, both methods are focusing on behavioural linkage between parents and the offspring rather than seeking to emulate specific genetic operators as observed in nature. In addition, unlike GAs, natural real-value representation is predominantly used. In the present state, EP and ES are very similar, despite their independent development over 30 years (the first communication between the EP and ES groups occurred in early 1992, just prior to the first annual EP conference [72]), and the historical associations to finite state machines or engineering field are no longer valid. In this study, ES approach is employed, so fur-

ther in this section Evolutionary Strategies are described, with special notes when the EP practice is different.

The commonly used notation (μ, λ) indicates that μ parents create $\lambda > \mu$ offspring by means of recombination and mutation, and the best μ individuals are selected from these λ 'children' to replace the parents. The $(\mu+\lambda)$ strategy uses selection from the union of parents and offspring (in this case $\lambda \geq \mu$). It can be seen that the first variant is not *elitist*, because it disregards the parents even if their performance is better than that of the offspring (see Section 4.2.3). However, Schwefel [186] argues that the (μ, λ) strategy is preferred over the $(\mu+\lambda)$, although some experimental findings indicate that the latter performs as well or better than the (μ, λ) strategy in many practical cases [22]. It should be noted that both schemes are the extreme instances of the general (μ, k, λ) strategy, where $1 \leq k \leq \infty$ is the maximum life-span (in generations) of an individual. Thus, the (μ, λ) is effectively the $(\mu, 1, \lambda)$ scheme, and $(\mu+\lambda)$ is (μ, ∞, λ).

In ES, selection is usually performed deterministically: μ best members are selected out of λ (or $\mu+\lambda$) individuals. EP employs tournament selection and prefers $(\mu+\lambda)$ strategy. Apart from tournament size in the latter case (see Section 4.2.3.4), the ratio λ / μ determines selection pressure. Typical population sizes λ of both ES and EP are relatively small in comparison with GAs, often not larger than 10 individuals, although population sizes up to 100 and even more are used as well.

4.4.1 Self-adaptation

One of the most important mechanisms that differs ES from the common GAs is endogenous control on genetic operators (primarily mutation). Mutation is usually performed on real-value parameters by adding zero mean normally distributed random values, as described in Section 4.2.5.2. The variance σ of these values is called *step size* in ES.

The question how to adapt the step size is a milestone of ES research. The adaptation rules used in practice can be divided into two groups: pre-programmed rules and adaptive, or evolved, rules. The pre-programmed rules express a heuristic discovered through extensive experimentation. One of the earliest examples of pre-programmed adaptation is Rechenberg's (1973) 1/5 rule, used in the (1+1)-ES strategy mentioned above. The rule states that the ratio of successful mutations (i.e. the mutations which produce the offspring performing better than the parent) to all mutations should be 1/5 measured over a number of generations. The mutation variance (step size) should increase if the ratio is above 1/5, decrease if it is below and remain constant otherwise. The variance is updated every k generations according to:

$$\sigma^{(g)} = \begin{cases} \sigma^{(g-k)} / c & n_s > 1/5 \\ \sigma^{(g-k)} \cdot c & n_s < 1/5 \\ \sigma^{(g-k)} & n_s = 1/5 \end{cases} \tag{4.16}$$

where n_s is the ratio of successful mutations and $0.817 \le c < 1$ is the adaptation rate. The lower bound $c = 0.817$ has been theoretically derived by Schwefel for the sphere problem [214]. The upper index in parenthesis henceforth denotes the generation number.

4.4.1.1 Mutative strategy parameter control

The idea of self-adaptive (evolved) control is simple and elegant. If we have such a powerful tool as an evolutionary algorithm, why not to let it evolve its own control by itself? In [186] Schwefel proposed to incorporate the parameters that control mutation into the genome. This way, an individual $\boldsymbol{a} = (\boldsymbol{x}, \boldsymbol{\sigma})$ consists of *object variables* (sometimes referred as *describing parameters*) $\boldsymbol{x}$ and *strategy parameters* $\boldsymbol{\sigma}$. The strategy parameters undergo basically the same evolution as object variables: they are mutated and then selected together, though only on the basis of objective performance, on which strategy parameters have indirect influence. The underlying hypothesis in this scheme is that good solutions carry good strategy parameters; hence, evolution discovers good parameters *while* solving the problem.

A simple example of this approach is self-adaptive mutation [21]. In this method, the step sizes are either increased by a certain value, decreased or kept the same, each with a probability of 1/3. In the next generation, on the average, 1/3 of the offspring will be closer to their parents than before, 1/3 will keep progressing at the same speed, and 1/3 will explore further areas. Depending on the fitness landscape, one of these three groups will do better than the others and, therefore, more individuals out of it will be selected to the next generation, where their step sizes are inherited.

Currently, a more generalised and flexible approach is commonly used. In its simplest case, $(1, \lambda)$-ES with one global step size, the mutation step from generation g to $g + 1$ are performed for each offspring $i = 1,\ldots,\lambda$ as [90]:

$$\begin{aligned} \sigma_i^{(g+1)} &= \sigma^{(g)} \exp\left(\tau \cdot \xi_i\right) \\ x_i^{(g+1)} &= \boldsymbol{x}^{(g)} + \sigma_i^{(g+1)} \boldsymbol{z}_i \end{aligned} \tag{4.17}$$

where

ξ_i is independent (for each i) realisations of a zero mean random number. Typically, ξ_i is normally distributed with standard deviation $1/\sqrt{2n}$ (n is the problem dimensionality, i.e. the length of the $\boldsymbol{x}$ vector).

$z_i \sim N(\mathbf{0}, \boldsymbol{I})$ is independent realisations of a normally distributed random vector. The notation $N(m, C)$ denotes the normally distributed random vector with the expectation m and covariance matrix C. In the case of uncorrelated values, the matrix C becomes a diagonal matrix with vector elements variances at the corresponding positions. $\boldsymbol{I}$ is the unity matrix. Therefore, all elements of z_i are independent and (0, 1)-normally distributed.

$\tau \propto \left(\sqrt{2\sqrt{n}}\right)^{-1}$ controls the adaptation rate.

Note that only one (best) individual $\boldsymbol{x}^{(g)}$ is used to create all λ offspring individuals. The exponent in (4.17) simply means that the mutation of the strategy parameters is multiplicative, with equal probability of multiplying and dividing σ by any given positive value. This mutation scheme has shown to be advantageous over the additive approach often used in EP:

$$\sigma_i^{(g+1)} = \sigma^{(g)} \cdot \left(1 + \alpha \cdot N(0,1)\right) \tag{4.18}$$

where $\alpha \approx 0.2$. However, some investigations show that the additive self-adaptation tends to outperform multiplicative mutations when noisy functions are considered [22].

The order of the equations (4.17) is important. The strategy parameters should be mutated first and the mutated values should then be used for mutation of objective variables. The reversed mechanism might suffer from generating offspring that have useful object variable vectors but poor strategy parameter vectors because these have not been used to determine the position of the offspring itself [22].

A few notes should be made about the problem dimensionality n. A great majority of practical tasks are multi-dimensional, often with quite different sensitivity to the parameters changes. When dealing with such problems, it is desirable to assign an *individual step size* to each parameter (dimension). This approach has been introduced by Rechenberg along with (4.17). In the case of individual step size adaptation, the first equation in (4.17) changes to:

$$\sigma_i^{(g+1)} = \sigma_i^{(g)} \exp\left(\tau' \cdot \zeta + \tau \cdot \xi_i\right) \tag{4.19}$$

where $\zeta \sim N(0, 1)$ is sampled once per generation for all individuals, and $\tau' \propto \left(\sqrt{2n}\right)^{-1}$ determines the overall adaptation rate.

The effect of individual step sizes for each dimension is illustrated in Fig. 4.8b (two-dimensional case). Clearly, added flexibility of individual step sizes allows to adapt better to the fitness landscape. However, the problem becomes sensitive to the choice of coordinate system, no invariance against search space rotation is provided. A more elaborate *correlated mutation* scheme ((μ, λ)-CORR-ES) allows for the rotation of hyperellipsoids, as shown in Fig. 4.8c. The rotation angles are usually mutated using additive mechanism (4.18) with the step size $\alpha \approx 0.0873 \approx 5°$ (refer to [20] for details).

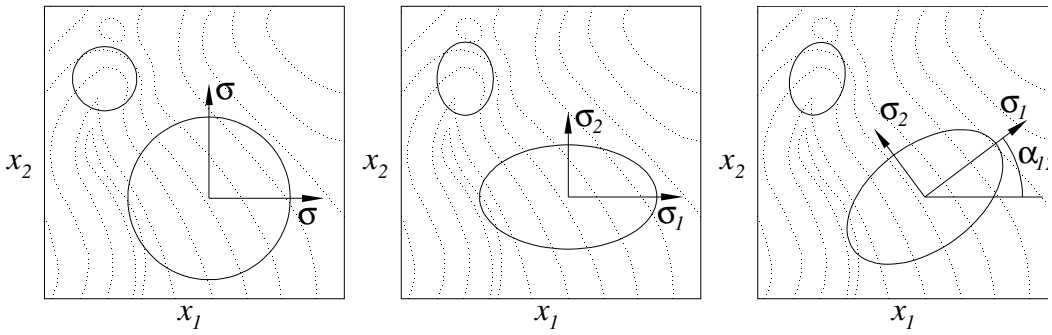


Fig. 4.8 The effect of mutation with self-adaptation of: a) one global step size; b) n individual step sizes; c) individual step sizes with arbitrary rotations ($(n^2 + n)/2$ strategy parameters). The ellipses represent one line of equal probability to place an offspring that is generated by mutation from the parent individual located at the centre of the ellipses. Two sample individuals are shown.

However, added strategy parameters increase the overall dimensionality of the problem. Instead of evolving n objective parameters, the algorithm has to evolve additional 1 (global step size) to $(n^2 + n)/2$ (correlated mutation) parameters. This enormous internal degree of freedom often enhances the global search capabilities of the algorithm at the cost of the expense in computation time, and it also reflects a shift from the precise adaptation of a few strategy parameters (as in case of one step size) to the exploitation of a large *diversity* of strategy parameters [22].

Although a clear understanding of the general advantages of one self-adaptation scheme compared to the other mechanisms is still missing, theoretical work [26] shows that the self-adaptation principle works for a wide variety of different probability density functions used for mutation of the step sizes, i.e. it is an extremely robust mechanism.

4.4.1.2 Derandomised self-adaptation

The growth of the number of dimensions associated with mutative strategy control (individual step size adaptation and correlated mutation) results not only in increase of computational time required to evolve such a large number of parameters. To achieve sensible adaptation, a longer genome requires an appropriate level of genetic diversity, which means a larger population size. Ostermeier et al. [163] empirically states that the population size has to scale linearly with the problem dimension n. When the number of parameters grows as $O(n^2)$, as in the case of correlated mutation, the computation cost may become prohibitively high so that even non-adaptive methods will have advantage.

In an attempt to overcome this limitation but to keep the flexibility of (μ, λ)-CORR-ES strategy, Nikolaus Hansen and Andreas Ostermeier introduced a completely derandomised method called *covariance matrix adaptation* ((μ, λ)-CMA-ES) [89, 90]. The basic idea is to adapt step sizes and covariances in such a way that the longest axis of the hyperellipsoid of

mutation distribution always aligns in the direction of the greatest estimated progress. However, unlike the correlated mutation scheme, all $(n^2 + n)/2$ strategy parameters are calculated deterministically on the basis of accumulated information about former mutation steps and their success (evolution path). The evolution path is searched for correlations between the step sizes and success, and the information is then used to modify σ_i in (4.19) instead of randomising it through ξ. This concept—utilising an evolution path rather than single search steps—is known as *cumulation*.

The cumulative step size adaptation mechanism relies on the assumption that if the mutation strength is below its optimal value, consecutive steps of the strategy tend to be parallel, and if the mutation strength is too high, consecutive steps tend to be antiparallel. For optimally adapted mutation strength, the steps taken by the evolution strategy are uncorrelated [15]. The evolution path $\boldsymbol{p}$ is accumulated as follows:

$$\boldsymbol{p}_i^{(g+1)} = (1-c)\,\boldsymbol{p}_i^{(g)} + \sqrt{c(2-c)}\,\boldsymbol{z}_i^{(g+1)} \tag{4.20}$$

where

$c \in (0; 1]$ determines the lifespan of the information accumulated in $\boldsymbol{p}$. After $\approx 1/c$ generations, the original information in $\boldsymbol{p}$ is reduced by the factor $1/\mathrm{e} \approx 0.37$. If $c = 1$, no cumulation takes place and $\boldsymbol{p}^{(g+1)} = \boldsymbol{z}^{(g+1)}$. $\sqrt{c(2-c)}$ is a normalisation factor to obtain identical variances of $\boldsymbol{p}^{(g+1)}$ and $\boldsymbol{p}^{(g)}$ (note $(1-c)^2 + \left(\sqrt{c(2-c)}\right)^2 = 1$).

$\boldsymbol{z}_i$ is the random vector as used in (4.17) for mutation of objective variables.

$\boldsymbol{p}^{(0)} = \boldsymbol{0}$.

The Hansen and Ostermeier's CMA-ES algorithm uses weighted recombination of μ selected individuals (this is denoted by $(\mu_\mathrm{w}, \lambda)$ notation). The recombination produces a single 'seed' individual

$$\langle \boldsymbol{x} \rangle_\mathrm{w}^{(g)} = \sum_{k=1}^{\mu} w_k \boldsymbol{x}_{k:\lambda}^{(g)} \tag{4.21}$$

to which λ independent random vectors are applied, producing λ offspring individuals for the next generation. This allows to track only one individual, the product of recombination, instead of keeping λ individual paths. The index $k{:}\lambda$ denotes the kth best individual (deterministic selection). The weights w_k are supposed to be normalised so that $\sum_{k=1}^{\mu} w_k = 1$ and (as a rule) should decrease with k ($w_1 > w_2 > \ldots > w_\mu$), although intermediate recombination ($w_1 = w_2 = \ldots = w_\mu = 1/\mu$) is also used. The actual choice of the weighting coefficients is problem dependent; the most popular choice is logarithmic distribution.

Likewise, as the evolution path $\boldsymbol{p}^{(g+1)}$ belongs to this 'virtual' weighted average individual $\langle \boldsymbol{x} \rangle_\mathrm{w}$ (4.21), a similarly defined *mean selected mutation step* is used:

$$\langle \boldsymbol{z} \rangle_{\mathrm{w}}^{(g+1)} = \sum_{k=1}^{\mu} w_k \boldsymbol{z}_{k:\lambda}^{(g+1)} \tag{4.22}$$

where $\boldsymbol{z}_i^{(g+1)}$ is from equation (4.20) and (4.25), the index $k{:}\lambda$ being the index of the kth best individual (as in (4.21)). The weights w_k are identical to those from (4.21).

Adaptation of mutation distribution is done by calculating the covariance matrix $\boldsymbol{C}$ of the mutation distribution. This matrix is used to generate random mutation vectors $\boldsymbol{z}_i$ (as in equation (4.17)); however, instead of a unit normal distribution $N(\boldsymbol{0}, \boldsymbol{I})$, the full covariance matrix is used: $N(\boldsymbol{0}, \boldsymbol{C})$. Nevertheless, like with the global step size adaptation, the overall variance of the mutation distribution σ is adapted separately from the shape of the distribution $N(\boldsymbol{0}, \boldsymbol{C})$, and the matrix $\boldsymbol{C}$ is scaled accordingly. The algorithm authors state two reasons for such separation [90]. First, changes of the overall variance and of the distribution shape should operate on different time scales. Due to the number of parameters to be adapted, the adaptation of the covariance matrix must operate on a time scale of n^2. Adaptation of the overall variance should operate on a time scale n, because the variance should be able to change as fast as required on simple objective functions. Second, if overall variance is not adapted faster than the distribution shape, an initially small σ can jeopardize the search process. The strategy detects a (nearly) linear environment, and adaptation (erroneously) enlarges the variance in one direction.

To facilitate this separation, the matrix $\boldsymbol{C}$ is decomposed into diagonal step size matrix $\boldsymbol{D}$ and orthogonal rotation matrix $\boldsymbol{B}$ so that

$$\boldsymbol{C}^{(g)} = \boldsymbol{B}^{(g)}\boldsymbol{D}^{(g)}\left(\boldsymbol{B}^{(g)}\boldsymbol{D}^{(g)}\right)^{\mathrm{T}} = \boldsymbol{B}^{(g)}\left(\boldsymbol{D}^{(g)}\right)^2\left(\boldsymbol{B}^{(g)}\right)^{\mathrm{T}} \tag{4.23}$$

(singular value decomposition). Columns of $\boldsymbol{B}^{(g)}$ are the normalised eigenvectors of the covariance matrix $\boldsymbol{C}^{(g)}$. Therefore, the distribution $N\left(\boldsymbol{0}, \boldsymbol{C}^{(g)}\right)$ can be obtained as

$$N\left(\boldsymbol{0}, \boldsymbol{C}^{(g)}\right): \ \boldsymbol{B}^{(g)}\boldsymbol{D}^{(g)}\boldsymbol{z}_i^{(g+1)} \tag{4.24}$$

where $\boldsymbol{z}_i \sim N(\boldsymbol{0}, \boldsymbol{I})$ is independent realisations of a normally distributed random vector (see (4.17)). The object parameter mutation is therefore done as follows:

$$\boldsymbol{x}_i^{(g+1)} = \langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(g)} + \sigma^{(g)}\boldsymbol{B}^{(g)}\boldsymbol{D}^{(g)}\boldsymbol{z}_i^{(g+1)} \tag{4.25}$$

for each offspring individual $i = 1,\ldots,\lambda$.

Both $\boldsymbol{C}^{(g)}$ and $\sigma^{(g)}$ are adapted using the evolution path $\boldsymbol{p}^{(g+1)}$ (4.20). However, to achieve a different adaptation rate as discussed above, the random vector $\langle \boldsymbol{z} \rangle_{\mathrm{w}}^{(g+1)}$ is scaled differently and different cumulation rates c are used. For adaptation of the covariance matrix $\boldsymbol{C}$, the *weighted mean selected mutation step* $\boldsymbol{B}^{(g)}\boldsymbol{D}^{(g)}\langle \boldsymbol{z} \rangle_{\mathrm{w}}^{(g+1)}$ is used in place of $\boldsymbol{z}_i^{(g+1)}$ in

(4.20), which has the same distribution as mutation steps omitting the overall step size $\sigma^{(g)}$ (compare (4.25)). For adaptation of the overall step size σ, the *conjugate evolution path* $\boldsymbol{p}_\sigma$ is calculated, where scaling of the mutation step with $\boldsymbol{D}^{(g)}$ is omitted as well (only axis rotation is relevant). Altogether, the transition of evolution path $\boldsymbol{p}_c$ for covariance matrix $\boldsymbol{C}$ reads

$$\boldsymbol{p}_c^{(g+1)} = (1-c_c)\,\boldsymbol{p}_c^{(g)} + c_w\sqrt{c_c\,(2-c_c)}\,\boldsymbol{B}^{(g)}\boldsymbol{D}^{(g)}\,\langle \boldsymbol{z}\rangle_{\mathrm{w}}^{(g+1)} \tag{4.26}$$

and that for the overall step size σ:

$$\boldsymbol{p}_\sigma^{(g+1)} = (1-c_\sigma)\,\boldsymbol{p}_\sigma^{(g)} + c_w\sqrt{c_\sigma\,(2-c_\sigma)}\,\boldsymbol{B}^{(g)}\,\langle \boldsymbol{z}\rangle_{\mathrm{w}}^{(g+1)} \tag{4.27}$$

where:

$c_c \in (0; 1]$ and $c_\sigma \in (0; 1)$ determine the cumulation time for evolution path and conjugate evolution path respectively (see (4.20)).

$c_w = \left(\sqrt{\sum_{k=1}^{\mu} w_k^2}\right)^{-1}$ is chosen so that under random selection[1] $c_w\,\langle \boldsymbol{z}\rangle_{\mathrm{w}}^{(g+1)}$ and $z_i^{(g+1)}$ are identically distributed with the same variance.

With (4.26), the adaptation of the covariance matrix is performed as follows:

$$\boldsymbol{C}^{(g+1)} = (1-c_{\mathrm{cov}})\,\boldsymbol{C}^{(g)} + c_{\mathrm{cov}}\,\boldsymbol{p}_c^{(g+1)}\left(\boldsymbol{p}_c^{(g+1)}\right)^{\mathrm{T}} \tag{4.28}$$

where $c_{\mathrm{cov}} \in [0; 1)$ is the change rate of the covariance matrix $\boldsymbol{C}$. If $c_{\mathrm{cov}} = 0$, no change takes place. Note that $\boldsymbol{p}_c^{(g+1)}\left(\boldsymbol{p}_c^{(g+1)}\right)^{\mathrm{T}}$ is a symmetrical $n \times n$ matrix with rank 1. Initialisation $\boldsymbol{C}^{(0)} = \boldsymbol{I}$.

The adaptation of the overall step size σ, as noted above, is done under the assumption that the optimal evolution steps should be uncorrelated. Therefore, at every generation, the current length of the conjugate evolution path $\left\|\boldsymbol{p}_\sigma^{(g+1)}\right\|$ is compared to the expectation of the length of a $N(\boldsymbol{0}, \boldsymbol{I})$-distributed random vector $\mathrm{E}\left(\left\|N(\boldsymbol{0}, \boldsymbol{I})\right\|\right)$. Under random selection, the actually realised step sizes will approach the latter value. However, if under actual selection the step sizes of 'good' individuals (accumulated in the evolution path) are smaller than expected, the overall step size should also decrease, and vice versa. Thus, the overall step size σ can be adapted as follows:

$$\sigma^{(g+1)} = \sigma^{(g)} \cdot \exp\left(\frac{1}{d_\sigma}\left(\frac{\left\|\boldsymbol{p}_\sigma^{(g+1)}\right\|}{\mathrm{E}\left(\left\|N(0, \boldsymbol{I})\right\|\right)} - 1\right)\right) \tag{4.29}$$

[1] Random selection occurs if the fitness function returns random numbers independent of $\boldsymbol{x}$.

Here $d_\sigma \geq 1$ is the damping constant which determines the change rate of $\sigma^{(g)}$, similarly to τ in (4.17). The expectation in denominator can be approximated as $\mathrm{E}\left(\left\|N(\mathbf{0},\boldsymbol{I})\right\|\right) = \sqrt{2}\cdot\Gamma\left(\frac{n+1}{2}\right)\Big/\Gamma\left(\frac{n}{2}\right) \approx \sqrt{n}\left(1-\frac{1}{4n}+\frac{1}{21n^2}\right) \approx \sqrt{n}+O(1/n)$ [89].

4.4.1.2.1 CMA-ES algorithm outline and parameter settings

Apart from population size λ and number of parents μ, the strategy parameters c_c, c_σ, c_{cov}, d_σ and the weighting coefficients $(w_1,\ldots,w_\mu)$ must be selected. Hansen and Ostermeier [90] suggest the following default setting:

λ	μ	$w_{i=1,\ldots,\mu}$	c_c	c_σ	c_{cov}	d_σ
$4+\lfloor 3\ln(n)\rfloor$	$\lfloor \lambda/2\rfloor$	$\ln\left(\frac{\lambda+1}{2}\right)-\ln(i)$	$\frac{4}{n+4}$	$\frac{4}{n+4}$	$\frac{2}{\left(n+\sqrt{2}\right)^2}$	$c_\sigma^{-1}+1$

Table 4.1: Default parameter settings for the (μ_w, λ)-CMA-ES

With the exception of λ (and in some cases μ), these settings are used in this study. The population size λ is usually chosen to be significantly larger for the objective functions which are affected by noise and/or dynamically change. For greater detail about the influence of the above parameters, refer to the original publication [90].

The initial settings are $\boldsymbol{p}_c^{(0)} = \boldsymbol{p}_\sigma^{(0)} = \mathbf{0}$, $\boldsymbol{C}^{(0)} = \boldsymbol{B}^{(0)} = \boldsymbol{D}^{(0)} = \boldsymbol{I}$; the initial point $\langle \boldsymbol{x} \rangle_{\mathrm{w}}^{(0)}$ can be set to any appropriate value according to the (expected) fitness landscape. Selecting the initial step size $\sigma^{(0)}$ is important; the actual value is purely problem dependent, however, it should not be too small. σ should not tend to increase significantly within the initial $2/c_\sigma$ generations, otherwise the initially learned distribution shape can be inefficient and may have to be unlearned consuming a considerable number of additional function evaluations.

After the initial settings and parameters are defined, the algorithm loop is performed until the termination criteria are met (e.g. number of function evaluations, search stagnation etc.):

1. Generate λ offspring individuals using (4.25);
2. Evaluate these individuals;
3. Select μ best individuals deterministically {this is usually done by sorting the population according to the members fitness};
4. Calculate the evolution path using (4.26);
5. Update the covariance matrix $\boldsymbol{C}$ according to (4.28);
6. Calculate the conjugate evolution path (4.27);
7. Update the overall step size σ (4.29);
8. Recalculate the matrices $\boldsymbol{B}$ and $\boldsymbol{D}$ from $\boldsymbol{C}$ via eigenvectors (4.23);
9. Repeat from step 1.

In addition, certain actions are recommended to increase numerical stability of the algorithm. First, the symmetry of the covariance matrix $\boldsymbol{C}$ should be enforced after its evaluation in step 5 (normally it is a symmetrical matrix; however, the discrepancies between the upper and lower triangles may arise due to round-off errors). Second, a maximum condition number for $\boldsymbol{C}$ of the order of $\sim 10^{14}$ should be ensured. If the ratio between the largest and smallest eigenvalue of $\boldsymbol{C}$ is greater than 10^{14}, the condition number may be limited by the operation $\boldsymbol{C}^{(g)} := \boldsymbol{C}^{(g)} + \left(\max_i \left(d_{ii}^{(g)} \right)^2 \cdot 10^{-14} - \min_i \left(d_{ii}^{(g)} \right)^2 \right) \boldsymbol{I}$ (d_{ii} is the ith diagonal element of the step size matrix $\boldsymbol{D}$). Finally, a minimum variance for the mutation steps should be ensured. If the length of the shortest principal axis of the mutation ellipsoid, $\sigma^{(g)} \cdot \min_i \left(d_{ii}^{(g)} \right)$, becomes smaller than a certain value (which is problem dependent), the axis length should be restricted.

The storage requirements of the algorithm is $O(n^2)$, which, for the practical values of n, is not a disadvantage. However, the computational requirements are $O(n^3)$. This may have a substantial impact on performance, especially noticeable if the objective function is not computationally demanding. To reduce the computational effort of the algorithm, the matrices $\boldsymbol{B}$ and $\boldsymbol{D}$ may be updated not after every generation but after $\sqrt{n}$ or even n/10 generations. This is possible without significant disturbance to the algorithm because change of C is comparatively slow (time scale n^2). This approach reduces the computational requirements from $O(n^3)$ to $O(n^{2.5})$ and $O(n^2)$ respectively.

Concluding this section, it can be said that CMA-ES algorithm is a state-of-the-art method for effective and robust optimisation of complex and non-separable multi-dimensional problems (those which cannot be solved as n one-dimensional problems), when calculation of the objective function derivatives is not possible. This method has invariance properties against linear transformations of the search space and of the objective function value. It is especially useful when large population sizes are not desirable. Testing the algorithm on specially designed test functions as well as on real world problems [90, 159] showed good performance, robustness and explorative properties.

4.5 Genetic Programming

Genetic programming (GP) is an evolutionary machine learning technique. It uses the same paradigm as genetic algorithms and is, in fact, a generalisation of GA approach. GP increases the complexity of the structures undergoing evolution. In GP, these structures represent hierarchical computer programs of varying size and shape.

GP is a fairly recent EA method compared to other techniques discussed before in this chapter. The first experiments with GP were reported by Stephen Smith [199] (1980) and Nichael Cramer [45] (1985). However, the first seminal book to introduce GP as a solid and practical technique is John Koza's 'Genetic Programming' [118], dated 1992.

Common GAs (see Section 4.3) operate with alphabet-based genomes which store a set of parameters being optimised. For fitness evaluation, these parameters are extracted from the genome and are passed to the fitness function as an input. In GP, each individual in a population is a program which is executed in order to obtain its fitness. Thus, the situation is somewhat opposite: the individual is a 'black box' with an arbitrary input and some output. The fitness value (often referred to as *fitness measure* in GP) is usually obtained through comparison of the program's output with the desired output for several input test values (*fitness cases*). However, fitness evaluation in GP is problem dependent and may be carried out in a number of ways. For example, the fitness of a program controlling a robotic animal may be calculated as the number of food pieces collected by the animal minus resources taken for search (e.g. path length). When seeking a function to fit the experimental data, the deviation (maximal one or in the least square sense or whatever) will be the measure of fitness.

One of the characteristics of GP is enormous size of the search space. GP search in the space of possible computer programs, each of which is composed of varying number of functions and terminals (i.e. data—variables, constants and functions without explicit arguments). The functions may be standard arithmetic operations, standard or domain-specific mathematical functions, and programming commands (e.g. flow control operators such as 'if … then'). Likewise, data may be of a range of types, including complex structured ones. It can be seen that the search space is virtually incomprehensible, so that even generation of the initial random population may represent some difficulties. If no bounds are placed on the size of the programs, the search space is infinite. Due to that, GP typically works with very large populations of hundreds and even thousands of members. Two-parent crossover is usually employed as the main genetic operator, while mutation has only a marginal role or is not used at all.

4.5.1 Genome representation in GP and S-expressions

Unlike linear chromosomes in GAs, genomes in GP represent hierarchical, tree-like structures. Any computer program or mathematical expression can be depicted as a tree structure with functions as nodes and terminals as leaves. For example, let us consider an expression for one of the roots of a square equation $a \cdot x^2 + b \cdot x + c = 0$:

$$\frac{-b+\sqrt{b^2-4ac}}{2a}$$

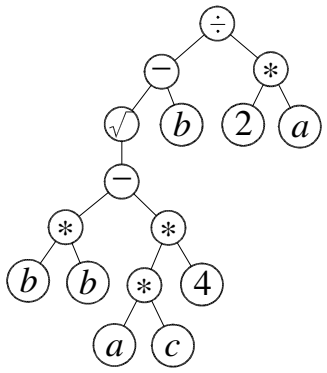


Fig. 4.9 Tree-like representation of an expression

Functions may have different number of arguments (in this example, two for arithmetic operations and one for square root); however, as a rule, they have only one output. By convention, the input arguments are calculated from the first (left) one to the last (right) one. In some cases, sub-trees may not be calculated at all. For example, the conditional operator 'if' takes three arguments, calculates the first one, and if the first argument evaluates to logical True (e.g. non-zero value), the operator calculates and returns the second argument, disregard-

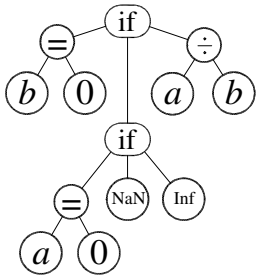


Fig. 4.10 Implementation of safe division a/b

ing the third one; otherwise it returns the third argument. Fig. 4.10 shows how safe division *a/b* may be implemented (providing that symbolic constants 'Inf' (infinity) and 'NaN' (not a number) are defined):

The convenience of such tree-like structures (apart from illustrative purposes) is that they can be easily modified on the sub-tree level. Any sub-tree can be taken out and replaced with another one, preserving syntax validity of the expression. This is how crossover is typically performed (this is discussed later in this section).

However, the trees such as shown in Fig. 4.9 and Fig. 4.10 should be encoded in some computer-readable form for actual GP implementation. This can be done in a number of ways. Some systems (e.g. MATLAB) provide built-in mechanisms for storage and operation on hierarchical structures. If this is not available, string representations are employed. An expres-

sion or a program can be encoded in a common for imperative languages way; for example, the formula for the root of a square equation from Fig. 4.9 can be written as

```
(sqrt(b*b - 4*a*c) - b) / (2*a)
```

Unfortunately, such representation, although being mathematically readable, is inconvenient to handle in a GP way. It has to be parsed to the tree-like form for every operation. Therefore, another syntax is traditionally used.

One can note that in the trees such as the ones above, an operation always precedes its arguments on the branch, e.g. instead of '*a* + *b*' it reads '+ *a b*'. This notation is known as *prefix notation* or *Polish notation*. It is used in the programming language LISP and its derivatives—certainly not the most human-friendly language but very flexible and useful in many areas, GP in particular. LISP has always been a popular tool for automatic program generation, so it has been used from the early stages of GP development [93] as well as by Koza in [118] and it still remains as one of the most popular languages in GP field.

LISP, as the name suggests (LISt Processing), has only one syntactic form, list, or *S*-expression ('symbolic expression'). List is a set of *atoms* (constants, variables, functions) or other lists enclosed in parenthesis. LISP evaluates all S-expressions it sees. An atom is a trivial expression and is evaluated to itself (it if is a variable, it is evaluated to its current value). A list is evaluated by treating the first element of the list as a function, and all the remaining items are passed to the function as arguments. For example, `(+ 1 2)` is a valid S-expression which is evaluated to 3. The example from Fig. 4.9 can now be expressed as

```
(/ (- (sqrt (- (* b b) (* (* a c) 4))) b) (* 2 a))
```

Although it seems longer than the above mathematical representation and not as human-readable, it is clear that it more closely resembles the tree structure from Fig. 4.9. What is important, it is relatively easy to manipulate with such expressions on sub-tree level of any depth: lists can be extracted by matching the corresponding pairs of parenthesis. Another list can be inserted in place of the extracted one or of any argument; moreover, the new list and the list being replaced may have different size and depth.

A tree (and thus its S-expression) has various characteristics which are important for GP. Two of them have already been mentioned: size and depth of the tree. The *size* of a tree is simply the number of its nodes and leaves (i.e. the number of atoms in the corresponding S-expression). The *depth* of a tree is defined as the length of the longest non-backtracking path from the root to an endpoint (calculated as the number of nodes and leaves on the way). The tree in Fig. 4.9 has the size 16 and the depth 7. A *full tree* is the tree in which *all* non-backtracking paths from the root to endpoints are equal in length. In the case of *binary trees* (those composed only of two-input functions), size l and depth d of a full binary tree are re-

lated as $l = 2^d - 1$. A *minimal tree* consists of a single long chain of functions, with all side branches terminating immediately in leaves. The size of a minimal binary tree is $l = 2d - 1$. The *mean depth* of a large randomly sampled population of programs of a given size l is $2\sqrt{\pi(l-1)/2} + O\left(l^{1/4+\varepsilon}\right)$ [68].

Tree shape can have an impact on the potential performance of a solution. Some problems are better solved with long sparse program trees, while other 'prefer' bushier trees [132]. Experiments show [176, 203] that in GP program trees in a population evolve towards shapes of intermediate density. However, they do not converge to the line of mean depth ('ridge line') but spread out like a random cloud [132]. Moreover, solutions generated to the one side of the ridge line (e.g. sparse trees) usually do not tend to cross the line and move to the other side. Thus, the initial population should contain a sufficient proportion of both bushy and sparse trees (see Section 4.5.2 below).

There are extensions to the tree-based GP. Most of them employ decomposition of the programs into sub-trees (modules) and evolving these modules separately. One of the most widely used methods of this kind is Koza's Automatically Defined Functions (ADF) [117]. In ADF approach, the program is split into a main tree and one or more separate trees which take arguments and can be called by the main program or each other. In another approach, code fragments from successful program trees are automatically extracted and are held in a library, and then can be reused in the following generations by any individual via library calls.

However, tree-based GP is not the only option. It is possible to express a program as a linear (though maybe recurrent) sequence of commands. One of the examples of linear GP systems is stack-based GP [168]. In stack-based languages (such as Forth) each program instruction takes its arguments from a stack, performs its calculations and then pushes the result back onto the stack. For example, the sequence `1 2 + .` pushes the constants 1 and 2 onto the stack, then '+' takes these values from the stack, performs the addition and pushes the result 3 back. The final dot extracts and prints out the result. The notation such as '1 2 +' is the opposite to that used in LISP and is called *reverse Polish notation* (or *postfix notation*). Typically, the program inputs are pushed onto the stack before the program is executed and its results are popped from the stack afterwards. Additional checks may be necessary to protect from both stack under and overflow.

Another example of linear encoding is register-based approach. Each instruction has access to a small number of registers, to which input data are supplied before execution. The output data is read in the final contents of one or more registers. A special case of this approach is machine-code GP, where the instructions are real hardware machine instructions and the program is executed directly. This may give a significant increase in execution speed.

4.5.1.1 Function set and terminal set

When designing a GP implementation, proper care should be taken for choosing the function and terminal sets. The function set $F = \{f_1, f_2, \ldots, f_{nf}\}$ is the set of functions from which all the programs are built. Likewise, the terminal set $T = \{a_1, a_2, \ldots, a_{nt}\}$ consists of the variables available for functions (constants are usually added arbitrarily if necessary). In principle, the terminals can be considered as functions with zero arguments and both the sets can be combined in one set of primitives $C = F \cup T$.

The choice of an appropriate set of functions and variables is crucial for successful solution of a particular problem. Of course, this task is highly problem dependent and requires significant insight. In some cases, it is known in advance that a certain set is sufficient to express the solution to the problem at hand. For example, all the function sets {`AND, OR, NOT`}, {`IF, OR, NOT`} and {`NAND`} are sufficient for realising any logical function, although the solutions produced by using them will be very different in shape and character. Similarly, it is known (nowadays) that the mass of a body is sufficient to establish a relationship between acceleration of the body and the force applied. However, in most practical real-world problems the sufficient set of functions and terminals is unknown. In these cases, usually all or most of the available data is supplied to the algorithm or iterative design is employed when additional data and functions are added if the current solution is unsatisfactory. As a result, the set of primitives is often far from the minimal sufficient set.

The effect of adding extraneous functions is complex. On the one hand, an excessive number of primitives may degrade performance of the algorithm, similar to choosing excessive genome length in GA. On the other hand, a particular additional function or variable may dramatically improve performance of both the algorithm and solution for a particular problem. For example, addition of the integral of error $\int(H - H_{set})dt$ as an input to altitude hold autopilot allows to eliminate static error and improve overall performance. Alternatively, the integrator function may be introduced along with both the current altitude reading H and the desired altitude H_{set}. The GP may discover the usefulness of the integration independently.

4.5.2 Initial population

Generation of the initial population in GP is not as straightforward as it usually is in conventional GAs. It has been noted above that the shape of a tree has (statistically) an influence on its evolution and that both sparse and bushy trees should be presented in the initial population. To this end, a so called 'ramped half-and-half' method, suggested by Koza [118], is typically used in GP. In this method, half of the trees in the population are generated as full trees and another half as random trees.

The ‘ramped half-and-half’ method employs two techniques for random tree generation: the ‘full’ method and the ‘grow’ method. Both of them start from choosing one of the functions from the function set *F* at random. It becomes the root of the new tree. Then, for each of the inputs of this function, a new primitive is selected with uniform probability. If the path from the root to the current point is shorter than the specified maximum depth, the new primitive is selected from the function set *F* for the ‘full’ method and from the union set *C* for the ‘grow’ method. If the path length reaches the specified depth, a terminal from the set *T* is selected at random for both methods. The process continues until the tree is complete, i.e. all the inputs are connected.

Koza suggests the maximum depth limit of 6 (equivalent to a maximum tree size of 63). However, in the ‘ramped half-and-half’ method, the current maximum depth parameter ranges linearly between 2 and the maximum allowed depth (i.e. 6). That is, 20% of the trees will have the maximum depth 2, 20% will have depth 3 and so on up to 6. Note that the shape of the trees generated by the ‘grow’ method depends on the relative size of the sets *F* and *T*. The longer the terminal set *T* as compared to the function set *F*, the sparser the trees will be. In contrast, the ‘full’ method always provides full trees of a given size. Thus, the ‘ramped half-and-half’ technique ensures that the initial population contains a wide variety of the trees of different sizes and shapes.

Another point that should be taken into account is the possibility of creating duplicate solutions in the initial population. Unlike traditional GAs, where the individuals are sampled directly from the search space, the trees in GP are built from a relatively small set of primitives and have (initially) a modest size. Therefore, it is not uncommon in GP to find duplicates in the initial population. This is usually not critical for practical implementations; however, the duplicate trees unproductively reduce genetic diversity of the population and thus may be eliminated by simple additional checking.

4.5.3 Genetic operators

4.5.3.1 Crossover

Crossover is usually the most important genetic operator in GP. Its classic variation [118] produces two children trees from two parent trees by exchanging randomly selected sub-trees of each parent (Fig. 4.11). Both parents are selected using one of the stochastic selection methods such as fitness proportional selection (4.2.3.2) and tournament selection (4.2.3.4). The crossover operation begins by choosing, using a uniform probability distribution, one random point in each parent independently to be the crossover point for that parent. The point may be a node as well as a leaf. Then, the sub-trees that have roots at the crossover points are removed from the parents, and the sub-tree from the second parent is inserted in

place of the removed sub-tree of the first parent; the second offspring is produced in a symmetric manner.

In terms of S-expressions, the sub-tree crossover is equivalent to exchanging the sub-lists of the parental lists. Considering the example from Fig. 4.11, the parental solutions are

`(- (* x x) (* (* x y) 2))` and `(/ (+ (* x x) y) 2)`

The sub-lists corresponding to the selected crossover fragments are emphasized. These sub-lists are swapped between the parents and the following offspring are produced:

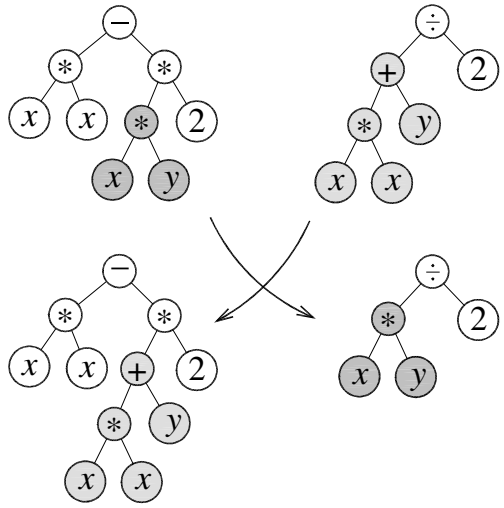


Fig. 4.11 Sub-tree crossover: $x^2 - 2xy$ crossed with $(x^2 + y)/2$ to produce $x^2 - 2(x^2 + y)$ and $xy/2$

`(- (* x x) (* (+ (* x x) y) 2))` and `(/ (* x y) 2)`

It can be noted that in such a simple operation, syntactic validity of the resulting expressions is always preserved. However, this is not enough for error-free program execution. The newly added fragment can potentially return a value which cannot be accepted by some functions, e.g. zero for division or non-positive value for logarithm. A single run-time error of this kind may render the whole program tree unusable, although it may contain very useful code fragments. In order to avoid this situation, all functions used for GP should be protected against incorrect input values. This may be done in several ways. One solution is to map unacceptable values onto domain of the functions, for example, to calculate the square root of the absolute value of the input to avoid negative arguments, or to constrain the denominator by a fairly small value such as 10^{-9}. Another, mathematically more acceptable option, is to assign appropriate symbolic values like 'undefined', 'inf' or 'NaN' whenever the result cannot be calculated, and to rewrite all functions so that they could handle such constants. The example in Fig. 4.10 represent this approach (note that it does not implement *handling* of the

symbolic constants but only returning them). The function which can accept *any* value which may possibly be supplied to it as an input is called *closed* function. The closure property of the function set is an important requirement for GP.

If the crossover points of both parents happen to be at the roots, the parents will simply be reproduced intact. A more interesting consequence may be observed if a parent is mated with itself (or with an identical individual)—a so called 'incest'. Unlike traditional GAs with fixed-length chromosomes where incest results in identical offspring, sub-tree crossover generally produces different individuals, because the two crossover points have generally different locations. This means that even if the population becomes dominated by one particularly fit solution at some stage of evolution (see Sections 4.2.3 and 4.2.3.2 in particular), it will not prematurely converge to this solution.

To avoid excessive growth of the branches of the program trees, a maximum depth value is usually established. If the crossover operation produces an offspring of impermissible depth, this offspring is disregarded and its parent is reproduced as is. Koza [118] uses the default maximum depth of 17. Even such a modest value allows creation of the trees of enormous size, up to $2^{17} - 1 = 131071$ (for binary trees).

4.5.3.2 Other genetic operators

Reproduction as such is simply copying of the selected individual into the next generation population. In classical GP [118], about 10% of the population is selected for simple reproduction and 90% is reproduced through crossover.

Mutation is typically a replacement of a randomly selected sub-tree of an individual with a new randomly generated sub-tree. A special case is *point mutation*, when a single random terminal is inserted in place of a sub-tree. In fact, point mutations sometimes happen during crossover operation. In general, mutation is less needed for GP than for GAs, because crossover alone can reintroduce genetic diversity (see above). In many practical applications, mutation is not used at all.

Permutation is changing of the order of arguments of a randomly selected function. The effect and usefulness of permutation is roughly the same as that of mutation.

Editing is changing the shape and structure of a tree while maintaining its semantic meaning. Usually this implies a mathematical simplification of the expression. For example, the sub-trees which can be immediately evaluated may be replaced with a corresponding terminal, e.g. the expression `(+ 2 3)` may be replaced with `(5)`. An example of mathematical simplification may be taken from Fig. 4.11. The first (left) offspring produced by crossover is $x^2 - 2(x^2 + y)$, which equals to $-x^2 - 2y$ and thus may be encoded simpler. However, the implications of such editing are twofold. Simplification of the solutions may reduce (sometimes

significantly) their length and thus reduce the computation time needed for their evaluation (see also Section 4.5.5) It should be noted that full simplification is a recursive process and by itself may require significant resources. In addition, shorter expressions are less vulnerable to disruption caused by crossover and mutation as there are less chances that they will be torn apart. This may help successful partial solutions to survive in a merciless evolution process. On the other hand, lengthy branches may be useful for GP search as they provide 'material' for construction of the new solutions. In other words, editing may reduce the variety of structures available for recombination. In general, as noted in the beginning of this chapter, parsimony is not the motto of any evolutionary technique and is not pursued by the algorithm.

Nevertheless, editing may be useful for final output of the results, for without it the solution may be incomprehensible for the human and inconvenient to handle (though even full simplification does not guarantee that the solution will be understood).

Some of the other genetic operators practically used in GP can be found in [130].

4.5.4 Convergence in GP

The question of convergence in GP is more complicated than in GAs. Generally, it is assumed in EA theory that the population converges when it contains substantially similar individuals [132]. Unlike conventional GAs, which have one-to-one mapping between genotype and phenotype, this rarely happens in GP. The search space in GP is essentially bigger than the phenotypic search space of the problem at hand. Any solution to the problem may be represented in an infinite number of ways (providing that the program length is unbound). For example, all the individuals $2x$, $2x - x + x$, $2x/1$ have the same phenotypic behaviour, and if one of them represent a suitable solution to the problem, all of them will be the solutions. Moreover, for practical engineering problems, strict mathematical identity is not required and a certain precision allowance is given, so that even $2x + 0.0001$ might be as good solution as the others.

Therefore, the population in GP may contain significantly different individuals (in terms of size and shape) and continue to evolve while yielding practically similar solutions to the problem. Nevertheless, all the individuals, though being distinctively different, have common ancestors and are related. Typically, over a number of generations the GP genotypes concentrate upon just one cluster that maps to the best phenotype. Apart from genetic drift (see 4.2.8), this has another reason. Some genotypes find it easier to resist the disruptive effect of genetic operators, so that their offspring have the same phenotype. Where this phenotype has the highest fitness, these genotypes quickly dominate the others. Thus, the population converges to contain just the descendants of one genotype-phenotype mapping.

However, the genotype cluster does not stabilise and continues to evolve. Since then, most of the highly fit individuals are produced by adding relatively insignificant branches to the successful core that came from the common ancestor. Therefore, each descendant genotype tends to be bigger than its parents. This results in a progressive increase in size known as bloat.

4.5.5 Bloat

The rapid growth of programs produced by GP is known since the beginning [118]. As already noted, this growth need not to be correlated with increase of fitness because it consists of the code that does not change the semantics of the evolving programs. The rate of growth varies depending upon the particular GP paradigm being used, but usually exponential rates are observed [155]. In fact, bloat occurs in most fitness-based search techniques that allow variable length solutions, in particular, in both tree-based and linear GP [132].

There are a number of different theories of bloat which are out of the scope of this research. The most important conclusions are briefly outlined here, and for greater detail the reader is referred to [131] or [132]. Firstly, it should be noted that GP crossover by itself does not change the average program size. Bloat arises from the interaction of genetic operators and selection, i.e. selection pressure is required.

One of the theories is that genomes attract junk code as a protection of the useful code from the effect of crossover. As the crossover point is chosen with a uniform probability, for the genomes with a lot of junk code it is likely that the junk code will be disrupted and not the core of the individual. Since the protective property increases with size and is inherited, there is a continuous tendency to increase the code size. It is interesting to note that the natural genomes found in biological systems appear to have a similar protective mechanism: the useful information is scattered among the large chunks of (apparently) unused code.

Another suggestion is that junk code lies towards the tips of the program trees. When these tips are removed from the successful individuals during crossover operation and are replaced with the branches of (on average) intermediate size, this is more likely to produce a fit offspring than in the case when a large sub-tree is removed.

Clearly, even a linear growth of the genome significantly hampers an extended use of GP. Thus, a number of anti-bloat techniques have been developed. Most of them are effective at preventing programs growing bigger and deeper. However, their effect on the ability of the GP algorithm to successfully evolve useful programs is less clear. Some experiments and trial runs may be required to avoid undesirable side effects.

The most commonly (if not always) used restrictive technique is size and depth limits. Its implementation is already described in Section 4.5.3.1. It should be noted that the actual

experiments (see e.g. [132]) indicate that the populations are quickly affected by even apparently generous limits. Another commonly used approach is to give some preference to smaller solutions. The 'penalty' for excessive length may be included in fitness evaluation. However, in order not to degrade the performance of the algorithm, this component should be small enough so that it would have effect only for the solutions with identical phenotypic performance.

Code editing, already discussed in Section 4.5.3.2, may also be used for fighting bloat, although it generally gives short-term results. Yet another approach is to construct genetic operators which intentionally produce shorter offspring than parents [130].

4.6 Summary

Evolutionary computation is an attractive and quickly developing technique. Although originated in the 1950s and 60s, it gained wide acceptance in the engineering field only since the late 80s. This is associated mostly with the growing computing power available, as the evolutionary methods are computationally expensive. In addition, methodological shortcomings of the early approaches and lack of intercommunication between different EA schools contributed to the fact that evolutionary computation remained relatively unknown to the engineering and scientific audience for almost three decades.

Despite computational demands, evolutionary methods offer many advantages over conventional optimisation and problem solving techniques. The most significant one is flexibility and adaptability of EAs to the task at hand. EAs can be applied to the optimisation of the discontinuous, rugged, non-stationary and noisy functions, with no requirements to calculate the derivatives. Moreover, the characteristics of the problem may be uncertain or unknown at all. Such robustness, as well as good global search characteristics, owes in the first place to the population-based approach to the search. EAs exploit not only the information contained in each individual, but also the information of the population as a whole.

Being population-based algorithms, EAs are easily portable to multi-processor architectures and have good scalability. Another related advantage is the natural ability to work with multi-parameter problems.

It should be kept in mind that EAs are essentially stochastic methods. No result is guaranteed in any particular run, even for the most optimal algorithm settings. Moreover, the traditional engineering approach to seek the solutions which are *correct, consistent, justifiable, certain, orderly, parsimonious* and *decisive* is not applicable to evolutionary computation. Still, formally incorrect, inconsistent, non-parsimonious solutions can be extremely good in terms of practical solving of the problem at hand.

All EAs have two core procedures which enable them to work. First, *selection* ensures survival of the fittest solutions, allowing the population to progress over a number of generations. Second, *reproduction* produces the new solutions from the selected ones, exploring the search space. Each of these operations can be implemented in various ways (Sections 4.2.3 through 4.2.6). Schema theorem and the building block hypothesis (Section 4.2.9) provide the theoretical basis for EAs.

The majority of current implementations of EAs descend from three strongly related but independently developed approaches: Genetic Algorithms, Evolutionary Programming and Evolution Strategies [54].

Genetic Algorithms (Section 4.3) have been introduced by J. Holland in 1975 [97] and subsequently studied by De Jong [52, 54], Goldberg [80] and others. They have been originally proposed as a general model of adaptive processes, although the largest application (for most EAs) remains the function optimisation. In many aspects, they mimic the evolution as observed in nature, including chromosome-like (alphabet-based) genome, large population sizes, stochastic selection methods and sexual mating. GAs are very basic and general, they allows adaptation to virtually any problem. On the downside is the requirement to develop a mapping between the problem domain variables and genome strings, which may be quite sophisticated in some cases.

Evolutionary Strategies and Evolutionary Programming (Section 4.4) originated more from classical numerical methods rather than from inspiration from biology. In the first place, they work with real numbers directly without encoding them into bit strings. As a rule, they use adaptive mutation of the variables without (or with little use of) sexual recombination. They are also characterised by smaller population sizes. However, they may be sensitive to mutation control strategy.

Genetic Programming (Section 4.5), being a significant development of GAs, share much of GAs' properties. In addition, GP is a powerful *machine learning* technique, which evolves solutions to a wide range of problems. Each individual represents a hierarchical program of varying size which may be executed in order to obtain its fitness.

In this study, a combination of the above methods is used to evolve a capable UAV recovery controller. An adapted GP approach is used to represent the control laws. However, these laws are modified more judiciously (yet stochastically) than commonly accepted in GP and evolved in a manner similar to ES approach. This allows to use smaller populations. The next chapter is dedicated to the actual implementation of this method.

Chapter 5. Controller Design

The following two chapters present a methodology which is developed to design a controller that satisfies the objectives of shipboard recovery of a fixed-wing UAV. The methodology itself is comprehensive and should be readily applicable for different types of UAVs and various task objectives. With appropriate modification of control law representation, the methodology can be applied to a broad range of control problems. Development of the recovery controller for the UAV *Ariel* may be considered as a design example to support the methodology.

The controller design process extensively employs simulation of the UAV, ship, environmental and other models described in Chapter 3. The simulations are performed automatically in an integrated environment under the control of evolutionary algorithms outlined in Chapter 4. Due to high demands on computational resources and due to the nature of the evolutionary design, some models are used in a simplified form during the initial stages of the design. Later on, they are replaced with full comprehensive models. This makes the design a multi-stage process which is finalised with thorough testing of the developed controllers and selecting the most suitable, best performing design.

This chapter focuses on adaptation of Evolutionary Algorithms for aircraft control problems. It starts from analysis of typical control laws and control design techniques. The historical review of modern control techniques is based on M. Crump's work [47]. Then, the structure of the UAV controller and the representation of the control laws suitable for evolutionary design are developed. This is followed by the development of the general evolutionary design algorithm, which is then applied to the UAV recovery problem.

5.1 Aircraft flight control

Not unlike the generic control approach, aircraft flight control is built around a feedback concept. Its basic scheme is shown in Fig. 2.13. The controller is fed by the difference ε between the commanded reference signal r and the system output y. It generates the system control inputs u according to one or another algorithm. In general, all these inputs and outputs are vectors (with the dimension of the control input u not necessarily being equal to the di-

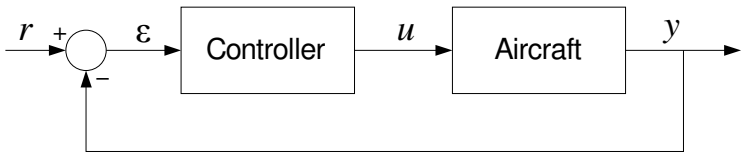


Fig. 5.1 Feedback system (one degree of freedom)

mension of the system output y), and a separate feedback exists for each degree of freedom.

One of the main tasks of flight control as an engineering discipline is design of the controllers which enable a given aircraft to complete a defined mission in the most optimal manner, where optimality is based on mission objective. A number of techniques of producing a required controller have been developed over the last hundred years since the first altitude hold autopilot was introduced by Sperry Gyroscope Company in 1912. Most of the techniques make certain assumptions about the controlled system (i.e. the aircraft), most notably linearity of its dynamics and rigidity of the airframe, to simplify synthesis of the controller. This somewhat limits the applicability and reliability of the resulting controller and makes thorough testing on the real system, which is a required stage of the process, a lengthy and complicated procedure.

To overcome these limitations, there is a need to address controller design from a different perspective, which is attempted in Evolutionary Design methodology. Nevertheless, for adequate analysis an overview of classic and modern control design techniques, as well as of the basics of control, is required. It is presented in this section.

In general, the controller can be mathematically represented as a set of *control laws*

$$\boldsymbol{u}_i = f_i\left(t, \boldsymbol{\varepsilon}_i\right) \tag{5.1}$$

or, for the autonomous controllers that have embedded task or pre-set reference (setpoint),

$$\boldsymbol{u}_i = f_i\left(t, \boldsymbol{y}_i\right) \tag{5.2}$$

where t is current time and $i \in [1; N]$ iterates through all degrees of freedom to be controlled. A large number of the flight controllers, including those considered in this study, is time-invariant so that t can be dropped from the equations.[1] Also, t is omitted as an argument of signals; it is implied that the signals are always functions of time: $\varepsilon(t)$, $u(t)$, $y(t)$, etc.

Both the aircraft and the controller may represent complex dynamic systems. Considering a general dynamic system, it can be found that the *state* of the system is often described by different variables than merely outputs or inputs. A state fully characterises the instantaneous condition of the system and is defined by the values of a set of *state variables* $\boldsymbol{x}$. The minimum number of state variables required is the *order* n of the system. In general, outputs represent some (or all) state variables or quantities derived from them.

The dynamics of such a system can be modelled via an nth order ordinary differential equation (ODE), or more conveniently, n first-order ODEs:

$$\dot{\boldsymbol{x}} = g\left(\boldsymbol{x}, \boldsymbol{y}\right) \tag{5.3}$$

[1] Note that at the same time the dynamics of the controlled system and thus of the system output y may be time-varying, in particular due to unsteady nature of the environment.

The output $\boldsymbol{y}$ is therefore a function of both state and inputs:

$$\boldsymbol{u} = f\left(\boldsymbol{x}, \boldsymbol{y}\right) \tag{5.4}$$

Note that for the sake of consistency these equations are written with respect to the controller, which has $\boldsymbol{y}$ as input and $\boldsymbol{u}$ as output. This would be opposite for the aircraft. Also, for convenience, the dynamics and control are henceforth considered separately for each degree of freedom, thus the subscript i is dropped.

There is a number of uncertainties present in real world systems. One of the complications of real systems is that the system output $\boldsymbol{y}$ is not available for the controller directly. The outputs (or the state), e.g. aircraft position, velocity, etc., can only be measured via respective sensors. This invariably adds instrumental errors and noise to the controller's input, making the feedback signal not equal to the real system output. Moreover, some of the state variables often cannot be measured at all (or with acceptable accuracy) and thus must be, if required, calculated by the controller indirectly using available measurements and knowledge about the system. Sometimes this is not possible and these state variables are *unobservable*. On the other hand, there may be additional measurements which are not directly controlled but which can help, if included in the control law, to improve stability and performance of the controlled system. For example, angular rates measurements can greatly enhance attitude control capability. To accommodate this as an integral part of the controller's logics, the control laws can be rewritten as

$$\boldsymbol{u} = f\left(\boldsymbol{x}, \boldsymbol{z}\right) \tag{5.5}$$

where $\boldsymbol{x}$ is the internal state of the controller and $\boldsymbol{z}$ is the vector of available measurements. These measurements include errors and noise of the sensors as well as the dynamic response of the system to both inputs and environmental disturbances.

An aircraft[1] generally has a very limited number of (motion) control inputs u_i, such as ailerons, elevator and rudder deflections and throttle setting—typically even less than the number of degrees of freedom. Only the respective number of system outputs (if any) can be controlled directly, for example, bank angle. Others are controlled indirectly through available inputs. This can significantly complicate the structure of the controller, especially if the objective output and its derivative cannot be controlled directly.

[1] By 'aircraft' or 'UAV', in this chapter, is understood the whole vehicle together with control actuators and sensors—everything except for the controller itself.

5.1.1 Types of feedback control

5.1.1.1 On-off control

The simplest feedback control is the *on-off control*, also referred to among engineers as *bang-bang control*. Perhaps the most common example of such control is a simple thermostat as found in home refrigerators and irons that switches the heating (or cooling) element on or off as the temperature passes through a defined threshold. This control law can be expressed as follows (for simplicity of illustrations, all control laws hereafter are written in scalar form suitable for single input single output (SISO) systems unless otherwise stated):

$$u = u_0 + \begin{cases} k, & z < a \\ 0, & z \geq a \end{cases} \tag{5.6}$$

where u_0 and k are arbitrary offset and gain suitable for the given system, and a is the threshold (set point). It can be extended to a multiple choice situation.

This law is not particularly suitable for direct flight control in normal situations. Indeed, with only two (or several) discrete control input levels, the system output will tend to oscillate about the set point, no matter how well damped the system is, because the control signal will not switch until the set point is already passed. Moreover, if the dynamics of the system is fast or a significant amount of noise in the system output is present, the controller will be switching also fast ('hunting'), possibly causing extensive wear to the control actuators. To prevent such behaviour, a special 'dead zone' (or 'deadband') is established around the set point where no switching occurs. Obviously, this reduces the accuracy of control.

However, the on-off control comes into play when event handling is required. A classic example of application of such control for aircraft is stall recovery. When angle of attack exceeds a specified value, a nose-down elevator command is issued. A similar logic can be implemented for overload protection or ground collision avoidance.

Another important area in the aerospace field where on-off rules can be successfully applied is internal switching between the controllers (or controller parameters). This approach is known as *Gain scheduling* [180, 181]. The technique takes advantage of a set of relatively simple controllers optimised for different points of the flight envelope (or other conditions). However, it was found that rapid switching (which may happen, in particular, for the same reasons as described above) may cause stability and robustness problems [189]. One of the popular simple solutions is to 'blend' (interpolate) the output of two or more controllers, which effectively turns simple on-off switching into a complicated control method.

5.1.1.2 PID control

The proportional-integral-derivative (PID) control is probably the most widely used type of control, thanks to the simplicity of its formulation and in most cases, predictable characteristics. A PID controller can be easily implemented in mechanical, hydraulic or pneumatic form as well as an electronic device. In a closed-loop system like that in Fig. 2.13, its control law is

$$u = K_{\mathrm{P}}\varepsilon + K_{\mathrm{I}}\int \varepsilon dt + K_{\mathrm{D}}\dot{\varepsilon} \tag{5.7}$$

where the parameters K_{P}, K_{I} and K_{D} are coefficients for proportional, integral and derivative components of the input error signal ε respectively. By adjusting these three parameters, the desired closed-loop dynamics can be obtained.

In essence, PID control uses the information from the present, past and future of the input signal. While the proportional term handles the current value of the input, the integral term accumulates the input over a period of time, allowing to eliminate the static error and increasing long-term precision. At the same time, the derivative component controls the response to a change in the system: its output is proportional to the slope of the input signal, thus projecting the current dynamics into the future. If the integral and/or differential coefficients are set to zero, the controller becomes a PD, PI or P-controller. In fact, stability can be ensured using only the proportional term; however, adding the integral and differential terms may improve quality of control, increasing accuracy and reducing overshoot and oscillations.

Probably the most problematic issue with the PID control is due to the differential term. While being important for good response time and high-speed dynamics, the differential component keenly suffers from both instrumental noise and sampling (calculation) errors. Indeed, even a small amount of noise can greatly affect the slope of the input signal. At the same time, numerical calculation of the derivative (for a digital controller in particular) must be done with a fairly small time step to obtain correct slope at a given point. Although low-pass filtering applied to the input signal smoothens the signal, it severely compromises the usefulness of the derivative term because the low-pass filter and derivative control effectively cancel each other out. Apart from using low-noise sensors (expensive, if available at all), an effective solution is to employ a separate sensor for measuring the derivative directly, if physically possible, instead of calculating it by the difference between the last and the current samples of the input. For example, when controlling the aircraft roll, the roll rate can be provided from a roll rate gyroscope, which delivers much more accurate and noise free readings than the numerical differentiation of the vertical gyro (angular positioning) output.

In contrast, the integral term averages its input, which tends to eliminate noise. However, a common problem associated with the integral control owes exactly to its ‘memory’

and is known as 'integral windup.' When a large input value persists over a significant amount of time, the integral term becomes also large and remains large even after the input diminishes. This causes a significant overshoot to the opposite values and the process continues. In general, integral control has a negative impact on stability and care must be taken when adjusting the integral coefficient. Limiting the integrator state is a common aid for the windup. A more elegant approach involves 'washing out' the integrator state by incorporating a simple gain feedback, effectively implementing a low-pass filter to the input signal.

PID control found wide application in the aerospace field, especially where near-linear behaviour takes place, for example, in various hold and tracking autopilots such as attitude hold and flight path following. The techniques of selecting the optimal PID coefficients are well established and widely accepted. Typically, they use the frequency domain as a design region and utilise *phase margins* and *gain margins* to illustrate robustness. However, design of a PID controller for nonlinear or multi-input multi-output (MIMO) systems where significant coupling between the different system inputs and outputs exists is complicated. As the aircraft control objectives evolved, new design and control techniques were developing. Although many of them essentially represent an elaborated version of PID control, they are outlined in the following sections.

5.1.1.3 Linear optimal control

The concept of optimality in mathematics refers to minimisation of a certain problem-dependent *cost functional*:

$$J = \int_0^T g\left(t, x, u\right) dt + h\left(x\left(T\right)\right) \tag{5.8}$$

where T is the final time, u is the control inputs and x is the state of the controlled system. The last term represents the final cost that depends on the state in which the system ends up. Optimal control is, therefore, finding the control law $u(t)$ that minimises J. In general, for an arbitrary system and cost functional, only numerical search can find the optimal (or near optimal) solution. However, for certain classes of systems (especially linear systems) and cost functionals, more effective design techniques can be devised.

A linear system is commonly written in a state-space form such as

$$\begin{aligned} \dot{\boldsymbol{x}} &= A\boldsymbol{x} + B\boldsymbol{u} \\ \boldsymbol{y} &= C\boldsymbol{x} + D\boldsymbol{u} \end{aligned} \tag{5.9}$$

where $\boldsymbol{x}$ is size n state vector, $\boldsymbol{u}$ is size m vector of control inputs, $\boldsymbol{y}$ is size r vector of outputs, and the matrices have the following dimensions: $\dim(A) = n \times n$, $\dim(B) = n \times m$, $\dim(C) = r \times n$, $\dim(D) = r \times m$. In many physically realisable systems there is no 'direct feedthrough'

term ($D = \mathbf{0}$), which is often assumed in control applications. The state-space representation is especially convenient for linear MIMO systems; the SISO case is obtained by setting $r = m = 1$. Note that the number of states n can be greater than 1 even for SISO systems. In future, it is implied that x, u and y can be vectors and thus the semi-bold typeset is omitted.

One of the first types of linear optimal control studied is *Linear quadratic control*. The *Linear Quadratic Regulator (LQR)* should control the system so that it minimises the cost

$$J = \int_0^\infty \left(x^{\mathrm{T}} Q x + u^{\mathrm{T}} R u \right) dt \tag{5.10}$$

where $Q = Q^{\mathrm{T}}$ and $R = R^{\mathrm{T}}$ are weighting matrices. This functional penalises the control energy (in quadratic form) as well as the amount of time it takes the system to reach zero state. In essence, the LQR is no more than a state feedback matrix gain of the form

$$u = -K_{\mathrm{c}} x \tag{5.11}$$

where K_{c} is a properly dimensionalised matrix (which may be time-varying), obtained as a solution of the algebraic Riccati equation. It can be noted that the LQR, as well as the cost functional, is expressed in terms of state variables instead of system outputs. As some of the state variables may not be directly observable through system outputs, some sort of state estimation may be needed. This problem has been elegantly solved by Rudolf Kalman in 1960 [110]. He introduced the concept of *internal state*, implying that one is concerned in the internal system behaviour as well as in the system input/output behaviour. He also developed an optimal filtering technique, commonly known today as the *Kalman filter*. This filter allows to estimate the system state on the basis of the measurements of the system outputs (possibly incomplete and noisy) and the knowledge of system dynamics.

One of the most widely used control design methods that utilise both LQR and Kalman filter is known as *Linear Quadratic Gaussian with Loop Transfer Recovery (LQG/LTR)*. The LTR enhancement attempts to automatically synthesise an appropriate Kalman filter given a regulator or vice versa. This technique generally almost fully recovers the desired properties of the full state feedback regulator.

An important aspect of Kalman's work is that it is a discrete time domain approach, extremely suitable for digital computer implementation. It uses linear algebra, which allows to handle MIMO systems with no extra cost (apart from computing power). The theory and application of these control techniques and Kalman filtering is detailed in many common control and signal processing textbooks, such as [14, 35, 36, 128, 139].

Being a powerful tool for control applications, the LQG/LTR method (as well as Kalman filtering in general) has several principal limitations. First, this is a linear technique, which application to nonlinear systems is difficult. The Kalman filtering is based upon the

premise of normal distribution of the measurement signals and the state variables under the influence of process and sensor noise. However, when the noise undergoes nonlinear transformation within the system, its distribution is no longer normal. A complete description of the conditional probability distribution (or density in the continuous case), which is necessary for optimal nonlinear filtering, requires a potentially unbounded number of parameters [126]. Sub-optimal solutions can be obtained by approximating the distribution [107, 202].

Probably the most widely used estimator for nonlinear systems is the extended Kalman filter (EKF) [201]. The EKF simply linearises the model of the system so that the basic linear Kalman filter equations can be applied. However, apart from certain difficulties in application, linearisation can produce a highly unstable filter if the assumption of local linearity is violated. To this end, a number of dedicated nonlinear filters that have similar properties to the Kalman filter have been proposed [108, 156, 157].

The LQG/LTR design has been attempted to produce a launch controller for *Ariel* UAV [47]. It was found that the technique is 'simple to use but hard to master,' the main difficulties being associated with finding the right combination of weighting matrices in order to produce a desired closed loop performance. Despite the effort, the controller lacked robustness to the changing aircraft dynamics and a large drop in performance was incurred.

Indeed, robustness is a challenging issue of modern control designs. The Kalman filter relies on the model of the system (in its predicting part), and the guaranteed performance of the controller can easily be lost when unmodelled dynamics, disturbances and measurement noise are introduced. Also, the well known in classic linear design phase and gain margin concepts cannot be easily applied to the multivariable systems that modern control is so suited for [47]. These problems led to the introduction of robust modern control theory.

5.1.1.4 Robust modern control

Unlike traditional optimal control, robust optimal control minimises the influence of various types of uncertainties in addition to (or even instead of) performance and control energy optimisation. Generally, this implies design of a possibly *low gain* controller with reduced sensitivity to input changes. As a consequence, robust controllers often tend to be conservative and 'sluggish.' On the other hand, they may be thought as the stabilising controllers for the whole set of plants (which include a range of uncertainties) and not only for the modelled system, which is a more demanding task.

The majority of modern robust control techniques have origins in the classical frequency domain methods. The key modification of the classic methods is shifting from eigenvalues to singular values (of the transfer function that describes the system), the singular value Bode plot being the major indicator of multivariable feedback system performance [61].

Probably the most popular modern robust control design techniques (particularly in the aerospace field) are H_2 and H_∞ control,[1] also known as the frequency-weighted LQG synthesis and the small gain problem respectively. They are similarly formulated in terms of the *norm* of the transfer function G of the closed-loop system. The H_2-norm of a Laplace transform matrix $G(s)$ is defined as

$$\|G\|_2 = \sqrt{\int_0^\infty \sum_i \left(\sigma_i\left(G(j\omega)\right)\right)^2 d\omega} \tag{5.12}$$

and the H_∞-norm as

$$\|G\|_\infty = \sup_\omega \bar{\sigma}\left(G(j\omega)\right) \tag{5.13}$$

where σ_i are singular values and $\bar{\sigma}$ is the greatest singular value of a given matrix (the function *sup* denotes the least upper bound [supremum] of a given set). The H_∞-norm can be interpreted as the peak value of the maximum singular value of the transfer function over the entire frequency range. In the SISO case, this is simply the peak value of the frequency response of that transfer function. In time domain, this translates to the maximum input to output gain of the system over an infinite time range, i.e. the maximum energy of the input compared to the energy of the output.

The robustness criterion is satisfied if $\|G\| < 1$ [143]. The optimal robust control (H_2 and H_∞) is formulated as finding the stabilising controller that minimises the respective norm. The H_2 design is an extension of the LQG technique outlined above. It is an iterative process that always delivers a stable controller. The H_∞ design is a relatively simple one-step procedure which allows to use design experience similarly to the classic design methods (the lack of engineering intuition in control designs where all problems are solved through matrix equations often upsets engineers). Both H_2 and H_∞ designs are often used together, with the H_2 synthesis being used at the first stage to determine what level of performance is achievable. Then, the H_∞ design framework is employed based on the outcome of the initial H_2 design. These techniques and the underlying theories are thoroughly described in several works, notably [60, 127, 143, 231].

The H_∞ control design found extensive use for aircraft flight control. One of the first such applications was the development of controllers for the longitudinal control of a Harrier jump jet [101, 102, 103, 104]. This work progresses from early paper designs through pilot-in-the-loop simulations to experimental flight status aboard a DRA Harrier. This work has been subsequently extended in [172] to fully integrated longitudinal, lateral and propulsive

[1] 'H' here stands for Hardy space and should be correctly typed in a special typeface; however, for technical reasons it is often substituted with semi-bold or italic H.

control. Other works include [111], where a lateral autopilot for a large civil aircraft is designed, and [112], where a robust longitudinal controller subject to aircraft weight and c.g. uncertainty is demonstrated.

A mixed H_2 / H_∞ approach is applied in [190] to design an autoland controller for a large commercial aircraft. The method employed here utilises the H_2 controller for slow trajectory tracking and the H_∞ controller for fast dynamic robustness and disturbance rejection. In [219], controllers generated using H_∞ methodologies are compared to a LQG/LTR controller, concluding that these two methods both produce controllers with similar performance.

Several H_∞ controllers have been tried to accomplish the UAV shipboard launch task [47]. It has been found that these controllers perform quite well in nominal situations. However, in the presence of large disturbances which place the aircraft well beyond its linear design operating point, the controllers performed poorly (sometimes extremely). At the same time, inability to include even simple static nonlinearities such as time delays and saturations made it difficult to synthesise a practical controller for this task within linear approach. Another deficiency found is common to all frequency domain techniques: the frequency domain performance specifications cannot be rigidly translated into time and spatial domain specifications. Meanwhile, time and especially spatial (trajectory) constrains are crucial for both launch and recovery tasks.

The time domain performance can be accounted for directly in the time domain l_1 design [28, 48, 49].->

It is often extended to include the frequency domain objectives, resulting in a mixed norm approach (particularly H_2 / l_1 design, detailed in [12, 13, 88]). However, l_1 design is plagued by the excessive order of the generated controllers. This is usually solved by reducing the problem to suboptimal control, imposing several restrictions on the system and performance specifications [37]. The application of l_1 approach to flight control is discussed in [198], with the conclusion that controllers with excessive order will generally be produced when using practical constraints.

There have been attempts to solve the H_∞ optimal control problems for nonlinear systems. However, these methods usually rely on very limiting assumptions about the model, uncertainties and disturbance structure. The mathematical development of nonlinear H_∞ control can be found in [50]. An example of nonlinear control of an agile missile is given in [223]. Despite a very complicated solution algorithm, this work is limited by a linear assumption on the vehicle aerodynamics, reducing the benefits gained from the use of nonlinear control. An analytical solution to the nonlinear H_∞ control problem is presented in [227, 228]; however, these works solve the problem considering the system inputs as the aerodynamic forces, even though their generation is a challenging nonlinear control problem in itself.

5.1.1.5 Nonlinear control

Linear control design techniques have been used for flight control problems for many years. One of the reasons why aircraft can be controlled quite well by linear controllers is that they behave almost linearly through most of their flight envelope. However, when the aircraft is required to pass through a highly nonlinear dynamic region or when other complicated control objectives are set, it has been found by several researchers that it is difficult to obtain practical controllers based on linear design techniques. The UAV shipboard launch and recovery tasks are substantially nonlinear problems. The sources of nonlinearities may be the aerodynamic forces generated at low airspeeds and high angles of attack (especially when wind disturbances are present); trajectory constraints imposed due to proximity of ground (water) and ship installations; kinematic nonlinearities when active manoeuvring is required; actuator saturations and some more.

In contrast to the linear systems, the characteristics of the nonlinear systems are not simply classified and there are no general methods comparable in power to those of linear analysis. Nonlinear techniques are quite often designed for individual cases, regularly with no mathematical justification and no clear idea of re-applicability of the methods. It also becomes difficult to assess the robustness of nonlinear designs [47]. Some of the more popular nonlinear control techniques are covered in textbooks [18, 84, 105].

The most basic nonlinear control law, the On-off control, has been described in Section 5.1.1.1 above. Gain scheduling is also mentioned there, noting that these controllers often lack robustness when the controllers are scheduled rapidly. Ensuring full-range robustness is a complicated task within this approach. As the recovery (as well as the launch) procedure is relatively short and there may be fast dynamic changes (particularly due to large scale turbulence), gain scheduling is not used in this work.

Another modern control technique remotely related to the on-off control is *variable structure* (also known as *sliding mode*) control [59, 215]. In this approach, a hypersurface (in state space) called *sliding surface* or *switching surface* is selected so that the system trajectory exhibits desirable behaviour when confined to this hypersurface. Depending on whether the current state is above or below the sliding surface, a different control gain is applied. Unlike gain scheduling, the method involves high speed switching to keep the system on the sliding surface. Examples of the variable structure control application to flight control problem can be found in [194, 195, 196]. The performance of the sliding mode and H_∞ controllers is compared in [138], where depth and heading control of an autonomous submersible is considered. The model included significant nonlinear dynamic effects and uncertainties. The work found that the H_∞ techniques were slightly more robust but significantly more complicated to use,

the main strength of the sliding mode control being the ability to accommodate known state dependent nonlinearities.

An inherent shortcoming of the sliding mode control is the requirement of full state information. If the feedback information is limited to a several measurable output signals, some form of observer is needed, which in itself may be a difficult problem for nonlinear systems.

A completely different approach is to enable applicability of the well known linear control methods to control nonlinear systems. This can be achieved using *nonlinear dynamic inversion*. This process, also known as *feedback linearisation*, involves online approximate linearisation of a nonlinear plant via feedback. Dynamic inversion gained particular attention in aviation industry in the late 1980s and 90s, aiming to control high performance fighters during high angle of attack manoeuvres (known as *supermanoeuvres*). One of the early applications is NASA High Angle of Attack Vehicle (HARV) [39]. In this work, quite good simulation performance results are obtained; however, with the inversion based on the same simulation model, any possible discrepancies are transferred into the controller, leading to questionable results in physical implementation with respect to incorrectly modelled dynamics.

Other notable works include [129, 200]. The latter work compares the dynamic inversion control laws with gain scheduled control laws, finding that the first deliver much better performance. In [185], however, a dynamic inversion controller is compared with a gain scheduled H_∞ controller, concluding that the H_∞ controller has better robustness to both measurement noise and model uncertainty while yielding similar response time. An example of nonlinear dynamic inversion for the control of highly manoeuvrable towed targets, applied within the *decoupling design* framework, is given in [29, 30]. The work showed successful trajectory tracking.

One problem of dynamic inversion is that of non-minimum phase zeros, which when inverted become unstable poles. This is partly addressed in [11] by separating the minimum and non-minimum phase dynamics and inverting only the minimum phase part. The stability and robustness of nonlinear inversion is examined in [151] with an analytical approach showing promising results, although on simplified models of aircraft and controllers.

5.1.1.6 Intelligent control

Intelligent control is a general and somewhat bold term that describes a diverse collection of relatively novel and non-traditional control techniques based on the so called *soft computing* approach. These include neural networks, fuzzy logic, adaptive control, genetic algorithms and several others. Often they are combined with each other as well as with more traditional methods; for example, fuzzy logic controller parameters being optimised using genetic algorithms or a neural network driving a traditional linear controller.

Neural network (NN), very basically, is a network of simple nonlinear processing elements (neurons) which can exhibit complex global behaviour determined by element parameters and the connections between the processing elements. The use of artificial neural networks for control problems receives an increased attention over the last two decades. It has been shown that a certain class of NN can approximate any continuous nonlinear function with any desired accuracy [208]. This property allows to employ NN for *system identification* purposes, which can be performed both offline and online. The result of system identification can be immediately used in *feedback linearisation* control as described earlier. This approach is employed in [133] for a helicopter trajectory tracking problem. Good performance results were obtained despite a large amount of uncertainties in modelled helicopter dynamics. A similar approach is used in [41, 115] for flight control of a tilt-rotor aircraft and the F-18 fighter.

Another area of intensive application of NN is fault tolerant control, made possible due to online adaptation capability of NN. In the recent work [166], an 'add-on' neural controller is developed to increase auto-landing capabilities of a damaged aircraft with one of two stuck control surfaces. A significant increase in successful landings rate is shown, especially when the control surfaces are stuck at large deflections.

Fuzzy logic control also gained some popularity among flight control engineers. This type of control relies on approximate reasoning based on a set of rules where intermediate positions between 'false' and 'truth' are possible. Fuzzy logic control may be especially useful when a decision must be made between several controversial conditions. For example, in [67] fuzzy logic is used for windshear recovery in order to decide whether energy should be given to altitude or airspeed, based upon the current situation.

An outer loop fuzzy logic controller for automatic carrier landing of the F/A-18 aircraft is presented in [209]. Although only a slight increase in terms of performance is achieved, the controller does exhibit 'intelligent' behaviour. Ervin [63] introduces a simple conceptual fuzzy logic controller for small model aircraft. In this work, the fuzzy rules and membership functions (that define the degree of truth as a function of an input) are developed on the basis of interviews with an expert pilot and then refined using the experience gained during the simulation tests. For a simple decoupled four-channel controller this may be appropriate. However, for more complex problems, optimisation of the membership functions according to given performance requirements represents significant difficulties. Such optimisation is often carried out using traditional as well as soft computing nonlinear multi-dimensional optimisation techniques. For example, in the set of works [31, 32], the membership functions of a fuzzy logic guidance controller are optimised using genetic algorithms with successful results.

A mixed intelligent control design is used in [230], where a neural network is used to obtain dynamical model of a helicopter using flight data and a combined controller is developed utilising a genetic-optimised PID controller and a human-defined fuzzy logic controller, showing good altitude control performance. The neural and fuzzy logic controllers are compared to an LQR controller for the helicopter station keeping (hovering) task in [147]. All types of controllers showed the ability to stabilise a naturally unstable (at this regime) helicopter even in strong turbulence conditions, with the LQR controller exhibiting slightly better time response. It is also noted that the manual design of the fuzzy set requires a good understanding of the system to be controlled.

Adaptive control term covers a set of various control techniques that are capable of online adaptation. A good survey of adaptive control methods is given in [17]. The applications of adaptive control is generally biased towards control for large time scales so that the controller has sufficient time to learn how to behave. This makes the relatively short-time recovery process unsuitable for online adaptation.

Evolutionary and *genetic algorithms* (EAs, GAs) are global optimisation techniques applicable to a broad area of engineering problems. They can be used to optimise the parameters of various control systems, from simple PID controllers [230] to fuzzy logic and neural network driven controllers [31, 109]. Another common design approach is evolutionary optimisation of trajectories, accompanied by a suitable tracking controller (e.g. [220]). An elaborated study of applications of EAs to control and system identification problems can be found in [214].

Unlike the majority of other techniques, Genetic Algorithms (in the form of *Genetic Programming*) are able to evolve not only the parameters, but also the *structure* of the controller. In one of the techniques, genetic programming is used to evolve so called *block structure controllers* [122, 124]. Block structure controllers consist of a network of modules, each representing a certain feature such as time delay, integrator, linear relationship, etc. The controller is 'wired' in a diagram that resembles electronic circuits.

In general, EAs require substantial computational power and thus are more suitable for offline optimisation. However, online evolutionary-based controllers have also been successfully designed and used. The *model predictive control* is typically employed for this purpose, where the controller constantly evolves (or refines) control laws using an integrated simulation model of the controlled system. A comprehensive description of this approach is given in [161].

For detailed description of EAs, please refer to Chapter 4.

5.1.2 Flight control for the UAV recovery task

Before entering into the discussion about particular aspects of UAV shipboard recovery problem, it may be useful to overview some of the traditional techniques of aircraft control during landing and recovery.

Aircraft control at this stage of flight (as well as at most stages) can be conventionally separated into two closely related, but distinctive tasks: guidance and flight control. Guidance is the high-level ('outer loop') control intended to accomplish a defined mission. This may be path following, target tracking, various navigation tasks, etc. Flight control is aimed at providing the most suitable conditions for guidance by maintaining a range of flight parameters at their optimal levels and delivering the best possible handling characteristics. For example, it may be damping of unwanted oscillations, speed control, sideslip control and many more. In the following sections, both these tasks are considered in the context of traditional landing technique.

5.1.2.1 Traditional landing

In a typical landing procedure, the aircraft follows a defined glide path. The current position of the aircraft with respect to the glidepath is measured in a variety of ways, ranging from pilot's eyesight to automatic radio equipment such as the Instrument Landing System (ILS). Basically, the objective of the pilot (or autopilot) is to keep the aircraft on the glidepath, removing any positioning error caused by disturbances and aircraft's dynamics. This stage of landing is known as *approach*. Approach may be divided into *initial approach*, in which the aircraft makes contact with ('fixes') the approach navigation system (or just makes visual contact with the runway) and aligns with the runway; and *final approach*, when the aircraft descends along a (usually) straight line. In conventional landing, final approach is followed by a flare manoeuvre or nose-up input to soften the touchdown; however, flare is typically not performed for shipboard landing due to random ship motion and severe constraints on the landing space.

As an example, the scheme of the ILS is illustrated in Fig. 5.2. Two aerials emit specially modulated signals that the aircraft can use to determine its displacement relative to the required glidepath. The glidepath is formed by the intersection of the equisignal zones of either aerial; the glideslope is usually 2.5 to 3.5 degrees and is set according to the requirements of the airport. Both vertical and horizontal displacements (lineup errors) are indicated to the pilot and can be used by autopilot for automatic landing. The ILS electronic equipment also measures rates of change of the errors (drift) and may also calculate accelerations to aid stabilisation on the glidepath.

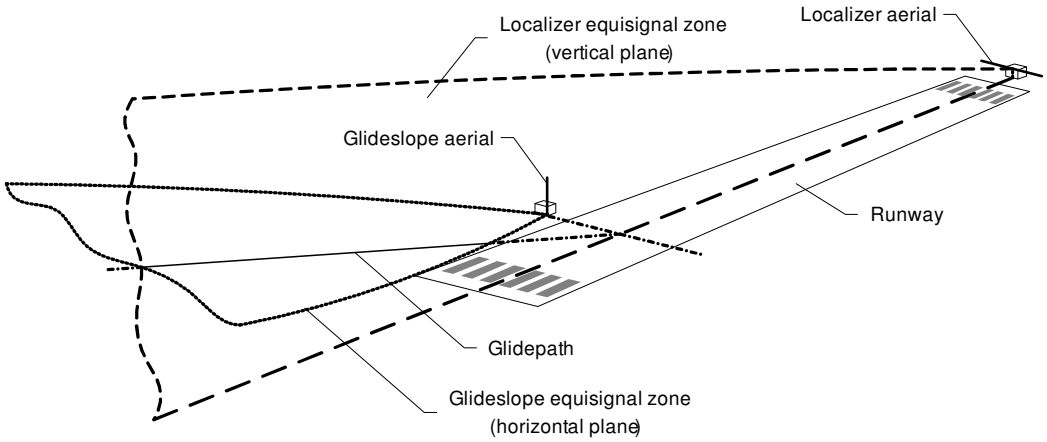


Fig. 5.2 Instrument Landing System (ILS) configuration

A similar technique is used for shipboard landing on aircraft carriers, with the error measurement provided by other technical means. Electronic systems such as ICLS *(Instrument Carrier Landing System)* and ACLS *(Automatic* (or *All-weather* in other sources) *Carrier Landing System)* employ tracking radars onboard the carrier, the error signals are transmitted to the aircraft via data link. The UCARS system designed specifically for UAVs has a similar arrangement; it is briefly described in Section 2.4.3.4.2. The visual *Fresnel Lens Optical Landing System* (FLOLS) relies on pilot's eyes for horizontal alignment (as the alignment with the landing deck centreline is distinctly visible, in good weather at least) and uses specially designed shipboard optical system for reporting vertical error. If the aircraft is above its normal position, the pilot sees the main light above the marked central line of the device, and vice versa. This is often aided by voice commands of the landing officer. In most modern shipboard systems, the measuring shipboard equipment is stabilised to average out the effect of periodic ship motion, or it can measure the ship motion and take it into account automatically.

Therefore, the guidance task during final approach involves trajectory tracking with both horizontal and vertical errors (and their rates) readily available. It is important to note that the errors being physically measured are *angular* deviations ε of the aircraft position (Fig. 5.3) as seen from the designated touchdown point (or more precisely, from where the radars or antennas or other guidance systems are located). They can be converted to linear errors Δh

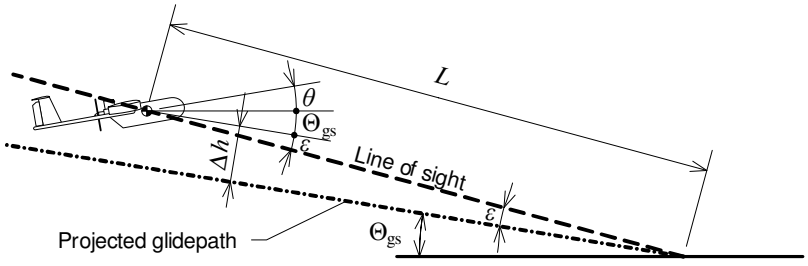


Fig. 5.3 Angular and linear glidepath errors

if the *distance L* to the aircraft is known. However, in many applications precise distance measurement is unavailable; for example, the Distance Measuring Equipment (DME), commonly used at civil airports, has the accuracy of only 185 m. Nevertheless, successful landing can be carried out even without continuous distance information (until the point where the flare must be performed, if applicable). This comes from the fact that exactly the angular errors are relevant to precise landing. Indeed, the goal is to bring the aircraft to a specified point on the runway (or on the deck) in a certain state (with required speed, attitude, acceleration, etc.) The choice of the landing trajectory only serves this purpose, considering also possible limitations and secondary objectives such as avoiding terrain obstacles, minimising noise level and fuel consumption and so on. In general, the actual position of the aircraft is less important at far distances but becomes increasingly important as the aircraft approaches the runway. Thus the angular errors provide a more adequate measurement of the current aircraft position with respect to the glidepath.

However, if the landing guidance system takes no account of distance and is built around the angular error only, it may cause stability problems at close distances, because the increasing sensitivity of the angular errors to any linear displacement effectively amplifies the system gain. This problem is eminent for both manned and unmanned landing systems, with the pilot-induced oscillations (PIOs) fairly common at the latest stages of carrier approach. Nevertheless, having a certain gain margin, an automatic system can adjust the error tolerance (or approximate the linear error) even with imprecise and discrete distance information without compromising stability. This is due to the fact that for typically very small glideslope and angular errors, the gain changes slowly.

5.1.2.1.1 Longitudinal control

To analyse the typical dynamics that the landing controller (or a human pilot) have to deal with, let us consider the longitudinal control of a typical aeroplane. In most cases, the dynamics of longitudinal and lateral motion is fairly decoupled at this regime, which allows to consider these motions separately.

As a rule, the aircraft configuration does not change during final approach. Flaps, slats, landing gear and other necessary landing equipment is adjusted beforehand. Therefore, the aerodynamic and dynamic characteristics of the aircraft itself remain constant (neglecting small mass change due to fuel burn and variations of atmospheric properties with altitude). Moreover, in the absence of large disturbances, the flight is quite steady, ideally with a constant airspeed. In such conditions, a linearised model can be considered.

Normally, an aeroplane has two control inputs in longitudinal motion: elevator and throttle. Both these inputs can be used for glidepath navigation. The elevator controls lift by

manipulating angle of attack, while the throttle controls lift via airspeed. Both elevator and throttle can be used simultaneously in a fully coupled control or one of these inputs may be used as a primary input while the other being fixed or set to maintain a specified flight parameter. In practice, two methods of such decoupled control are employed:

- Elevator is used as the main control while throttle holds airspeed (autothrottle);
- Elevator holds pitch attitude while throttle controls glidepath.

Minimisation of the landing speed is more often than not a necessity. For this reason, final approach is carried out with the minimum possible airspeed, only leaving an appropriate safety margin. This puts rigid constraints on airspeed variations at both ends. It is further complicated by the fact that the desired airspeed may be lower than (or close to) the so called minimum drag speed, at which the phugoid mode becomes unstable. For this reason, the autothrottle option seems more appropriate. However, with the autothrottle engaged, extra care should be taken to glidepath control, because the aircraft loses its natural stability of the flight path as a trimmed equilibrium state (at speeds greater than the minimum drag speed) when the phugoid mode is suppressed. Moreover, pitching moments associated with thrust changes may cause additional pitch and flight path instability, especially when the thrust line goes below centre of mass [170]. Perhaps for this reason, and also due to much longer periods of the phugoid mode as compared to short-periodic elevator control, airlines often prefer the pitch hold method for manual approach.

Now considering the elevator control of a trimmed on the glidepath aircraft with a constant airspeed, it can be shown [113] that the angular error ε control is equivalent to the 'angle of sight' (or 'line-of-sight angle') control if the landing spot (the 'target') is observable. The angle of sight λ is the angle between the longitudinal axis of the aircraft and the direction from the aircraft to the target landing spot (Fig. 5.3):

$$\lambda = \theta + \Theta_{\text{gs}} + \varepsilon = \theta_{\text{T}} + \Theta_{\text{gs}} \tag{5.14}$$

where θ is current (instantaneous) pitch angle, θ_{T} is the trimmed pitch angle, Θ_{gs} is the glideslope and ε is the error. It can be seen from the equation that the 'attitude hold' method of semi-automatic approach effectively controls the angular error. It should be noted, however, that the second relationship holds only as long as the trimming conditions are maintained.

The linear transfer function which describes the aircraft dynamics for elevator control of the angle of sight error ε is [113]

$$\frac{\varepsilon(s)}{\delta_{\text{e}}(s)} = \frac{K\left(s^2 + \overline{Y}^{\alpha} s + \overline{Y}^{\alpha}\frac{V_{\text{a}}}{L}\right)}{s^2\left(s^2 + 2\omega_{\text{s}}\xi_{\text{s}} s + \omega_{\text{s}}^2\right)} \tag{5.15}$$

where[1]

s — Laplace operator;

δ_e — elevator deflection;

K — system gain (pitch moment to elevator deflection);

ω_s, ξ_s — circular frequency and damping ratio of the short period motion;

$\bar{Y}^{\alpha}$ — dimensionless lift derivative by angle of attack;

V_a — airspeed;

L — distance to the target point.

It can be seen that at the far distances, where $\frac{V_a}{L} \to 0$, this transfer function reduces to the pitch control

$$\frac{\varepsilon(s)}{\delta_e(s)} \approx \frac{K\left(s+\bar{Y}^{\alpha}\right)}{s\left(s^2+2\omega_s\xi_s s+\omega_s^2\right)} = \frac{\theta(s)}{\delta_e(s)} \tag{5.16}$$

indeed proving the possibility of attitude hold approach. However, at closer distances where $V_a/L >> 1$, the angle of sight control turns out to be the altitude control:

$$\frac{\varepsilon(s)}{\delta_e(s)} \approx \frac{K}{s^2\left(s^2+2\omega_s\xi_s s+\omega_s^2\right)} = \frac{H(s)}{\delta_e(s)} \tag{5.17}$$

The second order integrator in this equation makes the control more difficult (especially for a human pilot), because it requires at least one derivative of the error signal, i.e. involves *lead* in the control. However, probably a more difficult problem is the change of the dynamics itself along the glidepath. It requires a controller with time-varying properties. This problem is avoided when linear error Δh is used instead of angular error ε (Fig. 5.3), although it leads to the more difficult altitude control (5.17). In order to prevent overtightening of error tolerances at far distances, the tolerances may be relaxed in accordance with the distance. This means, in effect, that the control gain will be reduced at far distances, again leading to a time-varying controller, though with a fixed structure.

5.1.2.1.2 Lateral control

The same considerations apply to lateral control. Lateral motion is controlled by two aerodynamic controls: ailerons and rudder. Similar to the longitudinal case, they can take up different roles, although by far the most optimal method in majority of cases is bank-to-turn control, where ailerons are used to roll the aircraft about its longitudinal axis and thus to produce a turn. The primary force responsible for turn is the horizontal portion of lift, which be-

[1] Please refer to Appendix A and Section 3.1.1.1 for explanation of the axes and units systems used in this work.

comes non-zero when the aircraft is banked. Angle of attack is slightly increased via appropriate elevator deflection to compensate the drop in vertical force. At the same time, rudder is used to compensate sideslip. In most cases, it is desirable to keep the sideslip angle β (and the sideforce Z or the respective load factor n_z) equal to zero. However, non-zero sideslip may be acceptable for some aircraft, particularly light aeroplanes and gliders. They can perform a 'flat turn' by applying rudder directly and using ailerons to maintain wing-level flight. More often, sideslip is used on these aircraft to induce additional drag in order to steepen the glidepath.

Trimming conditions include compensation of steady wind. Unlike longitudinal control, where steady wind can be compensated simply by additional elevator deflection, there are two ways to compensate crosswind:

- Banking the aircraft into the wind. To prevent turn, an opposite rudder deflection is applied, aligning the aircraft's longitudinal axis with the runway (or, generally, with the direction of flight)—so called 'slip landing.'
- Pointing the aircraft into the wind. The aircraft flies at an angle to the runway while its velocity is parallel to the runway—'crab landing.'

The first method involves non-zero sideslip and so is unfavourable from the point of view both aerodynamics and occupants' comfort. On the other hand, with the aircraft body and landing gear aligned with the runway, touchdown is easier. For this reason, large aircraft often combine these methods, utilising 'crab' technique for majority of the approach and then changing to slip before touchdown. For a UAV captured in-flight, however (which is the focus of this research), non-aligned yaw at the moment of recovery does not represent a serious problem. Therefore, lateral control can be performed entirely using ailerons, with the rudder keeping sideslip at zero.

The influence of distance to the landing spot on the dynamics is largely similar to the elevator control, requiring varying control at different distances.

5.1.2.2 UAV control for shipboard recovery

As it was seen from the discussion in the previous sections, landing of an aircraft is a well established procedure which involves following a predefined flight path. More often than not, this is a rectilinear trajectory on which the aircraft can be stabilised (trimmed), and the control interventions are needed only to compensate disturbances and other sources of errors. The position errors with respect to the ideal glidepath can be measured relatively easily. Shipboard landing on air carriers is principally similar; the main differences are much tighter error tolerances and absence of flare manoeuvres before touchdown. The periodic ship motion does have an effect on touchdown; however, it does not affect significantly the glidepath, which is projected assuming the average deck position. The choice of the landing deck size, glideslope,

aircraft sink rate and other parameters is made to account for any actual deck position at the moment of touchdown. For example, a steeper glideslope (typically 4°) is used to provide a safe altitude clearance at the deck ramp for its worst possible position (i.e. ship pitched nose down and heaved up). This makes unnecessary to correct the ideal reference trajectory on the fly.

In general, the final approach trajectory need not to be a straight line. The rectilinear approach has been selected for piloted aircraft (and many UAVs) largely due to simplicity of manual control, safety considerations, consistency of operation between different aircraft and relative simplicity of technical aids for instrumental approach. However, when the situation requires, other forms of trajectory should be considered. It was pointed out in the above sections that the two-stage design, finding an optimal trajectory and then synthesising a tracking controller that follows this trajectory, is a common control engineering practice. In particular, this approach has been successfully applied in the development of the catapult launch controller for the *Ariel* UAV [47]. In this work, control inputs (elevator deflections) have been optimised along the flight path so that the trajectory of the UAV under such control met the task objectives (quick gaining of a defined optimal climb angle and minimising altitude loss). After that, several controllers that implement the optimised control sequence as a function of sensor measurements have been synthesised using the novel *inverse system identification* technique. Even though the produced controllers were linear, they closely approximated the optimal control and the trajectory obtained in a nonlinear optimisation procedure.

However, the picture is different for the UAV shipboard recovery. As shown in Section 3.4.1, ship oscillations in high sea cause periodic displacement of the recovery window (the area where capture can be done) several times greater than the size of the window itself. This fact (and also the assumption that the final recovery window position cannot be predicted for a sufficient time ahead) makes it impossible to project an optimal flight path when the final approach starts. Instead, the UAV must constantly track the actual position of the window and approach so that the final miss is minimised. Therefore, it turns out that the UAV recovery problem resembles that of homing guidance rather than typical landing. While stabilisation on a known steady flight path can be done relatively easy with a PID controller, homing guidance to a moving target often requires a more sophisticated control. It can be said that homing guidance is concerned about optimising the guidance algorithm (and thus the control laws that implement it) instead of optimising the trajectory and tracking controllers. This task is addressed in the following section.

5.1.2.2.1 Proportional guidance

Not surprisingly, homing guidance found particularly wide application in ballistic missiles development, hence the accepted terminology owes to this engineering area. Since 1950s, a large number of guidance strategies have been proposed. However, nearly all the most elaborated guidance techniques critically rely on the information about the target's behaviour, attempting to predict, in one form or another, its trajectory. Consequently, they require measurement (or estimation) of the state of the target. If such estimation is erroneous, guidance performance drops significantly and the basic guidance strategies with lower requirements outperform their sophisticated counterparts [62].

The most popular of these basic guidance laws are known as Proportional Navigation (PN) Guidance. They have been successfully used for decades, and even today many tactical guided missiles employ PN for terminal guidance [197]. PN demands lateral acceleration of the pursuer (e.g. missile) to be proportional to the rate of rotation of the line of sight to the target:

$$a_{zk} = NV\dot{\lambda} \tag{5.18}$$

where N is the navigation constant (normally $N > 2$) and λ is the angle of sight (for the landing case, see (5.14) and Fig. 5.3). There are two most studied PN laws which differ with respect to the reference frame used. The so called Pure Proportional Navigation (PPN) is calculated in the pursuer's trajectory axes, thus the demanded acceleration a_{zk} is normal to the velocity vector $\boldsymbol{V}$. In contrast, the True Proportional Navigation (TPN) is drawn in the line-of-sight axes, so that a_{zk} is normal to the line of sight, and V in (5.18) becomes the closing rate V_c.

Obviously, in the case of PPN, $a_{zk} = \omega_k V$, where ω_k is the pursuer's turn rate (more precisely, turn rate of its velocity), therefore (5.18) can be written as

$$\omega_k = N\dot{\lambda} \tag{5.19}$$

which is the original definition of PPN [137].

It can be seen that PN does not use any knowledge of the target's trajectory (closing rate measurements required for TPN are usually available), nor it needs the time-to-go t_{go} or distance estimation nor any other predictions. This makes PN extremely robust. Nevertheless, even for such simple guidance laws, their characteristics are known largely empirically. Analytic solutions to the PPN problem are available only for $N = 1$ and $N = 2$ and a non-manoeuvring target (i.e. moving with constant velocity); and for any value of the navigation constant N and a non-manoeuvring target in the case of TPN [191]. It has been shown, however, that PPN does exhibit optimal behaviour and can accurately hit a non-manoeuvring target for $N = 3$ and that one can cope with a manoeuvring target by varying N between 3 and 5 [95].

In the context of UAV recovery, the difference between PPN and TPN is small because the UAV moves almost straight towards the 'target' from the beginning (compared to typical missile intercept scenarios) and thus the velocity vector and the line of sight almost coincide. Considering lateral motion, it is remarkable that if a coordinated level turn is used to obtain lateral acceleration, this acceleration is a function of only the bank angle γ:

$$a_{zk} = g \tan \gamma \tag{5.20}$$

thus the lateral acceleration command can be translated into the bank angle demand:

$$\gamma = \arctan \frac{NV\dot{\lambda}}{g} \tag{5.21}$$

In the longitudinal channel, normal acceleration corresponds to an appropriate elevator deflection (considering small perturbations and neglecting short period dynamics) and therefore can be easily controlled as well. It should be noted that an autothrottle may be necessary to maintain a safe airspeed in the presence of potentially large acceleration commands.

However, there are two major difficulties that can compromise the effectiveness of PN for UAV recovery. First, PN laws are known as generating excessive acceleration demands near the target. For a UAV with limited manoeuvrability, such demands may be prohibitive, especially at low approach airspeeds. On the other hand, the PN guidance strategy does not change during the flight, unlike direct angle of sight control (5.15). Several alternative guidance strategies with more favourable acceleration demands exist, e.g. augmented proportional navigation. However, it is unlikely they can sufficiently improve the guidance to an oscillating target such as ship's deck. These methods are typically built around minimising the so called zero effort miss, i.e. the expected miss distance if both the pursuer and the target will maintain their velocities. Non-zero effort miss where accelerations are considered in a similar manner may also be employed (see e.g. [169]). However, due to ship oscillations, both velocity and acceleration of the recovery window constantly change (including the sign), which impairs efficiency of these methods. Other class of guidance methods employs elaborated predictive models of the target, which we assume unavailable in this work (see Section 3.3).

Another potential deficiency of PN methods is the necessity to know the line of sight rate of rotation. In many cases it cannot be measured directly and can only be derived from sequential angular measurements. Like with any numerical differentiation, this process suffers intensely from instrumental noise, thus an estimation filter becomes a necessity. In the case of UAV recovery, this may be further complicated by the ship angular motion if the base tracking equipment is installed on the ship (which is likely, see Section 2.4.3.4): an unstabilised tracking system may add the ship's angular velocity to the angle of sight rate. Even greater difficulty may arise if the tracking radar or a similar device is mounted onboard the UAV: a

high level of turbulence may cause fast and irregular angular motion of the light vehicle so that even locking on the target may be difficult. This problem is addressed in Section 5.2.1.2.2.

5.2 UAV controller structure

From the analysis of Section 5.1.2.2 it becomes clear that the UAV control for the recovery stage should be a form of target tracking guidance strategy. The choice of the optimal strategy and its parameters, however, remains an open question. The objective of the rest of this chapter is therefore to synthesise such guidance strategy that enables reliable UAV recovery, and to produce a controller that implements this strategy.

As mentioned above, an analytical solution to a real-world guidance problem is very difficult, if possible at all. The evolutionary design (ED) method described in this chapter synthesises the control laws based on simulated *practical* experience, testing many designs in the conditions closely reflecting those the UAV will encounter in real flight. The method allows to evolve automatically both the structure and the parameters of the control laws, thus potentially enabling to generate a 'full' controller, which links available measurements directly with the aircraft control inputs (throttle, ailerons, rudder and elevator) and implements both the guidance strategy and flight control:

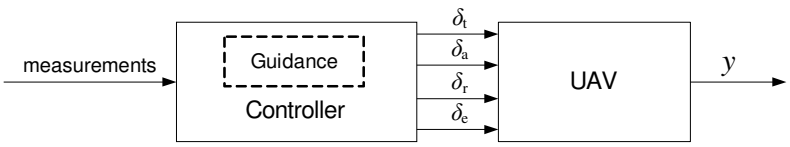


Fig. 5.4 Full controller with embedded guidance strategy

However, this approach, even though appealing at first and requiring minimum initial knowledge, proves to be impractical for two main reasons. First of all, computational demands of the evolutionary algorithms (EAs) soar exponentially with the dimensionality (complexity) of the problem. At the same time, the number of possible dynamic controllers is virtually unbounded, even with limited number of inputs (15–25) and outputs (4). It is therefore desirable to reduce complexity of the problem by reducing the number of inputs/outputs and limiting, if appropriate, possible structures of the controllers. This can be done, for example, by decoupling the longitudinal and lateral control and by using only one control in each channel for guidance (as described in Sections 5.1.2.1.1 and 5.1.2.1.2).

Another difficulty is the evaluation of the controller's performance. In ED, performance (*fitness* in EA terms) evaluation is made on the basis of simulation runs of the controller. For a full controller, many factors must be assessed, from the miss distance and other trajec-

tory characteristics to some particular indicators of control quality such as damping of short-period oscillations and control usage. All these factors must be weighted or combined in some way to obtain a collective estimate. Appropriate weights selection is a common problem for many control design techniques, it may be laborious and not so intuitive. At the same time, it has direct impact on algorithm convergence, and inappropriate fitness evaluation may hamper the design process significantly. Multi-objective search (see Section 4.2.1.1) can be employed here. However, it has its own shortcomings such as, generally, a larger population size required. In addition, the number of objectives in comprehensive UAV control is too large to be handled at once conveniently. Therefore, it is highly desirable to decompose the task into several simpler problems and to solve them separately.

A natural way of such decomposition is separating the trajectory control (guidance) and flight control as described in Section 5.1.2. The guidance controller issues commands u_g to the flight controller, which executes these commands by manipulating the control surfaces of the UAV (Fig. 5.5). These two controllers can be synthesised separately using appropriate fitness evaluation for each case. In addition, this approach allows to simplify the controllers and to facilitate design by reducing the number of input measurements z available to each controller: only the relevant signals can be fed into the controllers.

If necessary, both the guidance and flight controllers can be, in turn, subdivided into individual sub-controllers for each channel. It should be noted, however, that a decoupled model of the UAV may not be available, and the aim of ED is to produce controllers without modification of the controlled system. For this reason, all the controls of the UAV must be engaged during the tests, even if the respective control laws are not being developed and have not been produced yet. To circumvent this limitation, simple temporary controllers (e.g. proportional controllers) may be connected to the controls which are not currently used for evolution of the control laws. To protect them from excessive load, the respective test conditions may be relaxed. For example, when the longitudinal channel is being developed, simple proportional controllers which maintain wing-level flight with minimum sideslip may be connected to ailerons and rudder and the lateral turbulence component may be deactivated. However, extra care should be taken using this approach so as not to oversimplify the conditions for the controller being evolved and on the other hand, not to introduce unwanted behaviour

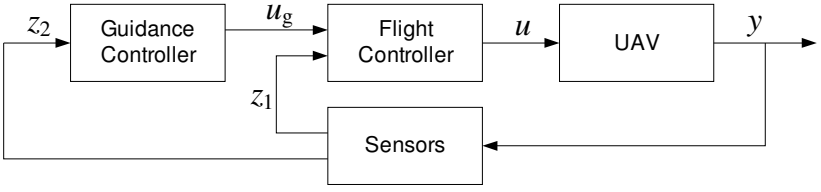


Fig. 5.5 UAV recovery control diagram

with these temporary controllers across the whole testing envelope. The evolutionary designs tend to adapt to the conditions they are tested against and to reveal obscure relationships, thus the evolved controller may incorporate into the design the changes of flight parameters caused by the temporary controllers. When these controllers are replaced with the final variants, some relationships will be lost and the evolved controller may become inoperable. This problem may be solved, at least partially, with the two-stage design when only the preliminary decoupled controllers are evolved, and then, as adequate performance is obtained and a suitable structure starts to develop, the evolution continues on a coupled design.

5.2.1 Guidance controller

At this stage, let us leave the internal structure of the controller to the automatic evolutionary design and rather look at the question of appropriate interface, i.e. set of inputs and outputs. As discussed above, it is desirable to keep the number of inputs and outputs to minimum, but without considerably compromising potential performance.

5.2.1.1 Outputs

An additional requirement to the outputs u_g is that these signals should be straightforwardly executable by the flight controller. This means that u_g should represent a group of measurable flight parameters such as body accelerations, velocities and Euler angles, which the flight controller can easily track. At the same time, the outputs should cause consistent changes of the trajectory of the UAV. Generally, such changes can be produced by an appropriate manipulation of the UAV velocity vector.

When discussing landing in Section 5.1.2.1.1, it has been said that there is a very limited capacity to control the trajectory by changing the airspeed. Even though this method may be applicable for steady path control during conventional landing, a more aggressive manoeuvring required for UAV recovery is deemed to be impossible using airspeed control. It has been reasoned that locking the airspeed using an autothrottle is the most suitable option for this task. Therefore, the absolute value of velocity may be considered approximately constant (ignoring the effect of wind gusts).

A specified change of trajectory can be obtained by controlled *rotation* of the velocity vector. This rotation can be commanded in two ways: by specifying the angular rate of rotation $\boldsymbol{\omega}_k$ and by applying a specified acceleration $\boldsymbol{a}_k$ normal to the velocity. These quantities are directly related through the velocity $\boldsymbol{V}_k$

$$\boldsymbol{a}_k = \boldsymbol{\omega}_k \times \boldsymbol{V}_k \tag{5.22}$$

and thus are interchangeable. Which one to use depends primarily on the convenience for the flight controller to measure and track the actual values of the respective signals. For an air-

craft, acceleration commands seem to be preferable as they will use almost full range of available body accelerations and therefore of the sensors scale, whereas body rotations due to trajectory changes are relatively small compared to the angular rates of short-period motion. This affects both the output sensors noise and the effect of turbulence measured by the sensors. In addition, accelerations are rather more intuitive to use.

A vector of normal acceleration can be specified in two ways: an absolute value plus an angle with respect to a certain origin (e.g. 0 = up, but note the vector is always normal to the trajectory, this definition vanishes as the trajectory approaches vertical); or two orthogonal components. Due to nature of position measurement system accepted in this work (Section 2.4.3.4.3), the latter representation, with the vertical and horizontal components, will be used. The angle and modulus representation might be more appropriate for a radar-based tracking system. Either way, they can be easily converted from one to another. Also, for convenience of use, all accelerations will be expressed as dimensionless load factors

$$n = a / g \tag{5.23}$$

where $g \approx 9.81$ m/s^2 is the gravity acceleration.

The acceleration demands are expressed in the *trajectory axes frame* $Ox_k y_k z_k$, in which the origin is located at the vehicle's centre of mass, Ox_k is directed along the inertial velocity vector, Oy_k is orthogonal to Ox_k and lies in the vertical plane (thus the notation 'vertical acceleration demand' should be taken conventionally as it may not be true vertical), and Oz_k completes the frame to the right-hand system, pointing horizontally to the right (looking along Ox_k). It differs from the body axes $Oxyz$ by the wind vector, aerial angles α and β, and the bank angle γ.

However, aircraft controls work primarily in the vehicle-fixed frame, making direct trajectory control more difficult. For this reason, it is deemed to be a reasonable approximation to consider the acceleration demands in the body-fixed frame $Ox_{k'} y_{k'} z_{k'}$, which is similar to the body frame $Oxyz$ but is rotated back by the angle γ so that $Oy_{k'}$ lies in the vertical plane. In other words, it is assumed, for the purposes of guidance, that the aircraft flies along its longitudinal axis Ox. This situation is illustrated in Fig. 5.6, with the aircraft shown directly from the back. Indeed, the aerial angles are typically very small (moreover, the sideslip β is usually deliberately kept at zero, see Section 5.1.2.1.2). Even combined with the wind, this approximation will, most likely, cause less difference in terms of reference frames than that between the PPN and TPN guidance methods (see Section 5.1.2.2.1), both of which show very similar practical performance.[1] It is expected that any discrepancies, should they appear significant,

[1] In the cases when the pursuer starts flying approximately towards the target, which is the case for the UAV final approach.

will be addressed by the automatically designed guidance control laws. This approach is believed to be more effective than making the flight controller track the true trajectory accelerations, because the latter will require the aerial angles information (which is noisy due to nature of vane sensors) as well as the wind estimation, severely compromising the quality and robustness of the flight controller.

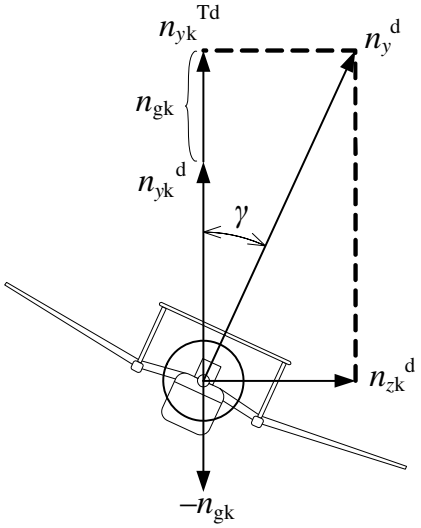


Fig. 5.6 Translating trajectory acceleration demands into body acceleration and bank angle

As a rule, an aeroplane cannot apply an arbitrary acceleration directly. In particular, aeroplanes are typically not designed to withstand substantial lateral body acceleration. Instead, they bank to a certain angle and apply normal body acceleration to change the flight path. This way, assuming the body lateral acceleration kept to minimum by the flight controller, the acceleration commands can be kinematically translated into the bank angle and the normal body acceleration demands, also taking into account gravity acceleration (Fig. 5.6):

$$\gamma^{\mathrm{d}} = \arctan\frac{n_{z\mathrm{k}}{}^{\mathrm{d}}}{n_{y\mathrm{k}}{}^{\mathrm{d}} + n_{\mathrm{gk}}} = \arctan\frac{n_{z\mathrm{k}}{}^{\mathrm{d}}}{n_{y\mathrm{k}}{}^{\mathrm{Td}}} \tag{5.24}$$

and

$$n_y{}^{\mathrm{d}} = \frac{n_{y\mathrm{k}}{}^{\mathrm{d}} + n_{\mathrm{gk}}}{\cos\gamma} = \frac{n_{y\mathrm{k}}{}^{\mathrm{Td}}}{\cos\gamma} \tag{5.25}$$

where γ^{d} is the bank angle demand, $n_y{}^{\mathrm{d}}$ is the normal body load factor demand, $n_{y\mathrm{k}}{}^{\mathrm{d}}$ and $n_{z\mathrm{k}}{}^{\mathrm{d}}$ are the vertical and horizontal load factor demands, n_{gk} is the load factor due to gravity force and $n_{y\mathrm{k}}{}^{\mathrm{Td}}$ is the total vertical load factor demand (including gravity). Taking into account the above assumption regarding the axes frames, the gravity load factor n_{gk} can be found directly through the pitch angle θ:

$$n_{\mathrm{gk}} = \cos\theta \tag{5.26}$$

The maximum bank angle during the low speed final approach (and often in the whole flight envelope) is restricted for safety reasons and due to operational limitations of the aircraft. In this work, the roll limits have been set to ±45°: experiments with the UAV model have shown that the aircraft is capable to pull about 1.4g to 1.5g normal acceleration at such low approach airspeed before exceeding the critical angle of attack, which corresponds to an approximately 45 degree steady level turn. This restriction has an important implication on

the calculation of the normal load and bank demands for the flight controller. The aircraft cannot turn upside down and apply a positive normal body acceleration to realise a *negative* vertical acceleration demand (the practice common to agile aircraft). For this reason, the two quadrant arctangent shall be used in (5.24) to calculate the bank angle demand. Then, *after* limiting the demand according to the restrictions, it can be supplied to (5.25) to obtain the correct (and possibly negative) normal load demand. For the same reasons, n_y^{d} is not calculated geometrically from the two acceleration demand components.

In a similar manner, the maximum realisable load factor n_y^{d} may limit the applicable bank angle, possibly even prior to the rigid restriction of 45°. However, unlike the bank angle which is not limited, as such, aerodynamically, the maximum load is quite sensitive to the current flight conditions. The main factor here is angle of attack α. Therefore, the flight controller shall correct the bank angle demand if the required load factor is unreachable. It should be noted, however, that when n_y limit is reached, it is a matter of preference how to dispose available horizontal and vertical accelerations. Both of them may be inadequate to the demands. If the bank angle is unchanged, both accelerations will be reduced proportionally. However, vertical acceleration may be considered more important, because, apart from guidance, it is responsible for the flight as such. Excessive roll with inadequate lift may result in crash. For this reason, roll correction may be beneficial.

In addition, bank angle limitation creates one more issue which should be taken care of if negative values of n_y^{d} are expected. In the presence of a non-zero horizontal acceleration demand $n_{z\mathrm{k}}^{\mathrm{d}}$, the aircraft will have to 'flip' to the opposite bank angle as the n_y^{d} changes its sign. This may produce unwanted behaviour if n_y^{d} is kept near zero and thus may change its sign back and forth often. To avoid unnecessary rolling, a relay-like block with the characteristics such as that shown in Fig. 5.7 may be introduced between $n_{y\mathrm{k}}^{\mathrm{d}} + n_{\mathrm{gk}}$ and $n_{y\mathrm{k}}^{\mathrm{Td}}$. The obtained value of $n_{y\mathrm{k}}^{\mathrm{Td}}$ should subsequently be used in formulas (5.24) and (5.25). This block will delay the transition through zero and thus the roll flip by a certain small value of load, reducing the amount of reverse actions.

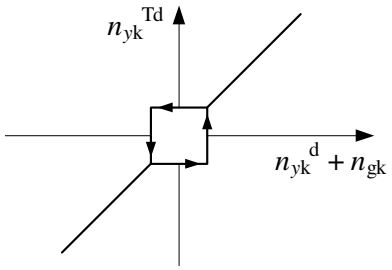


Fig. 5.7 Delaying the total vertical acceleration demand

In this work, however, it has been found that the *Ariel* UAV is incapable to produce a negative n_{yk}^{Td} at the approach airspeeds and realistic pitch angles due to limitation of positive (pitching down) elevator deflection. To this end, n_{yk}^{Td} has simply been limited to a small positive value of 0.001.

Altogether, the structure of the output part of the guidance controller becomes as shown in Fig. 5.8. With this scheme, the guidance laws produce general requests to change trajectory in horizontal and vertical planes. The kinematic converter then recalculates these requests to the form convenient for the flight controller. Both the bank angle γ and normal body load factor n_y can be relatively easily tracked, with the sensors providing direct measurements of their actual values. At the same time, this approach allows to evolve the horizontal and vertical guidance laws separately, which may be desirable due to different dynamics of the UAV's longitudinal and lateral motion and also due to computational limitations.

5.2.1.2 Inputs

Input measurements to the guidance controller should be those relevant to trajectory. First of all, this is all available positioning information (see Section 2.4.3.4). Also, some of the flight parameters may be useful for guidance. These are pitch and yaw angles and airspeed. They do not account for steady wind, but still provide substantial information regarding the current 'shape' of trajectory. For example, high pitch angle indicates intensive climb.

Regarding the yaw angle, it should be noted that it is measured relative to north direction. However, as the system should be independent of the world orientation, the yaw angle fed into the controllers is corrected by the 'reference' yaw ψ_0, which is perpendicular to the arresting wire in the direction of anticipated approach. This correction is fixed during the approach (as the ship heading is assumed to be constant) and is known to the UAV from the navigation data. This way, a zero yaw indicates that the UAV is pointed perpendicularly to the arresting wire (ignoring ship oscillations), which is the ideal condition in the absence of side wind and when the UAV moves along the ideal glidepath. This is similar to rotating the ground reference frame $O_g x_g y_g z_g$ by the correction yaw angle ψ_0. The rotated frame is referred as *approach ground reference frame*.

Apart from these directly measured (raw) signals, some derived quantities from them will be helpful. Two examples are the vertical and lateral velocity components with respect to

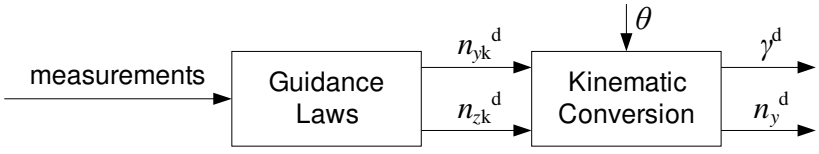


Fig. 5.8 Guidance controller

the approach ground reference frame. Ideally, this should be the inertial velocity components. They can be obtained from either GPS or onboard INS (or a mix thereof, see Section 2.4.3.4.1). However, in this work a 'minimalistic' measurement equipment is assumed. For this reason, the components of aerial velocity are calculated. They are obtained in the following algorithm. Firstly, the body components of airspeed V_a are found by rotating the aerial velocity vector $[V_a\ 0\ 0]^T$ by the aerial angles α and β using direction cosine matrix (DCM) A_{ab} (3.5). Secondly, the body components are expressed in the approach ground reference frame using DCM A_{gb}^T (3.3):

$$\boldsymbol{V}_{ag} = A_{gb}^T A_{ab} \begin{bmatrix} V_a \\ 0 \\ 0 \end{bmatrix} \tag{5.27}$$

The second and third components of $\boldsymbol{V}_{ag}$ will contain the required vertical and lateral velocities. In no wind conditions, they are equal to the inertial components.

5.2.1.2.1 Positioning information

Determination of the current UAV position and velocities with respect to the arresting wire is crucial for successful recovery. While previously discussed flight parameters may only help to improve the guidance quality, positioning carries direct responsibility for recovery. Obtaining useable positioning information is discussed in this section.

Section 2.4.3.4.3 described the technical aspects of the positioning system used in this study. The system is based on radio distance metering and provides ten independent raw measurements (Fig. 5.9): three distances d_1, d_2 and d_3 from the UAV to the radio transmitters located at both ends of the recovery boom which supports the arresting wire and at the base of recovery mast; three rates of change of these distances; distance differences $(d_1 - d_2)$ and $(d_3 - d_2)$; and rates of change of the differences.

All these measurements can be supplied to the guidance controller. The guidance laws evolution process is potentially capable to produce the laws directly from raw measurements, automatically finding necessary relationships between the provided data and the required output. However, this is unproductive for two main reasons. First, providing the pre-calculated parameters directly related to the UAV position may simplify the evolutionary search, reducing the computation time. Second and more importantly, the guidance laws evolved from raw measurements will be rigidly linked to the positioning system used, being inflexible and incompatible with any other positioning system. Not only completely different systems (such as radar tracking systems) will be hard to adapt to, but even incorporating the changes in geometrical parameters such as the base legs L_1 and L_2 (Fig. 5.9) may be difficult.

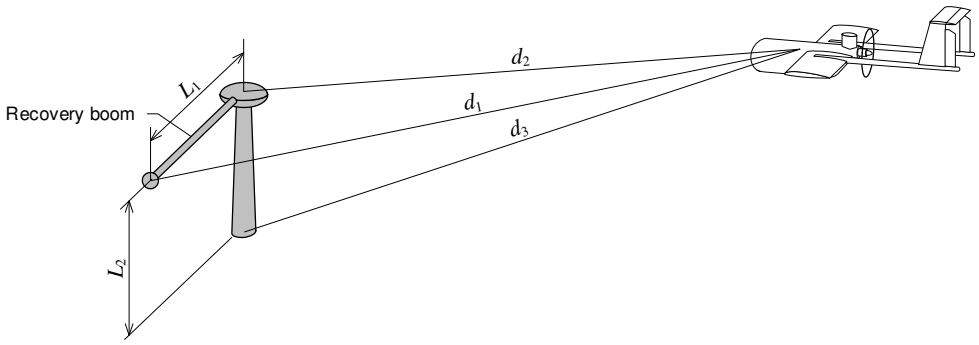


Fig. 5.9 Positioning scheme

For this reason, the positioning information is recalculated to reflect the actual position and velocities of the UAV in universal quantities such as metres and radians and with respect to a certain common origin point. If technically possible, this origin should be independent from actual implementation of the positioning system, which may be located at different places. A natural choice of such origin is the 'target' spot which the UAV should be aimed at. This point is located above the centre of the arresting wire (see Section 2.4.2). The elevation h_T of the target spot is determined by the cable sag at the moment of recovery and thus may vary depending on airspeed and final trajectory shape (Section 3.5.4). Ideally, it should be approximately one half of the effective cable sag, giving maximum error allowance both up and down. However, the actual cable sag cannot be measured in flight. Therefore, the target spot elevation is chosen to be a constant, and the value is determined in view of the expected cable sag obtained from simulation of the cable model. Section 3.5.4, Fig. 3.17 provides the necessary data. For normal approach speed, the cable sag varies between 3.5 and 4.5 m. Accordingly, the target spot elevation is chosen to be $h_T = 2$ m above the arresting wire.

Vertical and horizontal displacement. The vertical and horizontal position with respect to the target spot can be calculated by the following scheme (Fig. 5.10 for the vertical position). First, the displacement with respect to the second transmitter (located at the root of the recovery boom) L_0 is determined. Since

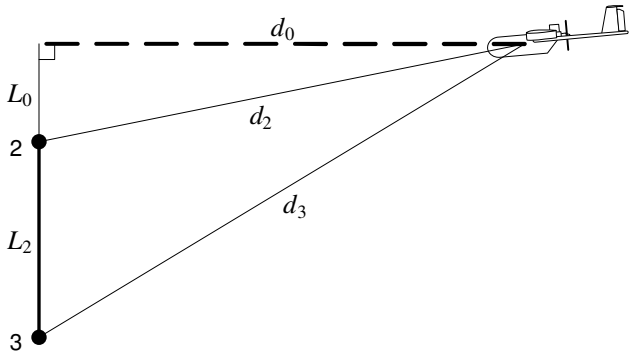


Fig. 5.10 To the calculation of spatial displacement

$$\left(L_2 + L_0\right)^2 + d_0^2 = d_3^2$$
$$L_0^2 + d_0^2 = d_2^2 ,$$

L_0 can be expressed as

$$L_0 = \frac{d_3^2 - L_2^2 - d_2^2}{2L_2}$$

As mentioned in Section 2.4.3.4, in practice this formula is very sensitive to separation of the

transmitters in space. For the accepted base leg L_2 = 5 m, at the distance of several hundred metres (in the beginning of final approach), any inaccuracies in measurement of d_2 and d_3 will cause large errors in L_0. This effect is known as dilution of precision. Generally, this is an unavoidable limitation of any triangulation-based system (including GPS). However, as the system provides an independent (and precise) measurement of the difference ($d_3 - d_2$), this formula may work better if rewritten as

$$L_0 = \frac{(d_3 - d_2)(d_3 + d_2) - L_2^2}{2L_2}$$

Having calculated L_0, the elevation relative to the target point can be obtained:

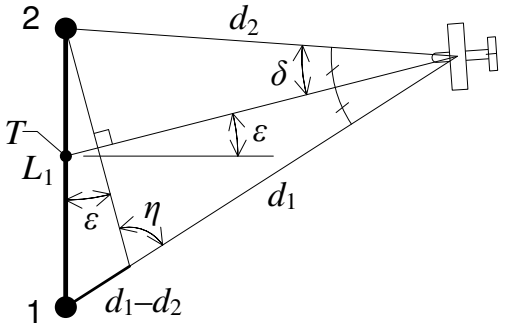


Fig. 5.11 To the calculation of angular displacement

$$\Delta H_T = L_0 - h_T = \frac{(d_3 - d_2)(d_3 + d_2) - L_2^2}{2L_2} - h_T \tag{5.28}$$

Note that ΔH_T is not the 'glidepath error' such as that shown in Fig. 5.3: there is no any glidepath defined to the guidance controller. This is simply a general measure of the vertical position of the UAV with respect to the arresting wire.

In a similar manner, horizontal displacement can be obtained, taking into account that the target is located in the middle of the arresting wire:

$$\Delta Z_T = L_0' + \frac{L_1}{2} = \frac{(d_1 - d_2)(d_1 + d_2) - L_1^2}{2L_1} + \frac{L_1}{2} = \frac{(d_1 - d_2)(d_1 + d_2)}{2L_1} \tag{5.29}$$

Angular position. As stated in Sections 5.1.2.1.1 and 2.4.3.4.3, angular measurements may provide a more adequate positioning information for the recovery task. Fortunately, these measurements can be approximated using only the distance differences. First, let us consider horizontal angular position, which is symmetrical with respect to the centre of the recovery boom (Fig. 5.11).

During the major part of the final approach, d_1, $d_2 >> L_1$ and therefore δ is small and $\eta = 90° - \delta \approx 90°$. In this case, the angle ε can be found straightforwardly

$$\varepsilon \approx \varepsilon_h \approx \arcsin \frac{d_1 - d_2}{L_1} \tag{5.30}$$

The angle ε is slightly smaller than the desired angular position ε_h because point T (the target point in this arrangement) is skewed towards the closest transmitter (#2 in Fig. 5.11). How-

ever, this skew is partially compensated by the approximation (5.30), which always overestimates ε since $\eta < 90°$. For small ε, which is expected for proper approach, the error tends to zero, and even for large ε, the estimated value will point the aircraft to a position between the transmitters. The error may become noticeable only at close distances (d_1, $d_2 \cong L_1$, i.e. less than half a second before capture) and large ε, which is an unusual situation most likely leading to a miss. Overall, this is an effective approximation which uses only one measurement. Exact calculation of the angular position would require distance measurement which is less accurate and could produce an inferior result.

Distance information is needed, however, to determine the vertical angular displacement, which is asymmetrical with respect to the transmitters. The target point T_v is elevated by h_T above the transmitter #2. The required angle ε_v[1] can be obtained by correcting the angle ε measured in a similar manner to (5.30), by the angle μ (Fig. 5.12):

$$\varepsilon_v = \varepsilon - \mu \approx \arcsin\frac{d_3 - d_2}{L_2} - \mu \tag{5.31}$$

μ can be determined from the triangle UTT_v, which is defined by the fixed side $TT_v = b \approx h_T + L_2 / 2$, angle $T = 90° - \varepsilon$, and the distance d_{23}, which can be found as the length of the median (assuming point T lies in the middle between the transmitters #2 and #3, as it is supposed to be):

$$d_{23} = \frac{1}{2}\sqrt{2d_2^2 + 2d_3^2 - L_2^2} \tag{5.32}$$

which for practical values of the distances and L_2 can be approximated as

$$d_{23} \approx \frac{d_2 + d_3}{2} = d_2 + \frac{d_3 - d_2}{2} \tag{5.33}$$

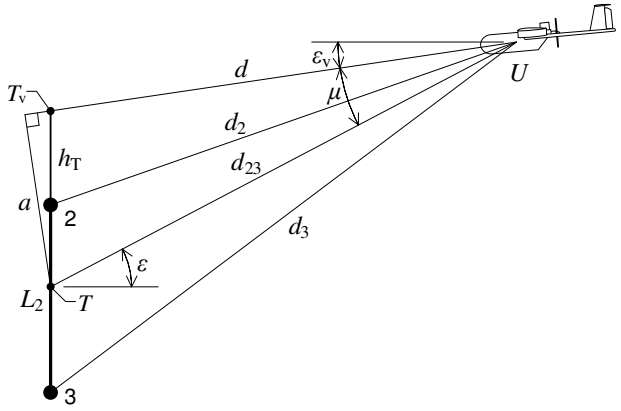


Fig. 5.12 To the calculation of vertical angular displacement

[1] The subscripts for ε stand for 'vertical' and 'horizontal' and are chosen to avoid ambiguity with the respective coordinates $\Delta H_T = \Delta Y_T$ and ΔZ_T, because $\varepsilon_v = \varepsilon_z$ correspond to ΔY_T and $\varepsilon_h = \varepsilon_y$ correspond to ΔZ_T.

Depending on the actual implementation of the positioning system, either way may deliver a more accurate result.

Using the laws of sines and cosines, μ is obtained as

$$\sin\mu = \frac{b\cos\varepsilon}{b^2 + d_{23}^2 - 2bd_{23}\sin\varepsilon} \tag{5.34}$$

This formula can be simplified by assuming that ε_v is small. Using the same argument as above for ε, it can be reasoned that until the very last moments of approach (when it is too late to correct the flight path noticeably), this assumption is sensible. Consequently, $a \approx b$ and

$$\sin\mu = \frac{a}{d_{23}} \approx \frac{h_T + \frac{L_2}{2}}{d_{23}} \ . \tag{5.35}$$

This formula is valid only for $d_{23} > h_T + L_2 / 2$. Finally,

$$\varepsilon_v \approx \arcsin\frac{d_3 - d_2}{L_2} - \arcsin\frac{h_T + \frac{L_2}{2}}{d_{23}} \ . \tag{5.36}$$

Translational and angular velocities can be obtained by differentiating the above equations by time:

$$V_{zT} = \Delta\dot{Z}_T = \frac{\left(\dot{d}_1 - \dot{d}_2\right)\left(d_1 + d_2\right) + \left(d_1 - d_2\right)\left(\dot{d}_1 + \dot{d}_2\right)}{2L_1} \tag{5.37}$$

$$V_{yT} = \Delta\dot{H}_T = \frac{\left(\dot{d}_3 - \dot{d}_2\right)\left(d_3 + d_2\right) + \left(d_3 - d_2\right)\left(\dot{d}_3 + \dot{d}_2\right)}{2L_2} \tag{5.38}$$

$$\omega_h = \dot{\varepsilon}_h \approx \frac{\dot{d}_1 - \dot{d}_2}{\sqrt{L_1^2 - \left(d_1 - d_2\right)^2}} \tag{5.39}$$

$$\omega_v = \dot{\varepsilon}_v \approx \frac{\dot{d}_3 - \dot{d}_2}{\sqrt{L_2^2 - \left(d_3 - d_2\right)^2}} + \frac{b\dot{d}_{23}}{d_{23}\sqrt{d_{23}^2 - b^2}} \ . \tag{5.40}$$

In the latter formula

$$b = h_T + \frac{L_2}{2} = 4.5\,\mathrm{m} \tag{5.41}$$

and d_{23} is defined by (5.33). It may be reminded that the differences $(d_1 - d_2)$ and $(d_3 - d_2)$ as well as their time derivatives $\left(\dot{d}_1 - \dot{d}_2\right)$ and $\left(\dot{d}_3 - \dot{d}_2\right)$ represent independent measurements and therefore the brackets in (5.37) and (5.38) should not be opened.

Additional measurements. Apart from the above position and velocity data, two other measurements may be useful for guidance. These are the total distance and closing velocity.

A general measure of the distance of the UAV from the recovery boom can be obtained from the readings of d_1 and d_2 using the formula similar to (5.32):

$$d = \frac{1}{2}\sqrt{2d_1^2 + 2d_2^2 - L_1^2} \tag{5.42}$$

Since this distance may be needed even close to the boom, it should not be simplified to the average of the source readings, similar to (5.33). A distance directly to the target spot could be calculated by multiplying (5.42) by cos ε_v; however, this would involve more measurements (see (5.36)) with little effect for guidance.

Closing velocity (or closing rate), again with respect to the centre of the arresting wire, may be obtained by differentiating (5.42). However, to avoid dependence on possibly inaccurate distance measurements, a simplified approach is used:

$$V_{CL} \approx -\frac{\dot{d}_1 + \dot{d}_2}{2} \tag{5.43}$$

For convenience and compatibility with the UAV inertial velocity, the minus sign makes the closing velocity positive when the UAV approaches the arresting wire.

5.2.1.2.2 Handling the periodic ship motion

Determination of UAV position discussed in the last section takes no account of ship motion. Meanwhile, it has enormous effect on the resultant calculated position.

The periodic ship motion (see Section 3.3) can be separated into translational and rotational oscillations. While it is desirable (and, in fact, required) that the UAV should track the translational component aiming at the actual position of the recovery window, angular displacement of the recovery boom have little effect on the size and position of the recovery window. In Section 3.4.1, amplitude of the recovery boom is considered. It can be seen that even in the worst practical conditions, angular amplitude of the boom never exceeds 30 degrees and the major portion of this amplitude is sourced in ship roll. In other two orthogonal planes, the amplitude rarely exceeds 5 degrees.

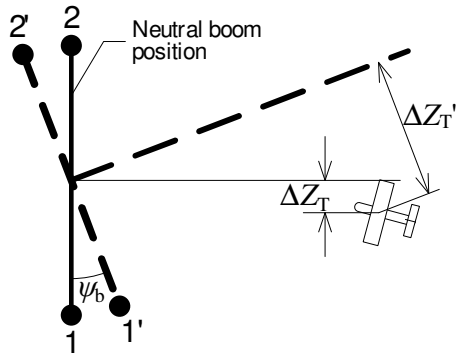


Fig. 5.13 The effect of ship angular motion on position measurement

At the same time, even minor angular displacement of the recovery boom may cause a huge change in position measurement. For example, 1° angle translates to a more than 5 m sideways displacement at the distance 300 m, i.e. about the size of the recovery boom (see Fig. 5.13). If angular motion is not compensated in one or another way, it is likely that the position readings will be unusable.

Nearly all shipboard positioning equipment used for similar purposes either compensates ship motion internally using a set of motion sensors or is stabilised on a movable platform. The positioning system modelled in this work is located directly on the recovery boom, providing relatively straightforward position measurement, but precluding it from the stabilisation alternative. Therefore, the only option is to take the angular ship motion into account in position calculations.

The data for motion compensation can be obtained from either the common ship's sensors or a set of dedicated sensors. The exact location of these sensors is unimportant because only angular measurements of the ship body motion are required: three Euler angles θ_s and three angular velocities ω_s. However, since the boom may be positioned in different ways (see Sections 2.3.6.2 and 3.4.1), the ship body measurements should be rotated to align with the actual position of the boom. As the neutral position of the boom is always horizontal, the rotation involves only one angle about the normal ship body axis y_s and can be expressed using the following direction cosine matrix:

$$\begin{aligned} A_{sb} &= \begin{bmatrix} \cos\psi_b & 0 & -\sin\psi_b \\ 0 & 1 & 0 \\ \sin\psi_b & 0 & \cos\psi_b \end{bmatrix} \\ \theta_b &= A_{sb}\theta_s - \begin{bmatrix} 0 \\ \psi_b + \theta_{sy} \\ 0 \end{bmatrix} \\ \omega_b &= A_{sb}\omega_s \end{aligned} \tag{5.44}$$

where θ_b and ω_b are Euler angles and angular velocities of the boom, and ψ_b is the boom angle with respect to the ship longitudinal axis. $\psi_b = 0$ corresponds to the forward position (Fig. 3.11), $-\pi/2$ to the left side position (Fig. 3.10). This angle is fixed during approach. The obtained boom yaw angle θ_{by} is corrected by the value $\psi_b + \theta_{sy}$ (where θ_{sy} is ship yaw) so that the neutral boom position is always $[0\ 0\ 0]^T$.

With the angles and angular velocities of the boom available, the calculations from the previous section can be corrected by adding the following values (assuming that the angles are small):

$$\delta H_T = \theta_{bx} d_{23} \tag{5.45}$$

$$\delta Z_T = \theta_{by} d \tag{5.46}$$

$$\delta \varepsilon_v = \theta_{bx} \tag{5.47}$$

$$\delta \varepsilon_h = \theta_{by} \tag{5.48}$$

$$\delta V_{y\mathrm{T}} = \omega_{\mathrm{b}x} d_{23} + \theta_{\mathrm{b}x} \dot{d}_{23} \tag{5.49}$$

$$\delta V_{z\mathrm{T}} = \omega_{\mathrm{b}y} d + \theta_{\mathrm{b}y} \dot{d} = \omega_{\mathrm{b}y} d - \theta_{\mathrm{b}y} V_{\mathrm{CL}} \tag{5.50}$$

$$\delta \omega_{\mathrm{v}} = \omega_{\mathrm{b}x} \tag{5.51}$$

$$\delta \omega_{\mathrm{h}} = \omega_{\mathrm{b}y} \tag{5.52}$$

It can be seen that, naturally, Cartesian position and velocity corrections directly depend on the distance and thus any inaccuracies in ship angle measurements will be greatly amplified. In contrast, angular corrections are free of this shortcoming and therefore, once again, angular positioning may be considered preferable. It is expected that the effect of noise placed upon all sensors will make the evolutionary design to choose the least affected readings if possible, even though instrument errors are not simulated.

Additional measurements, the distance d and closing rate V_{CL}, are also significantly affected by periodic ship motion. Whilst knowing actual distance is necessary, oscillations of the closing rate may be highly undesirable. In strong headwind conditions, for example, the actual closing rate with respect to the boom may periodically become close to zero and even negative, even though the average rate is positive. In addition, the approximation (5.43) tends to underestimate V_{CL} at close distances. This may destabilise the controllers which rely on closing rate measurements.

Since translational components of the ship motion play significant role in disturbances to the closing rate, compensation of the latter is cumbersome. It requires not only the ship angular velocity ω_{b} measurements, but also accelerometers readings and exact distance from the accelerometers to the boom centre. However, due to periodic nature of ship motion and almost steady approach of the UAV (assuming that airspeed is maintained and steady wind does not change dramatically), the oscillations may be averaged out by calculating the running average of the V_{CL} measurements (considering that the controller will be implemented in a digital form and the measurements will be digitised):

$$V_{\mathrm{CLa}} = \frac{1}{n(t) - n(t-s) + 1} \sum_{i=n(t-s)}^{n(t)} V_{\mathrm{CL}_i} \tag{5.53}$$

where $n(t)$ denotes the number of data sample corresponding to the time t. The window size s should cover at least one full period of ship motion. In the case of final approach, which is relatively short, the full previous history of measurements can be averaged. This can be done without storing the history in memory. In addition to eliminating the problem of ship motion, running average also reduces the noise of the readings.

It remains an open question *how* the corrections (5.45)–(5.52) can be implemented on a real system, considering that some (or all) position measurements are carried out onboard the UAV (see Section 2.4.3.4.3). Generally, there are two possibilities: to shift the phases of the transmitted signals according to current ship position and velocity so that the UAV receives the signals as if they are unaffected by the angular ship motion; and to transmit the ship angle and velocity data to the UAV within the measurement signals or using another real-time data link. The latter option seems to be easier; however, the details of implementation are unimportant in this study because the positioning system is considered ideal.

5.2.1.3 Terminal phase of approach

The recovery procedure, if successful, lasts until the cable hook captures the arresting wire. This happens when the UAV have moved about the full length of the cable *past* the recovery boom (see Fig. 2.11). However, position measurements may be unavailable beyond the boom threshold, and even shortly before the crossing the readings may become unreliable. In addition, the approximations from Section 5.2.1.2.1 become sufficiently inaccurate at very close distances and have rigid constraints such as $d_{23} > (h_T + L_2 / 2) = 4.5$ m. For these reasons, the terminal phase of approach, from the distance about 6–10 m until the capture (or detection of a miss), should be handled separately.

For a normal approach speed, the terminal phase lasts less than a second, of which normally less than half a second is spent before crossing the boom threshold and can potentially be used for guidance corrections. However, it is highly unlikely that any actions at this stage will sufficiently correct the guidance errors made at the earlier stages. Moreover, large errors detected at close distances will usually require vigorous steering, which may result in stall or other dangerous conditions just before capture, not to say about awkward body attitude. In addition, dynamic characteristics of the UAV may provoke unwanted behaviour which is normally unnoticed but becomes significant when immediate response is considered. For example, elevator deflection generates some amount of additional lift which has an opposite sign to the moment the elevator produces. Consequently, for a nose-up command normally issued to *climb*, the UAV initially *descends* until the wing gains enough lift to overcome the elevator force. In a usual situation, this descend is subtle; however, only a split second before crossing the boom, such reverse action may result in worsening the situation and even a crash into the boom.

It is therefore possible to disconnect the guidance controller several metres before the recovery boom without affecting the quality of guidance. The allowed error (approximately 2 m in all directions, determined by the lengths of the arresting wire and the cable) should ab-

sorb the absence of controlled guidance in the last 0.3 to 1 second (depending on the headwind) in most situations.

Two questions must be answered to complete the recovery procedure: how should the UAV be controlled during the terminal phase and how to determine the moment of changeover. An adequate answer to the first question is deemed to be the output

$$n_{yk}{}^{d} = n_{zk}{}^{d} = 0 \tag{5.54}$$

which means 'keep the last velocity.' These demands are channelled through the kinematic converter (see Fig. 5.8) to the flight controller as usual. Under this control, the UAV will continue to fly in the direction that the guidance laws produced at the moment of disconnection, making smooth transition from the guidance control. In addition, this control reduces any roll the UAV may have, which is desirable.

The moment of changeover is determined so as to keep the assumptions made in Section 5.2.1.2.1 well within the valid range and on the other hand, not to disrupt guidance notably. The point is chosen to be 7 metres before the recovery boom. The distance is determined from (5.42), which uses two different distance measurements. On a real implementation of the system, reliability questions should also be considered, with the backup means preventing both premature and late changeover.

Another issue to be considered on a real system is automatic miss detection. If the capture is unsuccessful and the UAV has not crashed, another attempt may be performed, which is one of the advantages of this recovery method. Since the distance readings may be unavailable, miss can be detected if the cable hook is not pulled after L_c / V_{CL} seconds, where L_c is the cable length, plus a necessary margin. In the case of a miss, the control is handed over to the main navigation controller, which diverts the UAV from the ship and guides the aircraft to a new approach entry point. Backup means for this important event should also be provided, possibly including the operator's 'go around' button.

5.2.2 Flight controller

Flight controller receives two inputs from the guidance controller: bank angle demand γ^{d} and normal body load factor demand $n_y{}^{d}$. It should track these inputs as precisely as possible by manipulating four aircraft controls: throttle, ailerons, rudder and elevator. These four outputs of the flight controller are connected directly to the respective actuators which drive the control surfaces and throttle of the UAV.

Apart from the demands, all available measurements from the onboard sensors are fed into the controller, except for positioning information which is relevant only for guidance. These are (see Section 3.1.2.4): body angular rates ω_x, ω_y, ω_z from rate gyros, Euler angles γ,

ψ, θ from strapdown INS, body accelerations n_x, n_y, n_z from the respective accelerometers, airspeed V_a, aerial angles α and β, actual deflection of the control surfaces δ_a, δ_r, δ_e, and engine rotation speed N_{rpm}. The barometric altitude data is not included due to insufficient accuracy. In addition, two time derivatives are calculated by first-order differentiation: $\dot{V}_a$ and $\dot{\alpha}$. Although these signals may be very noisy, they may be useful for high quality control.

All these inputs are grouped into several groups, and only the relevant measurements are provided to each of the four channels of the controller. This is described in detail in Section 5.3.2.1, where controller implementation is discussed.

For simplicity of design, the controller is subdivided into independent longitudinal and lateral components. The longitudinal component includes throttle and elevator channels, and the lateral component includes rudder and ailerons channels. Each sub-controller, as discussed in Section 5.1.2.1, has one primary control directly responsible for tracking the respective demand and one auxiliary control. In longitudinal branch, elevator tracks the input signal n_y^d, while throttle is responsible for maintaining a required airspeed. In lateral control, naturally, ailerons track γ^d, while rudder minimises sideforce by keeping n_z near zero.

It should be noted that the requirements to the flight controller regarding its tracking abilities are defined in terms of the actual task it will be used for. They may be different to the parameters of control quality commonly used in linear design, such as static error, damping ratio, overshoot, etc. For example, the controller may be allowed to have a significant static error if this way it delivers quicker high-frequency response or better robustness. All these characteristics will be automatically determined in the evolutionary design process.

5.3 Evolutionary design

The Evolutionary Design (ED) presented in this section is aimed at developing the control laws optimally suitable for both the defined mission and the controlled system. Generally, it takes no assumptions regarding the system and thus can be used for wide variety of problems, including nonlinear systems with unknown structure. In many parts, this is a novel technique, and the application of ED to the UAV guidance and control problems demonstrates the potential of this design method.

The core of evolutionary design is a specially tailored evolutionary algorithm (EA) which evolves both the structure and parameters of the control laws. This algorithm is used as a tool to create several control laws for the two UAV controllers described in the previous sections. When the design stage is done, the whole system is thoroughly tested to estimate its performance and robustness.

In order to understand the details of ED, a certain knowledge of evolutionary computation theory and practice is needed. The basics of EAs are presented in Chapter 4. It should be

noted that although EAs are inherently robust, selection of algorithm parameters may have significant effect on the algorithm's performance. However, there is little theory that suggests optimal parameters before the algorithm is run. Most of the parameters are selected on the basis of relevant experience and trial and error method. Nevertheless, since the algorithm is used for *creative* work only at the *design* stage, its performance is rather of secondary importance as long as the calculations take a sensible amount of time. The major requirements to automatic design methods are *quality* of the result and *exploration* abilities. By exploration is understood the scope of possible designs that the algorithm can comprehend. When EAs are used for design purposes, it is common to sacrifice quick convergence in favour of better exploration of the search space.

Although the basic framework of an EA is quite simple (see Section 4.1), there are three key elements that must be prepared before the algorithm can work. They are:

- representation of phenotype (control laws in our case) suitable for genetic operations (genome encoding);
- simulation environment, which enables to implement the control laws within the closed loop system;
- fitness evaluation function, which assesses the performance of given control laws.

These elements, as well as the whole algorithm outline, are addressed in the following sections. In the next chapter, the algorithm is applied to the UAV control problems.

5.3.1 Discussion

Earlier in this chapter, some of the most popular control techniques have been considered. It was stated that a great majority of classic control methods are essentially PID controllers (Section 5.1.1.2), often, however, with elaborated dynamic manipulation of the feedback coefficients and/or state estimation. In other words, general structure of such controllers, with the control signal being proportional to the measured (or estimated) error and, possibly, its derivative and integral, is similar, and the main effort is directed to the optimal choice of gains and optimal state estimation.

A distinction can be made between *static* and *dynamic* controllers. Simple proportional control such as $u(t) = K\,y(t)$ is static: at any instant, its output depends only on the current input. It has no state as such and no way to 'memorise' the past of the inputs. In contrast, full PID controller (5.7) is dynamic: its output is a function not only of the input, but also of the integrator state. In general, dynamic controllers can be expressed using two functions, one that describes the relationship between the state and the input (5.3), and another establishing the relationship between the output and both the state and the input (5.4). The linear optimal

control techniques (Section 5.1.1.3) employ linear dynamic controllers in the state-space form such as (5.9) (or its discrete variant).

A dynamic controller can be considered as a *filter* which filters its input so that a desired control output is obtained. On the contrary, static controller simply amplifies the input signals together with their noise content. Nevertheless, even simple linear static controllers can be successfully used for aircraft control in some cases. For example, roll stabilisation is often implemented as

$$\delta_a = K_1(\gamma_{def} - \gamma) + K_2\omega_x \tag{5.55}$$

where δ_a is the ailerons deflection, γ and γ_{def} are the current and desired roll angles, ω_x is the roll rate, and K_1 and K_2 are respective gain coefficients. The aircraft itself 'integrates' the ailerons deflection to produce current angle γ.

However, where more complex aircraft behaviour exist and/or when specific requirements to the control are set, dynamic control must be used. In particular, this is important when input measurements are considerably affected by noise. For this reason, representation of the control laws used for UAV control should allow evolution of dynamic controllers. Static control, if desired, can always be realised by setting the state equation to zero or removing state variables from the output function.

Whilst the structure of linear controllers is generally similar, nonlinear control offers great range of possible structures. The number of nonlinear structures is virtually unbounded, even considering a narrow class of them, and optimal choice of the structure is no less important and no less difficult than optimal selection of its parameters. Therefore, the ED algorithm should evolve structures as well as the numeric parameters. Moreover, as the parameters are closely related to the structure, both evolutions must proceed simultaneously. The example (5.55) shows a possible structure which linearly links three inputs with one output. K_1 and K_2 are the parameters of the structure, they are specific for this particular structure and have no meaning without it.

Parallel evolution of both the structure and the parameters of a controller can be implemented in a variety of ways. One of the few successfully employed variants is the *block structure controller evolution* by Koza et al [122, 124]. In this approach, the controller is assembled from a set of predefined basic blocks such as gain, integrator, lead, lag, etc. The algorithm evolves the topology of the system by applying random interconnection between the blocks, by introducing new blocks and deleting existing blocks. Direct structure altering operations are used sparingly (with the probability of the order of 1–5%), while the major source of structure variations is performed in a typical for genetic programming (GP) way, using two-parent sub-tree crossover (see Section 4.5.3.1). The parameters of each block (e.g. coeffi-

cients for gains, time constants for lags, etc.) are supplied as additional input(s) to the blocks and thus can be driven by regular branches of the program tree, including simple numerical constants (terminals). Numerical constants as such are not evolved, they are created at random for the initial population (and also in the scarcely applied mutation operations). Instead, the algorithm modifies the sub-trees during crossover and mutation, so that in the course of evolution the sub-trees are evaluated into the desired parameter values.

Although this method, in a sense, is quite straightforward and very general, a high price must be paid for the flexibility and convenience of the block structure controller evolution. Like any crossover-driven mechanism, the algorithm critically relies on diversity of the population, which leads to employment of very large populations. In the work [122], the algorithm has been applied to a simple SISO linear control problem. Even though it showed quick convergence (in terms of generations, whose number totalled only 32), the population size was as large as 66,000, which led to expenditure of more than $5 \cdot 10^{15}$ computer cycles, consuming 44.5 hours on a cluster of 66 DEC Alpha computers. Fitness evaluation of each individual was carried out on the basis of 10 simulation runs and included both time domain and frequency domain analysis.

In the current work, both the control problem and the controlled system is much more sophisticated. Not only it represents a real-world MIMO problem with all associated disturbances and multitude of environmental factors, but also the computer implementation is computationally more expensive. Even in basic configuration, a single simulation run may take 1 to 5 seconds (on a 2 GHz PC), which is approximately 20 times as expensive in terms of computer cycles as in [122]. Therefore, a different approach must be found, although, possibly, at the price of generality and self-sufficiency of the algorithm.

The first consideration that may help to save computational resources is fixing the numerical parameters. Instead of evaluating them in a costly sub-tree parsing process, the constants may be defined directly as real numbers. However, by doing that, the means of parameters evolution are lost. This is hardly a problem, because numerical optimisation can be reintroduced as a separate evolutionary process. Numerical optimisation in EAs is a well established practice and should pose even less problems than the GP-like evolution.

However, separate handling of numerical constants does not solve the main problem: large population size. Moreover, numerical evolution usually requires moderate population sizes but sufficiently large number of generations. For simultaneous numerical and structural evolution, a completely different approach is needed.

The ED algorithm enables to evolve suitable control laws within a reasonable time by utilising gradual evolution with the principle of strong casualty. This means that structure alterations are performed so that the information gained so far in the structure of the control law

is preserved. Addition of a new block, though being random, does not cause disruption to the structure. Instead, it adds a new dimension and new potential which may evolve later during numerical optimisation. The principle of strong casualty is often regarded as an important property for the success of continuous evolution [188].

This process can be illustrated in the following scheme. Suppose an approximation of a given flat shape is required. Approximation is made by a series of straight line segments, defined by the coordinates of N vertices (Fig. 5.14). A price is paid for every segment, and at start it is unclear how many segments is needed for a given accuracy requirement. Moreover, even for unlimited number of segments, introducing many segments right from the beginning is problematic because too many parameters must be optimised simultaneously. At best, it will lead to high computational demands, and in the worst case to a non-convergent search (remember that the algorithm does not 'know' the target shape and cannot place the points in accordance with it; it can only try a random location and obtain a measure of the resulting approximation from the fitness evaluation function).

Gradual evolution starts from a reasonable minimum number of points, say 3 or 4. Their location is optimised until a satisfactory intermediate solution is found. Then, another vertex may be introduced. However, if it is placed at a random location on the plane irrespective of the locations of the other vertices, the resultant polygon will nearly always deliver an inferior solution. It will either be perished in favour of former simpler solutions or if considered as a seed for further optimisations, require the optimisation to start all over again. For this reason, the new vertex should be placed initially on an existing segment (as in Fig. 5.14). This will not produce any immediate improvement and may even deteriorate the result slightly because more segments are used for the same approximation. However, further numerical optimisation should fairly quickly arrive at a better solution. Note that all points, including the 'older' ones, participate in further optimisation: their new optimal locations may be different. Nevertheless, the way of their evolution does not change dramatically because the newly added vertex have not disrupted the value of their present locations. After several steps of numerical optimisation, another vertex may be added, and the process continues until a satisfactory solution is found.

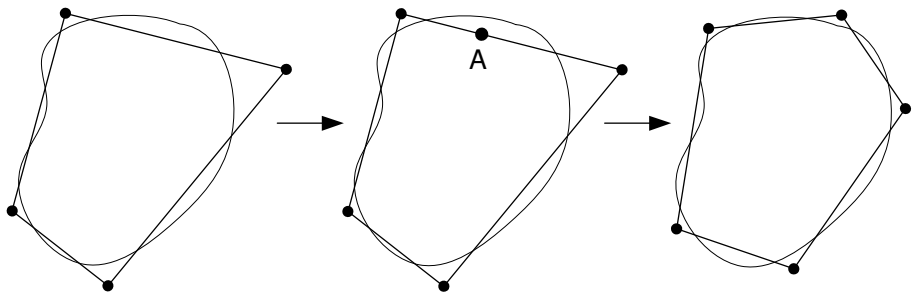


Fig. 5.14 Neutral structure alteration

Since the addition of new points is carried out as a separate dedicated operation (unlike sporadic structure alterations in the sub-tree crossover), it is termed *structure mutation*. Furthermore, structure mutation performed in a way described above is known as *neutral structure mutation*. The usefulness of neutral mutations has been demonstrated for the evolution of digital circuits [217] and aerodynamic shapes [159]. In the latter work, aerofoil of turbine stator blades is optimised involving shape optimisation process similar to that in the example above (using spline curves instead of straight lines).

There are several implications of this evolution scheme that should be taken into account. First, any existing point (or, in general, structure element), once introduced, cannot be deleted without disrupting the shape. Even though in some special cases simplification may be possible (for example, when two adjacent segments form a straight line), existing elements are usually not removed. This leads to ever-growing complexity of the solutions. However, similar (and perhaps even more pronounced) behaviour is observed in GP-like structure evolution as well (see Section 4.5.5).

Second, it is not necessary to wait until the existing intermediate structure is fully optimised before performing a new structure mutation. It is likely that the optimal parameters of the new structure will be different; even if not, numerical optimisation of these parameters will continue after structure mutation. Only at the last stage it may be worth waiting the full (or nearly full) optimisation of the final structure. As a rule, structure mutations are introduced with a relatively small probability during the course of parameters evolution. Alternatively, structure mutations are performed periodically once in a few generations to all or most of the members. Also, since the complexity and dimensionality of the structures in the population constantly grows during the evolution, it takes longer to optimise the growing number of parameters. For this reason, the number of structure mutations per generation should decrease. Conversely, at the initial stages, when only minimalistic designs are present, the population may be injected with more structure mutations.

As a result, the ED algorithm basically represents a numerical EA with the inclusion of structure mutations mechanism. The representation of the control laws, described in the following section, enables independent optimisation of the numerical parameters as well as the means for neutral structure mutations.

Of the variety of numerical EAs, outlined in Chapter 4, Evolutionary Strategies (ES) are chosen as a base for the ED algorithm. This type of EAs is optimised for real-value numerical evolution and allows use of relatively small population sizes. Unlike GP and common Genetic Algorithms (GAs), ES are mutation driven. This is convenient because the individuals in a ED population may represent very different structures which are incompatible with

each other in terms of their parameters, making any crossover operation within these parameters meaningless. The algorithm is described in greater detail in Section 5.3.5 below.

5.3.2 Representation of the control laws

Instead of building the control system from basic dynamic blocks as in the block structure controller paradigm [122], the controller is built from a combination of static functions and input signals, which are organised as a dynamic structure. This is done by constructing, in a similar manner, a number of *state equations* (5.3) and *output equations* (5.4). Considering the number input signals (more than 10 in some cases), this approach is believed to be more effective, especially if the general structure of the equations is chosen so that it always produces valid expressions suitable for control equations.

Since flight control is largely a continuous process, continuous representation of the state equations (5.3) is preferred over the discrete form $x_{n+1} = g(x_n, y_n)$. In addition, it gives independence of the sample rate used in the controller. This is convenient because the requirement to the bandwidth of the controller may be uncertain. Taking a safely small time step may be inappropriate for the computationally intensive ED because this has a significant impact on the computational demands (see Section 5.3.3). For this reason, maximally large time step (low sample rate) should be used during the evolution, while a smaller time step may be accepted for finer control during testing and validation.

The controller being evolved has m inputs, r outputs and n states. The number of inputs and outputs is fixed (although not all inputs may be used in a particular design). The algorithm allows varying number of states; however, in this work, the number of states is also fixed during the evolution. Again, some of the state equations may be left unused by some controllers. As a result, the controller comprises of n state equations and r output equations:

$$\begin{aligned} \dot{x}_1 &= g_1(\boldsymbol{x}, \boldsymbol{u}) \\ \dot{x}_2 &= g_2(\boldsymbol{x}, \boldsymbol{u}) \\ &\text{M} \\ \dot{x}_n &= g_n(\boldsymbol{x}, \boldsymbol{u}) \end{aligned} \tag{5.56}$$

$$\begin{aligned} y_1 &= f_1(\boldsymbol{x}, \boldsymbol{u}) \\ y_2 &= f_2(\boldsymbol{x}, \boldsymbol{u}) \\ &\text{M} \\ y_r &= f_r(\boldsymbol{x}, \boldsymbol{u}) \end{aligned} \tag{5.57}$$

where $\boldsymbol{u}$ is size m vector of input signals, $\boldsymbol{x} = [x_1, x_2, \ldots x_n]$ is size n vector of state variables, $y_{1\ldots r}$ are controller outputs (for convenience, the input and output variables are named as it is accepted for regular dynamic systems (as opposed to (5.3) and (5.4)), even though controller

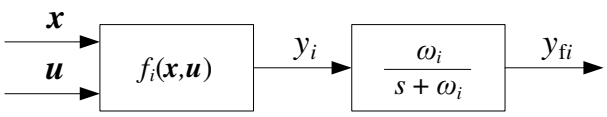


Fig. 5.15 Forming an output signal (i = 1…r)

outputs represent the inputs for the UAV). Initial value of all state variables is zero. All $n+r$ equations are built on the same principle and are evolved simultaneously. For structure mutations, a random equation is selected from this pool and mutated.

In addition, each output is passed through a simple first order low-pass filter. This is done to filter out the noise caused by presence of (possible noisy) input signals directly in the output equations (5.57). The bandwidth ω of this filter is optimised within the algorithm along with other numerical parameters. Therefore, the structure of each output becomes as shown in Fig. 5.15.

5.3.2.1 Representation of input signals

Input signals delivered to each particular controller may represent directly measured signals as well as the quantities derived from them, as discussed in Sections 5.2.1.2 and 5.2.2. To speed up the calculations involved, all inputs are calculated once at each time step and put in a common bank, where they are accessed by all controllers as needed. The inputs are grouped into several groups for easier access by different controllers. Each controller may have access to one or several groups, which is configured separately for each controller. For example, autothrottle controller may have access to the group of longitudinal flight parameters, but be restricted from using positional information, which is irrelevant for this controller. Excluding irrelevant inputs improves the evolution and makes the control laws more comprehensible.

Within each group, inputs are organised in the subgroups of 'compatible' parameters. Compatible parameters are those which have close relationship with each other, have the same dimensions and similarly scaled. The examples of compatible parameters are the pairs (n_x, n_{xg}), (ω_y, $\dot{\psi}$), (V_a, V_{CL}). As a rule, only one of the compatible parameters is needed in a given control law. For this reason, the probability of selection of such parameters for the structure mutation is reduced by grouping them in the subgroups. Each subgroup receives equal chances to be selected. If the selected subgroup consists of more than one input, a single input is then selected with uniform probability. Therefore, the inputs in a subgroup share the selection probability of the whole group.

For example, consider the group

{
(n_x, n_{xg})
θ
}

This group consists of two subgroups. If this is the only group available to a particular controller, the structure mutation procedure will select either subgroup with 50% probability. Subsequently, if the first subgroup is chosen, one of its members will be selected. Altogether, θ receives 50% chances to be selected, while both n_x and n_{xg} receive 25% each.

Therefore, every controller input may be represented by a unique *code* consisting of three indices: the number of group a, the number of subgroup b and the number of item in the subgroup c. The code is designated as `u(a,b,c)`. In the example above, θ has the code `u(1,2,1)` (considering the group as #1), while n_{xg} is represented by `u(1,1,2)`. When the equation parsing procedure encounters an input code, it obtains the current value of the respective signal from the bank of inputs, accessing it using the given indices.

In addition, any input (together with its coefficient, see Section 5.3.2.2 below) can be raised into any power from the list [–2; –1; –0.5; 0.5; 2]. This effectively implements the operations of inversion and square root. The exponent is chosen together with the input with a relatively small probability of about 6% each, the rest being occupied by the exponent 1. If the chosen exponent is not unity, it is included in the input code as the fourth parameter, e.g. `u(1,2,1,-0.5)`. Handling of this operation is more thoroughly described in the following section.

State variables are handled in a similar manner. They belong to a 'dummy' group #0, and the second index represents the number of the state variable. The third index is unused. For example, the code `u(0,2,0)` represents the second state variable of the controller. All state variables are treated as independent subgroups and thus share the probability of being selected for the structure mutation with the inputs.

5.3.2.2 Representation of control equations and the structure mutation

Each of the control equations (5.56) and (5.57) is encoded in a similar manner. To this end, only one single output equation of the form $y = f(\boldsymbol{u})$ will be considered in this section. State variables $\boldsymbol{x}$ are considered as special inputs and have no effect on the encoding.

The control equations representation allows to evolve a relatively narrow class of nonlinear equations, described below. Nevertheless, their structure is deemed to be adequate for typical control problems. This is done to speed up the search, which could otherwise be hampered by the multitude of possible operations. It proved to be more effective to include all meaningful quantities derived from source measurements as independent inputs (grouping

them as described in Section 5.3.2.1 above, see also discussion in Section 5.2.1.2.1) than to implement all the functions and operations which potentially allow to emerge all necessary quantities automatically in the course of evolution. The latter approach would require an extremely large number of structure mutations, which is possible only in a GP-like evolution with all associated computational requirements.

Another consideration is also performance related, but in a more technical sense. Each equation is calculated a few times at each time step during simulation, i.e. thousands of times per one fitness evaluation. Thus the encoding should be maximally compact and simple to allow quick parsing.

Finally, the encoding should allow a simple way to insert a new parameter in any place of the equation without disrupting its validity and in a way that this insertion initially does not affect the result, thus allowing neutral structure mutations.

Conceptually, the equation is a sum of input signals (also referred in this section as *variables* as it is accepted in mathematical expressions), in which:

- every input is multiplied by a numeric coefficient or another similarly constructed expression;
- the product of the input and its coefficient (whether numeric or expression) is raised to the power assigned to the input. In order to avoid singularities and bias to the positive values, the absolute value of the product is raised to the power, and the sign of the product is preserved in the result. In addition, to avoid singularity with negative powers, the absolute value of the product is limited on the lower bound by an arbitrary small constant 0.0001 (only for negative powers). These rules ensure the *closure property* of the operation, outlined in Section 4.5.3.1.
- a free (absolute) term is present.

The simplest possible expression is a constant:

$$y = k_0 .$$

A linear combination of inputs plus a free term is also a valid expression:

$$y = k_2 u_2 + k_1 u_1 + k_0 .$$

Any numeric constant can be replaced with another expression. An example of a full featured equation is

$$y = ((k_4 u_4 + k_3) u_3) \uparrow^{-0.5} + k_2 u_2 + (k_1 u_1) \uparrow^2 + k_0 . \tag{5.58}$$

The notation $\uparrow^n$ is used instead of power because the power operation is performed according to the special scheme specified above. For example, the $(k_1 u_1) \uparrow^2$ term could be written as $\text{sign}(k_1 u_1)(|k_1 u_1|)^2$.

It can be seen that a product of two (or more) variables can be introduced (as u_4u_3 in the above example). The operations of multiplication and addition are used in a consistent manner and thus can be encoded implicitly. Subtraction can be carried out by assigning negative values to the numeric coefficients, and division—by assigning a negative power.

The equation can be encoded directly as a tree-like expression, similarly to the genome representation used in GP. However, although MATLAB offers an effective way of representing hierarchical and recursive data (cell arrays), lack of passing the function parameters by reference[1] makes it inconvenient to update such data structures in a recursive function. To this end, a linear representation is developed. Another consideration taken into account is convenience of updating the numerical coefficients externally using the ES algorithm without parsing the expression. For this reason, the coefficients (and their respective strategy parameters) are kept in a separate array, and the expression contains only references (indices) to them.

The actual linear representation is as follows. The expression represents a linear (1D) cell array which may contain either input codes or indices of the numeric coefficients. It comes with two numeric vectors of equal length: the object parameters vector and the strategy parameters vector. The first vector contains the numeric coefficients for the expression, while the second vector contains their respective individual strategy parameters. (The purpose of strategy parameters is explained in Section 4.4.1.1). The expression is parsed according to the following algorithm:

[1] When parameter is passed to a function by reference (as opposed to passing by value), only address of the parameter in memory (the pointer) is actually supplied. Thus, any modification of the passed parameter will be reflected in the caller's function. The call $F(x)$ when x is passed by reference is equivalent to the assignment $x = F(x)$ when x is passed by value. Most programming languages allow the choice of either method.

1. Initialise the index pointer to the index of the first element.
2. Initialise result to 0.
3. Until the end of the expression is encountered, obtain the current item value in the following procedure:
   3.1. Pick up the current item and adjust the pointer to the next item.
   3.2. If the item is a number, interpret it as an index and pick up the respective value from the object parameters vector.
   3.3. Otherwise, if the item is an input code,
      3.3.1 Evaluate it by accessing the bank of input values.
      3.3.2 Pick up the next item and evaluate it (which involves a recursive run of the same procedure from step 3.1).
      3.3.3 Multiply the result to the input value.
      3.3.4 If the input is assigned with a non-unity exponent, raise the product to the specified power according to the scheme described above.
      3.3.5 Again evaluate the next item the same recursive way and add it to the intermediate result.
4. Add the obtained value to the result.
5. Continue the loop from step 3.

This algorithm can be illustrated by encoding the example (5.58). The respective internal representation of this expression is (for clarity, the input codes are assigned with one index instead of three, so that the code of the input u_i is `u(`i`)`):

Equation:	$y = ((k_4u_4 + k_3)u_3)\uparrow^{-0.5} + k_2u_2 + (k_1u_1)\uparrow^2 + k_0$
Expression:	`{ u(3,-0.5) u(4) 1 2 u(2) 3 u(1,2) 4 5 }`
Object parameters:	[k_4 k_3 k_2 k_1 k_0]
Strategy parameters:	[s_4 s_3 s_2 s_1 s_0]

This syntax somewhat resembles Polish notation with implicit '+' and '*' operators before each variable. An equivalent expression in LISP language would be

(+ (^ (* `u(3)` (+ (* `u(4)` k_4) k_3)) –0.5) (* `u(2)` k_2) (^ (* `u(1)` k_1) 2) k_0)

(note that it employs variadity of LISP's arithmetic operations, i.e. ability to take any number of arguments. The first '+' sums up all four terms of the original expression).

The rule 3.3.5 ensures presence of a free term in any sub-expression, such as k_3 and k_0 in the example above. In any correctly built expression, an input code is always followed by at least two items. The free term is important for neutral structure mutations, and the mutation procedure always preserves validity of the expression.

Structure mutations are carried out by replacing any numeric constant k_i with a sub-expression $(k_j u_j + k_i)$, where u_j is a randomly chosen input and k_j is its coefficient. For neutral mutations, the initial value of this coefficient is zero. However, if the input is assigned with a negative exponent n (which is chosen when the input is selected, see Section 5.3.2.1), the sub-expression becomes $((k_j u_j)\uparrow^n + k_i)$, and the initial value of k_j should be set to an appropriately large constant (e.g. 10^6) instead.

This operation can be performed very simply with the encoding described above. The new items, u_j and k_j, should be inserted (in that order) before a randomly chosen numeric coefficient. For example, consider the following evolution:

$$k_0 \;\rightarrow\; k_1 u_1 + k_0 \;\rightarrow\; (k_2 u_2 + k_1) u_1 + k_0 \,.$$

It corresponds to the following evolution of the genotype:

Expression:	{ 1 }	→ { u(1) 2 1 }	→ { u(1) u(2) 3 2 1 }
Object parameters:	[k_0]	→ [k_0 k_1]	→ [k_0 k_1 k_2]
Strategy parameters:	[s_0]	→ [s_0 s_1]	→ [s_0 s_1 s_2]

In addition, the new items can be inserted in front of the expression or before the last item. This adds a new summand to the whole expression. This can happen automatically when the last item is randomly selected as the insertion point; however, a new summand (i.e. linear effect of the variable (raised in a specified power if applicable) on the output) is expected to be generally more useful for flight control. Therefore, the probability of adding a new summand can be increased by handling this operation separately.

The exact algorithm of structure mutation is presented below. Apart from sampling of the initial population, this is the only structure alteration procedure.

1. Select an input (or a state variable) at random as described in Section 5.3.2.1: `u(a,b,c)`.
2. Obtain the initial values of the numeric coefficient and the strategy parameter (initial step size). The initial coefficient k is selected as described above, it can be either 0 or 10^6. The initial step size s is supplied together with the selected variable from a special table. It effectively defines the default scale of the variable.

3. Append the object parameters vector with the initial coefficient, and the strategy parameters vector with the initial step size. Obtain the index n of the newly added values (it will be the same for both vectors and equals to the length of either vector).
4. Form a sub-expression consisting of the selected variable code and the obtained index: { `u(a,b,c)` n }.
5. With 40% probability, set the insertion point (locus) to 1; otherwise, select a numeric value in the expression at random with equal probability among all numeric values present and set the locus to the index of this value.
6. Insert the sub-expression into the original expression at the chosen locus (*before* the item pointed).

This procedure may produce redundant expressions when the selected variable already exists at the same level, e.g. $y = k_2u_1 + k_1u_1 + k_0$. This is undesirable because it unnecessarily increases dimensionality of the problem (two coefficients k_2 and k_1 must be optimised instead of one for the same effect). Unfortunately, this condition cannot be reliably detected by simple inspection of the expression. For this reason, an algebraic simplification procedure has been implemented. It parses given expression, recursively collects the factors of each variable encountered and then rebuilds the expression. Rebuilding, however, is unnecessary if only checking for redundancy is needed. If after a structure mutation, simplification was successful, that is, returned a shorter expression, then the structure mutation was redundant. In this case, structure mutation is repeated.

In some cases, the free term of the whole expression may be disregarded. This is done, in particular, for all state equations (5.56), because integrating a non-zero constant can only lead to a growing state. Nevertheless, the free term is preserved in the expression for syntactic validity. If needed, it is subtracted from the result of full expression evaluation. Evolution of an additional parameter which has no effect on the phenotype draws virtually no extra resources. However, it can be used for other purposes, for example, as the bandwidth of the respective filter (Fig. 5.15).

5.3.3 Simulation environment

Fitness evaluation of the controllers is based on the outcome of one or more simulation runs of the models involved with the controller being assessed in the loop. Correct simulation set-up is vitally important for the success and performance of evolution.

Simulation environment for this study is constructed in the MATLAB/Simulink software. All the models described in Chapter 3 are implemented as Simulink library blocks and

can be arranged to form a required scenario. The whole model is formed by creating interconnections between inputs and outputs of the model blocks involved. The models participating in the evolution include:

- The *Ariel* UAV model (Section 3.1), which also includes separate
  - § Actuators model;
  - § Sensors model.
- The atmospheric model (Section 3.2), which includes:
  - § Standard atmosphere model;
  - § Steady wind model;
  - § Von Kármán turbulence model;
  - § Gust model.
- The ship model (Section 3.3);
- The static cable model (Section 3.5.4`);
- Controllers, which may be different according to the current task.

Since the bow location of the recovery boom, which is used for guidance law evolution, is typically not affected by ship airwake, and also due to the fact that ship airwake does not represent a significant difference for the flight controller as compared to regular turbulence, the ship airwake model is excluded. This also helps to reduce computational demands, which is critically important at the evolution stage. At the testing stage, however, this model is included.

The top-level Simulink diagram for the guidance task is presented in Fig. 5.16. The model records all necessary data to workspace for further analysis.

Initialisation of the model is specific for each task (flight or guidance controller evolu-

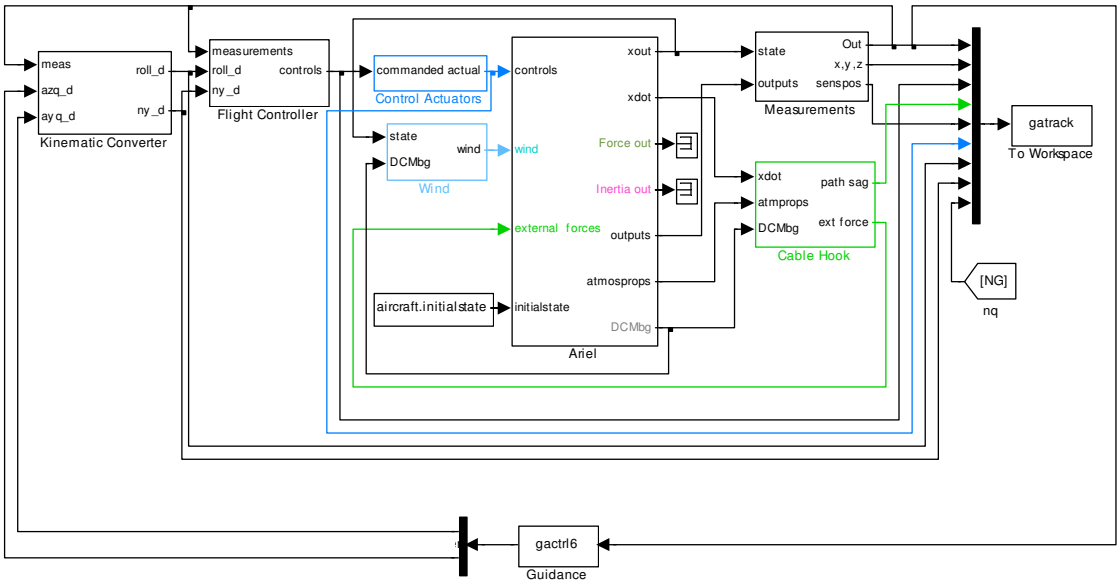


Fig. 5.16 Simulation model for the guidance task

tion). However, several common settings (mostly performance related) should be accentuated. First is the sample time (or sample rate) of integration. It has nearly proportional impact on EA performance, i.e. doubling the sample rate also doubles the time for each simulation run. For this reason, the sample rate should be kept at minimum, ensuring, however, numerical stability. The fastest and thus the most demanding dynamics in the model is contained in

- sensors (both the lags and noise);
- control actuators;
- turbulence model;
- and potentially, controllers.

These components should be especially carefully examined when adjusting the sample rate. The components which use white noise generators (sensors and turbulence) require a particularly large bandwidth for accurate noise representation. It is usually recommended[1] that the sample rate for noise generators should be two orders of magnitude greater than the bandwidth of the fastest dynamic element affected by noise in the system. However, this is an unacceptable requirement for the model used for evolution. In addition, mathematically accurate simulation of the effect of white noise is not the goal of the model. White noise itself is a rough approximation of the noise present in real sensors (the noise generated for turbulence model is immediately filtered by the von Kármán filters which have limited bandwidth and thus represents less of a problem). For the evolution, it should be enough to have only an acceptable level of disturbances in the signals, the more so as the exact levels of noise are not specified.

The experimentally determined minimum sample rate for the model is 100 Hz. The model is integrated using the 4th order Runge-Kutta method. Variable step solvers are useless when noise generators are present, which have a fixed (and maximally high) sample rate. For the reduced model without the fast blocks specified above, an adequate sample rate is 20 Hz. Such a model can be used for rough tests of the guidance strategy (however, it has not been employed in actual evolution).

It is important to turn off consistency checking on the Diagnostics pane of the simulation parameters in Simulink. Most of the time during simulation is spent for parsing the control equations. If consistency checking is enabled, Simulink forces the controller to evaluate the equations twice as many times as necessary, every second time with the same input. This increases simulation time by more than 90%. Bounds checking should also be disabled, although its effect is less noticeable.

[1] By the MATLAB documentation in particular.

If the Simulink Performance Tools are available, the Accelerator mode should be used for simulation. In this mode, Simulink generates C code of the model and pre-compiles it. The compiled code is then run for simulation. Even though the most time consuming operation, control equations evaluation, is not accelerated (because it is written as an S-function in MATLAB language), complexity and size of the model is high, and about twofold acceleration can be achieved.

5.3.4 Fitness evaluation

Fitness (or *objective value*, as it is referred in other optimisation techniques) is a measure of how well a given controller suited to a defined task. Fitness evaluation is based on objectives which are individual for each controller and therefore is calculated individually. However, some objectives are common for several control tasks. For example, the measure of control usage can be calculated for any controller in a similar manner. Several objectives may be weighted and combined in one common estimate. As discussed in Section 5.2, having several specific controllers makes it easier to define their particular objectives and reduces the number of these objectives as compared to one all-purpose autopilot.

Fitness evaluation of a controller can be divided into two main stages. First is preparation of the sample task and simulation of the model with the controller in the loop. The second stage is analysis of the results obtained from the simulation and evaluation of the fitness as such. These steps are often repeated several times, making the controller perform varying tasks in order to obtain a balanced measure of its performance. The overall fitness is usually calculated as the average of objective values for each task. Only by performing multiple simulations, a robust controller can be evolved. In general, this does not need to be done on every call of the fitness evaluation function. Only one (of a few) task may be chosen (usually at random) for every particular fitness evaluation. In the course of evolution, it is likely that every controller will be eventually tested against all the tasks and conditions available. However, this approach, even though speeding up per generation evolution, introduces additional 'disturbances' to the process and does not benefit the evolution on the whole. In addition, any instant 'snapshot' of the population fitness does not reflect real performance of each controller: the current best controller may become one of the worst in the next generation. If the sample tasks are significantly different, in every generation better fitness will be received by those individuals which were 'lucky' to receive an easier task, and the selection process will be biased. For this reason, multi-task fitness evaluation is used in this work.

Fitness value may reflect different levels of performance achieved by the controller, including 'hard bounds.' For example, if the controller crashes the UAV, a high penalty score may be assigned. However, excessive penalty for failure may significantly hamper the evolu-

tion process. Failures are normal during the evolution, especially at the initial stages, and a single accidental failure should not lead to immediate expulsion of the controller. Furthermore, for the sake of evolution, a distinction should be provided between 'total failure,' 'failure' and through to 'almost succeed.'

Hard bounds problem is especially demonstrative in the recovery guidance task. The target window is strictly determined. If the final UAV position fits into this window, the mission is completed successfully and a high fitness may be assigned. Conversely, if the position is outside the window, no matter how close to the border, the recovery attempt is unsuccessful. However, using even a moderate penalty for miss does not benefit the evolution of a capable controller. Instead, the distance from the centre of the target window should be used as the main component of the fitness. Early test evolutions which used rigid discretion between successful and unsuccessful attempts confirmed this thesis, showing scattered and unreliable convergence [114]. This is understandable, because with rigid discretion, the controllers that accidentally hit the window (due to favourable conditions, for example) may be preferred in the selection over those which show steady advance but are not yet capable enough.

Apart from the miss distance for the guidance task, other parameters of flight should be taken into account. One of them is control usage (or control effort): it is desirable to keep control usage at minimum. However, in a non-steady flight where no trim conditions may exist, zero control input[1] does not generally mean zero control. In fact, zero point for control deflection is chosen arbitrary. For this reason, calculation of the control usage as the integral of the square of control input over the entire flight, as it is often used in linear control techniques, is not appropriate. For near-steady flight, however, a rough estimate of control effort may be obtained with respect to the mean value of the control input, i.e. using the variance of the control signal (taking into account that the signal is discretised):

$$C_{\mathrm{c}}(y) = \frac{1}{n-1}\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2 = \mathrm{var}\left(y\right) \tag{5.59}$$

where y is the control signal discretised into n samples. This estimation is useful mostly on early stages to reject unstable controllers that cause the control signal to grow infinitely. To avoid excessively high (probably infinite) penalty for such behaviour, control signals are saturated. However, to provide better distinction between average-performing controllers that may become more stable later, the saturation bounds are 10 times higher than those of control actuators.

[1] 'Input' from the point of view of the aircraft.

A better estimation of the physical control usage as such can be obtained by analysing the *movements* of the control surfaces or, generally, the changes of the control signal. To analyse these changes, the signal can be differentiated using first-order differentiation:

$$\left(dy\right)_i = \frac{y_{i+1} - y_i}{t_\mathrm{s}}, \quad i = 1\mathrm{K}\ n-1 \tag{5.60}$$

Since the sample time t_s is fixed, it may be omitted from the equation and the difference may be reflected in the respective weight coefficient.

The variance of the differentiated signal can be used as a measure of control movements. However, variance will not detect unnecessary 'trembling' (or 'flapping') of the control signal, i.e. relatively high-frequency movements with small amplitude. While full frequency-domain analysis is desirable, this problem may be partially circumvented easier by moderating the estimate by the control usage estimate C_c:

$$C_\mathrm{f}(y) = \frac{\mathrm{var}(dy)}{C_\mathrm{c}(y)} \tag{5.61}$$

Indeed, for large control deflections, longer high-speed motions may be allowed, and vice versa. While control deflections are penalised separately as C_c, it is expected that large deflections may be needed for effective control.

Tracking abilities, if applicable, can be estimated by the weighted sum of the tracking error:

$$C_\mathrm{e}\left(z\right) = \frac{1}{n-1}\sum_{i=1}^{n}\left(z_i - z_i^\mathrm{d}\right)^2 \tag{5.62}$$

where z is the controlled signal and z^d is the desired value of the signal. Note that this estimation tends to penalise heavily any phase delay the controlled signal may have with respect to the reference signal, because phase delay results in persisting error when the reference signal constantly changes. This is expected to be rather beneficial because guidance to a moving target usually needs quick response.

The total fitness value is calculated as the weighted sum of all estimates:

$$F = W_\mathrm{c}C_\mathrm{c} + W_\mathrm{f}C_\mathrm{f} + W_\mathrm{e}C_\mathrm{e} + \mathrm{K} \tag{5.63}$$

The exact value of the weighting coefficients W, as well as the number of estimates taken into account, is individual for each controller. As a rule, the weighting coefficients are chosen empirically. It should be noted that the signals are dimensionalised and thus the coefficients may vary significantly. For example, aerodynamic surfaces deflection is measured in degrees and may range from –16 to 16 (for rudder and ailerons); considering 10-fold saturation limits

noted above, the peak values may reach ±160. At the same time, throttle range is 0.01 to 1, thus the cost for its usage will be about 30 times as low (for similar control dynamics).

Crashing of the UAV must be handled separately, because the above estimates may not reflect this condition. Moreover, the crashed controller may receive a relatively high score (especially the flight controller, for which trajectory is not analysed) because the estimation will be based on a short record of the signals. For this reason, if a crash is detected, a penalty is assigned, but the flight time before crash t_1 is rewarded:

$$F = 120000 - 1000t_1 \tag{5.64}$$

This approach enables the controllers to 'learn to fly' at the initial stages of evolution.

The fitness value obtained using the above formulas is the *cost* or *penalty* value: smaller values are better. This must be taken into account in the selection procedure.

5.3.5 Evolutionary Design algorithm outline

For Evolutionary Design, two algorithms based on Evolutionary Strategies have been implemented in MATLAB language. One is the extension of basic ES algorithm with individual mutative step size adaptation (ISA, Section 4.4.1.1). Another is based on Hansen and Ostermeier's covariance matrix adaptation algorithm (CMA-ES, Section 4.4.1.2). However, since the individuals in the population in ED are not uniform and, moreover, grow in length during the evolution, application of the CMA-ES algorithm is troublesome. It had been used to evolve 'protected' sub-populations after structure mutations (described below), where all individuals are uniform. Nevertheless, this showed no benefits for the evolution [114]. The CMA-ES algorithm employs cumulation of the evolution path, which is effective over a longer run, whilst sub-populations are raised for maximum 20–25 generations. Therefore, only the basic algorithm has been used in the final design procedure. It is outlined in this section.

The algorithm is implemented as a function with two input arguments: population size and number of generations to run. Other, more specific parameters are initialised within the program. The algorithm provides the ability to continue an unfinished or interrupted evolution. This is commanded by supplying a special flag instead of population size. The algorithm framework is as follows.

1. Initialise all parameters. If the evolution is continued, go to step 3.
2. Create initial population.
3. Evaluate fitness of each individual.
4. Save the full state to a temporary file for emergency recovery.
5. If the required number of generations is reached, return.
6. Sort the population according to the fitness of each individual.
7. If elitism is enabled, copy the best scored individual into the new population.
8. Until the new population is filled up,
9. Select the next individual from the sorted list (starting from the elite member).
10. Every (k_s)th generation, with probability P_s, perform structure mutation of the selected individual.
11. Reproduce n offspring individuals from the selected (and possibly mutated) individual.
12. Put these n new individuals to the new population
13. Continue the loop from step 8.
14. Increase the generation counter.
15. Continue from step 3.

Several steps of this algorithm need further clarification.

Initial population is created according to the specified task. By default, it is initialised with the control laws of the form $y = const$ with randomly chosen constants. Most of the controllers, however, are initialised with more meaningful control laws. For example, tracking controllers may be initialised with the laws $y = k_1\varepsilon + k_0$, where ε is the respective error signal and the coefficients k are sampled at random (taking into account default step sizes for the respective signal).

It can be seen that *selection* is performed deterministically. This is the most commonly used way in ES. As discussed in Section 4.2.3, deterministic selection is preferable for small to moderate population sizes. The populations used in this study were of moderate size, usually 24 to 49 members. Selection pressure is determined by the number of the offspring n of each individual. The smaller n, the lower the selection pressure. For nearly all runs in this work $n = 2$, which means that half of the population is selected. This is a rather mild level of selection pressure.

The parameters determining *structure mutation* occurrence, k_s and P_s, both change during the evolution. As discussed previously, the number of structure mutations should decrease as the complexity of the controllers grows and more time to optimise the coefficients is required. The probability of structure mutation P_s is normally high in the beginning (0.7 to 1.0)

and then decrease exponentially to moderate levels (0.4 to 0.6) with the exponent 0.97 to the generation number. Default value of k_s is set according to overall complexity of the controller being evolved. For simpler single-output controllers, as a rule, $k_s = 10$; for more complex controllers $k_s = 20$. However, in the beginning of evolution, k_s is reduced by half until the generation 20 and 100 respectively (other values have also been tried in some runs).

It can be noted that the fitness of freshly mutated individuals may be lower than that of the parents, even for the neutral structure mutations. To avoid removal of these individuals in the next selection procedures, they may be protected from removal (i.e. marked for independent selection) for a certain number of generations, until the newly added potential is developed. Alternatively, they can be evolved in separate sub-populations. This approach is used in [159]. In that work, small population size with strong selection pressure is used in the main algorithm ((2,10)-ES strategy) and such protection is vital. In the current study, however, mild selection pressure and larger population sizes are employed due to great diversity of the individuals in the population. With these parameters, protection proved to be unnecessary.

Reproduction is performed simultaneously with mutation according to (4.17), as it is typically done in ES, with the exception that this operation is performed separately for each selected member, and the member's individual step sizes (stored in the strategy parameters vector) are used for mutation.

Elitism has been employed in most of the evolutions where deterministic fitness evaluation is possible. In deterministic cases, fitness of the elite member is not re-evaluated in the next generation to save time. Elitism has a substantial effect on the growth of average complexity of the population's members because it always participate in reproduction (apart from being propagated intact to the next generation). It produces the offspring (often highly fit ones) every generation, gradually displacing new, more complex individuals. For this reason, the number of structure mutations should be increased when elitism is employed (preferably by decreasing k_s rather than by increasing P_s). For $n = 2$, it is advisable to use odd population sizes when elitism is employed and even sizes otherwise.

Evolution is monitored and controlled manually. The program displays the average and current best fitness during the run. The process may be terminated when the current best fitness has reached a plateau. For the elitist strategy, however, a sufficiently large number of generations should be allowed before stagnation is determined, because elitism naturally results in plateau-like convergence graph. The program also saves full history of evolution as well as the best individuals from each generation, so that the process can be easily traced. In some cases, earlier designs may be preferred if their structure is significantly simpler whilst the performance is only marginally lower.

It is important to note that initial stages of evolution are critically important for the success of the whole process. Evolution tends to fix and develop current achievements, thus the initially found successful design has high chances to dominate in the resulting solution. The designs may be different from run to run. If the evolution stagnates at early stages, it is better to restart the process, even though a successful structure may eventually be found. However, early stagnation happens quite rarely and usually indicates inappropriate number of structure mutations or insufficient default step sizes.

5.4 Concluding remarks

In this chapter, two important topics are discussed. The first is the analysis of control required for the shipboard UAV recovery method proposed in Chapter 2. Existing control and control design techniques are outlined. Current methods of guidance and control in relevant tasks, such as conventional landing and homing guidance, are considered. Based on this, general framework of the UAV control system is designed. It consists of two separate controllers: the guidance controller and the flight controller. Altogether, they comprise six control laws, synthesis of which is the objective of Evolutionary Design.

The second topic is compilation of the Evolutionary Design methodology. The core of this technique is a specially tailored evolutionary algorithm, which simultaneously evolves both the structure and the parameters of a set of control laws. The algorithm is accompanied by the descriptions of the internal control laws representation and fitness evaluation procedure. The simulation environment used for fitness evaluation of the UAV controllers is also presented. Special attention is given to minimisation of computational requirements, because ED involves thousands of simulation runs during the evolution.

In the next chapter, the ED is applied to synthesis of the actual control laws for the UAV recovery task. It is followed by testing and verification of the designed controller, which allows to validate the new design method.

Chapter 6. Controller Synthesis and Testing

In this chapter, Evolutionary Design is used to evolve controllers for the UAV shipboard recovery task. The structure of the UAV control system, as well as the Evolutionary Design approach, are detailed in Chapter 5. It is important to keep in mind that the ED algorithm is an engineering *tool* which is employed for creative work and may exhibit complex behaviour. Furthermore, it is an essentially stochastic technique and the results are generally not replicable, even though all resulting designs may show very similar performance from run to run. The engineer is free to choose any appropriate design which shows satisfactory performance, making the decision on the basis of the engineering task in question.

After synthesising all the controllers which constitute the control system architecture, the system is tested in maximally diverse conditions to measure its robustness and performance in order to verify and validate its design. The results are presented in this chapter.

6.1 Controller synthesis

The UAV control system is synthesised in several steps. First, the flight controller (Section 5.2.2) is produced. This requires several stages, since the flight controller is designed separately for longitudinal and lateral channels. When the flight controller is obtained, the guidance control laws are evolved.

Application of the ED algorithm to control laws evolution is fairly straightforward. First of all, the simulation model must be prepared. It provides the 'testbed' for fitness evaluation. The model for the guidance laws evolution is presented in Section 5.3.3. In the model for the flight control evolution, the guidance controller is replaced by synthetic test control signals which are designed for each task individually.

Apart from the model, fitness evaluation procedure, which is a multistage process, must be written. Its basic blocks are outlined in Section 5.3.4. The evaluation procedure consists of three main steps: 1) preparation of the sample task for the controller, 2) execution of the simulation model for the given sample task and 3) analysis of the obtained performance and evaluation of the fitness value. Implementing the sample tasks for each controller (the 'lessons' of a kind) may require some effort and even creativity. Indeed, at the early stages, when the flight controller only starts to develop, real-world objectives cannot be set forth. In particular, half-developed controller is not able to cope with the full range of disturbances. Nevertheless, the sample tasks should 'teach' the controller so that a fully capable design evolves. If the system and the controller were linear, they could be tested against simple step inputs. Nonlinear design, however, requires a more elaborated approach.

Another component of fitness evaluation is the cost weighting coefficients, which determine the effect of various flight parameters recorded during simulation run upon the fitness value. Since evolutionary algorithms are particularly robust at finding global optimum, precise adjustment of these coefficients is usually not necessary. As a rule, they should reflect the order of magnitude of the influence of the respective cost component. They can be relatively easy tuned by hand in one or two test runs of the algorithm. For example, if the evolved controller shows good tracking but exhibits oscillatory behaviour (too aggressive control with low damping), the cost for control activity (C_f) may be increased (see Section 5.3.4).

When both the model and fitness evaluation are prepared, the final evolution may be started. Typically, the algorithm is run for 100–200 generations (depending on complexity of the controller being evolved). The convergence and the resulting design is then analysed and the evolution, if necessary, is continued. The evolution may be repeated a few times to obtain the most suitable design. Since the evolution process does not tend to produce parsimonious solutions (as pointed out in Section 4.1), simpler designs are often preferred, even at the cost of slightly lower performance.

6.1.1 Step 1: PID autothrottle

Keeping a safe airspeed is vital for the UAV. Before the main control laws evolution can proceed, a more or less accurate airspeed hold must be ensured.

As a rule, high degree of coupling exists between airspeed and the elevator channel. Since the throttle channel is 'slow' due to high inertia of both the aircraft and the engine, it is usually quite useful to have 'feedforward' links from the elevator channel and/or flight parameters of short period longitudinal motion. For example, nose-up elevator deflection or pitch attitude may indicate (forthcoming) climb, and throttle may be adjusted beforehand. However, at this stage, elevator control is not yet developed, and the aircraft response to elevator is not thoroughly considered. Meanwhile, working autothrottle is a prerequisite for successful elevator control. For this reason, a simple PID variant of autothrottle is first evolved. At the next stage, its evolution is continued in a full form together with the elevator control law.

The PID structure of the controller may be ensured by appropriate initialisation of the initial population and by disabling structure mutations. Therefore, the algorithm works as a numerical optimisation procedure. The structure of the autothrottle control law is following:

$$\begin{aligned} \dot{x}_1 &= k_1 x_1 + k_2 \Delta V_a \\ \delta_t &= k_3 x_1 + k_4 \Delta V_a + k_5 \dot{V}_a + k_6 \end{aligned} \tag{6.1}$$

where $\Delta V_a = V^d - V_a$ is the airspeed error signal, δ_t is the throttle position command and $k_{1\ldots6}$ are the coefficients to be optimised. Also, the low-pass output filter (Fig. 5.15) is employed.

Its bandwidth is controlled by the excluded from the state equation free term, as detailed in Section 5.3.2.2. To ensure numerical stability of the model which have fixed sample rate, this value is constrained by the range [1; 120] rad/s.

Presence of the $k_1 x_1$ term in the state equation allows to evolve the integrator anti-windup mechanism implementing low-pass filtering of the error signal. This may be especially important for throttle, because it easily reaches saturation limits. It should be noted that due to presence of close saturation limits (on both throttle position and rate), overall behaviour of the system may be far from linear, even though the PID control law is linear.

Dimensionality of the problem may be reduced without compromising the PID control as such by excluding either k_2 or k_3 coefficient (one of them may be replaced with unity). However, trial runs indicate that this rather impairs the evolution. It appears that having a common gain for the state is beneficial.

Sample task for the PID autothrottle is designed as follows. A 30-second flight is allocated for performance measurement. Such a long flight is needed because throttle response is quite slow (see Section 3.1.2.4). Since robustness of the PID controller to discrepancies in the UAV model is not topical at this stage (it will be addressed in further evolution), and because structure of the controller is fixed so that irrelevant measurements cannot be attracted, only a single simulation run for each fitness evaluation is performed.

Simple static controllers are connected to both rudder and ailerons:

$$\begin{aligned} \delta_{\mathrm{r}} &= 10 n_z + 20 \omega_y \\ \delta_{\mathrm{a}} &= 40 \gamma + 5 \omega_x + 0.4 z_{\mathrm{g}} \end{aligned} \tag{6.2}$$

where δ_{r} is the rudder deflection, δ_{a} is the ailerons deflection, n_z is the lateral load factor, γ is the bank angle, ω_x and ω_y are roll and yaw rates, and z_{g} is lateral position of the UAV in the inertial ground frame. These control laws provide near wing-level flight along the x_{g} axis (to the north) with minimum sideslip. The value z_{g} is not measurable by the onboard sensors, it is obtained directly from the simulation environment only for the longitudinal channel development. Without this term, the UAV (being a single-propeller aircraft) gradually deviates to the right due to static error of the control. In principle, this poses no problems, the term is included only for the convenience of flight path analysis. Low gain for z_{g} ensures adequate stability near the vertical plane $z_{\mathrm{g}} = 0$ while minimising roll demands. All gains in (6.2) are chosen empirically analysing test simulation flights. The disturbances in lateral motion are disabled at this stage, thus precise adjustment of these coefficients is not necessary. The most prominent disturbances come from thrust changes, which produce noticeable variations of propeller reaction and gyroscopic moments.

It should be noted that for compatibility with the original UAV model [47, 153], the aerodynamic control surfaces deflections are measured in degrees while all other angles are measured in radians. This results in seemingly large gain coefficients, although they are chosen rather small to ensure stability in all circumstances during evolution. It is also noted that the original data [153] apparently contain some errors and omissions which are propagated (again for compatibility reasons) to the current model. One of them is, apparently, excessively strong nose-up aerodynamic pitching moment m_{z0}. This is compensated by shifting the tailplane incidence angle by +7 degrees (providing additional nose-down moment). Similar correction is used in the work [47]. In addition, the recommended approach airspeed is reduced from 26 m/s to 22 m/s. At 26 m/s and flaps 40°, the UAV flies (level) at negative angles of attack with nearly maximum weight. This is not unusual in itself, but provides inadequate margin for negative stall angles while having excessive margin for positive angles. Reduced airspeed may have double effect on recovery. On the one hand, it increases cable hook sag (see Section 3.5.4), allowing greater vertical guidance error. In addition, it reduces impact speed, making capture less stressful for the airframe and more reliable. (However, as the process of capture is not modelled in this work, this is only a potential improvement). On the other hand, it reduces available load factors and slows down aerodynamic response of the aircraft (simply stated, reduces manoeuvrability) which makes guidance more difficult. It might be more beneficial for guidance to reduce flaps setting to 20° and keep the recommended approach speed. Nevertheless, the recommended flap position was retained. Varying approach speed and active flaps control may be recommended for further development of the control system.

Elevator inputs provide the main source of disturbances for training the autothrottle. A 2.5-degree nose-up step input is executed at time $t = 5$ s. It produces nearly steady climb without reaching stall angles of attack and saturation on the throttle (except for, possibly, dynamic transition moments). At $t = 14$ s, a similar nose-down elevator step is commanded, entering the UAV into glide with small descent angle. In addition, a 3 m/s tailwind gust is imposed at $t = 23$ s. Generally, manoeuvres and wind disturbances are the most prominent sources of airspeed variations, therefore they are included for autothrottle assessment.

Initial airspeed is set to 23 m/s, i.e. 1 m/s above the required airspeed. This provides an additional small scale disturbance. Initial altitude is 100 m, providing enough elevation to avoid crash for any sensible throttle control.

Also, sensors noise and turbulence are added. They provide small scale high frequency disturbances. These are especially important for correct treatment of the derivative term and adjustment of the low-pass output filter. Airspeed readings in itself are quite noisy, and the airspeed derivative, which is obtained by numeric differentiation, is even more noisy. Exclud-

ing noise characteristics may result in excessive reliance on the derivative term. Noise is applied only to the relevant measurements. A relatively mild level of turbulence corresponding to steady wind 5 m/s (see Section 3.2.2) is used along the longitudinal and normal body axes, while the lateral component is disabled. Steady wind itself is irrelevant and thus is not included.

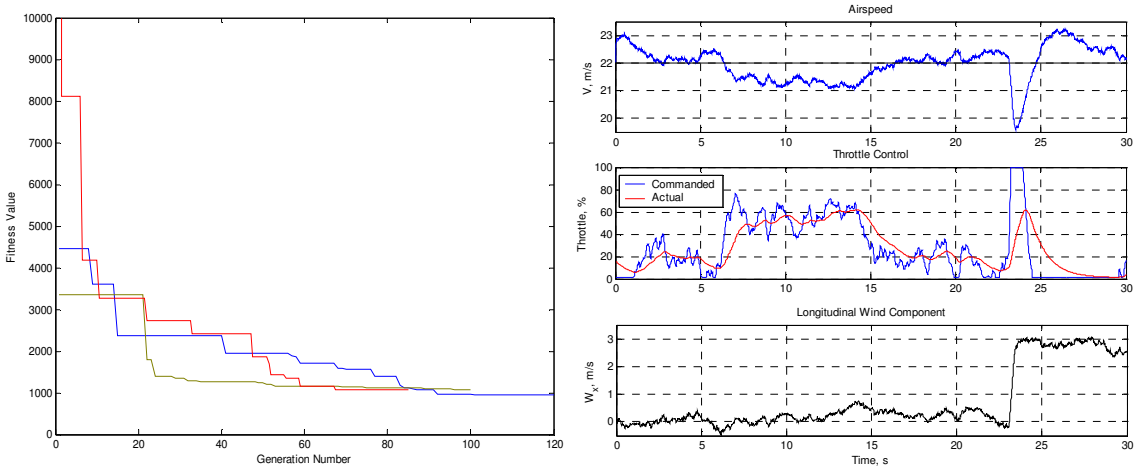
Algorithm settings are as follows. Since the structure is fixed, large population size N is unnecessary. Population size of 25 members has been used in most runs. $N = 13$ showed similar results in terms of fitness evaluation demands (red line on the convergence graph in Fig. 6.1a; adjusted to the population size 25 for comparison). Fitness is evaluated deterministically because all random signals are repeatable from run to run. Elitism is enabled. Fitness is calculated with the following weighting coefficients (see Section 5.3.4):

$$F_t = 2000C_e(V_a) + 1000C_c(\delta_t) + 2000C_f(\delta_t) \tag{6.3}$$

It should be noted that the *commanded* signal δ_t (which enters the actuator) is used for evaluation. In addition, as pointed out in Section 5.3.4, the saturation bounds are increased 10-fold for better evaluation, making [–4.5; 5.5] for throttle. The normal bounds [0.01; 1.0] are applied afterwards, within the actuator.

Convergence graphs for three independent runs of the ED algorithm are presented in Fig. 6.1a. They show the best fitness values for each generation. On average, 2000 to 2500 simulation runs is needed to achieve satisfactory performance. Because this is only an intermediate controller, full optimisation is not necessary. It will be continued in the next stage.

The best controller obtained in the experiments is the following:



a) *b)*

Fig. 6.1 PID autothrottle current-best fitness (a) and sample response (b)

$$\begin{aligned} \dot{x}_t &= -2.9966 x_1 + 85.571 \Delta V_a \\ \delta_t &= 0.012616 x_1 + 0.089324 \Delta V_a - 0.044004 \dot{V}_a + 0.23827 \\ \omega_t &= 5.5283 \end{aligned} \tag{6.4}$$

ω_t is the output filter bandwidth in rad/s. Sample response of this controller is illustrated in Fig. 6.1b.

6.1.2 Step 2: Longitudinal control

With a simple autothrottle available, elevator control may be developed. Elevator control provides tracking of the normal body load factor demand n_y^d (see section 5.2). Also, evolution of the autothrottle is continued with no restrictions on its structure.

Probably the most difficult problem in synthesis of the flight controller components is design of the sample tasks on which the controllers will be trained. In simple cases where the control law structure is known, such as that of the PID throttle evolved in the previous section, an appropriate sample task can be developed fairly easily using the experience with similar controllers. However, in general design of the sample tasks may be challenging. It is desirable that the sample task closely reflect the conditions the controller will be subject to in a real guidance task. Unfortunately, this is not possible at the early stages of multistage design process. Guidance laws are unknown when the flight controller is being designed. They can be evolved only after a capable flight controller has been produced, because the flight controller is an executive system for the guidance controller. In the meantime, a replacement pseudo-guidance task must be used for the flight controller training. However, it should be kept in mind that a too 'artificial' sample task can inure the controller to 'wrong habits' which may emerge unexpectedly in a real flight. It may be reminded that, in general, response of a nonlinear system to an arbitrary signal cannot be predicted using the response to another signal. Even though the aircraft often exhibits almost linear behaviour and its response is not unpredictable, the most general approach is attempted in this research, making as few assumptions about the system as possible (keeping in mind, nevertheless, that this is flight control as such).

As a suitable replacement of the guidance law, a simple static altitude hold is used:

$$n_y^d = 0.03(H^d - H) + \cos\theta \tag{6.5}$$

This task is similar to glidepath control (see Section 5.1.2.1.1) when altitude error is utilised. A low gain is chosen to ensure guaranteed stability. The $\cos\theta$ term takes into account gravity component, assuming that roll angles are small (see Section 5.2.1.1, (5.26)). However, real guidance in high sea will most likely require a more active control, especially closer to the recovery boom. For this reason, a synthetic 'chirp' signal $c(t)$ is added starting from 10th sec-

ond of flight. This signal represents a sine wave which frequency increases linearly from $f_0 = 0.01$ Hz at $t_0 = 10$ s to $f_1 = 0.5$ Hz at $t_1 = 20$ s. For $t < 10$ s, $c(t) = 0$, providing 'training' for smooth control. Chirp signal is often used for response analysis of nonlinear systems. Altogether, the reference input signal is

$$n_y^{\mathrm{d}}(t) = 0.03(H^{\mathrm{d}} - H(t)) + \cos\theta(t) + 0.2c(t) \ . \tag{6.6}$$

The preset altitude H^{d} is 5 m above the initial altitude $H_0 = 100$ m. This causes a small positive step input in the beginning of flight. Similarly to provisional ailerons control law (6.2), precise altitude readings are not available from the UAV sensors; they are taken directly from the UAV model state vector.

Noise is applied to all readings available for the controller. Flight time is limited to 20 s, which is close to normal final approach flight time (10–15 s), but allocates more time for better 'training' of slow throttle control. Other conditions are similar to those used for PID autothrottle evolution. These include lateral control (6.2), initial position and turbulence parameters.

An important consideration that must be taken into account is robustness of the controller to possible discrepancies between the model and the real system. Evolutionary design does not generally provide robust solutions if tests for robustness are not included in fitness estimation. Robust controllers can be evolved, however, by explicit variation of the model during evolution. This can be done either randomly or deterministically. Large number of model parameters makes it difficult to provide variability for all of them. Nevertheless, the most important parameters which are expected to have significant contribution to the control may be involved. For the longitudinal control, they include:

- both lift and drag;
- aerodynamic moment from elevator deflection;
- power plant characteristics, particularly thrust coefficient.

In the evolution of longitudinal control, these parameters are varied as follows. Three test flights in similar conditions are allocated. For each flight, the respective coefficients are varied deterministically by a moderate amount of 10% either up or down; one of the flights is carried out in default configuration. A relatively small variation is used because all these parameters are vital for the aircraft, and substantial downgrade of them (especially lift) may result in inability to continue flight. Indeed, reduction of lift coefficient by more than 15% proved to be fatal for the UAV in current conditions. In the work [47], which featured linear robust controller design for the same UAV model, reduction of lift coefficient by 5% resulted in unacceptable deterioration of performance. Therefore, it is not likely that these parameters will significantly differ from reality if the aircraft is ever going to fly. Nevertheless, the com-

bination of parameter variations is constructed so that it delivers a substantial change in aircraft characteristics. When lift coefficient is increased, drag coefficient is decreased, providing sufficient variation of the lift-to-drag ratio. Increased lift coefficient is also accompanied by reduction of propeller thrust coefficient (otherwise throttle saturates most of the time) and increase of pitching moment due to elevator deflection, which provides overreaction for normal acceleration as compared to default configuration. Overall fitness of the controller is calculated as an average of three fitness values obtained for each flight.

Elevator control law is initialised as follows:

$$\begin{aligned} \dot{x}_2 &= 0 \\ \delta_e &= k_1 x_2 + k_2 \Delta n_y + k_3 \end{aligned} \tag{6.7}$$

where $\Delta n_y = n_y^d - n_y$ is the load factor error and the coefficients $k_{1\ldots3}$ are sampled at random for the initial population. For simplicity, first-order dynamic control law is used. However, when combined with the throttle control, both control laws may include both state variables x_1 and x_2, forming a tightly coupled longitudinal control.

Inclusion of the $k_1 x_2$ term in the output equation is important, even though the state equation is initially unused. Without this term, any modifications of the state equation will have no effect on the phenotype (i.e. on the control law). This leads to attraction of meaningless 'junk code' to the state equation. In the future, nearly any attempt of the structure mutation mechanism to include x_2 to the output equation will be fatal or at least useless due to irrelevant atavistic code in the state equation.

Fitness estimation combines both throttle control fitness (6.3) and elevator control fitness. They are constructed in a similar manner with the following weighting coefficients:

$$F_{te} = F_t + F_e = F_t + 500000 C_e\left(n_y\right) + 50 C_c\left(\delta_e\right) + 5000 C_f\left(\delta_e\right) \tag{6.8}$$

The weights put slightly more emphasis on elevator control than on throttle control, because the former is more important for guidance. Autothrottle only supplements it to provide safe flight conditions.

Relatively large population size of 49 members has been used in all runs. Initial structure mutation probability P_s was set to 1.0; final value $P_s \to 0.6$ (see Section 5.3.5). Structure mutations happened every (k_s = 5)th generation for the first 20 generations, then k_s = 10 until 100th generation, then k_s = 20. Satisfactory controllers were obtained after about 100 to 120 generations, which requires approximately 15000 simulation runs. The best performing controller has the following form:

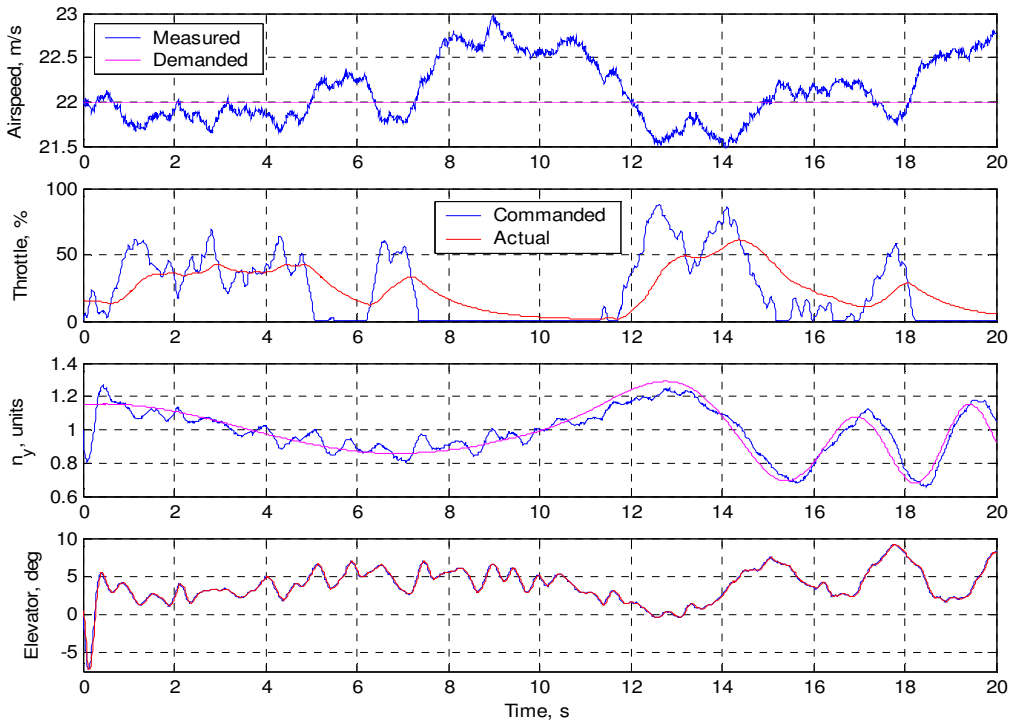


Fig. 6.2 Longitudinal control sample response

$$
\begin{aligned}
\dot{x}_1 &= \left(\left(-252.611\omega_z - 172.0765\right)\alpha - 4.411\right)x_1 + 75.0613\Delta V_a \\
\dot{x}_2 &= -0.77101x_2 - 228.8894\Delta n_y \\
\delta_t &= 0.0011588x_1 + 1.0565\Delta V_a - 0.6989\dot{V}_a + 0.24651 \\
\delta_e &= 41.5594\theta + \left(-140.8115\alpha\right)\uparrow^{0.5} + 0.89016x_2 - 26.1045\Delta n_y + 50.1835\omega_z + 3.871 \\
\omega_t &= 2.1467 \\
\omega_e &= 37.2585
\end{aligned}
\tag{6.9}
$$

ω_t and ω_e are the output filter bandwidth in rad/s for throttle and elevator respectively. The operation $\uparrow^n$ is described in Section 5.3.2.2. Refer to Appendix A for other notations. Response of this controller to the sample input (in default configuration) is presented in Fig. 6.2.

6.1.3 Step 3: Lateral control

Lateral control consists of two channels: ailerons control and rudder control. As specified in Section 5.2.2, ailerons manage the bank angle demand γ^d, while rudder is responsible for minimising the lateral acceleration (or load factor) n_z. As a rule, for an aerodynamically stable aircraft such as the *Ariel* UAV, lateral control is fairly simple and is not as vital for flight as longitudinal control. For this reason, both control laws, for ailerons and rudder, are evolved simultaneously in one step.

The set-up is largely similar to that used in the previous step. The just evolved longitudinal control laws (6.9) are connected to throttle and elevator. Both ailerons and rudder control laws are initialised in a similar manner to (6.7):

$$\begin{aligned} \dot{x}_3 &= 0 \\ \dot{x}_4 &= 0 \\ \delta_\mathrm{a} &= k_{11}x_3 + k_{12}\Delta\gamma + k_{13} \\ \delta_\mathrm{r} &= k_{21}x_4 + k_{22}n_z + k_{23} \end{aligned} \tag{6.10}$$

Static altitude hold (6.5) is employed to maintain a safe altitude. Sample input signal γ^d is designed to obtain adequate estimation of both steady state (or nearly steady state) performance and active manoeuvring. From $t = 0$ until $t = 8$ s, a 30° roll demand is supplied. At $t = 8$ s, the demand changes instantly to 10° to the opposite direction, and starting from $t = 10$ s, chirp signal $c(t)$ similar to that used in (6.6) is added. To avoid bias to either side, the direction of the first roll changes between runs, so that the roll commands in the first run are +30/−10°, in the next run −30/+10° and so on. Fitness is calculated as follows:

$$\begin{aligned} F_\mathrm{ar} = F_\mathrm{a} + F_\mathrm{r} = 50000C_\mathrm{e}(\gamma) + C_\mathrm{c}(\delta_\mathrm{a}) + 2000C_\mathrm{f}(\delta_\mathrm{a}) + \mathrm{K} \\ 200000C_\mathrm{e}(n_z) + 10C_\mathrm{c}(\delta_\mathrm{r}) + 2000C_\mathrm{f}(\delta_\mathrm{r}) \end{aligned} \tag{6.11}$$

In order to obtain a robust controller, relevant parameters of the UAV model are varied similarly to the previous case, but with an increased magnitude of 15%. The parameters involved are efficiency of both control surfaces (moment derivatives $m_x^{\delta_a}$ and $m_y^{\delta_r}$) and rolling moment due to sideslip angle m_x^β. Control efficiency is varied inversely: higher efficiency of ailerons is accompanied by lower efficiency of rudder, and vice versa. This provides more than a simply linear variation of lateral characteristics. This is further complicated by increased m_x^β for less efficient ailerons.

Algorithm settings are the same as for longitudinal control design. Fairly quick convergence (within 70–80 generations) was observed, with slow further progress. The winning solution is obtained in the following form:

$$\begin{aligned} \dot{x}_3 &= -4.9025x_3 \\ \dot{x}_4 &= -44.57\omega_x - 39.2775x_4 + \left((4.6845x_4)\uparrow^{-0.5} + 69.3478\right)n_z \\ \delta_\mathrm{a} &= 12.5873x_3 - 26.7812\Delta\gamma + 8.1188\omega_x + 0.57032 \\ \delta_\mathrm{r} &= -1.6916x_4 - 1.33n_z + 4.1764\omega_y + 21.2955\beta - 0.65184 \end{aligned}$$

Considering that the initial state is zero, state variable x_3 is always zero and not surprisingly, the ailerons are driven by simple static control law similar to (5.54). Therefore, the controller can be rewritten as

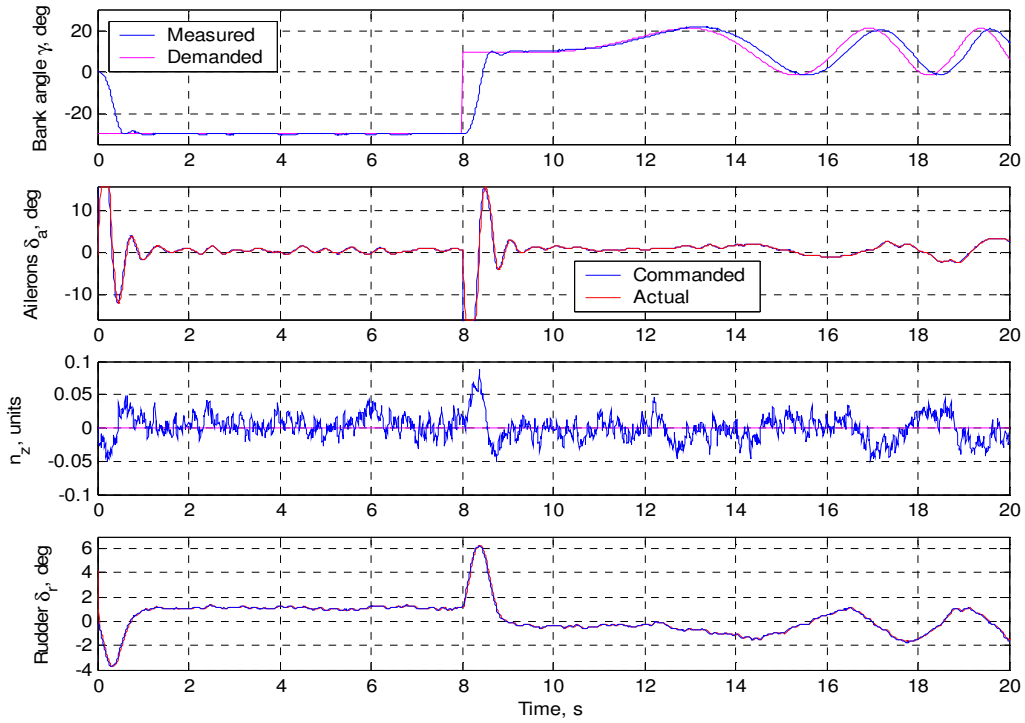


Fig. 6.3 Lateral control sample response

$$
\begin{aligned}
\dot{x}_4 &= -39.2775x_4 - 44.57\omega_x + \left(\left(4.6845x_4\right)\uparrow^{-0.5} + 69.3478\right)n_z \\
\delta_{\mathrm{a}} &= -26.7812\Delta\gamma + 8.1188\omega_x + 0.57032 \\
\delta_{\mathrm{r}} &= -1.6916x_4 - 1.33n_z + 4.1764\omega_y + 21.2955\beta - 0.65184 \\
\omega_{\mathrm{a}} &= 26.2 \\
\omega_{\mathrm{r}} &= 30.1
\end{aligned}
\tag{6.12}
$$

Sample response of this controller is presented in Fig. 6.3.

6.1.4 Step 4: Stall prevention mechanism

Active guidance control may produce high acceleration demands n_y^{d}, which may lead to exceeding the maximum allowed angle of attack $\alpha_{\max}$ and result in stall. The easiest way to avoid this is constraining the demand by a certain value $n_y^{\mathrm{d.max}}$. However, this is unfavourable for several reasons. Firstly, the maximum realisable load factor depends on a number of factors (apart from $\alpha_{\max}$), most prominently on airspeed V_{a} and aircraft weight mg. Even with autothrottle engaged, airspeed may vary substantially due to a limited amount of control (saturation bounds and slow response) and wind disturbances. Weight, at the same time, is not directly measurable and can only be estimated. Moreover, the relationship between load factor n_y and angle of attack α may vary due to uncertainties in the aerodynamic model. Therefore, it is desirable to control angle of attack directly. Secondly, when high demands n_y^{d} present, angle of attack should be kept to the maximum as close as possible (leaving a specified safety

margin, though). Excessive limiting, by a guaranteed safe $n_y^{\mathrm{d.max}}$ in particular, results in a considerable performance downgrade.

An elevator control law which, apart from tracking the load demand n_y^{d}, limits angle of attack in a specified range, could be evolved, in principle, in one step during longitudinal control synthesis (Section 6.1.2). However, this approach is inefficient and complicated. First, it would require to extend the control law structure with additional nonlinear operations such as limiters, triggers, relays, conditional statements and probably some others. Stall prevention may be considered as a triggered operation; as long as angle of attack is within safe limits, control quality (n_y^{d} tracking) should not be affected by stall prevention mechanism. However, such extension is hardly needed for other control laws. It can significantly impede evolution with little effect on control quality. In addition, it would require considerable complication of control equations encoding and more computational resources for parsing and evaluation.

Second, fitness evaluation of the control law with stall prevention would be more complex and time consuming. The sample tasks must reflect the conditions which the controller is being designed for. Therefore, apart from simulations to assess tracking quality, separate flights are needed to assess stall prevention. Separate estimation is required because these objectives are contradictory: adequate limitation of α invariably results in tracking degradation and vice versa.

Finally, the UAV model used in this study (Section 3.1.2) is inadequate for correct simulation of post-stall regimes. Evolution may choose to use post-stall angles of attack if this is not deliberately penalised and the aircraft behaves more or less consistently. For correct results, entering post-stall regimes should be explicitly avoided. Sample tasks for longitudinal control evolution (Section 6.1.2) are chosen so that angle of attack does not exceed 5–7° for any sensible control, i.e. well below the stall angle. Strict penalty for exceeding $\alpha_{\max}$ in order to evolve full control law with limiter in undesirable like any hard bound (see Section 5.3.4).

As a result, a separate control law which limits angle of attack is developed. The limiter should work only when dangerous angle of attack is reached, being deactivated in normal conditions. The simplest control law of this kind is a fixed positive (nose-down) elevator deflection triggered when $\alpha_{\max}$ is reached. This is rather crude but effective method of stall recovery. However, it represents only an interim solution. As soon as the aircraft departed from stall angles and elevator intervention is ceased, persisting acceleration demand will again cause the aircraft to stall. Even combined with a relay, this may produce unstable oscillatory behaviour. In addition, direct intervention produces uncontrolled pitching velocity and potentially excessive elevator activity. In general, this method is useful for stall *recovery* but not for stall *prevention*. It has been used in [47] to supplement the launch controller. Meanwhile, it is expected that potentially dangerous conditions leading to stall may persist over a significant

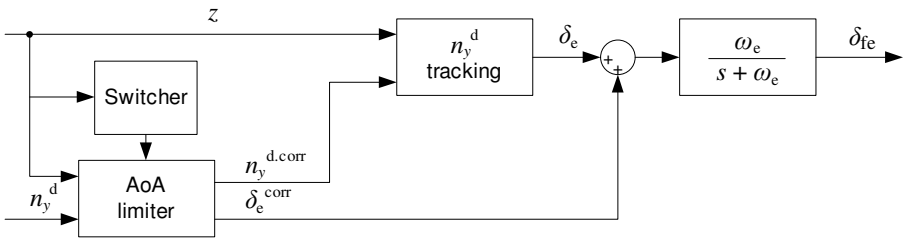


Fig. 6.4 Stall prevention mechanism (z is the measurements vector)

amount of time during final approach, in particular high acceleration demands and large scale turbulence. Therefore, a stall prevention mechanism is necessary.

Stall prevention works primarily by correcting the reference signals (n_y^{d} in this case) so that safe angles of attack are maintained. In addition, extra damping of rising high pre-stall angles of attack is provided to reduce overshoot. This mechanism consists of two blocks (Fig. 6.4): the angle of attack limiter, which holds a specified angle of attack, and the switching rule, which engages the limiter when necessary.

Stall angle of attack for the *Ariel* UAV with flaps at 40° is approximately 10 degrees. Leaving a small margin, $\alpha_{max} = 9°$ is defined. Additional damping is provided starting from $\alpha = 5°$ and progressively increases until $\alpha = 10°$:

$$\delta_e^{corr} = \begin{cases} K(\alpha)\dot{\alpha} & ,\ \dot{\alpha} > 0 \\ 0 & ,\ \dot{\alpha} \le 0 \end{cases} \tag{6.13}$$

where $K(\alpha) = 0$ for $\alpha < 5°$ and linearly increases to a maximum value K_{max} at $\alpha \ge 10°$. Damping is independent of the switching rule. K_{max} is optimised together with the evolution of the angle of attack limiter control law.

The control law is evolved in a similar manner to other laws described above. Low-pass output filtering is not used, because the output signal is eventually passed through the elevator's filter (Fig. 6.4). The objective of the control law is to maintain $\alpha = \alpha_{max}$ by correcting n_y^{d} as long as this demand is greater than required for safe flight. Also, free term is disregarded in both state and output equations.

Engaging and disengaging of the limiter is carried out by the following switching rule. The limiter is engaged immediately as the angle of attack reaches α_{max}. At this moment, the current value of n_y^{d} is 'frozen' and further corrections are made with respect to this 'frozen' value n_y^{d0}. This is done in order not to be disturbed by the changes of the n_y^{d} signal. As long as n_y^{d} is greater than the possible maximum, its value is irrelevant.

Switching the limiter off is more sophisticated. Momentary reduction of angle of attack below α_{max} is not a reliable sign of stall recovery; this may be due to dynamic oscillatory transition process. Therefore, the limiter is disengaged when the following condition is satisfied:

$$\alpha < 10°$$
$$\text{AND } n_y^{\text{d}} < n_y^{\text{d.corr}}.$$

In other words, the limiter switches off when the input demand falls below the corrected demand, but only if the angle of attack is already small enough. At higher angles of attack, small corrected demands $n_y^{\text{d.corr}}$ are allowed, because this may be needed to achieve better dynamics of the transition. A small 1° allowance (compared to the switch-on bound $\alpha_{\max} = 9°$) is given to prevent locking on high angles of attack in the case when α is kept by the limiter slightly higher than required (due to static error, for example). When the limiter is disengaged, it passes the load demand through: $n_y^{\text{d.corr}} = n_y^{\text{d}}$.

Two sample scenarios are employed to evolve the limiter control law. One involves quick 'jump' to high angles of attack by issuing a large step-like load demand. Another uses a slowly rising ramp signal so that the maximum angle of attack is reached within approximately 3 seconds. In order not to produce extreme climb angles, initial attitude is set to 10° nose down, and after 2 seconds, 45° roll is commanded. This results in almost level flight with throttle in the middle of its range.

Fitness is calculated similarly to all tracking controllers with respect to the command $\alpha = \alpha_{\max}$. Two exceptions are made, however. First, overshoot is penalised twice as heavily as undershoot, because exceeding the maximum angle of attack is considered to be hazardous:

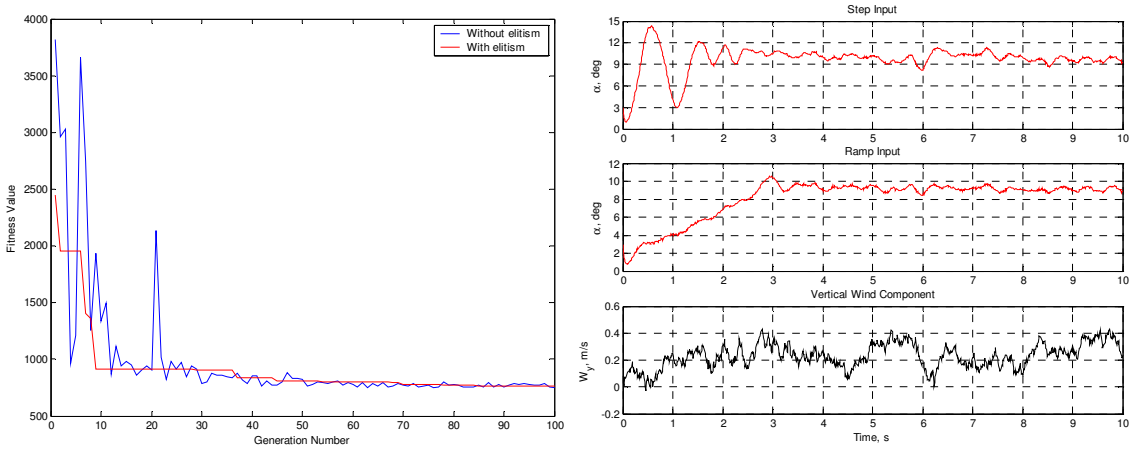
$$\begin{aligned} \Delta\alpha_i &= \alpha_i - \alpha_{\max} \quad i = 1\text{K } n \\ a_i &= \begin{cases} \Delta\alpha_i & , \Delta\alpha_i < 0 \\ 2\Delta\alpha_i & , \Delta\alpha_i \geq 0 \end{cases} \\ C_{\text{a}} &= \frac{1}{n-1}\sum_{i=1}^{n} a_i^2 \end{aligned} \tag{6.14}$$

Second, since the limiter has no direct control on elevator, the amount of longitudinal oscillations is taken into account as the variance of pitch rate. Altogether, the fitness value is calculated as

$$F_{\text{a}} = 10^6 C_{\text{a}} + 0.1 C_{\text{c}}(\delta_{\text{e}}) + 1000 C_{\text{c}}(\omega_z) + 1000 C_{\text{f}}(\delta_{\text{e}}) \tag{6.15}$$

This estimation puts more emphasis on tracking than on control usage (as compared to main elevator control law, see (6.8)), because the limiter operates in near-stall regime and quick response is necessary, even at the cost of large control activity.

Algorithm settings and initialisations are similar to the case of longitudinal control evolution (Section 6.1.2), except that smaller population of 24 or 25 members is used because only one control law is evolved. Both elitist and non-elitist strategies have been tried with largely similar results. Quick convergence, within 40–50 generations, was observed in all cases, with slow further progress. Examples of current-best convergence are presented in Fig.



a) *b)*

Fig. 6.5 Angle of attack limiter current-best fitness (a) and sample responses (b)

6.5a. It has been noted that non-elitist strategy tend to produce more complex solutions with marginally better performance. A simpler solution obtained in elitist strategy has been selected:

$$\begin{aligned} \dot{x}_5 &= \left(93.01\omega_z - 15.0798\right)\left(\alpha - \alpha_{\max}\right) \\ n_y^{\,\mathrm{d.corr}} &= n_y^{\,\mathrm{d0}} - \Delta n_y^{\,\mathrm{d}} = n_y^{\,\mathrm{d0}} - 0.63208 x_5 - 25.246\left(\alpha - \alpha_{\max}\right) \\ K_{\max} &= 60 \end{aligned} \tag{6.16}$$

The parameters $K_{\max}$ and $n_y^{\,\mathrm{d0}}$ are detailed above. Sample responses of this controller are presented in Fig. 6.5b. Quite large overshoot for step input can be explained by inadequate elevator efficiency for positive deflection (nose-down moment), probably due to excessive aircraft nose-up moment, as explained in Section 6.1.1. By the moment the angle of attack reaches $\alpha_{\max}$, elevator is already deflected fully down. The following two or three periods of oscillations are caused by alternate switching between the control laws.

6.1.5 Step 5: Guidance

At this point, flight controller synthesis is completed and guidance laws can be evolved. As discussed in Section 5.2.1, guidance controller comprises two control laws, for the vertical and horizontal load factor demands, $n_{y\mathrm{k}}^{\,\mathrm{d}}$ and $n_{z\mathrm{k}}^{\,\mathrm{d}}$ respectively. This is equivalent to acceleration demands

$$a_{y\mathrm{k}}^{\,\mathrm{d}} = g\, n_{y\mathrm{k}}^{\,\mathrm{d}}; \quad a_{z\mathrm{k}}^{\,\mathrm{d}} = g\, n_{z\mathrm{k}}^{\,\mathrm{d}} \tag{6.17}$$

These demands are passed through the kinematic converter to form the flight controller inputs $n_y^{\,\mathrm{d}}$ and γ^{d}.

It is expected that despite a substantial difference between the longitudinal and lateral dynamics of the aircraft, basic principles of guidance in vertical and horizontal planes should

be largely similar. Although operational limitations of the UAV may have significant effect on guidance, the dynamics of target motion is usually the most contributing factor. Motion of the recovery boom is considered in Section 3.4.1. It can be seen that while amplitudes of longitudinal and lateral oscillations are sufficiently different in some cases, their dynamics is similar, and even the smallest amplitude (at sea state 6) is greater than the allowed guidance error. For this reason, both guidance laws are evolved simultaneously in one step.

In a sense, evolution of guidance laws is easier prepared than that of the flight controller. This is the final objective with clearly defined goals and there is no need to prepare 'synthetic' sample tasks. The evolution can be carried out in the most comprehensive environment with 'real' (though simulated) recovery tasks. However, as noted in Section 5.3.3, several simplifications are made in order to reduce computational demands and speed up the evolution. These include replacement of the full cable model with the static model (see Section 3.5.4), absence of ship airwake model and reduction of the integration sample rate to a minimum which ensures numerical stability of integration but may be insufficient for correct simulation of noise. In addition, only one boom position is considered (Section 3.4.1).

It is important to note that when the controller is evolved in a stochastic environment with many random factors, the evolution progress becomes highly stochastic as well. Since EAs are stochastic in itself, there may seem little difference here. However, several implications should be taken into account. First, every fitness value carries less information about the individual's performance than it does in deterministic cases. Therefore, current best fitness in a generation is not a reliable indicator of the best solution. Every generation may contain a number of solutions which received good score only by chance. Only the solution which show sustained progress across generations may be considered good. However, since the solutions change from generation to generation, it is not always possible to track the solution's history back. Testing on a sufficient number of randomly chosen conditions is required to select the winning solution. This also applies to the selection procedure in the EA. It is desirable to obtain the fitness value on the basis of multiple simulation runs, reducing thereby sampling errors and making fitness value more deterministic and reliable. Excessive number of simulation runs, however, may slow down the evolution considerably, thus an optimal number should be found. In addition, relatively mild selection pressure should be used when fitness evaluation is stochastic.

Second, for the same reason, stagnation of progress of the current best fitness is not a reliable indicator of stagnation of the evolution. Every generation may produce individuals with very good fitness which will not sustain further selections. In some cases, current best fitness may appear stagnated from the very first generation. Average population fitness may be considered as a better measure of evolution progress.

Third, elitism has little sense in a sufficiently stochastic environment and is usually not employed. If it is desirable, for some reason, to use elitist strategy, fitness of the elite solution must be recalculated every generation.

In the guidance task, the random factors include:

- Initial state of the UAV. The approach corridor entry point and the position allowance, which determine initial position of the UAV, are specified in Section 2.4.2. Initial attitude is zero roll, –1° pitch, and yaw is set so that velocity is directed towards the recovery boom (i.e. side wind component is initially compensated by the navigation controller). Initial airspeed is normally distributed with mean 22 m/s and standard deviation 1 m/s. Initial controls position approximately corresponds to trim setting for steady descend with 2°40' glideslope and required airspeed (only throttle position is important due to slow response).
- Sea State. It has direct influence on ship motion (see Section 3.3.1). It is also closely related to wind magnitude and turbulence intensity.
- Atmospheric parameters, particularly steady wind and turbulence. For simulation, they are sampled as follows. Wind direction is selected uniformly within $[0; 2\pi]$, i.e. any wind direction may be chosen with equal probability. Wind magnitude at 6 m altitude is selected using normal probability distribution. The mean value is calculated as 2.5·(Sea_State) m/s (see Table 3.2). Standard deviation is 1 m/s. Wind magnitude at the actual altitude varies according to (3.33) (Section 3.2.2). At 40 m, it is approximately 1.4 times that at 6 m. It is additionally ensured that wind magnitude does not exceed 20 m/s, otherwise the UAV may never reach the ship if substantial headwind component is present. Turbulence parameters are set as specified in Section 3.2.2.
- Initial state of the ship. Since ship motion is simulated deterministically (Section 3.3.1), initial phases of all six components of periodic motion are chosen independently at random from $[0; 2\pi]$ with uniform probability to provide randomisation of ship position. Also, wave heading angle (see Section 3.3.2) is selected at random from the interval $[0; \pi]$. Ship steady motion (forward speed) is not specified at this stage, because it can be replaced with appropriate steady wind with the same effect (no measurement fed into the controllers, including positional measurements, is linked to the ground frame; ship velocity is assumed to be constant during approach).

For the evolution of guidance controller, Sea State is fixed to the maximum specified level 6. Amplitude of periodic ship motion is determined with respect to the Significant Wave Height (SWH), which is 1.6 times the average wave height (see discussion in Section 3.4.1).

It is expected that the guidance law capable to recover the UAV at sea state 6 will cope with less hostile conditions. Another consideration is that only large disturbances (especially ship amplitude) may lead to development of suitable guidance laws. Guidance to a near stationary position is trivial and can be successfully performed in a number of ways. However, it may come out that these conditions (sea state 6) are too violent for reliable recovery. This may lead to a non-convergent evolution. In this case, the conditions may be relaxed.

The above environmental conditions cover a wide range of actual conditions which the UAV may encounter during recovery. It is expected that in normal operation only a limited scope of them may be realised. For example, strong tailwind is not typical for recovery (as well as for any landing). Conservativeness of the environmental factors applied allows to evolve a controller which is robust to environmental uncertainties. Meanwhile, robustness to unmodelled dynamics of the UAV is addressed in the flight controller development. For guidance development, it is assumed that the flight controller tracks the reference commands reasonably well for any specified discrepancies in the UAV model. Therefore, only the reference UAV model is used in the guidance controller evolution. Robustness and operational envelope of the obtained solution will be determined during the testing phase.

Fitness is calculated as follows:

$$\begin{aligned} F_{\mathrm{g}} = {} & 40\Delta h_1^2 + 20\Delta z_1^2 + 50\left|\psi_1\right| + 25\left|\gamma_1\right| + \mathrm{K} \\ & 500C_{\mathrm{c}}\left(n_{y\mathrm{k}}{}^{\mathrm{d}}\right) + 500C_{\mathrm{c}}\left(n_{z\mathrm{k}}{}^{\mathrm{d}}\right) + 100C_{\mathrm{f}}\left(n_{y\mathrm{k}}{}^{\mathrm{d}}\right) + 100C_{\mathrm{f}}\left(n_{z\mathrm{k}}{}^{\mathrm{d}}\right) \end{aligned} \tag{6.18}$$

where Δh_1 and Δz_1 are vertical and horizontal miss distances, and ψ_1 and γ_1 are final yaw and bank angles (note that yaw is corrected so that zero yaw is perpendicular to the arresting wire, see Section 5.2.1.2). Greater weight for vertical miss than that for horizontal miss is used because vertical miss allowance is smaller (approximately 3–4 m vs. 5 m) and also because vertical miss may result in a crash into the boom if the approach is too low. Horizontal miss, on the other hand, is usually safer for the aircraft. Analogous, final yaw receives greater weight because it determines the amount of lateral pendulum oscillations after recovery (see Section 2.4.3), which are hazardous due to possible collision with the ship, while final roll is less relevant to the success of recovery.

Second power applied to the miss distance heavily penalises large miss while tolerates small miss. When large miss distances (greater than 2–4 m) are present, other components of the cost F_{g} are negligible. This helps to evolve general principles of guidance at the early stages of evolution and then to refine them taking into account control effort.

Additionally, crash of the UAV is treated according to (5.63). A crash is detected if

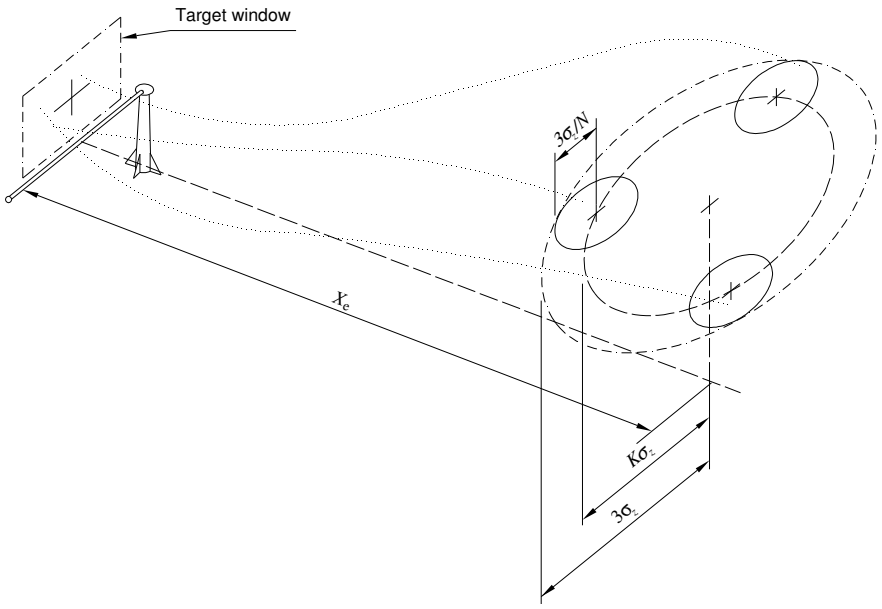


Fig. 6.6 To the explanation of distribution of the initial positions of the UAV for fitness evaluation (not to scale). Solid ellipses represent 3σ-lines of equal probability density of each position distribution. Vertical dimensions are defined in a similar manner with respect to σ_y. Dotted lines represent possible successful trajectories.

$H < h_s$, i.e. the cable hook touches the mean water level;[1]

OR $|\psi| \geq 90°$, i.e. the UAV completely diverts from the ship;

OR $|\gamma| \geq 70°$ (dangerous excessive roll. Commanded roll is limited to 45°, see Section 5.2.1.1).

$N = 3$ simulation runs are performed for each fitness evaluation. To provide a more comprehensive estimation of guidance abilities, initial positions of the UAV in these N runs are maximally distributed in space, maintaining, at the same time, a sufficient random component. This is done as follows. Nominal standard deviation of the initial position is specified in Section 2.4.2 as $\sigma_y = 5$ m for elevation and $\sigma_z = 10$ m for lateral displacement. For the purposes of fitness evaluation, these deviations are reduced N times, and the mean values are spread evenly on the ellipsis with the centre at the specified ideal position and the semi-axes $K\sigma_y$ and $K\sigma_z$, where $K = 3 - 3/N$. Position of the first point on the ellipsis is chosen at random. This way, 3σ bound of equal probability density of the original distribution is maintained (Fig. 6.6). Distance X_e is selected at random for each run as specified in Section 2.4.2.

Algorithm initialisation is the same as for longitudinal flight control laws evolution (Section 6.1.2), except that elitism is not used and the population size is 48 members. The initial population is sampled with the control laws of the form

[1] Detection of collision with a wave is not possible because waves as such are not modelled. At the same time, taking a guaranteed margin above wave crests may be too conservative: in a rough sea, the recovery boom level may periodically appear to be below the highest crest. Assuming that waves are not too high to overflow gunwales and thus there is enough elevation of the boom above the water, this condition (going below the crests) does not necessarily lead to crash. Further analysis of trajectory is needed to assess the chances of such collision.

$$\begin{aligned} \dot{x}_6 &= 0 \\ \dot{x}_7 &= 0 \\ a_{yk}{}^{d} &= k_{11}x_6 + k_{12} \\ a_{zk}{}^{d} &= k_{21}x_7 + k_{22} \end{aligned} \tag{6.19}$$

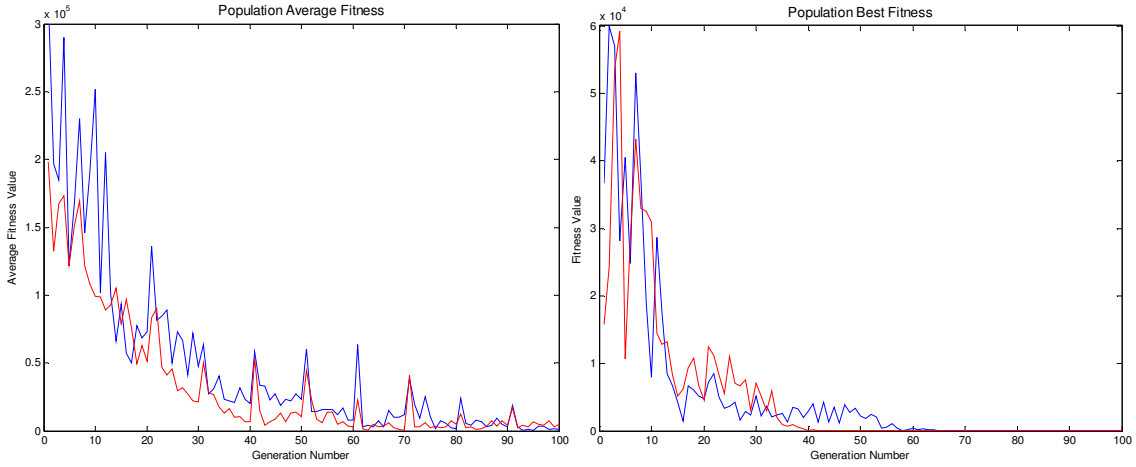
where all coefficients k are chosen at random. Necessity of the state variables in the output equations is explained in the annotation to (6.7). For convenience, the control laws are expressed in terms of accelerations, which are then converted to load factors according to (6.17). Since these control laws effectively represent 'empty' laws $y = const$ (plus a low-pass output filter with randomly chosen bandwidth), structure mutation is applied to each member of the initial population. Further structure mutations are performed as specified in Section 6.1.2. With such initialisation, no assumption about possible guidance law is taken.

The results obtained from several consecutive runs of the ED algorithm are rather expectable but nevertheless interesting. Quickly, within 30 to 50 generations, the algorithm invariably discovers the core of proportional navigation guidance law (PN, Section 5.1.2.2.1) and then only refines it. Population average and current-best convergence graphs for two most different runs are shown in Fig. 6.7. Peaks on the average fitness convergence graph represent the effect of structure mutations. It can be seen that even when the best fitness stagnates, progress of the population continues for at least 30–40 generations.

Removing unused state variables, the basic structure of proportional guidance is obtained as

$$\begin{aligned} a_{yk}{}^{d} &= k_{11}\omega_v + k_{12} \\ a_{zk}{}^{d} &= k_{21}\omega_h + k_{22} \end{aligned} \tag{6.20}$$

where k_{11}, $k_{21} < 0$ due to sign convention of the measurements (position and angles are meas-



a) *b)*

Fig. 6.7 Population average fitness (a) and best fitness (b) for two independent evolutions of guidance controller

ured with respect to the ship), and k_{12}, $k_{22} \approx 0$. Refer to Section 5.2.1.2.1 and Appendix A for other designations.

Most of the control laws produced also contain velocity component in the horizontal guidance law, which may be interpreted as a damping term:

$$a_{z\mathrm{k}}{}^{\mathrm{d}} = k_1\omega_{\mathrm{h}} + k_2 V_{z\mathrm{T}} + k_3 \tag{6.21}$$

Two of the laws also include integral of the target angular velocity. It may be noted that airspeed or closing velocity have never been included in any resultant guidance law.

From the final populations, the best solution is identified by calculating fitness of each member using $N = 25$ simulation runs, taking into account also success rate (the percentage of successful recoveries. The controller which has substantially better success rate may be preferred over that with better average fitness). Two best performing controllers from different runs are the following (unused state variables are removed):

$$\begin{aligned} \dot{x}_6 &= -3.548\omega_{\mathrm{v}} \\ a_{y\mathrm{k}}{}^{\mathrm{d}} &= 0.916x_6 - 54.3\omega_{\mathrm{v}} - 0.1261 \\ a_{z\mathrm{k}}{}^{\mathrm{d}} &= -107.68\omega_{\mathrm{h}} + 0.0534V_{z\mathrm{T}} + 0.4756 \\ \omega_{\mathrm{fv}} &= 53.52 \quad \omega_{\mathrm{fh}} = 81.89 \end{aligned} \tag{6.22}$$

and

$$\begin{aligned} a_{y\mathrm{k}}{}^{\mathrm{d}} &= -56.036\omega_{\mathrm{v}} - 0.424 \\ a_{z\mathrm{k}}{}^{\mathrm{d}} &= -182.47\omega_{\mathrm{h}} + 0.3079V_{z\mathrm{T}} + 0.608 \\ \omega_{\mathrm{fv}} &= 11.845 \quad \omega_{\mathrm{fh}} = 42.085 \end{aligned} \tag{6.23}$$

where ω_{fv} and ω_{fh} are output filter bandwidths for vertical and horizontal guidance laws respectively. The first controller achieved 100% success rate in the 25 test runs with average fitness 69.52 (it is worth noting that this controller was only 18th best based on fitness in its final population; at the same time, the best scored controller showed only 88% success rate and fitness 110.1. This illustrates the amount of uncertainty present in the environment). The second controller, (6.23), demonstrated slightly worse but still acceptable result with 92% success rate and fitness 98.24.

Considering dominance of Proportional Navigation driven solutions, it would be reasonable to employ and optimise the pure proportional navigation (PPN) law for recovery and to compare it with the obtained solutions. If PPN delivers an adequate level of performance and robustness, it can be used instead of the above laws as a simple and effective guidance method which uses minimum number of measurements.

The proportional guidance is implemented in the following form:

$$a_{yk}^{\,d} = k_1 V_{CLa} \omega_v$$
$$a_{zk}^{\,d} = k_2 V_{CLa} \omega_h \qquad (6.24)$$

Low-pass filtering is employed as usual, the bandwidths are optimised in the evolution process together with the navigation constants k_1 and k_2. Average closing rate V_{CLa} (5.53) is used as the most suitable measure of trajectory velocity (out of available measurements), which is measured with respect to the ship for the recovery task.

For this evolution, small population with 12 members is used because the structure is fixed (structure mutations are disabled) and the number of parameters to be optimised is only four. In addition, this structure can be rigidly embedded into the controller, which eliminates the need of parsing of the control equations and speeds up simulation more than twice. This allows to use more simulation runs to obtain a fitness estimation and thus to reduce the amount of uncertainty in selection.

Two evolutions with $N = 4$ and $N = 10$ runs have been performed. In both cases, current-best fitness shows little information about the progress of evolution. Only for $N = 10$ some progress is observable during the first 30 generations (Fig. 6.8). In every generation there are controllers which deliver acceptable fitness values (less than about 100) in the specified N runs. At the same time, population average fitness is not very illustrative as well.

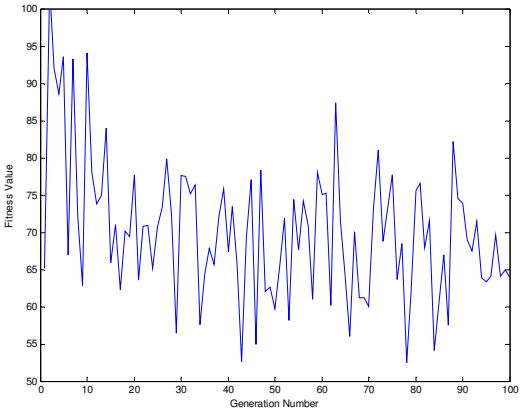
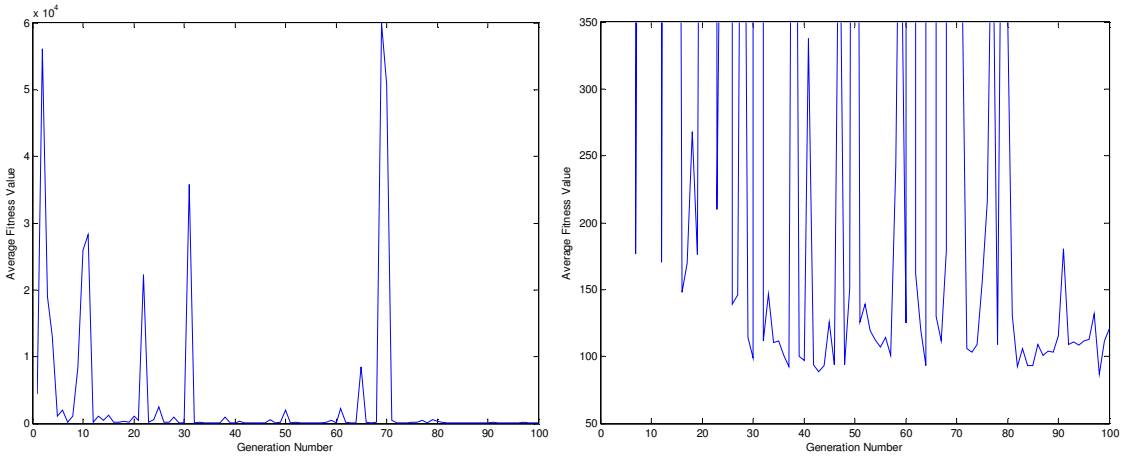


Fig. 6.8 Population best fitness in PN optimisation with N = 10

From time to time, evolution produces unstable controllers which deliver very large miss (or even crash) with correspondingly large fitness values. Even a single unstable controller extremely biases the population average fitness (Fig. 6.9a). However, ignoring the peaks produced by this bias, a general picture of evolution progress is observable (Fig. 6.9b). It can be seen that progress is apparent during the first 40–45 generations, although it stabilises long after that (near 80th generation in this case). A better readable convergence could be obtained by excluding the unstable controllers (with fitness value greater than about 1000) from average calculation.

Alternatively, stability of the controllers in the population can be enforced by constraining the navigation constants being optimised in a sensible range (for example, [–6; –2]). However, this approach significantly degrades the evolution process—which may seem counterintuitive. Nevertheless, as discussed in Section 4.2.2, handling constraints within the algorithm is often undesirable. In the case of ES (which class the ED algorithm belongs to), rigid



a) *b)*

Fig. 6.9 Population average fitness in PN optimisation (N = 10): full range (a) and restricted to meaningful values (b)

restriction of objective parameters results in that the individuals with unsuitable strategy parameters have high chances to survive. In general, exactly the strategy parameters are responsible for producing out-of-scope objective values. As a result, evolution of strategy parameters is nearly blocked. In the case of PN optimisation, this is further complicated, apparently, by vaguely defined optimum: different navigation constants may produce controllers with very similar accuracy.

The best PN controller have been selected using the fitness values of each member of the final population averaged over 100 simulation runs. It is the following:

$$\begin{aligned} a_{\mathrm{yk}}{}^{\mathrm{d}} &= -3.27 V_{\mathrm{CLa}} \omega_{\mathrm{v}} \\ a_{\mathrm{zk}}{}^{\mathrm{d}} &= -3.18 V_{\mathrm{CLa}} \omega_{\mathrm{h}} \\ \omega_{\mathrm{fv}} &= 106.5 \quad \omega_{\mathrm{fh}} = 21.7 \end{aligned} \tag{6.25}$$

This controller received the fitness 88.85 and showed success rate 95%.

In order to obtain a less stochastic evolution and potentially a more refined solution, PN law has also been optimised in a more benign environment, using Sea State 2 (‘smooth sea’) set-up. Unlike the case of full evolution of guidance laws, there is no risk to evolve an incapable (in rougher conditions) law, because the structure is fixed. However, although sufficiently steadier evolution indeed has been observed, the obtained solutions were highly optimised only for calm environment (showing fitness about 9–12), demonstrating poor results at Sea State 6 (fitness 180–2500, success rate not more than 84%). In particular, all controllers had low output filter bandwidth (mostly near the limit 1 rad/s), which reduces control activity but impedes active manoeuvring.

Concerning the output filters, it can be concluded, by comparing (6.22), (6.23) and (6.25), that the environment provides inadequate means for precise adjustment of the band-

widths. This is partly due to insufficient simulation of noise (see Section 5.3.3) and also emphasis on guidance accuracy. It is likely that as long as the bandwidth is not too narrow (greater than 10–20 rad/s), the filter has little effect on guidance accuracy. Indeed, trajectory parameters change relatively slowly, thus the noise can be filtered out easily with any appropriate filter. At the same time, guidance output has only indirect effect on control activity: it is processed by the flight controller and passed through its filters. To confirm this proposition, the controller (6.25) has been tested in the same conditions with $\omega_{fv} = \omega_{fh} = 15$ rad/s. The results are very similar to the original controller: fitness 94.36 and success rate 96%, which is within the statistical error. For this reason, this bandwidth will be used for all guidance controllers during the testing phase.

The controllers (6.22) and (6.25) are selected as the best obtained solutions of the guidance controller synthesis. They will be used in the following section for thorough testing of the whole system. Many other solutions from the final generations, including (6.23), have comparable performance and can be employed if the two selected controllers appear to have unsatisfactory characteristics.

6.1.6 Possible hardware implementation issues

The implementation of any controller is an important part of the design process which should receive due attention. In this research, a physical implementation was not possible due to a lack of hardware. However, possible issues which may arise during hardware implementation should be addressed. Some of them have been already discussed in the relevant sections and thus they are only summarised here. Unlike the majority of design methods, the Evolutionary Design enables to take most of the issues into account at the stage of controller synthesis, provided that they are included in the simulation model. Nevertheless, not all system parameters can be modelled at this stage due to either computation cost considerations or lack of actual data.

Computational engine. Even though continuous form of control equations is used, the controller is designed with digital onboard implementation in mind. Nowadays, digital implementation offers the most effective solution even for relatively simple systems. However, computing power of the onboard processor can be significantly lower than that of the computer used for simulation. On the other hand, faster specialised devices such as digital signal processing (DSP) units may be available. If the actual hardware for the onboard controller is specified, it can be incorporated into the simulation model from the beginning, or can be used for detailed testing of the developed controller. Cross-platform simulation is a well established technology today and should pose little problem. With appropriate hardware interface, even the actual controller may be used in the simulation loop if necessary. In this research,

however, implementation requirements were not specified, thus only a general mathematical model has been used.

Computing power of the onboard computer must be adequate to provide real-time calculation of all built-in control laws with the rate (frequency) that ensures numerical stability and provides minimum delay. However, this rate may be sufficiently lower than that used for the simulation. As discussed in Section 5.3.3, high sample rate is needed for correct simulation of noisy signals and physical elements with fast dynamics, such as actuators and sensors. In a real system, their simulation is unnecessary, and the controller can be executed with lower frequency, especially if signal filtering (a potentially demanding procedure) is left to external specialised processors. The optimal rate, nevertheless, should be determined carefully. A too low rate may lead to catastrophic degradation of performance, while a too high rate demands excessive resources and may produce round-off and other computational issues. Different sample rates can be simulated within the simulation model used for evolution and/or testing of the controller. In particular, Simulink software package allows to simulate multi-rate systems in a single framework.

Controller activation and deactivation. The control is handed over to the recovery controller from the mission (or navigation) controller. The latter is supposed to guide the UAV to the area specified in Section 2.4.2 and bring the aircraft to the landing configuration. An additional but necessary condition is reliable reception of the signals of the positioning system (Section 2.4.3.4.3). As soon as these conditions are satisfied, the recovery controller may be activated. However, since initial state of the controller is zero and position of the controls may vary, transition between the regimes may be somewhat brutal. In the beginning of final approach, this is not expected to be a problem, because the controller has plenty of time to recover. In this study, immediate switching is simulated. Nevertheless, a more complicated method can be employed to ensure a smooth switch. One of the examples is weighted blending (crossfading) of the outputs of the two controllers, gradually passing control from one to the other within a certain time.

Controller deactivation is discussed in Section 5.2.1.3. At a specified distance from the arresting wire, the acceleration demands of the guidance controller are replaced with zero. This provides a smooth transition, which is very important a few moments before capture. If miss is detected, the control is handed over back to the navigation controller, which performs go-around manoeuvre. For physical design, reliability issues associated with controller activation and deactivation should receive close attention. Backup means, including, possibly, manual control should be provided for both miss detection and controller activation/deactivation.

Another reliability issue is loss of the positioning signals. This may happen not only in the case of equipment failure, but also due to environmental factors such as obstruction of the

antennas by a high wave or ship installations. Since positioning signals are crucial for guidance, even short-term loss of the signals must be handled methodically. As a rule, temporary loss (up to about a second in duration) should not lead to failure of recovery, unless the loss is detected only a few metres before the boom. In the latter case, however, little can be done. Probably the best strategy is to keep the last commanded direction, which is achieved the same way as deactivation of the controller is handled, by issuing zero acceleration demands (5.54). This can be elaborated by keeping the last issued acceleration demands for a few fractions of a second in order to handle radio interference and momentary loss of the radio signals (milliseconds in duration). Maintaining the demands for a longer time may lead to extreme deviation of the flight path. This is especially dangerous if descend has been commanded before the loss is detected. For this reason, a positive vertical acceleration demand may be supplied to prevent further descend and possible crash. If reception is not regained within a specified time, missed approach is detected and go-around is performed. All these cases should be thoroughly investigated for actual design of the recovery controller.

6.2 Controller testing

Testing is a crucial stage of any controller development process as the designer and end user must be satisfied that the controller meets its performance requirements. In addition, an allowable operational envelope must be obtained. Testing of a developed controller can be performed either within the physical system (in flight or static) or in a comprehensive simulation environment. Either method has several advantages and disadvantages. As a rule, both of them are employed in the development of a real system, with the physical tests performed largely to confirm the results of the preceding simulation tests.

In physical implementation, the controller is working on a true model and hence there are no modelling errors to introduce uncertain effects into the system. In addition, physical testing accurately reflects the operational use of the controller. On the other hand, physical testing involves large time and cost demands for multiple tests. Moreover, the failure of the controller at any stage may lead to a dangerous situation and even to loss of the aircraft. Another disadvantage is that the testing conditions will most likely not provide the full range of the specified operating conditions. As noted above, lack of hardware does not allow to carry out physical tests for this research, thus only simulation tests are performed.

Testing within a simulation environment has the benefit that many different tests may be applied rapidly with no severe consequences in the event of controller failure. The whole range of operating conditions may be tested, with different combinations of any selected conditions. On the downside is that potentially large modelling errors may be introduced, which may bias the results and cause an otherwise excellent controller to fail when implemented

physically. In addition, testing situations which cannot occur in practice are possible in the simulation environment.

In this work, two main types of simulation tests are conducted. The first type is a *robustness* test. It is aimed at ensuring the controller has good robustness to modelling uncertainties and to test whether the controller is sensitive to specific perturbations. The second type is a *performance* test, where operating conditions are varied and the performance of the controller is evaluated. In a large part, the tests are consistent with the approach taken in [47] for development of the launch controller for similar UAV model. The details of these tests are presented in the following sections. These details mainly contain combined statistical results and do not present individual cases. However, samples of several different trajectories are given in Fig. 6.11 to Fig. 6.18 and Fig. 6.20 to Fig. 6.21.

6.2.1 Simulation model

One of the advantages of Evolutionary Design is that the design process is built around testing of largely the same simulation model as used for final testing. Ideally, fitness evaluation of a candidate solution is performed during evolution considering the same objectives as for the final testing and in a similar manner. Therefore, final testing is needed mostly for verification purposes, to ensure that evolution indeed produced a controller which satisfies design requirements. However, due to highly stochastic environment and high computational demands, thorough testing during the synthesis stage is troublesome. For this reason, a full-scale simulation testing is required.

In Section 5.3.3, the simulation environment used for evolution of the guidance controller has been described. For the final testing, the same model is used, although it is upgraded to include more elements. On the top level (Fig. 5.16), however, the models are identical. The modifications include the following:

- All control laws are embedded into the controllers, avoiding parsing of the equations at each time step and saving calculation time.
- Ship airwake model is included into the Wind subsystem. It supplements the existing steady wind, turbulence and gust models.
- Static cable model is replaced with the full dynamic model (Section 3.5). They have similar interface, plus the full model uses exact position of the arresting wire (obtained from the simulation environment) to detect successful capture or miss. Since this model is the most computationally demanding part of the simulation environment, only 8 lumped masses with gradually decreasing link lengths towards the end are used to represent the cable. Initial state of the cable corresponds to steady level flight with the recommended approach airspeed of 22 m/s.

- Simulation sample rate is increased to 200 Hz. This improves adequacy of the noisy signals and ensures robustness of the controllers to change of sample rate.

Every component of the Wind subsystem (steady wind, turbulence, ship airwake, gusts) can be enabled or disabled separately to provide simulation of different scenarios.

6.2.2 Robustness tests

Testing the controller for robustness to model uncertainty involves perturbing the model and determining the effect upon controller performance. The perturbations can be performed in several ways. Physical quantities such as mass and wing area can be changed directly. Dynamics of the system can be varied by introducing additional dynamic elements and by changing internal variables such as aerodynamic coefficients. For a realistic test, all perturbations should be applied simultaneously to identify the worst case scenario. However, single perturbation tests (the *sensitivity analysis*) allow to analyse the degree of influence of each parameter and help to plan the robustness test more systematically. Therefore, single perturbation tests are first performed. After that, multiple perturbation tests are carried out, where all parameters of the model are perturbed randomly within the identified limits.

6.2.2.1 Single perturbation tests

In this type of test, a single model variable is perturbed by a set amount and the effect upon the performance of the controller is determined. A crucial element of the test is therefore evaluation of the performance and determining the threshold where the performance becomes unacceptable.

Performance evaluation can be measured in a manner similar to fitness evaluation of the guidance controller (equation (6.18)). However, since performance of the whole UAV controller must be assessed and its internal signals are unimportant, the last four terms (which penalise activity of the output signals $n_{yk}{}^{d}$ and $n_{zk}{}^{d}$) may be excluded. Nevertheless, control activity must be taken into account. Many of the model parameters have a significant effect on control activity while having little effect on guidance accuracy. For example, as will be illustrated later, delay of the measurement signals may produce unacceptable oscillatory behaviour of the controller, even though still providing safe recovery. Therefore, the cost of control activity is included the same way as it has been used for the flight controller evolution (see (6.8) and (6.11)).

In addition, two other penalties are applied in order to obtain a better assessment of the trajectory parameters. Changing of some of the model parameters may significantly degrade capability of the autothrottle. Whilst decreased airspeed will result in degradation of trajectory and even a crash (due to proximity of stall regimes and reduced manoeuvrability), overspeed

may be left unnoticed if only the final miss is taken into account. At the same time, the recovery gear is designed to absorb only a limited amount of energy. The maximum allowed impact speed V_{imp} in this work is assumed to be 30 m/s (see Section 2.4.3). Therefore, the impact speed (the relative speed of the UAV with respect to the recovery boom at the moment of capture) over this limit is heavily penalised, so that overspeed of about 2–2.5 m/s renders the recovery attempt unsuccessful even for otherwise perfect approach. Considering that the required approach airspeed is 22 m/s, this is rather a generous limit which normally enables the approach with sufficient tailwind component.

Another additional parameter taken into account in performance measurement is the minimum altitude H_{min} attained during the approach. It is expected that some of the model variations, for example increasing mass and reducing wing lift capabilities, may result in a too low approach, at the same time providing accurate terminal guidance by slightly more aggressive manoeuvring. Nevertheless, low approach increases the risk of a collision with high waves. As specified in Section 6.1.5, crash is detected if altitude falls so that the hook touches the mean water level (approximately 4 m). However, highest waves may reach 7.5 m and even more (see Table 3.2 in Section 3.3.1) at Sea State 6.[1] Considering that the reference level of the recovery boom is 16 m and allowing sufficient margin for vertical travel, the altitudes below 10 m will be penalised.

Altogether, the performance cost (PC) is calculated as follows:

$$\begin{aligned}\mathrm{PC} = {} & 40\Delta h_1^2 + 20\Delta z_1^2 + 50\left|\psi_1\right| + 25\left|\gamma_1\right| + 50 f_{\mathrm{V}} + 20 f_{\mathrm{H}} \mathrm{K} \\ & 10C_{\mathrm{c}}\left(\delta_{\mathrm{a}}\right) + 10C_{\mathrm{c}}\left(\delta_{\mathrm{r}}\right) + C_{\mathrm{c}}\left(\delta_{\mathrm{e}}\right) + 200C_{\mathrm{f}}\left(\delta_{\mathrm{a}}\right) + 200C_{\mathrm{f}}\left(\delta_{\mathrm{r}}\right) + 200C_{\mathrm{f}}\left(\delta_{\mathrm{e}}\right)\end{aligned} \tag{6.26}$$

where

$$f_{\mathrm{V}} = \begin{cases} V_{\mathrm{imp}} - 30 & , V_{\mathrm{imp}} > 30 \\ 0 & , V_{\mathrm{imp}} \le 30 \end{cases}, \quad f_{\mathrm{H}} = \begin{cases} 10 - H_{\mathrm{min}} & , H_{\mathrm{min}} < 10 \\ 0 & , H_{\mathrm{min}} \ge 10 \end{cases} \tag{6.27}$$

Impact speed V_{imp} and minimum altitude H_{min} are measured in m/s and metres respectively. Other designations are as in (6.18) (see also Section 5.3.4). Unlike fitness evaluation in the flight controller evolution (Sections 6.1.1 to 6.1.4), the commanded control deflections δ_{a}, δ_{r} and δ_{e} are saturated as required for control actuators. Throttle command is not included because it has very little impact due to slow response.

The absolute value of the PC obtained using (6.26) is not very illustrative for comparison between the results. Smaller values indicate better performance, but the ‘ideal’ zero value

[1] Considering average height of the highest 1/10 waves. Table 3.2 presents the heights of the highest 1/3 waves (the Significant Wave Height), which is 1.275 times as low. Note that the height of a wave is defined as the vertical distance between trough and crest, i.e. double the amplitude assuming harmonic waves.

is unreachable because a minimum level of control activity is always present even in very calm environment. For this reason, a *Normalised Performance Cost* (NPC) will be used:

$$\mathrm{NPC}=\frac{\mathrm{PC}}{\mathrm{PC}_{\mathrm{ref}}} \tag{6.28}$$

where $\mathrm{PC}_{\mathrm{ref}}$ is the reference Performance Cost obtained for the reference (unperturbed) model with the guidance controller being considered. NPC > 1 indicates deterioration of performance. However, the performance with the perturbed model may be better than the reference performance, thus NPC < 1 is also possible.

The next question to be considered is which scenarios to use for performance evaluation. In Section 6.1.5, where the guidance controller is designed, it has been shown that the environment delivers a great deal of uncertainty. To obtain a reliable estimate of performance, several tens (up to a hundred) of simulation runs in various conditions should be performed at every point. However, this would imply a prohibitively high computational cost, as this test needs to be carried out at about 20 to 60 points for each of more than 40 model variables for each controller to be estimated. Meanwhile, there is no single 'typical' scenario that could encompass most of the real flight conditions.[1] For these reasons, four different scenarios will be used for evaluation of a single Performance Cost. These scenarios are selected so that most of the possible real world conditions are covered. The range of disturbances and the initial ship phase are chosen to provide a moderately conservative estimation. All random parameters (which include the turbulence time history and the ship initial state) are reproducible between the PC estimations, therefore PC is calculated deterministically. The overall PC is obtained by averaging the individual PCs between the four recovery scenarios. These scenarios are the following:

1. *Calm environment.* No wind, no turbulence; initial position of the UAV is at the ideal reference point (Section 2.4.2): distance 300 m, elevation 14 m, zero sideways displacement. A small amount of ship motion corresponding to Sea State 2 (SWH)[2] is, however, included. This scenario is useful to analyse the performance in benign environment and also to soften the conservative bias to the difficult conditions (the other three scenarios).

[1] For robustness tests of the launch controller in [47], a single 'middle of the range' launch scenario has been used. However, even though the launch case is simpler than recovery (the longitudinal channel is the most important for successful launch), a single scenario with tailwind did not provide adequate estimation of robustness for certain parameters. Although the most critical perturbations are believed to be estimated adequately, some side effects of such a limited evaluation are observable. For example, the tests showed that ailerons' efficiency (rolling moment due to ailerons deflection $m_x^{\delta a}$) could be reduced to almost zero without a negative effect on launch. If crosswind had been included in the testing scenario, this mishap would be discovered.

[2] The reference (SWH) after a Sea State definition denotes that the amplitude of motion is set with respect to the Significant Wave Height at the specified Sea State. See Section 3.3.2 and also discussion at the end of Section 3.4.1.

2. *Tailwind and a high start.* High tailwind is normally not expected during approach because it is usually preferable to choose the opposite direction of approach (with headwind) if tailwind component is present. Therefore, a moderately low tailwind of 5 m/s is used. It is important to note that similarly to the guidance controller evolution set-up (see page 250), the wind setting (here and in the other scenarios) defines the wind magnitude at 6 m altitude. Wind magnitude at the actual altitude varies according to (3.33) (Section 3.2.2). At 40 m, it is approximately 1.4 times that at 6 m. Turbulence and ship airwake parameters are set according to the wind magnitude as specified in Section 3.2.2. Ship motion corresponds to Sea State 5 (SWH) ('rough sea'). The tailwind is combined with a 10 m too high initial elevation (total elevation above the recovery boom is 24 m instead of 'ideal' 14 m, i.e. two specified standard deviations; with the 16 m boom height, the initial altitude is 40 m) and a 1 m/s excess of airspeed (total 23 m/s). In addition, the initial position is displaced 20 m (again two standard deviations) to the right. This combination induces a rapid descent. Even for the reference model, throttle is saturated on the idle setting almost all the time during the approach and the UAV nevertheless accelerates. Any model perturbation which produces less drag is likely to attain an overspeed penalty.
3. *Headwind and a low start.* This case is opposite to the previous one. Wind magnitude is 10 m/s (at 6 m), which is approximately 65% the absolute maximum the UAV (with the fixed autothrottle setting) can cope with in order to reach the ship.[1] Typical flight time in this scenario is about 30 seconds, compared to less than 14 s in the first scenario and 10 s in the second scenario. Initial position is 10 m below and 20 m to the left of the ideal position. Initial airspeed is 1 m/s lower than required. Sea State 5 (SWH) is set for the ship motion. Turbulence intensity is defined similarly to the previous scenario. This scenario is specific in that the flight path is likely to go below the recovery boom level and thus the UAV may need to climb in order to be recovered.
4. *Crosswind.* Wind magnitude, Sea State and turbulence settings are the same as in the previous scenario. Wind direction is 90° from the right (with respect to the recovery boom, i.e. for the approach direction to the north, easterly wind is applied). Initial position is displaced 20 m horizontally in the direction of the wind, i.e. to the left, which requires the UAV to 'overcorrect' the crosswind. Initial altitude and airspeed are exactly as specified (14 m elevation and 22 m/s respectively).

[1] Considering the variations of wind magnitude with altitude.

The reference trajectories (amongst several others) for these scenarios are shown in Fig. 6.11 to Fig. 6.14 (blue lines) and also in Fig. 6.15 to Fig. 6.18.

The first test determines the robustness to time delays in the measurement signals. This test will be used as an illustration of the method employed, with only the results presented for the following tests. The threshold of the acceptable performance will be illustrated in this test as well.

Time delays occur in a physical system due to several factors, which include delays associated with the measurement devices, delays in encoding, decoding and transmitting the signals to the controllers, delays in controller computation and delays in the actuator systems. In a well designed system, these delays should be small and in a well designed controller should have minimal effect on the performance.

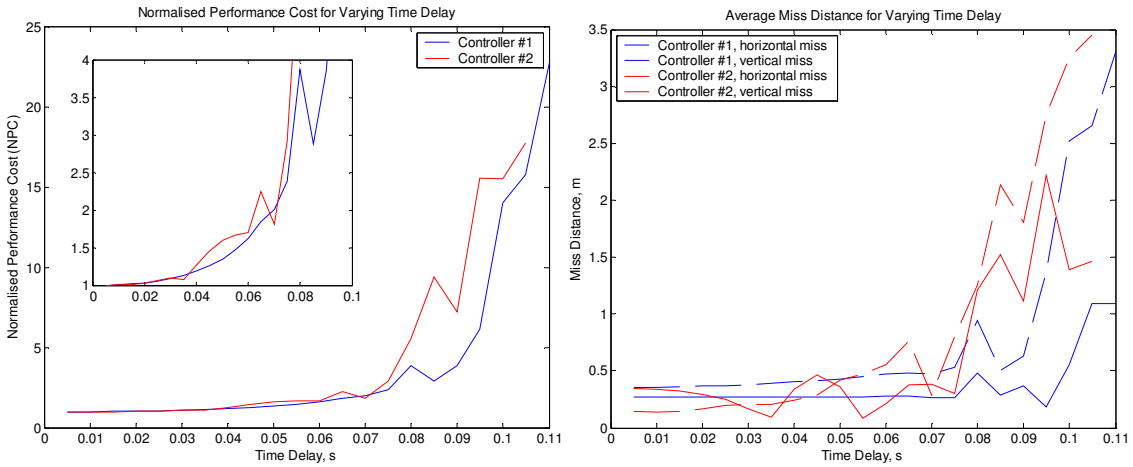
Some of these delays (or lags) are modelled within the respective subsystems of the simulation environment. These include sensors and control actuators models (see Section 3.1.2.4). In addition, integration of the control equations adds one time step delay (0.01 s for the controllers synthesis and 0.005 s for testing). It is general practice to lump all other unmodelled delays into a single delay applied either to the measurement signals or to the control output. In this case the delay has been applied to the measurement signals. For the controller synthesis, the delay was 0.02 s, providing the pure processing delay between the measurement signals and the control output of 0.03 s (30 milliseconds). Adding to that the lags present in the sensors model and actuators model,[1] this provides an adequate and rather conservative approximation of the delays expected in a real system. For this reason, controller synthesis did not include specific means to evolve robustness to time delays. The more important is to analyse the actual robustness of the controllers.

In the robustness test procedure, the time delay is varied between the minimum value 0.005 s (5 milliseconds)[2] and a value that would reflect the absolute maximum that this variable would achieve. The test stops when unacceptable performance is detected. This happens when a specified threshold NPC value is reached or a failure in any of the testing scenarios occurs. For more precise computation, the time delay increments between the tests are the multiplies of the simulation time step, i.e. 0.005 s. Therefore, the delay is always an integer number of simulation time steps.

The tests are carried out for the two controllers selected in Section 6.1. The controller #1 is (6.25) (pure Proportional Navigation guidance) and the controller #2 is (6.22). Fig. 6.10a shows the NPC evaluated for these controllers for varying time delay. The miss distance (av-

[1] First-order lag with time constant 0.04 s for all sensors and second-order lag with time constant 0.016 s and damping ratio 0.6 for aerodynamic surfaces actuators.

[2] Not including the internal processing delay within the controller, which is another 0.005 s.



a) *b)*

Fig. 6.10 NPC (a) and miss distance (b) for controllers with time delays

eraged over the four scenarios) is shown in Fig. 6.10b. It can be seen that in terms of guidance accuracy, delays up to 0.07 s for the second controller and up to 0.09 s for the first controller cause little effect. However, considering the NPC value, which also takes into account control activity, sufficient degradation is observable from 0.035 second delay. This is expectable, because delays usually cause oscillations in the closed-loop system.

To get a better understanding of these results, the trajectories and control deflections for the first controller are presented in Fig. 6.11 to Fig. 6.14. Rudder deflections are not shown because they are little affected by the delays. It can be seen that although the trajectories corresponding to 0.75 and 0.09 s delays (black and yellow paths) may be considered acceptable in terms of guidance accuracy, they produce excessive control usage. Delay of 0.06 s (green path) causes noticeable oscillations as well; however, they fade out within 6–7 seconds in a calm environment (Fig. 6.11), which may be considered as an appropriate level. In a turbulent environment, degradation is less pronounced, because even the reference controller (with smallest delay) produces sufficient control activity. Therefore, the highest acceptable NPC value should be between 1.618 (green line, delay 0.06 s) and 2.384 (black line, delay 0.075 s). In this work, NPC = 1.8 is chosen as the threshold. This is closer to the green path.

For the selected NPC threshold, the maximum time delay can be defined as 0.065 s for the controller #1 and 0.06 s for the controller #2.

It should be noted, however, that perturbation of any one of the parameters does not provide a comprehensive picture of the effect on performance. For this reason, the effect of perturbation of empty mass m_{empt} will be closely examined as well. Unlike time delay, this perturbation should primarily affect the trajectory rather than the control activity. Note that although the mass changes, the inertia tensor (3.13) does not and hence the dynamics of angular motion is unaffected.

m_{empt} is varied from 50% of the reference value (20.45 kg) to 200% linearly with 2% steps. This is done separately for increasing and decreasing the variable until the threshold NPC = 1.8 is crossed, or a failure occurs, or the limit (200% or 50% respectively) is reached. Several trajectories for increased mass are presented in Fig. 6.15 to Fig. 6.18.

Somewhat surprisingly, the trajectories with greater mass go higher than the nominal trajectory. However, examining the elevator control deflections, the reason becomes clear. Changing the empty mass displaces the centre of mass backwards (apparently because payload is concentrated in the nose), producing nose-up pitching moment. This requires additional positive elevator deflection to restore the balance. However, despite the correction introduced in Section 6.1.1, the positive elevator deflection limit is very close and this additional deflection quickly causes saturations on elevator during the flight. This, in turn, causes degradation of guidance accuracy, which is especially visible in the tailwind scenario. Nevertheless, for moderate NPC values (up to ≈1.6 at least, the green path), the accuracy is perfectly acceptable, even though the trajectories are very different. Only for NPC > 2.1 (black line) does accuracy deteriorate so that miss becomes likely. Therefore, the threshold NPC = 1.8 proves to be reasonable. It may be noted again that unlike the previous case (time delay perturbation), the same threshold is obtained primarily due to growing miss distance, while the penalty for control activity plays only a minor role.[1]

The parameters corresponding to aircraft geometry and configuration are tested in a similar manner. The range is increased to the scale factors between 0 and 10 (0 to 1000%) with step 0.05. NPC is linearly interpolated at the intermediate points. The allowable perturbations (as factors to the original values) are summarised in Table 6.1.

Parameter	Lower limit factor		Upper limit factor	
	Controller 1	Controller 2	Controller 1	Controller 2
Empty mass (m_{empt})	0.50*	0.50*	1.54	1.52
Rolling moment of inertia (I_x)	0.20	0.20	10*	2.43
Yawing moment of inertia (I_y)	0*	0*	10*	10*
Pitching moment of inertia (I_z)	0*	0*	2.17	1.91
xy cross product of inertia (I_{xy})	0*	0*	10*	10*
Wing area (S)	0.74	0.87	1.55	1.60

* *The extreme value tested*

Table 6.1 Allowable perturbations of UAV inertial properties and geometry

[1] This penalty may be even smaller for the worst cases, because when the control sets on the deflection limit (saturates), it has no activity.

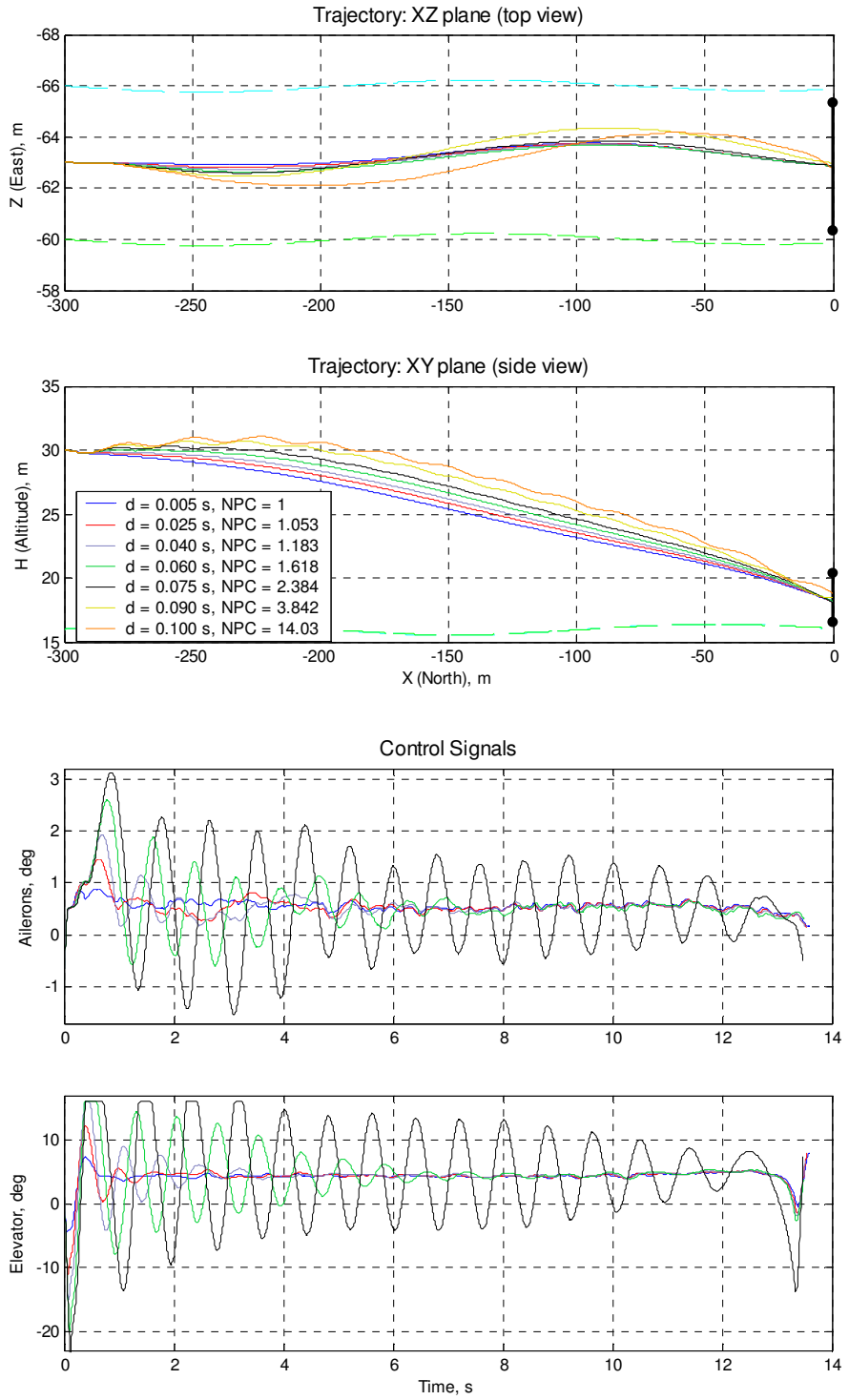


Fig. 6.11 Scenario 1: No Wind. Flight path and control signals for controller #1 with time delays (d). The legend on the trajectory graph applies to all figures. NPC is averaged over all four scenarios. Control signals for the worst two cases (with NPC > 3.8) are not shown, they exhibit extremely aggressive oscillations. Dashed cyan and green lines on the trajectory graphs represent the 'unrolled' along the flight path traces of the tips of the recovery boom (lateral position on the top view and vertical position on the side view). The bar on the right hand side illustrates the size of recovery window. The height and position of the window may be slightly different for different trajectories because it is determined by the shape of the arresting cable-hook. However, in most cases this is barely noticeable on the graphs since the difference in flight time and terminal part of trajectory is small. Note that the horizontal and vertical scale on the trajectory graphs is different.

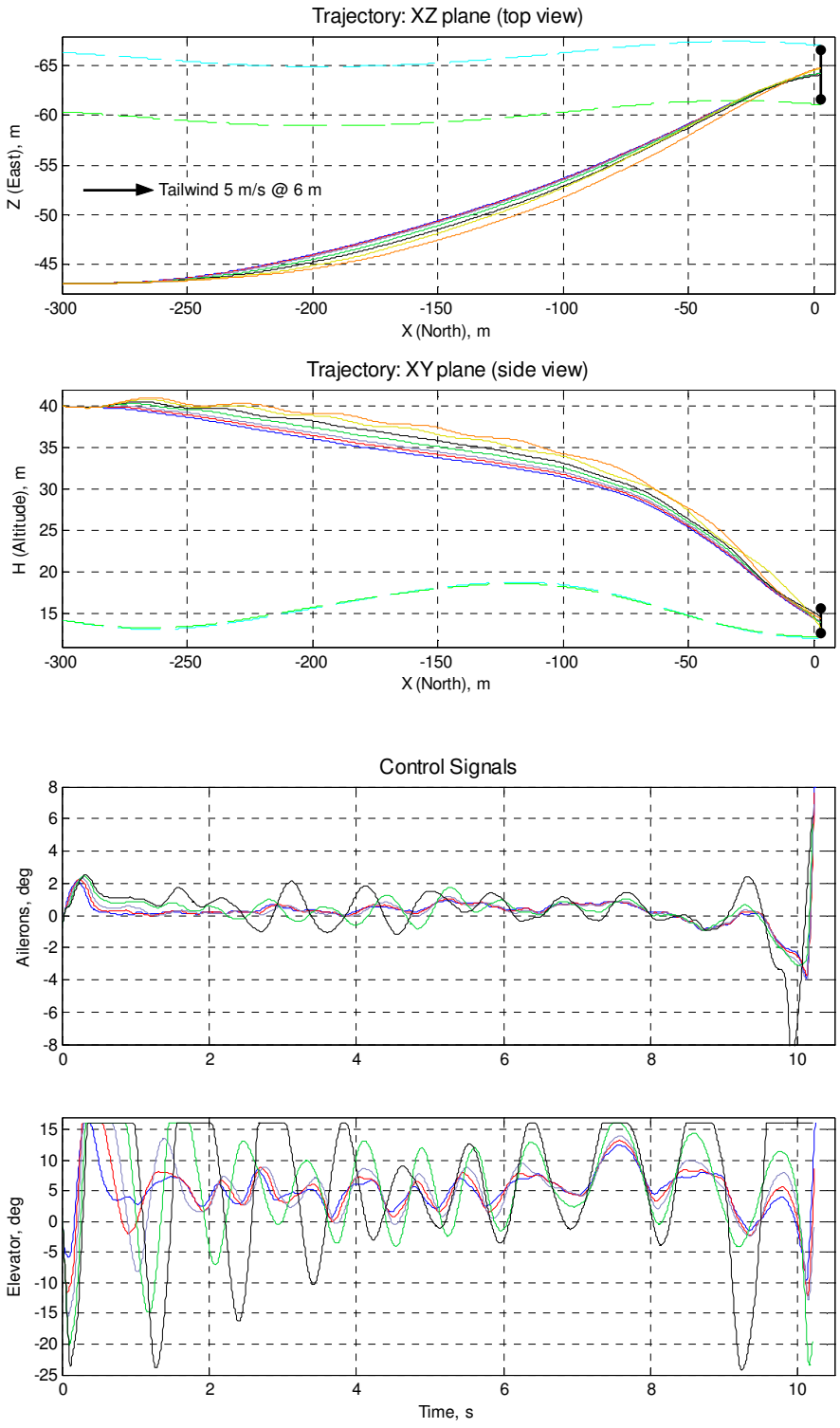


Fig. 6.12 Scenario 2: Tailwind. Flight path and control signals for controller #1 with time delays. See legend and annotation to Fig. 6.11.

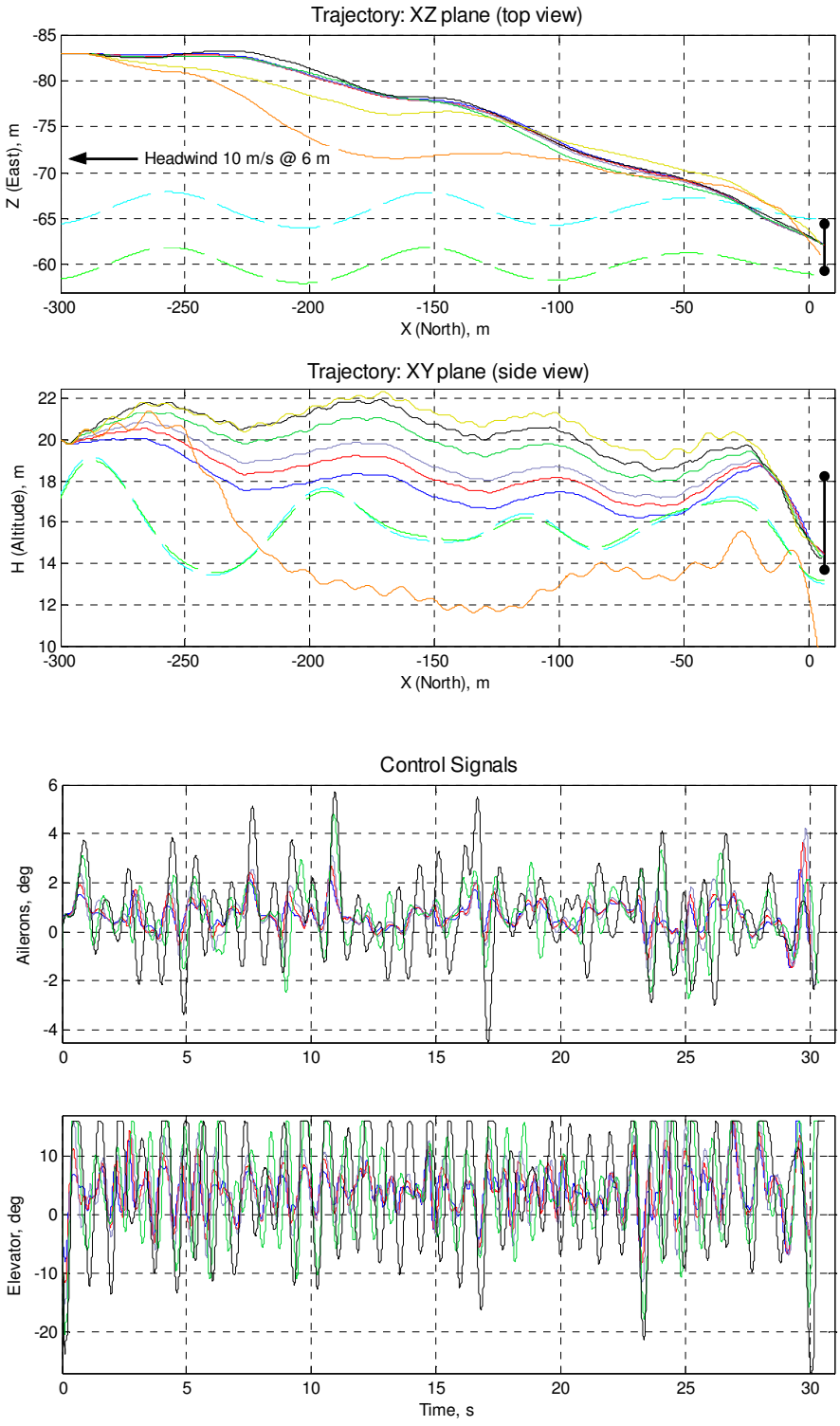


Fig. 6.13 Scenario 3: Headwind. Flight path and control signals for controller #1 with time delays. See legend and annotation to Fig. 6.11.

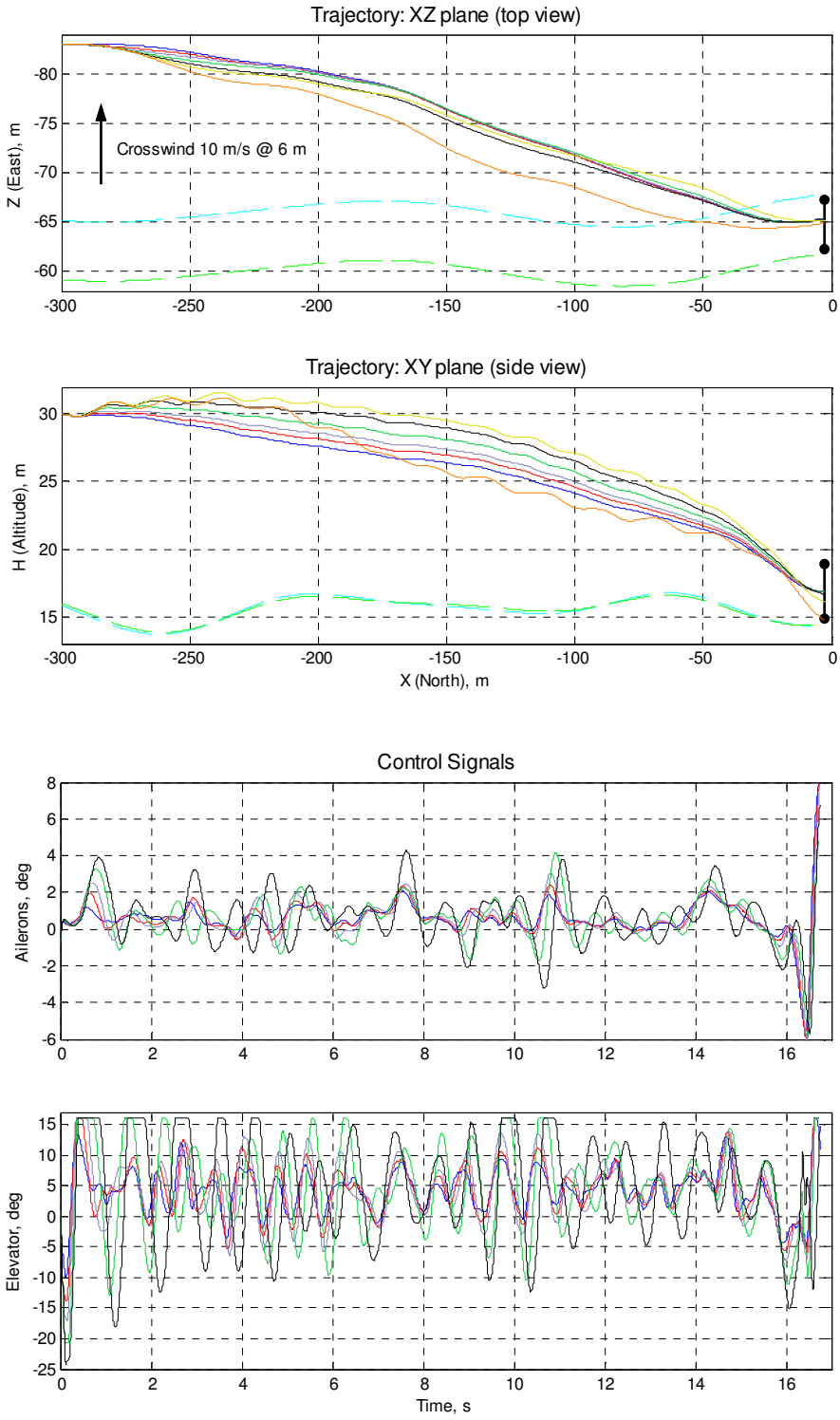


Fig. 6.14 Scenario 4: Crosswind. Flight path and control signals for controller #1 with time delays. See legend and annotation to Fig. 6.11.

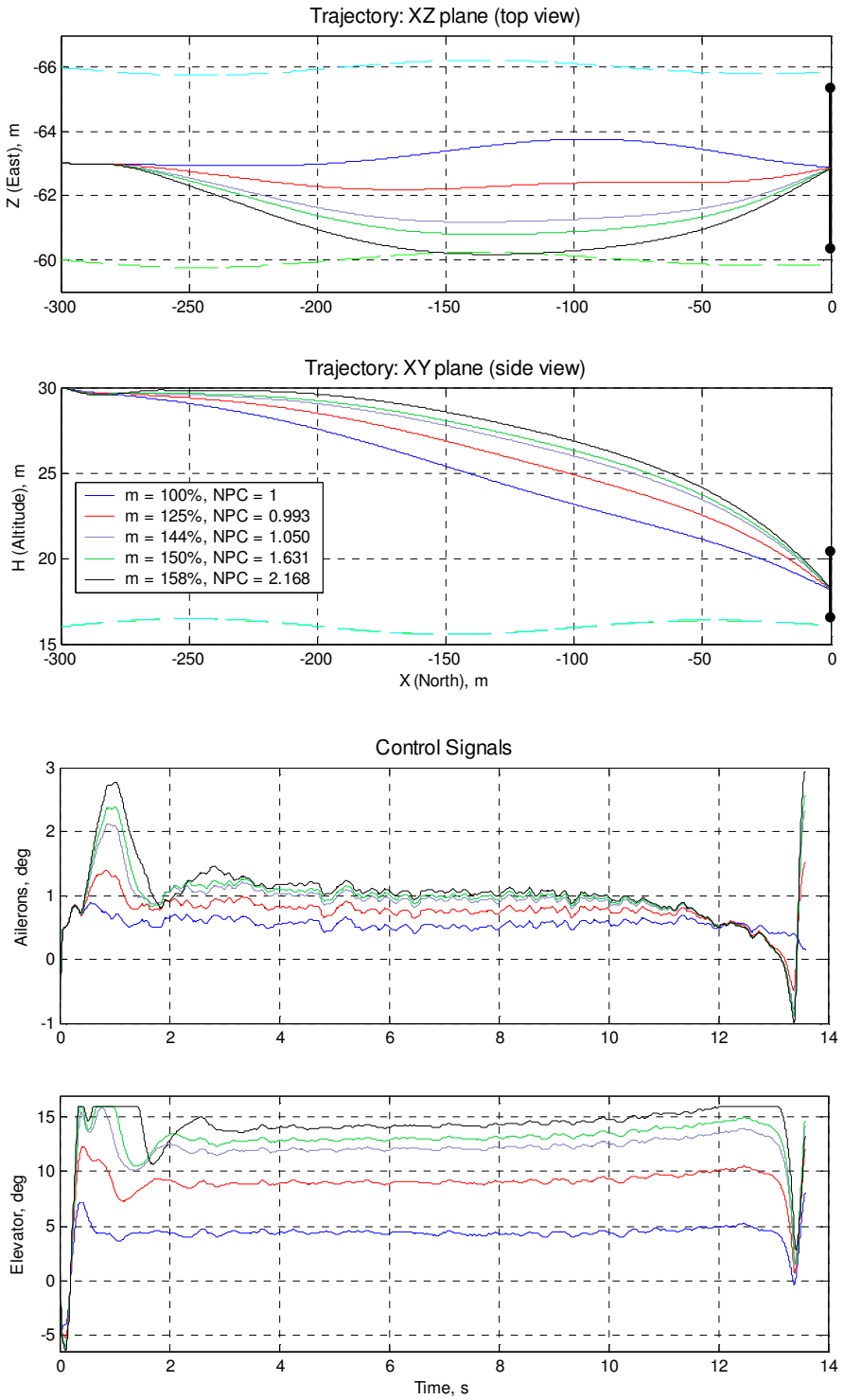


Fig. 6.15 Scenario 1: No Wind. Flight path and control signals for controller #1 with varying empty mass. The legend on the trajectory graph applies to all figures. NPC is averaged over all four scenarios. Dashed cyan and green lines on the trajectory graphs represent the 'unrolled' along the flight path traces of the tips of the recovery boom (lateral position on the top view and vertical position on the side view). The bar on the right hand side illustrates the size of recovery window. See also annotation to Fig. 6.11.

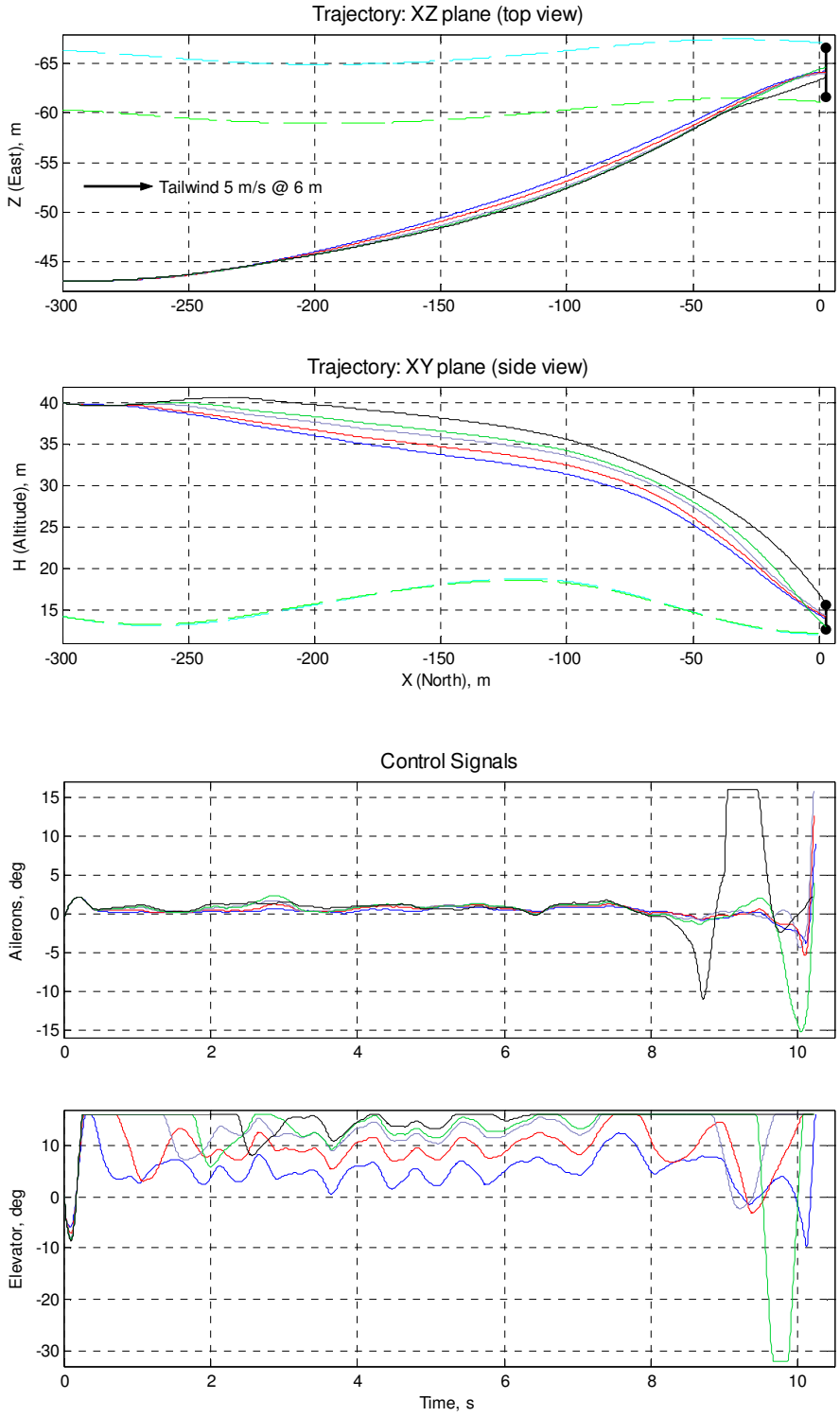


Fig. 6.16 Scenario 2: Tailwind. Flight path and control signals for controller #1 with varying empty mass. See legend and annotation to Fig. 6.15.

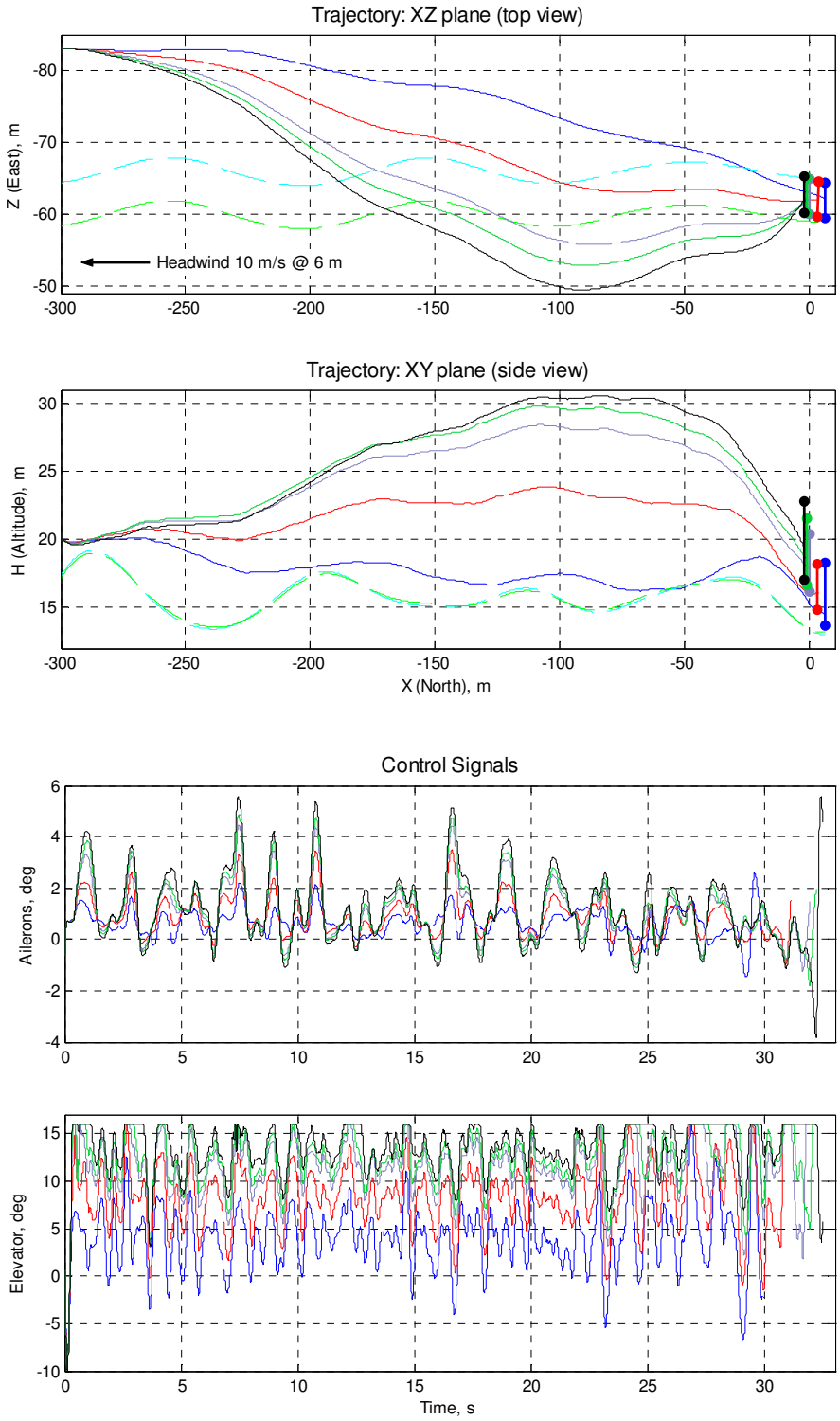


Fig. 6.17 Scenario 3: Headwind. Flight path and control signals for controller #1 with varying empty mass. See legend and annotation to Fig. 6.15. Note that in this case the final position is sufficiently different for different flights due to significant variation of the trajectory and flight time. The recovery windows are shown separately for each flight in respective colours.

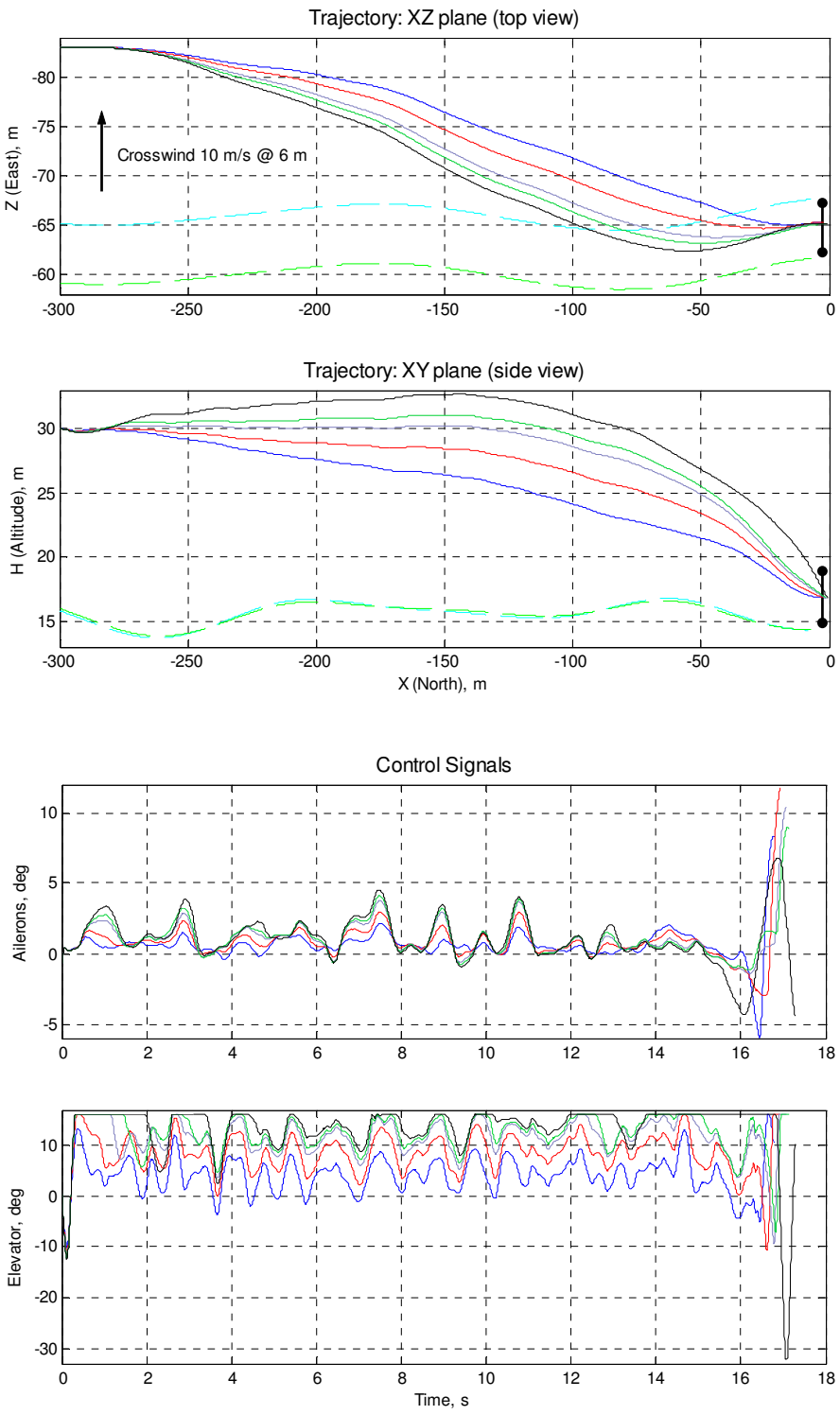


Fig. 6.18 Scenario 4: Crosswind. Flight path and control signals for controller #1 with varying empty mass. See legend and annotation to Fig. 6.15.

This table shows that both controllers maintain good performance over a wide range of perturbations. For some parameters, no noticeable drop in performance is experienced within the testing limits. These limits indicate quite large perturbations that can reasonably be expected. The main perturbations that cause a performance degradation are those that affect the physically attainable trajectory and not the operation of the controller, e.g. increasing weight and decreasing wing area. However, as follows from the above analysis, careful adjustment of elevator efficiency and/or horizontal stabiliser incidence may help to increase the tolerance to increased weight. There is still sufficient margin in angles of attack and engine power.

Robustness to perturbations of the power unit parameters will be investigated next. This is done similarly to the previous test. Refer to Section 3.1.2.3 for details about the propulsion model. The results are given in Table 6.2. As it can be seen, the power plant is certainly not the limiting factor for the recovery operation. The maximum power may be reduced at least threefold without noticeable negative effects. Other parameters allow a great variation as well.

Parameter	Lower limit factor		Upper limit factor	
	Controller 1	Controller 2	Controller 1	Controller 2
Engine power (P_{max0})	0.18	0.27	10*	10*
Propeller power (P)	0*	0*	4.07	9.23
Propeller thrust (T)	0.30	0.28	7.77	8.25
Propeller normal force (F_z)	0*	0*	10*	10*
Propeller gyroscopic moment (M_G)	0*	0*	10*	10*

* *The extreme value tested*

Table 6.2 Allowable perturbations of UAV power unit parameters

Variations of the aircraft aerodynamic parameters are slightly different. It is expected that many of the parameters may be scaled down to zero safely. At the same time, the range of allowable perturbations may be very large. Therefore, the following approach is taken. For scaling down, the scale factor changes linearly from $1–k$ to 0 with step $k = 0.1$. Some of the more sensitive parameters such as lift coefficient C_y are tested with a smaller step $k = 0.05$. For scaling up, the factor changes exponentially with exponent $1+k$. That is, the parameter grows by 10% (or 5%) with respect to the previous test point. The maximum factor is 100 (more precisely, $1.1^{49} \approx 106.7$ in most cases). Note that as most of the aerodynamic data is contained in lookup tables, the entire table is scaled by the perturbation. A summary of the robustness results is presented in Table 6.3.

Parameter	Lower limit factor		Upper limit factor	
	Controller 1	Controller 2	Controller 1	Controller 2
C_x	0.25	0.15	2.95	3.46
$C_x^{\delta_a}$	0*	0*	100*	70
C_x^{β}	0*	0*	100*	32.5
$C_x^{\delta_e}$	0*	0*	8.90	8.80
$C_x^{\delta_r}$	0*	0*	100*	100*
C_y	0.76	0.74	1.70	1.70
C_y^{β}	0*	0*	100*	100*
$C_y^{\delta_e}$	0*	0*	1.80	1.70
$C_z^{\delta_a}$	0*	0*	100*	100*
C_z^{β}	0*	0*	6.50	4.80
$C_z^{\delta_r}$	0*	0*	12.5	9.5
$m_x^{\delta_a}$	0.29	0.42	3.28	3.28
m_x^{β}	0*	0*	5.44	6.80
$m_x^{\delta_r}$	0*	0*	28.3	50.0
m_{z0}	0*	0*	1.62	1.56
m_z^{β}	0*	0*	100*	100*
$m_z^{\delta_e}$	0.70	0.74	1.79	1.83
$m_y^{\delta_a}$	0*	0*	42.0	10.7
m_y^{β}	0*	0*	55.8	55.2
$m_y^{\delta_r}$	0*	0*	16.8	16.5
$C_z^{\omega_x}$	0*	0*	14.8	5.31
$C_z^{\omega_y}$	0*	0*	14.9	8.80
$C_z^{\dot{\beta}}$	0*	0*	28.5	9.60
$C_y^{\omega_z}$	0*	0*	9.44	9.38
$C_y^{\dot{\alpha}}$	0*	0*	16.5	14.8
$m_x^{\omega_x}$	0*	0*	24.9	11.4
$m_x^{\omega_y}$	0*	0*	100*	58.8
$m_x^{\dot{\beta}}$	0*	0*	100*	100*
$m_z^{\omega_z}$	0.20	0.34	3.23	3.05
$m_z^{\dot{\alpha}}$	0*	0*	12.1	5.55
$m_y^{\omega_x}$	0*	0*	15.4	12.1
$m_y^{\omega_y}$	0*	0*	100*	100*
$m_y^{\dot{\beta}}$	0*	0*	100*	100*

* *The extreme value tested*

Table 6.3 Allowable perturbations of UAV aerodynamic coefficients

This table shows that both controllers have a surprising amount of robustness, even though only a few aerodynamic parameters have been taken into account during the controller synthesis in order to obtain a robust design. It is important to note that changes in aerody-

namic configuration have a direct effect only on the flight controller performance, which is the same for both guidance controllers being tested. Nevertheless, different guidance controllers may have different tolerance to degradation of the flight controller performance. This is clearly seen from the test results, with the controller #1 having better overall robustness to perturbations of majority of the parameters. An important exception is drag coefficient C_x, to which the controller #2 demonstrates sufficiently higher robustness.

The majority of the aerodynamic parameters may be safely scaled up and down more than an order of magnitude. The exceptions are those parameters which directly linked to aircraft capabilities (such as lift coefficient C_y and drag coefficient C_x) and primary aircraft response to control inputs (rolling moment due to ailerons deflection $m_x^{\delta_a}$ and pitching moment due to elevator deflection $m_z^{\delta_e}$), which is expectable. Other parameters that may cause robustness problems are pitch damping $m_z^{\omega_z}$ and pitching moment m_{z0}. However, they allow a generous range of perturbation as well. It is interesting to note that rudder efficiency $m_y^{\delta_r}$ may be reduced to zero, indicating that the aircraft possesses high natural yawing stability and active rudder control is not necessary. Comparing Table 6.3 with the respective analysis of the launch controllers in [47, Table 7.4], it may be concluded that the requirements to basic aircraft capabilities, particularly lift and drag, are tougher for launch. This could be expected because after launch the UAV must climb rapidly to gain a safe altitude and lift is crucially important. At the same time, the recovery controllers have a lower limit on drag while the launch controllers do not. This is because the UAV descends during the final approach and too low a drag causes the aircraft to gain unneeded speed.

Finally, robustness to sensor noise will be examined. Since the exact amount of noise for each sensor had not been specified, a reasonable noise power has been placed upon all measurement signals during controller synthesis, considering the nature of the respective sensors. Aerial sensors such as the angle of attack sensor and the airspeed sensor included a fairly large amount of noise, while gyroscopic sensors featured lower level of noise (see Section 3.1.2.4). However, such an approach is not guaranteed to be adequate. This is even more so considering that the simulation time step is too large for correct simulation of white noise (see discussion in Section 5.3.3). For this reason, testing the robustness to noise characteristics is especially important.

The approach in this test is largely the same as in the case of aerodynamic parameters. The noise power is iteratively scaled up by a certain factor and a standard four-scenario test is performed. However, since the maximum allowable amount of noise may be several orders of magnitude higher than the initial value, a larger step $k = \sqrt{2} \approx 1.41$ is used. The maximum scale factor is limited to 10^6. Testing for reduced amount of noise is not performed, because less noise will not apparently cause any drop in performance. The signals sourced from simi-

lar sensors (for example, three Euler angles or three UAV angular velocities) are processed simultaneously. The results are presented in Table 6.4. These results show absolute noise power (intensity) attained, in respective units. It should be kept in mind that due to possible inadequacy of noise simulation, the results are accurate only to the order of magnitude. It can be concluded that both controllers have very close robustness to noise in the UAV state measurement signals, which is expectable as the flight controllers are identical in both cases. The discordance is attributed mainly to the difference in nominal performance of the two controllers. At the same time, sensitivity to noise in positioning signals is noticeably higher for the second controller.

Signal (units)	Maximum noise power	
	Controller 1	Controller 2
Airspeed (V_a, m/s)	0.125	0.17
Aerial angles (α, β, rad)	$1.5 \cdot 10^{-5}$	$2 \cdot 10^{-5}$
Load factors (n_x, n_y, n_z, units)	$4.6 \cdot 10^{-4}$	$5 \cdot 10^{-4}$
Angular velocities (ω_x, ω_y, ω_z, rad/s)	$2 \cdot 10^{-5}$	$2.2 \cdot 10^{-5}$
Euler angles (γ, ψ, θ, rad)	$2.5 \cdot 10^{-5}$	$2.8 \cdot 10^{-5}$
Radio distances (d_1, d_2, d_3, m)	0.24	0.25
Radio distances rates (m/s)	0.04*	0.04*
Distance differences (d_1–d_2, d_3–d_2, m)	0.018	0.01
Distance differences rates (m/s)	$7.4 \cdot 10^{-4}$	$5.2 \cdot 10^{-4}$
Ship angular velocities (ω_s, rad/s)	$2.7 \cdot 10^{-5}$	$1.1 \cdot 10^{-5}$
Ship Euler angles (θ_s, rad)	Not used	0.01*

* *The extreme value tested*

Table 6.4 Maximum allowable noise power

Overall, these tests have not exposed any significant robustness problems within the controllers. Large variations in single aircraft parameters caused very few control problems. However, such variations cannot realistically judge the performance of the controllers under simultaneous perturbations of multiple parameters. In order to compare the practical robustness of the aircraft, the following tests are performed.

6.2.2.2 Simultaneous multiple perturbation tests

These tests involve simultaneous perturbation of all the aircraft variables by a random amount. This style of testing requires many simulations to ensure adequate coverage of the testing envelope.

Normally distributed random scale factors are used for perturbations. The chosen standard deviations are approximately one tenth of the maximum allowed perturbations identified in the previous robustness tests. This means that the majority of perturbed models have only small differences from the nominal model, although occasional models may have a greater perturbation. The same settings have been used in [47] for testing of the launch controller. By running a large number of tests, the effects of simultaneous perturbations may be evaluated.

None of the perturbations will be large enough to cause a significant drop in performance individually, as there is a minuscule chances that the maximum perturbation values will be exceeded (a random number 10 standard deviations away from the mean).

For each of the two controllers, 1000 tests (4000 simulations considering four scenarios) have been carried out. Note that the perturbations were the same for both controllers in each case. This allows to compare the performance between the controllers in similar conditions. The obtained results are somewhat contrasting. For the first controller, 97.2% of the cases tested have NPC < 1.8, i.e. experienced only a small degradation in performance. It is interesting to note that the best NPC is approximately 0.68, indicating that some perturbations deliver superior performance. Four cases (0.4%) have only marginally higher NPC (less than 1.87). Another six cases (0.6%) attained NPC between 2.0 and 5.2 with approximately uniform distribution. The rest of the tests (1.8%) resulted in a crash of the UAV with correspondingly very high NPC. The majority of these cases feature simultaneous one-sided variation of closely related parameters, for example, reduction of both wing area and lift coefficient. This indicates the aircraft capability limit rather than a control problem. This is confirmed by the fact that similar effect is observed for both controllers in these conditions. Only in two of these cases did the second controller manage to recover the UAV (not in all scenarios though, but at least avoided a crash), with nevertheless poor NPC. The worst case saw reduction of wing area by as much as 80% of the allowed perturbation (8 standard deviations!)

Some insight into unfavourable mutual dependency of the model variables can be obtained by calculating the correlation coefficients for the values of model variables that caused unsatisfactory performance. To do this, the model parameters corresponding to 24 cases with poor performance (NPC > 2) have been extracted from the table of results and correlation coefficients have been calculated using the MATLAB's function `corrcoef`. Considering that the original random perturbations are uncorrelated, high correlation coefficients indicate that simultaneous coordinated variation of the respective pair of parameters may produce poor results. The strongest correlation (coefficients between 0.75 and 0.98)[1] is observed between the following pairs of parameters: yawing moment of inertia I_y and propeller gyroscopic moment M_G; yawing moment coefficient due to sideslip m_y^{β} and rolling moment coefficient due to sideslip rate $m_x^{\dot{\beta}}$; roll damping coefficient $m_x^{\omega_x}$ and yawing moment coefficient due to roll rate $m_y^{\omega_x}$; and finally rolling moment coefficient due to yaw rate $m_x^{\omega_y}$ and yawing moment due to sideslip rate $m_y^{\dot{\beta}}$. It should be noted that this analysis is somewhat superficial and should be used only as a guide for further analysis. In particular, degradation of performance

[1] All cases involve positive correlation coefficients, implying that simultaneous increase (or decrease) of the respective parameters takes place. No strong negative correlations have been found.

may be caused by the effect of simultaneous correlated variation of more than two parameters. Nevertheless, the above pairs are indeed physically related and may cause coupling problems between the longitudinal and lateral channels.

One of the cases resulted in a miss in one of the scenarios whilst having NPC < 1.8. This is not impossible considering that NPC is averaged over four scenarios. Although the cost function (6.26) is designed so that a miss should generally produce average NPC > 1.8, this is not always ensured, especially when the UAV missed by a few centimetres.

The second controller showed much poorer robustness. 90.4% of test cases showed NPC < 1.8. At the same time, only 69.8% of the tests reported successful recovery. Such a large proportion of cases which acceptable NPC value but without capture (about 20%) indicates that the controller delivers sufficiently inferior guidance strategy when subject to model uncertainty. This could be expected because this controller uses more positioning measurements. The majority of these cases involve a miss in the tailwind scenario, highlighting a potential guidance problem. The number of cases with a crash in one or more scenarios is 2.8%, which is a significant increase over the first controller as well.

From these tests, the first controller is clearly preferable. However, it must also be tested over a wide range of scenarios. This will test its performance robustness.

6.2.3 Performance tests

The procedure of performance testing is very similar to that of the robustness tests. In this case, however, it is the operating conditions that are perturbed rather than the system model. Single perturbation tests are not performed because they have little relevance to the real system. For example, high sea wave is extremely unlikely to occur without high winds, and thus the performance under ship motion alone is not indicative. Nevertheless, some tests concerned about several most important variables will be conducted in Section 6.2.4 in order to determine the operational envelope.

Since the nominal performance estimated by (6.26) may vary significantly in different conditions, the success in each case is judged by several factors separately. The performance is considered satisfactory if:

$$\begin{aligned} &\text{horizontal miss distance is less than 1.5 m;} \\ \text{AND}\ &\text{vertical miss distance is less than 1 m;} \\ \text{AND}\ &\text{capture is detected;}[1] \\ \text{AND}\ &\text{impact speed is less than 30 m/s.} \end{aligned} \tag{6.29}$$

[1] In limited circumstances, the arresting wire may be missed even if the miss distance is well within the normal allowances. Such cases include a strong upwind or crosswind gust just before capture, a terminal approach with ascending trajectory and high airspeed, approach with strong crosswind and probably several others.

A recovery which satisfies these conditions will be referred to as *accurate recovery*. Note that these conditions leave a considerable margin taking into account currently used dimensions of the recovery gear (lengths of the arresting wire and cable), therefore the 'pure' success rate based only on capture detection will also be considered (*successful recovery*).

The Performance Cost is calculated similarly to (6.26) excluding the cost for control activity and control effort (see also (6.27)):

$$\mathrm{PC} = 40\Delta h_1^2 + 20\Delta z_1^2 + 50|\psi_1| + 25|\gamma_1| + 50 f_{\mathrm{V}} + 20 f_{\mathrm{H}} \tag{6.30}$$

This provides an estimation of the overall guidance quality. PC = 0 indicates a perfect recovery.

The performance test is very similar to the tests performed during the guidance controller evolution. Most of the environmental conditions are varied the same way (see page 250). These include initial states of the UAV and the ship and most of the atmospheric properties. Note that semi-deterministic variation of the UAV position that has been used in fitness evaluation procedure is not employed. Several additional factors are also included in order to allow a more realistic simulation of the real world environment. First of all, tests are performed for Sea States 2 to 6 (the controller evolution used only Sea State 6). A series of tests is carried out for each Sea State in order to determine the performance in different sets of conditions. Secondly, ship speed is allowed to vary in addition to random periodic ship motion. This is especially important for correct simulation in a calm environment, as a high ship drive may generate a significant amount of turbulence in the airwake even though the ambient atmospheric turbulence may be nearly zero. In the tests, ship speed is set randomly between 0 and 12 m/s with uniform distribution. The maximum value is approximately 85% of the maximum speed of ANZAC frigate (27 knots).

Wind settings are generated according to the algorithm used for controller evolution (page 250), with additional checking that tailwind does not exceed a moderate value of 7 m/s (at the initial altitude). It is implied that an opposite approach direction will be chosen if greater tailwind component exists. In order to avoid an excessive number of cases with the extreme wind speed, the random values that are generated for wind speed are not saturated at the defined bounds. Instead, the wind parameters are sampled again until a satisfactory combination is found. Note that wind speed is measured with respect to the ship (the *wind over deck*), therefore the UAV may need to compensate a fairly high 'virtual' wind even in a dead calm sea if the ship speed is significant. For the default boom location (see Fig. 3.11), this will be primarily a crosswind. However, other boom locations will also be investigated.

Finally, the gust model (Section 3.2.3) is employed. Up to 5 gusts are included over the simulation time. The time of occurrence is uniformly randomly distributed over a 40 second

interval (the exact flight time is unknown in advance; however, a great majority of flights is less than 40 s). It is not impossible that two or more gusts superimpose. The ramp time (the time in which the gust magnitude reaches the maximum and in which it falls to zero) is a normally distributed random value with a mean of 0.3 s and standard deviation of 0.2 s, with a lower bound of 0.05 s. The time length of each gust is given by an absolute normal distribution with a standard deviation of 3 seconds, ensuring additionally that the length is greater than the double ramp time chosen previously. This implies that any gust will reach its maximum magnitude. The magnitude of the gust depends on the Sea State, although not as strong as wind speed does. The magnitude is given by an absolute normally distributed random number with a standard deviation of 0.3 m/s for Sea State 0 to 1 m/s for Sea State 6, with linearly distributed values for intermediate Sea States. The direction of the gust is selected from $[0; 2\pi]$ with equal probability, whilst the inclination is normally distributed with a standard deviation of 11.25°. Largely similar gust settings have been used for testing of the launch controller [47], although with no regard of Sea State.

It is believed that these variations provide realistic enough simulation of real world environmental conditions. Where possible, standard design values are used [3, 4], particularly wind speed, turbulence parameters and ship random motion, with moderate randomisation.

Given these variations, multiple simulations may be conducted and the performance of the UAV may be analysed using the recovery success rate and the performance cost (6.30). Although the controller #2 experienced some robustness problems during previous tests, its performance is tested as well to obtain a more comprehensive picture. Similar to robustness tests, the same random variations are used for both controllers. Simulation time is limited to 120 seconds to protect the model from possible conditions where the ship is out of reach. Even though the headwind component is limited, exceptional crosswind in a combination with high ship speed may result in such conditions.

6.2.3.1 Default recovery boom position

The configuration used for controller synthesis will be investigated first. It uses a fixed recovery boom located at the bow of the ship (Fig. 3.11). The tests start from *Sea State 2* (SWH)[1] ('smooth sea'). 200 simulation runs have been performed in these conditions. As expected, 100% of the tests were successful for both controllers. The first controller showed marginally better PC (average 24.7 vs. 26.7 for the second controller). Average miss distance did not exceed 20 centimetres horizontally and 9 centimetres vertically. The flight time varies from 9.3 s to 22.8 s, mostly due to ship drive. Analysis of the flight time histories shows that in the cases of a headwind (which corresponds to crosswind for the ship), ship airwake plays a

[1] All tests are performed with respect to the Significant Wave Height. See footnote 2 on page 263.

dominant role in atmospheric disturbances during the last 2–3 seconds of flight. The tests for Sea States 0 and 1 are not performed because the results are expected to be perfect as well.

The tests at *Sea State 3* demonstrated very similar results. All of the recovery attempts were successful. A very minor increase in Performance Cost is observed (average 26.6 and 28.4 for the controllers #1 and #2 respectively). Maximum flight time, however, increased to 33.5 s due to greater wind.

The tests at *Sea State 4* ('moderate sea') saw first failures. In order to obtain a more reliable statistics, 500 simulations have been carried out for each controller. For the first controller, 4 complete failures[1] have been detected. Physically, these are not crashes. All these cases involved nearly maximum ship speed (more than 11.5 m/s) and a strong (for Sea State 4) crosswind of about 8 m/s in the approximately opposite direction to the ship (i.e. headwind for the ship). This forces the UAV to turn into the wind and the simulation eventually stops on the condition that yaw angle reaches 90°. This means that the aircraft moves parallel to the ship, having no means for further approach. Even if some tailwind component is present (with respect to the approach direction), such attitude of the UAV makes recovery almost impossible. In two other cases with similar conditions, the UAV manages to approach the recovery boom, which takes more than 90 and 100 seconds respectively, but misses it due to prolonged exposure to strong downwash in the ship's airwake and unfavourable cable position. No other specific problems have been identified. In other test cases (98.8%) the UAV has been successfully recovered. Applying tougher performance requirements (6.29), 97.6% cases can be considered successful. Four of six cases that do not fall in this category had excessive impact speed, while two other cases exceeded the required miss distance. Performance Cost (considering the successful cases) shows only a small degradation over the previous test conditions, averaging to 30.9. Average miss distance is approximately 0.25 m.

The second controller is slightly less successful. It demonstrates similar problems to the first controller in all the cases analysed above, plus entails three other cases with large miss distances. They are attributed to headwind combined with the large scale atmospheric disturbances, again mostly due to ship airwake. Among the successful 98.2% cases, average PC is 31.8. Miss distances are very close to those attained by the first controller as well. In 96.8% of the cases, the requirements (6.29) are satisfied. Two of the cases rejected as inaccurate have excessive speed and the rest have the miss distance larger than required.

The main problem identified in this analysis can be solved straightforwardly. The difficulty in reaching the ship arises when the longitudinal component of the wind over deck ex-

[1] The term 'complete failure' is used to indicate either a crash or a condition where the UAV did not reach the recovery boom within 120 seconds. This is in contrast with failure where the UAV reaches the boom but fails to capture the arresting wire.

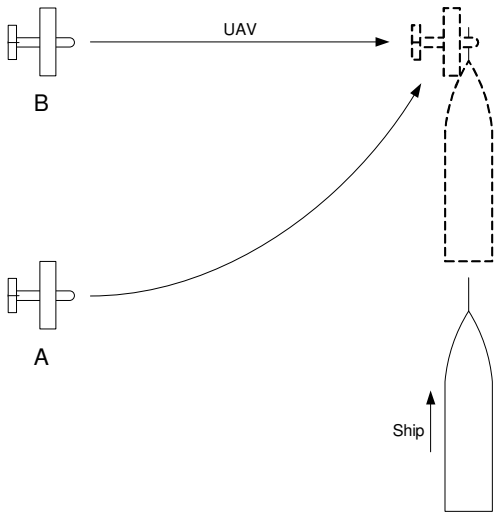


Fig. 6.19 Compensation of ship drive

ceeds approximately 19 m/s. Wind over deck can be measured onboard the ship. If this figure is reached, the ship can reduce its speed for a few minutes. Alternatively, if in addition a moderate crosswind component is present (with respect to the ship), the approach can be done from the upwind side.

However, a more elegant solution can be found. The final approach may be included into the navigation task solved by the common navigation (mission) controller. That is, instead of guiding the vehicle to the approach corridor entrance on the perpendicular to the current location of the recovery boom (Fig. 6.19, A; see Section 2.4.2), the rendezvous point is estimated and the corridor entrance corrected in view of the future ship location (Fig. 6.19, B). Considering the Proportional Navigation nature of both controllers, this correction should not distract the guidance, assuming that the ship moves uniformly. If the correction is ideal, the UAV will fly along a straight line under the PN law, as shown in Fig. 6.19, B. This can be further elaborated to include the crosswind component (for the UAV), so that the UAV will not need to compensate it during the final approach. However, for the guidance laws employed this may have a negative effect, because these laws do not use the wind data nor any other information to pre-plan the flight path. The guidance controllers will make the UAV to fly with no regard of such a correction.

Correction of the approach corridor entrance location can be calculated as follows. The ship crosswind component W_{zs} is measured onboard the ship. It will constitute the tail or headwind for the UAV, having a direct effect on the approach time. Wind measurements should be taken approximately at the altitude of the flight path (this can be done by installing the wind vane at the highest antenna on the ship), or the increase of wind speed at the actual altitude should be estimated and taken into account. The flight time is estimated taking into account the approach distance X_e (normally 300 m), the measured wind speed W_{zs} and the normal approach airspeed of the UAV V_{a0}, which is 22 m/s. Finally, the distance that ship will travel in this time is calculated taking into account the ship forward speed V_s, which is known from the ship navigation data. The respective coordinate of the corridor entrance is then corrected by this distance:

$$\Delta x_s = \frac{X_e V_s}{V_{a0} + W_{zs}} \tag{6.31}$$

where the correction Δx_s is given in the ship axes frame. It is added to the lateral coordinate Z_e of the entry point (see Fig. 2.14 in Section 2.4.2).

It should be noted that this solution will not solve the problem completely. Crosswind in excess of about 19 m/s will still render the approach unfeasible in many cases. This is the aircraft limit considering that the airspeed is about 22 m/s and not the limitation of the controller. Even if the navigation controller would take the crosswind into account and planned the approach accordingly, any error involving wind compensation (i.e. when the UAV moved too far downwind) would be difficult to correct. Nevertheless, the crosswind is now expressed in world coordinates with respect to the UAV and not to the ship. This enables the ship to maintain its speed in strong headwind conditions and thus extends the operating envelope.

The above ship drive correction has a negative effect on the positioning system employed in this work. In this configuration, the aircraft flies at a substantial angle (up to 40–50°) to the recovery boom. However, ship motion is compensated in the positioning signals under the assumption that this angle is small (see Section 5.2.1.2.2). In the default configuration (forward boom located at the bow, Fig. 3.11), ship roll is subtracted from the measurements of the vertical UAV position and velocity. When the UAV is at an angle to the boom, the influence of ship roll is diminished but ship pitch emerges. Unfortunately, this cannot be compensated without knowing the actual UAV position with respect to the ship at every moment of the approach. As it turned out in the guidance controller synthesis, the guidance controller does not need the position data, it only uses the relative velocities. However, if this data is available to the ship (only the angle (azimuth) is needed), it can be used to correct compensation of ship motion. Some of the positioning systems (e.g. radar based) can naturally provide the angular location of the aircraft. This can be as well implemented in the current system using the measured distance differences (Section 5.2.1.2.1). Alternatively, a constant correction corresponding to the pre-planned azimuth can be used. For pure Proportional Navigation law, the azimuth will be nearly constant if the initial UAV position is exactly as planned by (6.31). However, this may produce unsatisfactory results if the actual angle is different due to inaccurate wind measurements and various disturbances, which is especially likely to happen close to the boom, exactly where precise positioning is needed. To provide correct measurements for the guidance controllers, the correction using the measured UAV azimuth has been implemented within the positioning system.

To verify the effect of the above upgrades, another set of 500 simulations have been performed for each controller. Apart from the correction of the initial UAV position, the conditions were similar to the previous test. Note that a random component of the lateral position have been included as usual. This makes an allowance for inaccuracies of the navigation planning. The results are somewhat mixed. In terms of reliability, a significant improvement

is achieved. No complete failures happened with the first controller and only one complete failure occurred with the second controller. This failure happened for the same reasons as previously. The UAV had a sufficient miss to the right a few metres before the boom but could not compensate it due to unfavourable combination of wind and ship speed. This is a good illustration to the above discussion about the aircraft limitation. In a less hostile environment, such a correction does not represent any problem. Apart from this case, two recovery failures (misses) have been detected for either controller. All of them are attributed to atmospheric disturbances combined with unfortunate boom velocity and position. In two of these cases, a strong gust occurred 2–3 seconds before crossing the boom. In other cases, ship airwake played the major role. In all these cases, a go-around was possible.

Considering the successful captures, some interesting results are observed. The controller #1 experienced a noticeable drop in guidance accuracy. A significant portion of the cases does not satisfy the requirements (6.29), with mere 93.2% (466 flights) considered fully successful (compared to 97.6% in the previous test without the corridor entrance correction). Many cases (2.6%, 13 flights) indicate excessive impact speed. In 19 cases the miss distance is larger than required. The largest average miss distance is 0.45 m, almost twice as large as without the ship drive compensation. As can be expected, mostly the horizontal accuracy has suffered. Average PC is 35.1 (compared to 30.9 in the previous test). It is interesting to note that the second controller did not exhibit such accuracy degradation. On the contrary, it demonstrates improvement which could be expected from the first controller as well. 97.8% of cases can be considered fully successful, complying with (6.29). Only two cases exceeded the desired accuracy and six cases indicated excessive impact speed. The average miss distance is 0.26 m in both planes and the average Performance Cost is 29.0. Analysing the guidance law (6.22), it may be reckoned that such robustness to different configurations in the horizontal plane is attributed to the horizontal relative velocity V_{zT} term.

In view of these results, the tests in the other two Sea States are conducted in both configurations, with and without the ship drive compensation. Considering that a greater range of conditions should be covered and that the negative effects on performance may be complicated in a hostile environment, 2000 simulations have been performed for Sea States 5 and 6.

In *Sea State 5* ('rough sea'), ship roll amplitude reaches 15° and decrease in performance is considerable. Without the ship drive compensation, the picture is similar to the previous case. Controller #1 achieves slightly better results in terms of both success rate and accuracy. 91.1% of recovery attempts were successful and in 85.6% of cases attained the required accuracy. Among the successful captures that do not satisfy the requirements are 39 cases (1.95%) with excessive impact speed. At the same time, a significant percentage of attempts (5.55%) resulted in a complete failure. Average PC reached 40.4, with average miss distance

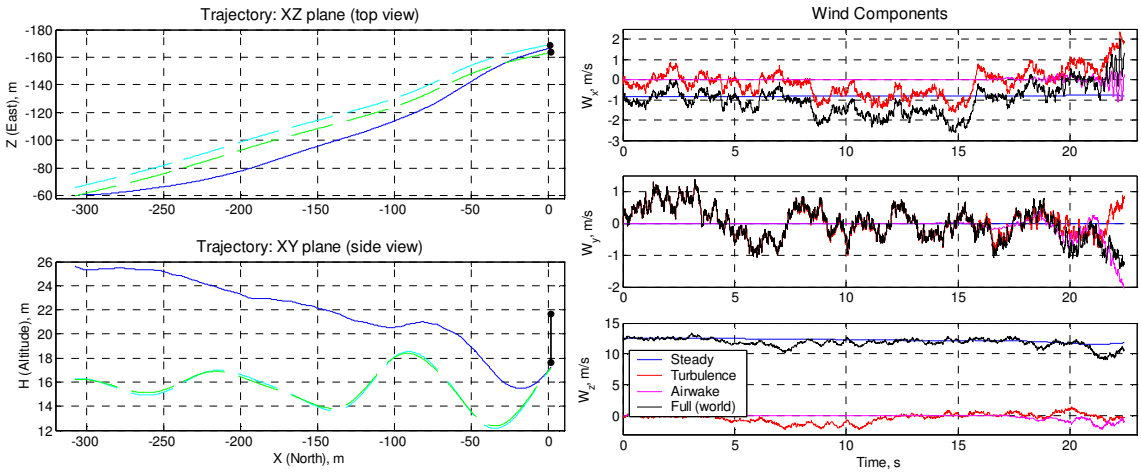


Fig. 6.20 An example of failure due to ship motion and airwake. Dashed lines on the trajectory graphs represent the 'unrolled' along the flight path traces of the tips of the recovery boom (lateral position on the top view and vertical position on the side view). The bar on the right hand side illustrates the size of the recovery window.

(excluding complete failures, for which miss distance is not applicable as such) of about 0.4 m. The respective figures for the second controller are: 87.1% captures, 81.8% of cases satisfied accuracy requirements, 1.25% had excessive speed, 6.7% complete failures. Average PC is 39.6, average miss distance is 0.44 m horizontally and 0.59 m vertically.

About two thirds of complete failures are attributed to inability for the UAV to reach the ship. However, unlike the situation with Sea State 4, many failures (about one third) can be credited to unfavourable recovery boom motion which could not be compensated and to unsuitable UAV attitude at the moment of capture. This is often combined with unfortunate gusts and other atmospheric disturbances in the last seconds of flight. An example of failure of such nature is shown in Fig. 6.20.

With the ship drive compensation enabled, both controllers behave similarly to Sea State 4. The first controller experiences a drop in performance, while the second controller improves its results. The performance results are the following (the data in brackets are for the controller #2): 90.6% (91.6%) captures, 71.7% (86.0%) of recoveries are accurate enough to satisfy (6.29), average PC = 60.8 (43.0), average miss 0.78 m (0.39 m) horizontally and 0.69 m (0.56 m) vertically, 2.4% (0.6%) captures with an excess of speed. Nevertheless, the number of complete failures decreased dramatically for both controllers, totalling 1% (2.9%).

The results for *Sea State 6* ('very rough sea'), the worst case considered with ship roll up to 22.5°, follow the established pattern. The following results are obtained without the ship drive compensation: 69.5% (67.0%) captures, 56.3% (56.2%) accurate recoveries, average PC = 58.0 (50.5), average miss 0.87 m (0.58 m) horizontally and 0.85 m (1.0 m) vertically, 2.4% (1.9%) captures with excessive speed. Considering the percentage of complete failures, which is 14.5% (16.1%), it may be concluded that Sea State 6 is indeed extreme conditions.

Note that the results are very close for both controllers, which may indicate approaching physical limitations of the vehicle.

The ship drive compensation still sufficiently improves the number of complete failures, which is in this case 5.2% (8.2%). The great majority of these failures are of the same type that shown in Fig. 6.20. However, other results are on the same level for the second controller and deteriorate for the first controller. This implies that when previously the UAV could not reach the boom properly, the ship drive compensation enables it to be done but does not preclude from a large miss. The accuracy results are the following: 63.2% (67.6%) captures, 39.0% (54.6%) accurate recoveries, average PC = 80.4 (57.5), average miss 1.7 m (0.63 m) horizontally and 1.2 m (1.0 m) vertically, 4.6% (2.0%) captures with excessive speed.

The obtained results are summarised in Table 6.5.

Sea State	2		3		4				5				6			
Ship drive compensation	No		No		No		Yes		No		Yes		No		Yes	
Controller	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
Successful recoveries, %	100	100	100	100	98.8	98.2	99.6	99.4	91.1	87.1	90.6	91.6	69.5	67.0	63.2	67.6
Accurate recoveries,[1] %	100	100	100	100	97.6	96.8	93.2	97.8	85.6	81.8	71.7	86.0	56.3	56.2	39.0	54.6
Recoveries with large miss,[1] %	0	0	0	0	0.4	0.8	3.8	0.4	3.4	4.0	16.2	4.2	9.3	7.7	18.2	9.2
Recoveries with overspeed,[1] %	0	0	0	0	0.8	0.6	2.6	1.2	1.95	1.25	2.4	0.6	2.4	1.9	4.6	2.0
Complete failures,[2] %	0	0	0	0	0.8	1.0	0	0.2	5.55	6.7	1.0	3.0	14.5	16.1	5.2	8.2
Average PC [3]	24.7	26.7	26.6	28.4	30.9	31.8	35.1	29.0	40.4	39.6	60.8	43.0	58.0	50.5	80.4	57.6
Average horizontal miss,[4] m	0.14	0.20	0.18	0.21	0.26	0.27	0.45	0.27	0.42	0.44	0.79	0.39	0.87	0.58	1.7	0.63
Average altitude miss,[4] m	0.09	0.06	0.13	0.11	0.22	0.24	0.32	0.26	0.40	0.59	0.69	0.56	0.85	1.00	1.2	1.0

Percentages are given with respect to all attempts. Notes: [1] Considering (6.29). [2] See footnote 1 on p. 283. [3] For successful recoveries. [4] Excluding complete failures.

Table 6.5 Recovery performance statistics in default configuration

Several conclusions can be made analysing this data. Overall, the performance is quite reasonable. A degradation at high Sea States is expectable and is attributed to an extreme level of uncertainty present in the environment. The ship drive compensation proves to be effective, considerably reducing the number of complete failures where the UAV is unable to reach the ship and extending the operating envelope. This improvement overcomes the negative effect of some deterioration of guidance accuracy for the first controller, which may be compensated by using a longer recovery boom. The second controller, generally showing inferior results, performs quite well with the ship drive compensation. However, it should be noted that this controller always shows considerably higher rate of complete failures. Analysis of trajectories reveals that this is due to lower profile of approach and a characteristic feature of the positioning system. The UAV initially descends deeper than with the first controller, continuing the approach almost level to the boom. In some cases which involve a substantial horizontal deviation and a rapid rise of the boom due to ship motion a few seconds before in-

tended capture, the UAV position violates the operating conditions of the positioning system (see Section 5.2.1.2.1). The system then supplies incorrect information and the UAV diverts from the boom. Another consequence of the low profile is higher chances of a collision with a wave and in some configurations with ship superstructures. Although waves are not modelled, this should be taken into account for actual design.

The same tendency to a low approach has a small positive effect. Steep dive is rarely needed at the terminal part of approach. Fairly often, the UAV even needs to climb at the final stage. For this reason, the number of cases with a too high impact speed is notably lower than that for the first controller. As noted earlier, the aircraft accelerates with idle engine even at very moderate glide angles. However, this can be considered as a deficiency of the UAV *Ariel* design, which is not intended for such an active manoeuvring.[1] Since this UAV is used in this study only as an example, this effect may be ignored.

6.2.3.2 Pivoting recovery boom

The next configuration to be considered involves the adjustable boom which can be rotated 70 degrees to either side (Fig. 3.11). This construction has been proposed to enable an almost unconstrained choice of approach directions, which is helpful to avoid crosswinds for the UAV during the final approach. A negative property of the orientating boom is large amplitude of its motion at the extreme angles, including significant lateral rocking due to ship roll motion. These amplitudes are analysed in Section 3.4.1. It should be kept in mind that an adjustable boom is a more complex and expensive installation than a fixed boom, hence it should be used only if it offers a substantial advantage in terms of performance.

To analyse performance of the controllers with the pivoting recovery boom, only an extreme boom position (70° to the left) will be considered. If it gives a noticeable positive effect, other positions may be investigated. In general, intermediate positions are expected to produce the results between those obtained in the current test and those corresponding to the neutral boom position (Table 6.5), with the extreme angle being the worst case in terms of boom amplitude of motion. What is to be investigated is whether an absence of crosswind will overcome the effect of greater boom oscillations and deliver a better guidance accuracy and success rate than those already achieved. Of the two possible approach directions with the given boom position, the approach from the stern of the ship is simulated. In reality for non-zero ship speed, the majority of recoveries will be attempted from this direction as it usually entails lower impact speed due to headwind component with respect to the ship. This direc-

[1] It may also be reminded that the approach speed is reduced with respect to the design value. This is discussed in Section 6.1.1.

tion is also expected to be slightly more difficult for precise approach due to potentially longer exposure to ship airwake.

Since the results for Sea States 2 and 3 were perfect in terms of success rate, the analysis starts from Sea State 4. Although the ship drive compensation is less useful in this configuration because the velocity component to be compensated is only $V_s \cos 70° \approx 0.34V_s$, it is employed in view of the results obtained in the previous test. The tests with no compensation are performed as well. Since the boom is supposed to be positioned so that little or no cross-wind component exists (this may be ensured using a simple vane to determine the wind over deck direction), wind azimuth is selected differently to the previous configuration. Considering the limitation of 70°, in the worst case (for tail or headwind with respect to the ship) wind azimuth will be 20° relative to the approach direction. In view of this, relative wind azimuth is chosen as a normally distributed random number with a standard deviation of 10°. Head or tailwind direction may occur with equal probability. The hard bounds are applied as usual (see page 282), ensuring that no severe tailwind is present and the headwind, if any, allows the aircraft to reach the ship. Other test parameters are the same as used in the previous section.

The collated results are presented in Table 6.6. Testing stopped at Sea State 5, because almost no improvement over the previous case is achieved. The first observation is that the second controller performs very poorly, no matter with or without the ship drive compensation. The main reason for this is much greater lateral amplitude of the boom (nearly four times that with the neutral boom position) for which the lateral guidance law is apparently not optimal. Combined with the problems identified earlier, it renders the second controller of little use in this configuration. Several crashes into the water have also been observed for this controller at Sea State 5.

Sea State	4				5			
Ship drive compensation	No		Yes		No		Yes	
Controller	1	2	1	2	1	2	1	2
Successful recoveries, %	99.6	89.6	98.8	91.8	90.4	63.6	91.2	73.2
Accurate recoveries,[1] %	97.2	81.2	95.6	82.8	68.0	47.2	74.5	56.4
Recoveries with large miss,[1] %	1.6	8.4	1.2	7.4	18.8	15.6	13.2	14.8
Recoveries with overspeed,[1] %	0.8	0	1.6	1.2	2.4	0.8	5.2	2.8
Complete failures,[2] %	0.4	0	0	2.2	2.0	11.6	1.2	5.2
Average PC [3]	24.1	45.1	25.0	40.3	56.8	70.1	51.4	55.5
Average horizontal miss,[4] m	0.46	0.68	0.51	0.55	1.0	1.0	0.90	1.0
Average altitude miss,[4] m	0.20	0.60	0.21	0.44	0.43	1.4	0.39	1.1

Percentages are given with respect to all attempts. Notes: [1] Considering (6.29). [2] See footnote 1 on p. 283. [3] For successful recoveries. [4] Excluding complete failures.

Table 6.6 Recovery performance statistics with the boom at 70°

As expected, the effect of the ship drive compensation is less pronounced than with the neutral boom position due to smaller ship velocity component to compensate and low cross-

wind. For the same reasons, the occurrence of complete failures is lower without the compensation as compared to the tests with neutral boom position. In addition, the success rate in some cases is slightly better than with the fixed boom. At the same time, the accuracy in terms of Performance Cost and miss distances is considerably lower. This is again due to greater boom oscillations. As a result, it may be concluded that such performance does not justify installation of the adjustable boom.

6.2.3.3 Side recovery boom

The last configuration that deserves thorough investigation employs the side boom as illustrated in Fig. 3.10. As discussed in Section 2.3.6.2, this location is more convenient from the point of view of the infrastructure of the frigates currently in service. On the other hand, it is more difficult to provide a necessary clearance to the ship structures with the side boom. Other potential difficulties include high amplitude of motion of the boom and prolonged exposure of the approaching UAV to the ship airwake.

In this case, the ship drive compensation is not necessary, because the UAV approaches along the ship's velocity vector. The performance tests have been conducted in this configuration similarly to those for the default boom location, starting from Sea State 3. The results are shown in Table 6.7. As it can be seen, the controllers exhibit quite poor performance, with the second controller prone to failures even in Sea State 3. Investigation of the trajectories allows to attribute such a behaviour of the controller #2 to its bias to the low altitudes and to the right, which is due to presence of the absolute terms in the guidance laws (6.22). In some circumstances, it causes failure of the positioning system (see page 289).

Sea State	3		4		5		6	
Controller	1	2	1	2	1	2	1	2
Successful recoveries, %	100	86.5	98.6	88.4	72.4	48.8	23.8	27.2
Accurate recoveries,[1] %	100	82.0	91.8	80.6	46.0	31.2	10.2	11.8
Recoveries with large miss,[1] %	0	4.5	6.0	7.6	23.6	16.8	13.2	15.4
Recoveries with overspeed,[1] %	0	0	0.8	0.2	2.8	0.8	4.6	1.4
Complete failures,[2] %	0	0	0	0	0.8	7.6	6.2	16.2
Average PC [3]	14.1	34.1	37.1	42.4	72.2	76.4	109	114
Average horizontal miss,[4] m	0.27	0.68	0.60	0.79	1.36	1.33	3.73	2.35
Average altitude miss,[4] m	0.19	0.66	0.27	0.73	1.26	2.27	2.13	2.47

Percentages are given with respect to all attempts. Notes: [1] Considering (6.29). [2] See footnote 1 on p. 283. [3] For successful recoveries. [4] Excluding complete failures.

Table 6.7 Recovery performance statistics with the side boom

There are two main reasons for such a poor performance of both controllers in this configuration. The first is very strong lateral oscillations of the recovery boom. Unlike the default boom position, ship motion causes not only translational displacement of the boom but also significant rotations (see analysis in Section 3.4.1). Apart from being an additional dis-

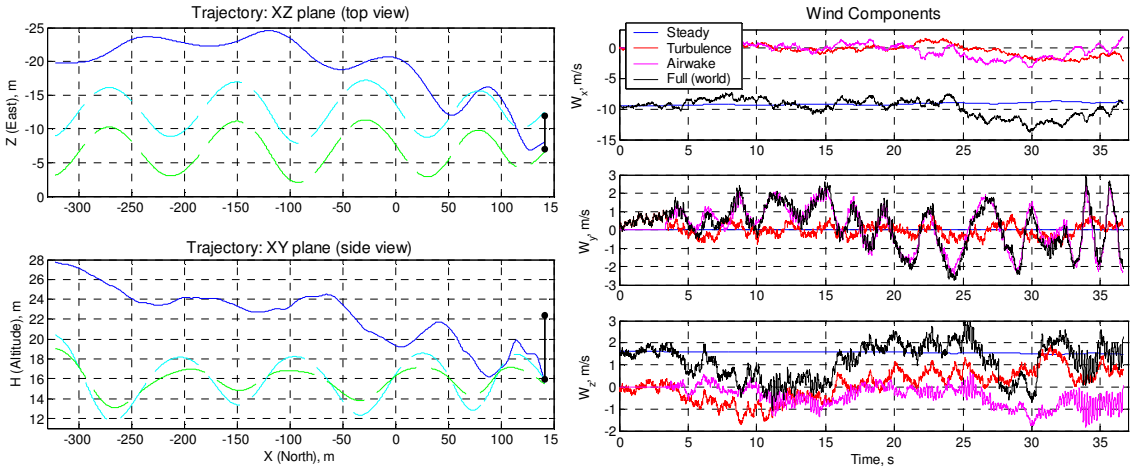


Fig. 6.21 An example of failure due to ship motion and airwake, the side boom configuration, Sea State 5. Dashed lines on the trajectory graphs represent the 'unrolled' along the flight path traces of the tips of the recovery boom (lateral position on the top view and vertical position on the side view). The bar on the right hand side illustrates the size of recovery window corresponding to final UAV state.

turbance of the target point as such, this interferes with the signals of the positioning system, which does not compensate this component of motion. It may be noted, however, that the order of this interference can be estimated to be not more than 8–10% in terms of accuracy (cosine of roll amplitude, see Section 5.2.1.2.1). To give a picture of the level of disturbances present in this configuration, a sample test case is demonstrated in Fig. 6.21. Note the traces of the tips of the boom (dashed lines). One can compare this graph with any of Fig. 6.12 to Fig. 6.18, they are generated for the same Sea State and thus for the same ship amplitude but in the default configuration.

High lateral translational oscillations are especially destructive in combination with a tailwind, which produces a higher closing rate. As indicates the analysis of failures, the UAV often reaches its physical limits, banking left and right with maximum rates at close distances in order to catch up with the current boom location.

The second reason is severe disturbances due to ship airwake, especially its vertical component W_y. A sample can be seen in Fig. 6.21 on the right. In free air, vertical turbulence is normally not as high as the horizontal components. However, ship motion generates a lot of vertical disturbances (wind changes up to 5 m/s within a second can be seen in Fig. 6.21; if not for the strong downwash in the last second of flight, this case would be successful). These disturbances come primarily from the periodic component of the airwake model (Section 3.2.4). At the same time, as noted in Section 3.2.4.2, validity of this component is questionable for high amplitudes. For this reason, the same tests have been repeated without the airwake model. The results are shown in Table 6.8 (Sea States 4 to 6):

Sea State	4		5		6	
Controller	1	2	1	2	1	2
Successful recoveries, %	100	98.6	89.2	86.4	25.8	29.6
Accurate recoveries,[1] %	95.6	89.2	60.8	62.4	8.8	12.8
Recoveries with large miss,[1] %	2.8	9.0	27.2	23.2	15.4	16.6
Recoveries with overspeed,[1] %	1.6	0.4	2.8	0.4	14	3.8
Complete failures,[2] %	0	0	0	0	0	2.2
Average PC [3]	25.9	40.4	62.2	61.9	104	113
Average horizontal miss,[4] m	0.51	0.55	1.16	0.89	3.17	2.68
Average altitude miss,[4] m	0.20	0.39	0.50	0.77	1.83	2.21

Percentages are given with respect to all attempts. Notes: [1] Considering (6.29). [2] See footnote 1 on p. 283. [3] For successful recoveries. [4] Excluding complete failures.

Table 6.8 Recovery performance statistics with the side boom, airwake excluded

Comparing it with Table 6.7, some interesting observations can be made. First, the contribution of airwake indeed proved to be huge. The results for Sea States 4–5 improved dramatically, approaching the performance level in the default configuration. In terms of complete failures rate, these are the best results achieved. At the same time, accuracy and success rate for Sea State 6 did not change noticeably. This indicates that in these conditions ship amplitude is the limiting factor (in Sea State 6, it is approximately 1.5 times as high as in Sea State 5).

In view of this, if the side boom location is to be chosen for any particular reason, a thorough investigation of ship airwake should be conducted. Only with a comprehensive and validated airwake model, reliable simulation results can be obtained. Other configurations are generally less affected by airwake and the respective results are expected to be more consistent with reality.

6.2.4 Operating envelope

The results obtained in the previous section can be used to ascertain the operating envelope of the recovery system. By examining the conditions under which recovery is successful and those under which recovery fails, it is possible to arrive at a set of limiting conditions for recovery. In addition, another set of tests can be conducted to determine the limiting conditions.

It is important to keep in mind that recovery is a stochastic procedure with many uncertain factors such as unpredictable gusts, turbulence and ship position. There is no envelope under which successful recovery is guaranteed, although the probability of failure in calm seas and mild winds may be extremely low. For this reason, the operating envelope can be considered as a probability field in the multidimensional space of operating conditions. Every point in this field shows the probability of successful recovery under the respective conditions. Some of the conditions may set hard bounds, beyond which recovery is physically impossible. Other conditions may set soft bounds, where probability of success gradually de-

creases as the conditions change. By applying a specified tolerance, the allowable range of conditions can be defined. Tougher tolerance will define a narrower operating envelope. Usually, it is up to the end user to take up the risk and to define an allowable rate of failures.

In the previous section, the effect of wind, turbulence and ship motion has been covered comprehensively and the success rate under the typical sets of conditions has been obtained. It can be seen, for example, that in Sea State 5 successful recovery can be expected in about 90% of cases (in default configuration). However, Sea State is only a general description of a wide range of conditions. It is a measure of severity of the atmospheric and sea conditions. Sea State encompasses the amplitude of motion of the ship, average wind speed and level of turbulence. Meanwhile, the actual values of some of the parameters may have a significant effect upon the success rate. For example, recovery with a headwind may deliver a substantially higher success rate than recovery with a crosswind of the same magnitude in the same Sea State. If this information is known, it may help to plan the recovery procedure more carefully. This is especially important for the parameters which are controllable to some extent, such as the aforementioned wind direction (with respect to the flight path).

From the previous analysis it can be concluded that wind speed and direction play major role in determining the success of recovery. Apart from affecting the guidance accuracy and hence the success rate, wind sets hard bounds to the recovery process. For this reason, additional investigation is conducted in this section to determine the limits of operation of the aircraft in terms of wind parameters.

The absolute limits are generally determined by the airspeed of the aircraft. As a rule, it is not productive to expect the UAV to cope with the winds that are greater than its airspeed. Whilst it is not impossible to provide recovery in greater wind (relying on the airstream instead of compensating it), practicality of this feature remains dubious. At the very least, navigation control will be difficult if possible at all in such a strong wind. Therefore, considering that the approach airspeed is 22 m/s, this is set as the extreme value for testing.

Simulations are performed in the default configuration (fixed recovery boom at the bow) for the controller #1 with the ship drive compensation enabled. This is identified as the most suitable combination, considering the results of the robustness and performance tests. The simulation scenario involves explicitly defined steady wind parameters, Sea State and ship speed, while all other parameters are chosen similarly to the performance tests. In this case, Sea State determines only the amplitude of ship motion. Note that turbulence intensity is determined by the wind speed. Since the steady wind model includes variation of the wind speed with altitude, wind speed is recalculated to the reference altitude of 6 m. In order to provide the maximum wind of 22 m/s at the actual altitudes of flight (about 15–40 m), the maximum reference wind at 6 m is taken as 16 m/s. At the altitudes above 36 m, average wind

speed may exceed the normal airspeed of the vehicle (note that the nominal altitude at the beginning of approach is 30 m with the allowed standard deviation of 5 m). If the initial altitude happens to be higher than 36 m, the UAV will never reach the ship (in the case of headwind) unless it descends to lower altitudes. Even then, it may take too long time (more than the limit 120 s), considering that at the boom level (16 m) the wind will be greater than 19 m/s. This fact may contribute to the success rate for extreme winds.

During the tests, wind is varied as follows. Wind azimuth changes deterministically from 0 (tailwind) to 180° (headwind) with step 30°. For each azimuth, wind magnitude is varied from 6 to 16 m/s with step 2 m/s. However, to exclude high tailwinds which are not normal for recovery, the range of magnitudes for azimuths 0° and 30° is 5 to 8 and 5 to 9 m/s respectively with a smaller step of 1 m/s. Wind magnitudes below 5–6 m/s are not tested, it is expected they will not represent a noticeable problem.

For each wind setting, 100 simulations are performed. The number of successful recoveries, as well as the number of accurate recoveries that satisfy the conditions (6.29), is retained. It determines the success rate of recovery in the respective conditions. The random factors include initial UAV position, initial phase of ship motion, turbulence (including airwake) and gusts. In some cases, the combination of these conditions may not be consistent within the typical real world environment. For example, severe winds rarely occur in calm seas, although locally this is not impossible. Nevertheless, such conditions allow to ascertain the effects of wind more informatively.

On the basis of these results, contour plots with the lines of equal probability of success are generated. Due to the very large amount of computation required (nearly 4000 simulations for each case), only the results for two Sea States (2 and 5) and for an idle ship are presented. They allow nevertheless to trace the influence of ship motion on the success rate for given wind parameters. Sea State 2 represents a calm sea, where ship motion has very little effect on guidance accuracy (see Table 6.5). Therefore, the pure effect of wind can be observed. In contrast, ship motion in Sea State 5 is significant enough to cause occasional failures even in low winds, but on the other hand it is not as critical as in Sea State 6 to overshadow the effect of wind.

The plots for the above conditions are presented in Fig. 6.22. The two plots on the left hand side illustrate the chances of capture of the arresting wire by the UAV. They do not take into account the impact speed, which may be quite high for tailwinds. Nevertheless, these graphs are useful to determine the guidance abilities of the controller. It can be seen that capture is guaranteed for crosswinds up to 12 m/s with any practicable tailwind, even in rough sea. For headwinds, ship motion becomes increasingly important. In calm seas, wind up to 14 m/s does not represent serious guidance difficulties. Some degradation is observable only

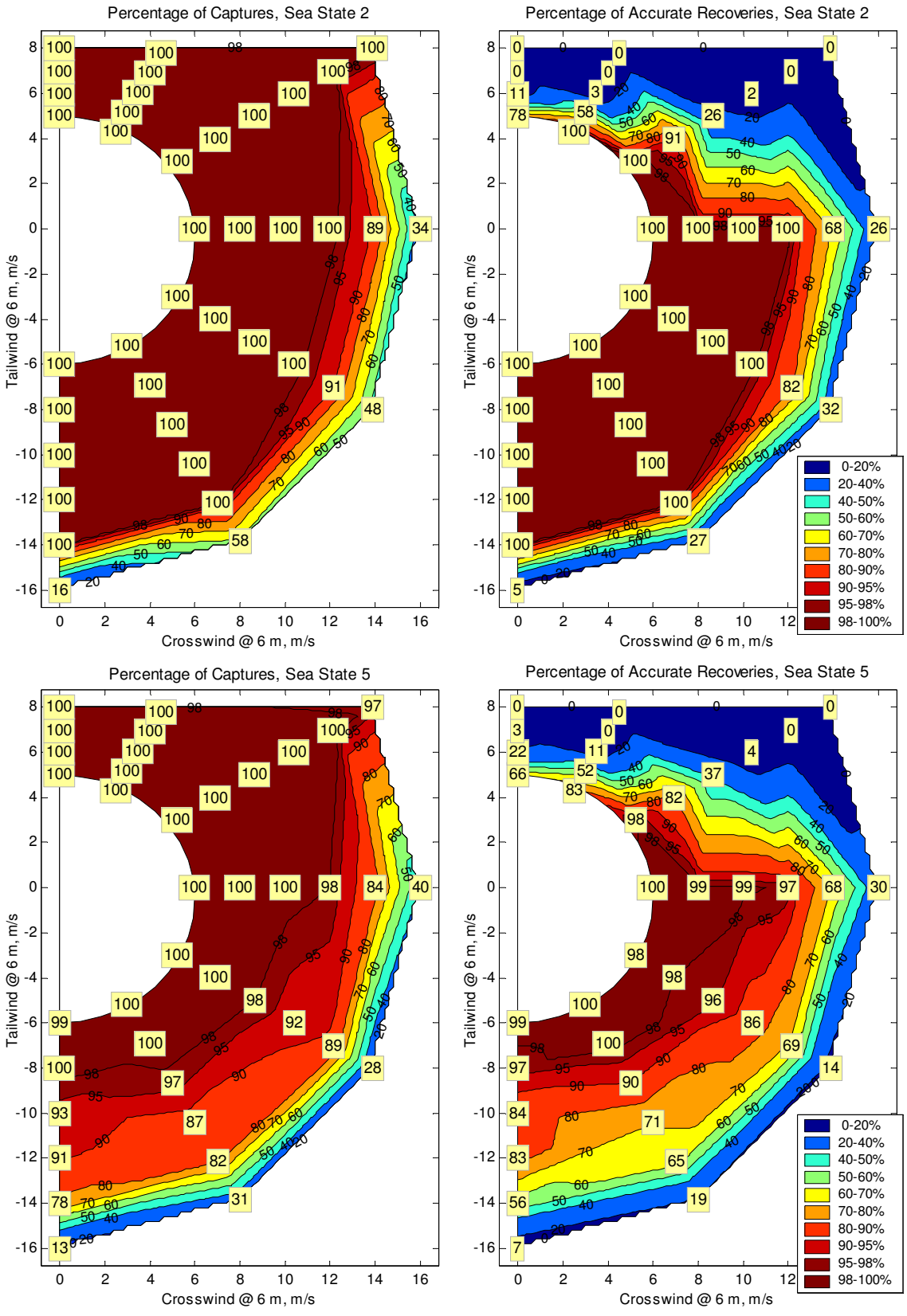


Fig. 6.22 Probability of successful capture of the arresting wire and probability of meeting the accuracy requirements (6.29) for different winds and Sea States. The contours are interpolated from the data points shown in boxes. White area is untested.

when the crosswind component approaches 12 m/s. An abrupt drop in capture probability starts as wind speed exceeds 14 m/s, indicating the physical limitation of the aircraft. In rough sea, headwind plays a dominant role in determining the success rate. It indicates that low closing rates are unfavourable for guidance to a moving target. This somewhat counterintuitive result comes from the absence of guidance at the last metres of approach (see Section 5.2.1.3). This is an important observation, which may lead to redesign of the terminal guidance algorithm and/or positioning system when physical implementation is considered. However, even for winds up to 14 m/s in any direction, there are about 80% chances of capture. For stronger winds, the chances fall quickly, similar to the case of low amplitudes of ship motion

It is important to note that these data consider the nominal dimensions of the recovery gear accepted in this work (see Sections 2.4.2 and 2.4.3), i.e. 10 m long cable hook and 6 m long arresting wire with 0.5 m safety margins. If one is to limit the length of the arresting wire to 3 m (plus possible margins), right hand plots should be considered. They represent the probability of successful capture, taking into account tighter tolerances to the miss distance and also impact speed. The tolerance to the vertical miss (1 m) is strict, but it cannot be easily translated into the cable length requirements (see footnote on page 281).

In a calm sea, tougher miss tolerances provide almost the same rate of success, which can be seen comparing the left and right plots for tailwinds. The 95% and 98% probability envelopes are very similar. Only extreme winds do cause a noticeable difference, apparently due to the high level of turbulence. Presence of a headwind, however, quickly causes violation of the maximum allowable impact speed, making safe recovery for tailwinds greater than about 6 m/s very unlikely. It should be noted that this limit is fully determined by the design requirements. There is little that any controller can do, considering that the UAV already flies with nearly minimum airspeed.

With greater amplitude of ship motion, the changes in success rate due to tougher tolerances are more evident. Only for wind speed below 6–7 m/s and tailwind not more than 2–3 m/s, safe recovery can be almost guaranteed. This implies, however, that in mild turbulence accurate and reliable guidance is possible even to an intensely oscillating ship. In a standard for Sea State 5 wind speed 25 kn (12.5 m/s) [3], there are 70% (at an azimuth of 150°) to 97% (for pure crosswind) chances of accurate recovery, which agrees well with the results of performance tests (Table 6.5). In the presence of a tailwind, impact speed becomes the dominant factor, and the picture is similar to that at Sea State 2. Interestingly, the results for Sea State 5 demonstrate noticeable improvement for tailwind scenarios as compared to Sea State 2. This is due to violent ascending manoeuvres which take place in some cases where significant altitude correction must be made in the last moments. Such manoeuvres cause the speed to fall, with the autothrottle initially unable to compensate it due to the slow response of the engine.

In a calm sea, these manoeuvres are rarely needed, and the aircraft is almost sure to exceed the speed limit if the tailwind component is greater than a certain value (about 5.5–6 m/s).

Overall, the controller demonstrates quite consistent and predictable behaviour across the operating envelope, showing good performance even in severe conditions. In a calm sea, where no active guidance is needed, the envelope is limited by the physical limitations of the aircraft and recovery equipment. In a rough sea, the UAV tends to withstand a crosswind better than a headwind due to limitations of the positioning system. With appropriate modification of the system, the operating envelope can be extended. This is, however, a highly specific technical work which requires a dedicated investigation (see Section 2.4.3.4).

6.3 Concluding remarks

In this chapter, an application of the Evolutionary Design is demonstrated. The aim of the design was to develop a controller which provides recovery of a fixed-wing UAV onto a ship under the full range of disturbances and uncertainties that are present in the real world environment. The design process relies on the system models described in Chapters 2, 3 and 5. The methodology and the necessary theoretical background are given in Chapters 5 and 4.

The first part of the chapter is dedicated to the synthesis of the controller, which is subdivided into several specialised blocks. Due to such subdivision, controller synthesis is a multistage process. However, the approach employed for synthesis of each block is very similar. Evolutionary algorithm is used as a tool to evolve and optimise the control laws. One of the greatest advantages of this methodology is that minimum or no a priori knowledge about the control methods is used, with the synthesis starting from the most basic proportional control or even from 'null' control laws. During the evolution, more complex and capable laws emerge automatically. As the resulting control laws demonstrate, evolution does not tend to produce parsimonious solutions. Many solutions contain apparently atavistic 'dead branches' and the parts that would seem strange and unjustified for an engineer used to classic designs. As pointed out in Chapter 4, this is a normal behaviour for any evolutionary process driven by fitness, and it should be *accepted* before attempting to analyse the obtained solutions.

Nevertheless, where the situation permits, simple control laws do emerge, as demonstrated by the ailerons control and to some extent the guidance laws. This is never guaranteed, however, and if the engineer has a strong preference to simple designs, this feature may be included in fitness (heavily penalising lengthy solutions), or a simpler solution may be selected from a number of the solutions obtained in different runs of the algorithm, even if it delivers slightly inferior performance. It should be kept in mind that Evolutionary Design is a stochastic process and may produce sufficiently different solutions every time. Since the first way significantly impedes the process of evolution, the second approach has been used in this

work. The experience shows that even though the solutions may be seemingly different between the runs, evolution tends to converge to almost the same fitness value (see for example Fig. 6.1a, Fig. 6.5a, Fig. 6.7), demonstrating remarkable robustness and indicating that a near optimal solution can be found. In very limited cases, however, it may take too long time for the evolution to discover the core of a potentially optimal solution, and the process does not converge. More often than not, this hints at a poor choice of the algorithm parameters. However, this may happen even with nearly optimal settings. Due to the nature of the Evolutionary Design algorithm, in which the solutions only grow during the evolution, it becomes less likely that the optimal structure will be eventually discovered at the later stages. For this reason, the algorithm does not include automatic detection of stagnation and is monitored and terminated manually. It is not very uncommon to interrupt a slowly progressing evolution and to start the process again.

The most important and difficult problem in Evolutionary Design is preparation of the fitness evaluation procedure. Evolutionary Algorithms may be extremely effective for solving difficult problems, but nothing will replace the definition of the problem itself. For an EA, fitness evaluation procedure is such a definition. For a single stage ED and for the final stage of a multi-stage ED, the fitness value may be based on the performance of a solution in a comprehensive environment as it is defined in the original problem statement. However, this may not be applicable at the initial stages. Therefore, special intermediate problems must be defined. Correct definition of such problems (sample tasks) is crucially important. An unsuitable task may lead to development of controllers which are highly optimised for this specific task but may fail in a complete system. This demands good understanding of what is required from these intermediate controllers. In some cases, the sample task in itself may lead to a multi-stage evolution, which is demonstrated by the two-stage evolution of the autothrottle. Initially, a PID structure is quickly optimised, and only then a complex structure is evolved together with the tightly coupled elevator control.

Computational considerations are also of the utmost importance. Robustness of EAs comes at the price of computational cost, with many thousands of fitness evaluations required. For problem solving algorithms such as the ED algorithm (in contrast to optimisation algorithms), supercomputing power is often a necessity. Unfortunately, it has not been available for this research. This led to a significant simplification of both the simulation models and the fitness evaluation procedures employed during the evolution. Some of the models were replaced by their simpler approximate equivalents, and variations of many of the model parameters in order to obtain a robust controller were not conducted. However, with some understanding of flight dynamics, variation of the most important aerodynamic parameters has been included into the fitness evaluation procedure, which allowed to obtain a remarkable robust

controller. It should be noted that even with such simplifications, the design problem is highly nonlinear and multidimensional.

Controller testing constitute the second part of this chapter. Due to a lack of hardware, the testing is limited to the simulation environment. Obviously, for a comprehensive design for physical implementation, the hardware would be available and physical testing of the controller would also be conducted. In addition, physical design would imply a more thorough specification of the parameters, both for the systems involved and for the operational environment. In this work, many of the model parameters were chosen arbitrarily.

Controller testing has several goals. First, it demonstrates the capabilities of the controller in terms of performance and robustness in various conditions. Second, it enables to identify the issues in both the controller design and the systems involved that affect performance and to suggest ways for further improvement. Finally, it allows to validate the design methodology applied.

The simulation testing covers the entire operational envelope and highlights several conditions under which recovery is risky. All environmental factors—sea wave, wind speed and turbulence—have been found to have a significant effect upon the probability of success. Combinations of several factors may result in very unfavourable conditions, even if each factor alone may not lead to a failure. For example, winds up to 12 m/s do not affect the recovery in a calm sea, and a severe ship motion corresponding to Sea State 5 also does not represent a serious threat in low winds. At the same time, strong winds in a high Sea State may be hazardous for the aircraft. Overall, the performance is impressive, with a reliable recovery possible in very rough environmental conditions, in which the UAVs currently in service are typically not operated.

Limitations of the local positioning system employed in this research result in two serious issues identified during the tests. Specific handling of the terminal phase of approach significantly degrade the chances of success for strong headwinds and high Sea States. If this issue is fixed, the operational envelope may be considerably enhanced for headwinds. The system also fails to operate correctly at a high azimuth of the aircraft with respect to the recovery boom and close distances. Whilst in such conditions safe recovery is already unlikely, correct detection of this situation may be used for issuing a go-around directive to prevent a possible crash. In addition, some degradation of positioning accuracy is observed where a large lateral angular motion of the boom is present. It should be noted, however, that the positioning system is the least specified system used in this study (in fact, it has not been specified at all). It is used only as a design example to provide the means of local positioning, which is a key element for recovery. Its principle has been chosen in view of simplicity of implementation and potential availability for small inexpensive UAVs.

Probably the most important consideration in the context of this research is validity of the Evolutionary Design methodology. Whilst it is evident that ED did produce a capable controller that satisfies the original problem statement, several important observations can be made. First of all, in the absence of similar solutions to be compared with, it is unclear how close the result is to the optimum. Considering remarkable robustness of EAs for arriving at the global optimum, it may be believed that the controller is very close to the optimal solution. However, comparing performance of the two generated guidance controllers in different configurations, it may be concluded that there is still room for improvement. The main reason for that is believed to be the limited scope of the testing conditions used in the evolution process, which is mainly due to shortage of computational resources. The evolutions were conducted only in the default recovery configuration in a single Sea State setting with no ship drive. Not surprisingly, the resulting performance has been found to be much better in this configuration, although this is attributed also to the more favourable conditions (lesser boom oscillations and small exposure to airwake). It is understandable why a more universal Proportional Navigation controller showed superior results during the tests across the whole range of conditions, although the second controller originally had a better fitness. In general, the evolutionary approach can provide a highly optimal solution but does not guarantee robustness to untested conditions. For this reason, maximally diverse conditions should be included in the testings for fitness evaluation if sufficient computing power is available.

Analysis of the guidance laws produced by the ED algorithm can provide a good illustration of the above. The guidance law of the controller #2 (6.22) contains, apart from the PN core, absolute terms that cause the aircraft initially to descend and to deviate to the right. During the tests, it has been found that this behaviour is responsible for several types of failures. Then what are these terms for, how could they grow to such relatively large values? The fact is that this question is, in general, not applicable to the solutions born in an evolutionary process. There is no particular reason why one or another feature appeared in the structure. It may be purely unjustified. Apparently, this guidance law happened to be near optimal in the conditions persisted during the evolution, and this is a sufficient reason. However, as soon as the conditions change, this solution is no longer optimal. In this case, the problem could be identified relatively easily, but in general this may not be so obvious. Once again, careful planning of the fitness evaluation procedure is crucially important in order to obtain a robust design.

On the whole, Evolutionary Design is a useful and powerful tool for complex nonlinear control design. Unlike most other design methodologies, it tries to *solve* the problem at hand automatically, not merely to optimise a given structure. Although ED does not exclude necessity of a thorough testing, it can provide a near optimal solution if the whole range of conditions is taken into account in the fitness evaluation. In principle, no specific knowledge

about the system is required, and the controllers can be considered as ‘black boxes’ whose internals are unimportant. Successful design of the controller for such a challenging task as shipboard recovery demonstrates great potential abilities of this novel technique.

Chapter 7. Conclusions

The growing use of Unmanned Aerial Vehicles (UAVs) in both military and civil applications has led to an increase in operational roles and areas of their deployment. A need for UAV operation off maritime platforms has been identified long ago, but at present it is only partially satisfied. One of the most (if not the most) difficult engineering challenges that hampers wide deployment of shipboard UAVs are the problems associated with launch and recovery.

Due to the extremely confined area available on maritime platforms (ships, oil rigs etc.), maritime operation generally requires the aircraft to either have Vertical Take-Off and Landing (VTOL) capability or some form of launch and recovery assistance. VTOL capability often places severe limitations upon range, speed and endurance of the aircraft. For this reason, operation of traditional fixed-wing UAVs with assisted launch and recovery may be preferable. However, this approach requires a substantial research effort put into the development of such assistance.

The launch assistance usually takes the form of catapult launching or rocket assisted takeoff. This subject is comprehensively covered in [47], where the controller for autonomous catapult launch of a fixed-wing UAV is developed. The current research supplements this work, offering a technology of autonomous recovery of similar class UAVs. Unlike launch, the techniques of UAV recovery are very diverse. These techniques are analysed and a new method of recovery is proposed in the current work to enable autonomous recovery of small to medium-size UAVs in extreme environmental conditions. The research contained in this body of work has demonstrated that fully autonomous recovery of fixed-wing UAVs to a moving ship is possible even in very difficult conditions, in which the UAVs currently in service are not usually operated.

A major effort has been directed in this work towards development of a methodology that is capable of designing a controller that will provide guidance and flight control for the aircraft during the recovery stage. Since it was unclear initially which type of guidance and control method is the most suitable for this recovery method, a problem solving evolutionary algorithm has been applied to produce automatically not only the optimal parameters of the controller, but also the whole structure of the control laws. As a result, the developed methodology is potentially suitable to generate controllers for a wide range of control problems, which are characterised by the need of global optimisation in the multidimensional design space.

This chapter discusses the methodology developed and the application of this methodology for design of a recovery controller. The limitations of both the methodology and the recovery method are examined and possible areas for future research are suggested.

7.1 Discussion

This research has produced several key results. They can be grouped into two distinct areas: those relating to the recovery technique and those relating to the design methodology.

7.1.1 UAV recovery

The first aim of this research was to propose a recovery technique that could be used to recover a small to medium-size fixed-wing UAV onto a moving ship with limited deck space in the presence of large atmospheric disturbances. The analysis done in Chapter 2 concluded that none of the existing techniques is readily suitable for this goal. A new recovery method, termed the *Cable Hook Recovery*, has been proposed and discussed. It features a relatively long flexible line (the arresting cable) carried onboard the aircraft with a self-latching hook attached at the loose end. A damped arresting wire, similar to those used on large aircraft carriers, is stretched over a boom or in a similar manner outside the deck space on the ship. For recovery, the UAV extends the arresting cable and passes over the ship's arresting wire. The lock captures the arresting wire and the aircraft is stopped in midair. The aircraft is then winched up and recovered onto the deck in a manner typical for crane operations.

Due to relatively large miss allowances available in this method and unimportance of the final attitude of the aircraft, this technique is believed to have a significant advantage over the methods currently employed or proposed for this purpose, especially in the case of severe weather conditions. However, like with any new techniques, it has a great deal of uncertainties, and its real performance and capabilities are largely unclear. Complete design of this system requires lots of effort and work of a team of experienced engineers. In this research, full engineering design of such a system was not considered. Instead, a detailed conceptual design of the system has been conducted and the initial design requirements for possible future development of the system have been determined.

In order to demonstrate the benefits that implementation of the Cable Hook recovery may deliver, the aerial part of this system has been modelled and the UAV controller that utilises its abilities has been developed. The parts included into the model are the UAV with the recovery controller, the arresting cable, the atmosphere (including turbulence and airwake) and a simple model of ship motion. The simulation testing described in Chapter 6 have shown that this recovery technique potentially allows to recover the UAV even in very harsh conditions, with the wind speed approaching the airspeed of the aircraft and the amplitude of ship

motion reaching 22° (on roll), assuming that the shipboard recovery gear works well and provides necessary deceleration of the vehicle and further recovery. Understandably, in extreme conditions safety of the recovery may be compromised. However, there are considerable chances that recovery will be successful, and even if not—as one of the attractive features of this recovery method—another attempt will be possible.

Several conclusions can be made about the issues related to the elements of the system. The first one relates to the aircraft itself. While no specific restrictions have been identified regarding the aircraft design that may be required for this particular type of recovery, some points should be kept in mind when considering actual design of the UAV and the recovery system. A considerable amount of rapid manoeuvres may be necessary for successful recovery in a high sea. Therefore, the aircraft must possess a sufficient level of manoeuvrability at a low approach speed. If a necessary level cannot be achieved at the required speed, then either the approach speed may be increased and the recovery gear reinforced, or greater dimensions of the recovery gear (lengths of the cable and arresting wire) may be provided, which will allow the UAV to have a greater miss. Both ways have disadvantages. It may be noted that for the controller produced in this study, manoeuvrability requirements are expressed in terms of trajectory angular rates, thus a higher speed will entail proportionally higher requirements to the body accelerations of the aircraft.

A specific issue with the UAV used as a design example in this study (*Ariel*), but which also may arise with other aircraft, is concerned about vertical manoeuvrability. Both steep climbs and descents may be commanded by the controller, particularly at the later stages of approach. The aircraft speed control should respond quickly enough to these conditions. This may be particularly difficult for slowing down in a dive, as majority of the UAVs do not have controllable airbrakes. If acceleration in such a condition is unavoidable, a greater allowance for the impact speed must be given. This may lead to reinforcement of the aircraft.

Another system which has a direct impact on the recovery performance is the local positioning system. This system is a key element for recovery, as the UAV must know the exact current location of the arresting wire in real time (or rather the relative angular velocity for the controller produced for the *Ariel*). The positioning system employed in this work has been chosen in view of simplicity of implementation and potential availability for small inexpensive UAVs. However, it imposed several restrictions on recovery, particularly relating to the headwind conditions. To overcome this, it is desirable that the actual system provide positioning information until the very moment of capture.

7.1.2 Evolutionary Design methodology

The development of a design methodology that allows to produce a controller capable of recovery of a UAV using the proposed Cable Hook recovery technique was the major aim of this thesis. As noted above, this was not an engineering project with a physical implementation, and the actual simulation results have merely illustrative purposes. It is likely that the actual design will be carried out using different specifications, including another UAV, positioning system and recovery equipment. The design methodology should be flexible and able to produce possibly the most optimal controller for any reasonable initial requirements.

The methodology is based on Evolutionary Algorithms (EAs). Application of EAs to full control design is a relatively new undertaking. Unlike the numeric optimisation techniques employing EAs, problem solving EAs are still rarely used in control community. This research aimed, among other things, to demonstrate the power and flexibility of evolutionary approach in a full cycle design process. The developed methodology has successfully been used to synthesise a controller capable of recovery of the UAV *Ariel* in a wide variety of conditions. The methodology possesses a large amount of flexibility, with possible applications to a broad range of control tasks.

The Evolutionary Design methodology attempts to *solve* the problem at hand, instead a traditional approach which tends to optimise an already given control structure. The main task of an engineer therefore shifts to an appropriate problem definition. Given a defined problem, the ED algorithm is then applied to produce a near optimal solution automatically. For many engineering tasks, the original problem is too complex to be solved directly within a sensible time, given that the computational resources available are limited. In this case, the original problem may be broken into several simpler ones and the ED then applied to each problem individually. Such an approach is demonstrated in Chapter 6 with a five-step synthesis of the recovery controller.

The solutions produced by ED (as well as by any problem solving evolutionary technique, e.g. Genetic Programming (GP)) have specific properties which make them unusual for traditional engineering way of thinking. The evolution does not tend to produce logically correct, justified, consistent or parsimonious solutions. Nevertheless, these solutions may be nearly optimal in terms of solving the problem at hand. As a consequence, behaviour of such solutions in the conditions different to those they were evolved in may be unpredictable. For this reason, the problem statement to be solved by ED (which takes the form of the fitness evaluation procedure) should include all the conditions expected in the real environment. This may lead to significant increase of the computational power required. However, with some understanding of the controlled system, the range of training conditions can be reduced to a minimum without compromising robustness of the solution. Indeed, a flight controller with

very good robustness to a great number of model uncertainties has been evolved using carefully selected model variations.

Despite still being computationally expensive, ED achieved remarkable efficiency in terms of computational cost. Where a supercomputer was needed to solve a relatively simple control problem using GP, a significantly more complex problem has been solved on a personal computer using ED. This is in part due to efficient encoding of the control laws, developed for this technique.

In addition, it has been confirmed that evolutionary methods are able to invent human-competitive solutions, with the Proportional Navigation guidance laws emerging during the evolution.

7.2 Limitations of this research

Like any research, this work has several limitations. These limitations exist in both the design methodology developed and the application of this methodology. These limitations will be discussed in this section.

7.2.1 Limitations of the Evolutionary Design

The first point to note is that ED, as well as any evolutionary method, does not produce inherently robust solutions in terms of both performance and unmodelled uncertainties. As noted above, behaviour of the solutions in the untested during the evolution conditions may be unpredictable. A sufficient coverage of all conditions included in the evolution process invariably requires a large number of simulations, which entails a high computational cost. When the amount of uncertainties in the system and the environment becomes too large, computational cost may become prohibitively high. At the same time, subdivision of the problem into smaller ones and reduction of the number of environmental factors in order to reduce the amount of uncertainties require good understanding of the system. This severely compromises one of the main advantages of ED, which states that little, if any, a priori knowledge of the system is needed.

Another limitation is that evolutionary techniques have little analytical background. Their application is based largely on empirical experience. In particular, most algorithm settings such as population size, selection pressure, probability of genetic operators etc. are adjusted using trial and error method. Whilst EAs are generally robust to these settings, it should be expected that the settings used for controller synthesis in this work may be far from optimal for another application, and their optimisation will take a significant time.

Being a stochastic method, ED may produce very different solutions each run. Although for some applications this is a desirable property, it makes cross-validation of the re-

sults almost impossible. The only way to verify and validate the obtained solutions is comprehensive testing in a full simulation environment or within a real system.

A more specific limitation is the range of possible control laws, which is determined by the encoding of the control equations and by genetic operators applied. For this research, encoding has been optimised for calculation speed and allowed to handle only a certain class of nonlinear equations. For applications with sufficiently different control objectives and/or different systems, the encoding and the relevant genetic operators may need to be modified. However, this is a common approach in EA community, and this modification does not contradict the ED methodology as such.

7.2.2 Limitations of the developed recovery controller

The limitations of the controller produced in the ED procedure are generally not major and do not affect the validity of the results presented. The main limitations are due to limited validity of the system models available. For a complete control system design, higher fidelity models will be essential.

The aircraft model is limited in several ways. It may be quite inaccurate when used in extreme conditions, particularly at high angles of attack and sideslip. In most cases, the aircraft does not exceed stall angles during the approach; however, a validated at high angles dynamic model may be required for correct implementation of the angle of attack limiter. The sensors and actuators models are rather primitive and allow only a qualitative estimation of the effects of lags and noise present in these systems.

The cable model is the only dynamic model of the components of recovery gear considered. Whilst it is built upon widely accepted principles, it may be insufficient for complete simulation of the recovery process, including the capture and post-capture dynamics. Accurate simulation of the process of capture may reveal some constraints which have not been taken into account in this work. For example, the cable was allowed to slide at a high angle to the arresting wire (this happens when the UAV approaches in a strong crosswind), implying that capture is still possible. However, in reality capture may not be guaranteed in these circumstances.

One important simplification used for the cable model is uniformity of the atmospheric disturbances along the cable, which may not be valid for the cable of a considerable length. Implementation of a more realistic model may requite a significant redesign of the wind model. In addition, parameters of the cable were chosen arbitrarily as a first approximation, only to provide initial control system design requirements. These parameters may be reconsidered in an iterative design process of the actual system.

The ship model is quite simple and includes behavioural simulation of the periodic ship motion observed on a real ship. It uses a harmonic approximation to all six components of motion, which are fully uncoupled. At the same time, random ship motion is not absolutely periodic and a significant amount of coupling should exist between the components. This may have either a beneficial or adverse effect on recovery. The amplitudes of periodic motion used in this research reflect a fairly conservative scenario for each Sea State (using the Significant Wave Height setting, which is 1.6 times the average wave height); however, larger displacements may occur occasionally in real world conditions. Apart from a more realistic simulation of ship motion, inclusion of a wave model in order to enable simulation of wave clearance during the approach would allow a greater confidence in the recovery success probabilities. In addition, due to the deterministic nature of the ship model used, it is not suitable for any controllers which try to predict the future angular position of the ship. Although prediction has not been employed in this work, it may be suggested for further research. This also includes any possible simulations with a human pilot in the loop (as a backup means, for example).

The wind model provides standardised parameters of atmospheric properties, steady wind and free-air turbulence, based on accepted military standards. Whilst standardised parameters are convenient for compatibility with other research and cross-validation, they may not represent the whole range of possible operating conditions. The effects of temperature, precipitations, excessive humidity (which is likely at sea) and many other factors have not been studied. At the same time, correct simulation of these effects would require comprehensive models of other systems involved, particularly of the UAV engine.

An attempt has been made in this research to produce a general ship airwake model. Unfortunately, lack of validation data and ready airwake models was unavoidable, because such models are either in development or publicly unavailable. Although the designed model is based on a well established military standard, it includes many new parameters whose effect is not validated. At the same time, it does not include interaction with the ship model, although it is clear that the actual ship position does influence the picture of air currents around the ship. Nevertheless, the simulation experiments have shown that airwake plays major role in determining the recovery success rate, particularly for the side recovery boom configuration. A validated airwake model will be essential for further development of the recovery system.

Like with any new technology, lack of some accurate models (particularly of the recovery gear) is unavoidable without a thorough engineering analysis of the respective systems and possibly, a hardware testbed. Inaccuracies found in other models, whilst being slightly detrimental to the research, do not compromise applicability of these models to the aircraft recovery simulation. These models still represent reasonable approximations to the type of

behaviour that would be experienced by an aircraft being recovered. In particular, the ship model will still provide sufficient movement of the recovery boom and the wind model will perturb the aircraft. In addition, the aircraft model represents an aircraft with severe performance limitations, which gives rather a conservative picture of achievable operating envelope.

One of the limitations of this research is the lack of physical implementation of the recovery equipment and controller. This physical implementation would comprehensively prove the advantages of the controller design and the recovery method. However, the high cost, long development process and unavailability of the aircraft[1] required for such a demonstration are outside the scope of this research.

7.3 Further work

The work presented in this thesis has exposed several possible areas for future research. These are related to both the Evolutionary Design methodology and its application to the recovery controller design, as well as to the shipboard UAV operation in general.

7.3.1 Research opportunities on the Evolutionary Design

The ED methodology proved to be easy to apply and extremely suitable for current application. However, the main effort has been directed towards the practical implementation and application of this design method, with little investigation into its general functionality and efficiency. As stated above, application of evolutionary methods is largely based upon empirical evidence of their work, with little theoretical support available. Whilst expanding the theoretical background in this area is desirable, it is possible that this is not feasible to the extents suitable for practical applications, considering the current state of evolutionary science. Therefore, further practical applications of this method, possibly to other control problems, may provide the necessary background and further validation of the methodology.

An investigation into the most effective algorithm settings, such as the population sizes and probabilities of genetic operators, will also be beneficial. Extension and optimisation of the control laws representation may help to expand the area of possible applications as well as improve efficiency of the algorithm.

A large room for improvement exists in the computer implementation of the algorithm. Whilst every effort has been made in this work to optimise the internal algorithmic efficiency, significant optimisation of the coding (programming) is possible. First of all, the control equation parsing algorithm can be rewritten in a lower level programming language such as C. This alone may give an approximately 2 to 6-fold increase in computation speed. Another

[1] Development of the UAV *Ariel* has stopped and the only aircraft has reportedly crashed.

area for improvement is parallelisation of the algorithm. Evolutionary methods are extremely suitable for parallel computation. The concept of distributed parallel computation, where the calculations are distributed over a network (possibly global) of different computers, can also be implemented relatively easily. This may allow to achieve a supercomputing power without entailing high costs.

7.3.2 Application of the recovery controller design methodology to other UAVs

An immediate area for further research is the application of the controller design methodology presented in Chapters 5 and 6 to other UAVs. The methodology should be immediately applicable for the design of a recovery controller for a fixed-wing UAV. However, high fidelity dynamic models of the aircraft and other components are essential in this process. Since exact specifications to the recovery equipment are not yet defined, a multistage iterative design process may be needed to obtain the optimal requirements to the dimensions of the arresting wire and cable. Their requirements may be different for different UAVs.

One of the limitations identified in this research that affects the operating envelope is passive speed control. In the designed controller, the autothrottle always maintains a recommended airspeed. However, a higher airspeed may be more optimal in the presence of a strong headwind. Evolution of a 'smart' autothrottle may be integrated into the design process. This will not change synthesis of the flight controller significantly. The only needed upgrade is redesign of the throttle sample task, with a varying (instead of a constant) reference signal. The guidance controller will include an additional control law for speed control, plus an appropriate modification of the fitness evaluation procedure.

The methodology can also be used for application to other recovery methods. For example, it should be readily applicable to produce a controller for the Net Capture recovery, which requires very similar guidance. Especially beneficial may be application of ED to the methods which imply very complex and highly nonlinear dynamic control, such as the Deep Stall recovery. For this type of control, the methodology may be even easier, with direct synthesis of the full flight controller attempted. However, the structure of the control laws may need to be modified to include additional nonlinear elements.

7.3.3 Physical implementation

An obvious area for future research is the physical implementation of both the recovery equipment and a controller developed using the presented methodology on a flying vehicle. The application of the methodology presented in this thesis for the UAV *Ariel* is unlikely to achieve flight status since development of the aircraft has stopped. However, it is feasible

that this methodology will be applied to produce a hardware controller. This would allow to demonstrate the robustness of the generated solution.

Physical implementation of the recovery equipment is no less interesting. Since this recovery method is new, a hardware testbed may be needed at the first stage to prove the declared capabilities of the method. At first, it may be implemented on a ground-based station to investigate possible issues with the capture itself. If it proves to be successful, a test deployment on a ship may be conducted, which allows to investigate the operational capabilities of the Cable Hook recovery method.

7.3.4 Other issues arising from this research

A serious issue regarding the local positioning system has been identified during extensive testing of the recovery controllers. As discussed above, the system employed in this work does not allow accurate guidance at close distances to the recovery boom, which causes sufficient deterioration of the recovery success rate in headwind conditions. Since an accurate local positioning system is essential for the majority of shipboard recovery methods, a further investigation into possible design concepts and the feasibility analysis of such a system is desirable. The system should provide accurate positioning data until the very moment of capture. In addition, it should be inexpensive and lightweight enough to be fit on a small UAV.

7.4 Concluding remarks

Autonomous recovery of a fixed-wing UAV on a moving ship with a limited deck space is a challenging task, with the problems arising from difficulty of capture and safe deceleration of the aircraft, ship motion and large atmospheric disturbances. The recovery method proposed in this research enables recovery of small to medium-size fixed-wing UAVs in severe weather conditions without imposing sufficient restrictions on the UAV design.

The methodology presented in this thesis can be used to develop a highly optimal controller capable of autonomous recovery of the UAV in adverse conditions. The Evolutionary Design method, which is the core of this methodology, potentially enables to produce controllers for a wide range of control problems. It allows to solve the problems automatically even when little or no knowledge about the controlled system available, which makes it a valuable tool for solving difficult control tasks.

Appendix A. Nomenclature

The notations used throughout this study employ highly systematic approach characteristic to scientific publications. Where possible, the name of the respective axis is used to indicate direction instead of common names that describe the nature of the variable. For example, X is used for drag force and ω_x for body roll rate. This approach is also used in several engineering schools.

The axes systems used in this study are described in Section 3.1.1.1. They generally follow the Forward-Up-Right convention.

The subscripts which refer to axes or variables are typed in *italics*. Other subscripts which refer to axes frames and names are typed in a straight font. Subscripts of different types may be combined in a single notation without using sub-subscripts, e.g. $\omega_{x_a} \equiv \omega_{xa}$ (roll rate in aerial axes).

- Subscripts for axes frames[1] (if applicable)
  - (no subscript) = body frame
  - g = ground (inertial) frame
  - a = aerodynamic (wind) frame
  - k = trajectory frame
  - w = wind-over-deck frame (ship airwake model)
- Superscripts
  - d = demand
  - (any variable) = derivative by the variable (in aerodynamic parameters)
    e.g. $m_z^{\omega_z}$ – coefficient of pitching moment derivative by pitch rate ⇔ coefficient of pitching moment due to pitch rate ⇔ pitch damping coefficient)
- Notation

α = angle of attack

β = sideslip angle

γ, ψ, θ = Euler angles (roll, yaw, pitch)

$\delta_a, \delta_r, \delta_e$ = ailerons, rudder, elevator deflection angle

δ_t = throttle setting

[1] Excluding the dynamic cable model.

Θ = trajectory angle (flight path slope)

Θ_{gs} = glideslope

$\boldsymbol{\theta}_s$ = ship Euler angles

λ = angle of sight

ρ = air density

ω = angular rate

ω_x, ω_y, ω_z = roll, yaw, pitch rates

ω_s = ship angular rate

A_{ij} = Direction Cosine Matrix for rotation from frame i to frame j, e.g.

A_{ab} = aerial to body frame DCM

a = acceleration

b_a = mean aerodynamic chord

C = coefficient

C_x, C_y, C_z, = drag, lift, sideforce coefficients

C_T, C_P, C_Z, C_G = propeller thrust, power, normal/sideforce, gyroscopic moment coefficients

d_i = distance from the UAV to the ith transmitter

F = force

g = gravity acceleration (9.80665 m/s^2)

H = altitude

h_s = cable sag

h_a = effective cable sag

h_T = elevation of the target point above the arresting wire

L_1 = length of the recovery boom

L_2 = height of the recovery boom supporting mast

l = wingspan

M = moment (torque)

M_x, M_y, M_z = rolling, yawing, pitching moment

m = mass

m_x, m_y, m_z = coefficients of rolling, yawing, pitching moment

n = load factor

P = engine power

q = dynamic pressure

S = wing area

s = Laplace variable

V = speed

V_a = airspeed

V_{CL} = closing rate

V_y = climb rate

V_{wd} = wind over deck magnitude

W = wind velocity

X, Y, Z = drag, lift, sideforce

Appendix B. Basic Ariel UAV characteristics

The basic design characteristics of the full scale Ariel aircraft are shown in Table B.1 and Table B.2. The data are taken from [153]. The aircraft aerodynamic data can be found in the same source as analytic functions and in [47] as graphical data.

Wing area, m	1.16
Wing span, m	3.02
Mean aerodynamic chord, m	0.387
Wing section	NACA 64_2-215 mod
Horizontal tailplane area, m^2	0.3
Available payload volume, m: Length	0.55
Width	0.30
Height	0.29
Maximum takeoff mass, kg	32.5
Available payload, kg	10 to 12

Table B.1 Ariel UAV main design parameters

Parameter	*Value*	*Longitudinal arm*	*Vertical arm*
Empty mass	20.45 kg	−0.29 m	−0.046 m
Empty I_x	6.4 kg·m^2	n/a	n/a
Empty I_y	13.3 kg·m^2	n/a	n/a
Empty I_z	8.0 kg·m^2	n/a	n/a
Empty I_{xy}	−0.85 kg·m^2	n/a	n/a
Empty I_{xz}	0 kg·m^2	n/a	n/a
Empty I_{yz}	0 kg·m^2	n/a	n/a
Payload	10 kg	0.4 m	−0.15 m
Fuel	2.05 kg	0.05 m	−0.15 m
Maximum takeoff mass	32.5 kg	−0.056 m	−0.085 m
c.g. limits	n/a	−0.15 to −0.01 m	n/a

Note: Arms are measured forward and up from the datum point at the wing leading edge

Table B.2 Ariel UAV inertial data

List of Publications

1. M. R. Crump, S. Khantsis, C. Bil and A. Bourmistrova. *Aspects of launch and recovery of fixed-wing unmanned air vehicles (UAV) from small surface vessels*, ed. D. J. Mee, in *10th* Australian *International Aerospace Congress* proceedings. Brisbane, Engineers Australia, 2003.
2. S. Khantsis, A. Bourmistrova. *UAV controller design using evolutionary algorithms*, ed. S. Zhang, R. Jarvis, in *AI 2005: Advances in Artificial Intelligence: 18th Australian Joint Conference on Artificial Intelligence* proceedings. Sydney, Australia, Springer-Verlag: Lecture Notes in Computer Science, 3809: p. 1025-1030, 5-9 Dec. 2005.
3. S. Khantsis, A. Bourmistrova. *Stochastic Design of a Non-Linear Controller for Autonomous UAV Recovery*, ed. A. Zerger, R. M. Argent, in *MODSIM 2005 International Congress on Modelling and Simulation* proceedings, Modelling and Simulation Society of Australia and New Zealand, Dec. 2005.

Bibliography

1. *Climatic Information to Determine Design and Test Requirements for Military Systems and Equipment. Military Standard MIL-STD-210C*. United States Department of Defence, 9 Jan. 1987.
2. *Design and Airworthiness Requirements for Service Aircraft. Design and Construction. Defence Standard 00-970 Issue 2 Part 1 Section 4*. United Kingdom Ministry of Defence, 1 Dec. 1999.
3. *Design and Airworthiness Requirements for Service Aircraft. Flight. Defence Standard 00-970 Issue 2 Part 1 Section 2 Leaflet 5: Atmospheric Disturbances*. United Kingdom Ministry of Defence, 1 Dec. 1999.
4. *Flying Qualities of Piloted Aircraft. Military Handbook MIL-HDBK-1797*. United States Department of Defence, 19 Dec. 1997.
5. *Flying Qualities of Piloted Aircraft. Military Specification MIL-F-8785C*. United States Department of Defence, Nov. 1980.
6. *Machovec nylon 3-strand twisted rope specifications.* Online publication available at http://www.machovec.com/rope/nylon_3strand.htm Accessed 9 Dec. 2005.
7. *UAV Forum body shop.* Online publication available at http://www.uavforum.com/library/bodyshop.htm Accessed 2005 30 Dec.
8. *Unmanned Aerial Vehicles: Outrider demonstrations will be inadequate to justify further production*. GAO/NSIAD-97-153. United States Government Accounting Office, Washington, Sep. 1997.
9. C. M. Ablow, S. Schechter. *Numerical simulation of undersea cable dynamics.* Ocean Engineering, 10(6): p. 443-457, 1983.
10. AeroVironment Inc. *Pointer FQM-151A Unmanned Aerial Vehicle (UAV) System.* Online publication available at http://www.aerovironment.com/area-aircraft/prod-serv/ptrdes.pdf Accessed 12 Jun. 2005.
11. S. Al-Hiddabi, N. McClamroch. *Output tracking for nonlinear non-minimum phase VTOL aircraft*, in *IEEE Conference on Decision and Control* proceedings, 1998.
12. T. Amishima, T. Bu and M. Sznaier. *Mixed L_1 / H_2 controllers for continuous time systems*, in *American Control Conference* proceedings, 1: p. 649-654, 1998.
13. T. Amishima, M. Sznaier. *Mixed H_2 / L_1 controllers for continuous-time MIMO systems*, in *American Control Conference* proceedings, 5: p. 3362-3366, 1999.
14. B. Anderson, J. Moore. *Linear Optimal Control.* Prentice-Hall, 1972.

15. D. V. Arnold, H.-G. Beyer. *Random dynamics optimum tracking with evolution strategies*, in *Parallel Problem Solving from Nature VII*, ed. J. J. Merelo Guervós, et al, Springer-Verlag: Berlin Heidelberg, p. 3-12, 2002.
16. I. L. Ashkenas, T. S. Durand. *Simulator and Analytical Studies of Fundamental Longitudinal Control Problems in Carrier Approach*, in *AIAA Simulation for Aerospace Flight Conference* proceedings, Aug. 1963.
17. K. J. Astrom, B. Wittenmark. *A survey of adaptive control applications*, in *IEEE Conference on Decision and Control* proceedings, 1: p. 649-654, 1995.
18. D. Atherton. *Nonlinear Control Engineering.* Van Nostrand Reinhold Company, 1975.
19. A. Babister. *Aircraft Stability and Control.* Oxford, 1961.
20. T. Bäck. *Evolutionary algorithms in theory and practice.* Oxford University Press, 1996.
21. T. Bäck, et al, eds. *Handbook on Evolutionary Computation*. Oxford University Press, 1997.
22. T. Bäck, U. Hammel and H.-P. Schwefel. *Evolutionary computation: Comments on the history and current state.* IEEE Transactions on Evolutionary Computation, 1(1): p. 3-17, Apr. 1997.
23. J. E. Baker. *Adaptive selection methods for genetic algorithms*, in *International Conference on Genetic Algorithms and Their Applications* proceedings: p. 100-111, 1985.
24. J. E. Baker. *Reducing bias and inefficiency in the selection algorithm*, in *Second International Conference on Genetic Algorithms* proceedings: p. 14-21, 1987.
25. N. A. Barricelli. *Symbiogenetic evolution process realised by artificial methods.* Methodos, 9(35-36): p. 143-182, 1957.
26. H.-G. Beyer. *Toward a theory of evolution strategies: self-adaptation.* Evolutionary Computation, 3(3): p. 311-347, 1996.
27. H.-G. Beyer, K. Deb. *On self-adaptive features in real-parameter evolutionary algorithms.* IEEE Transactions on Evolutionary Computation, 5(3): p. 250-270, 2001.
28. F. Blanchini, M. Sznaier. *Rational L_1 suboptimal compensators for continuous time systems.* IEEE Transactions on Automatic Control, 39(7): p. 1487-1492, 1994.
29. A. Bourmistrov. *Feedback control design for aerial towed systems*. PhD Thesis, RMIT University: Melbourne, 1997.
30. A. Bourmistrov, R. Hill and P. Riseborough. *Nonlinear control law for aerial towed target.* Journal of Guidance, Control and Dynamics, 18(6): p. 1232-1238, 1995.
31. A. Bourmistrova. *Knowledge based control system design for autonomous flight vehicle*. PhD thesis, RMIT University: Melbourne, 2001.

32. A. Bourmistrova, I. Herszberg and P. Riseborough. *Membership function optimisation with genetic algorithms*, in *EMAC 2000* proceedings. Melbourne, Australia, 2000.
33. A. Brindle. *Genetic algorithms for function optimisation*. Doctoral dissertation and Technical Report TR81-2, Department of Computer Science, University of Alberta: Edmonton, 1981.
34. G. Brown, R. Haggard and J. Fogleman. *Parafoils for shipboard recovery of UAVs*, PAPER 91-0835, AIAA, 1991.
35. R. G. Brown, P. Y. C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering.* 2nd ed. John Wiley & Sons, 1992.
36. A. J. Bryson, Y.-C. Ho. *Applied Optimal Control.* Hemisphere, 1975.
37. T. Bu, M. Sznaier and M. Holmes. *A linear matrix inequality approach to synthesizing low order L_1 controllers*, in *IEEE Conference on Decision and Control* proceedings, 1996.
38. B. Buckham, M. Nahon. *Dynamics simulation of low tension tethers*, in *IEEE Oceans* proceedings, 2: p. 757-766, 1999.
39. D. J. Bugajski, D. F. Enns. *Nonlinear control law with application to high angle-of-attack flight.* Journal of Guidance, Control and Dynamics, 15(3): p. 761-767, 1992.
40. G. S. Bushgens, ed. *Aerodynamics, stability and controllability of supersonic aircraft (in Russian)*. Nauka. Fizmatlit: Moscow, 1998.
41. A. J. Calise, R. T. Rysdyk. *Nonlinear adaptive flight control using neural networks.* IEEE Control Systems Magazine, 18(6): p. 14-25, 1998.
42. Carnegie Mellon University. *Genetic Algorithms FAQ.* Online publication available at http://www-2.cs.cmu.edu/Groups/AI/html/faqs/ai/genetic/top.html Accessed May 2005.
43. A. Chipperfield, P. Flemming. *Genetic algorithms in control systems engineering.* Journal of Computers and Control, 24(1), 1996.
44. D. A. Coley. *An Introduction to Genetic Algorithms for Scientists and Engineers.* University of Exeter, 1998.
45. N. L. Cramer. *A representation for the adaptive generation of simple sequential programs*, ed. J. J. Grefenstette, in *International Conference on Genetic Algorithms and their Applications* proceedings. Carnegie-Mellon University, Pittsburgh, PA: p. 183-187, 1985. Available online at http://www.sover.net/~nichael/nlc-publications/icga85/index.html
46. W. J. Crowther. *Perched landing and takeoff for fixed wing UAVs*, in *Symposium on Unmanned Vehicles (UV) for Aerial, Ground and Naval Military Operations* proceed-

ings. Ankara, Turkey, 9-13 Oct. 2000. Available online at http://www.eng.man.ac.uk/Aero/wjc/papers/Perched_Landing_Nato.pdf

47. M. R. Crump. *The dynamics and control of catapult launching Unmanned Air Vehicles from moving platforms*. PhD thesis, RMIT University: Melbourne, 2002.
48. M. A. Dahleh, J. B. Pearson. *L_1 optimal compensators for continuous-time systems.* IEEE Transactions on Automatic Control, 32(10): p. 889-895, 1987.
49. M. A. Dahleh, J. B. Pearson. *l^1-optimal feedback controllers for MIMO discrete-time systems.* IEEE Transactions on Automatic Control, 32(4): p. 314-322, 1987.
50. M. Dalsamo, O. Egeland. *H_∞ control of nonlinear passive systems by output feedback*, in *IEEE Conference on Decision and Control* proceedings, 1, 1995.
51. L. Davis, ed. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold: New York, 1991.
52. K. A. De Jong. *An analysis of the behaviour of a class of genetic adaptive systems*. Doctoral dissertation, University of Michigan, 1975.
53. K. A. De Jong. *Genetic algorithms are NOT function optimisers*, in *Foundations of Genetic Algorithms 2*, ed. L. D. Whitley, Morgan Kaufmann: San Mateo, CA, 1993.
54. K. A. De Jong, W. M. Spears. *On the state of evolutionary computation*, ed. S. Forrest, in *5th International Conference on Genetic Algorithms* proceedings. San Mateo, CA, Morgan Kaufmann: p. 618, 1993.
55. K. Deb, R. B. Agrawal. *Simulated binary crossover for continuous search space.* Complex Systems, 9: p. 115-148, 1995.
56. K. Deb, H.-G. Beyer. *Self-adaptation in real-parameter genetic algorithms with Simulated Binary Crossover*, ed. W. Banzhaf, et al., in *Genetic and Evolutionary Computation Conference* proceedings. San Francisco, CA, Morgan Kaufmann: p. 172-179, 1999.
57. K. Deb, D. Joshi and A. Anand. *Real-coded evolutionary algorithms with parent-centric recombination*, KanGAL Report 2001003, 2001. Available online at http://www.iitk.ac.in/kangal/dhiraj/tech_rep2001003.pdf
58. K. Deb, Kanpur Genetic Algorithm Lab. *Real coded genetic algorithm.* Online publication available at http://www.iitk.ac.in/kangal/real.htm Accessed May 2005.
59. R. DeCarlo, S. Zak and G. Mathews. *Variable structure control of nonlinear multivariable systems: a tutorial.* Proceedings of the IEEE, 76(3): p. 212-232, 1988.
60. J. C. Doyle, K. Glover, P. P. Khargonekar and B. A. Francis. *State-space solutions to standard H_2 and H_∞ control problems.* IEEE Transactions on Automatic Control, 34(8): p. 831-847, 1989.

61. J. C. Doyle, G. Stein. *Multivariable feedback design: concepts for a classical/modern synthesis.* IEEE Transactions on Automatic Control, AC-26(1): p. 4-16, Feb. 1981.
62. E. Duflos, P. Penel and P. Vanheeghe. *3D guidance law modeling.* IEEE Transactions on Aerospace and Electronic Systems, 35(1): p. 72-83, 1999.
63. J. C. Ervin. *Fuzzy logic control of a model airplane*, in *IEEE International Conference on Systems, Man and Cybernetics* proceedings, 3: p. 2320-2325, 1998.
64. L. J. Eshelman, J. D. Schaffer. *Real-coded genetic algorithms and interval schemata*, in *Foundations of Genetic Algorithms 2*, ed. L. D. Whitley, Morgan Kaufmann: San Mateo, CA, 1993.
65. A. Estabrook, R. MacDougal and R. Ludwig. *Unmanned Air Vehicle. Impact on CVX design*, Technical Report TD 3042, SPAWAR System Center: San Diego, Sept. 1998. Available online at http://www.spawar.navy.mil/sti/publications/pubs/td/3042/td3042.pdf
66. B. Etkin, L. Reid. *Dynamics of Flight - Stability and Control.* John Wiley and Sons, 1996.
67. M. A. Fernandez-Montesinos, G. Schram, R. Vingerhoeds, H. Verbruggen and J. Mulder. *Windshear recovery using fuzzy logic guidance and control.* Journal of Guidance, Control and Dynamics, 22(1): p. 178-180, 1999.
68. P. Flajolet, A. Oldyzko. *The average height of binary trees and other simple trees.* Journal of Computers and System Sciences, 25: p. 171-213, 1982.
69. T. C. Fogarty. *Varying the probability of mutation in the genetic algorithm*, ed. J. D. Schaffer, in *Third International Conference on Genetic Algorithms* proceedings: p. 104–109, 1989.
70. D. B. Fogel. *Autonomous automata.* Industrial Research, 4: p. 14-19, 1962.
71. D. B. Fogel, J. W. Atmar. *Comparing genetic operators with Gaussian mutations in simulated evolutionary processes using linear systems.* Biological Cybernetics, 63: p. 111, 1990.
72. D. B. Fogel, J. W. Atmar, eds. *Proceedings of the First Annual Conference on Evolutionary Programming*. Evolutionary Programming Society: La Jolla, CA, 1992.
73. D. B. Fogel, A. J. Owens and M. J. Walsh. *Artificial intelligence through simulated evolution.* New York: Wiley, 1966.
74. S. Forrest, M. Mitchell. *What makes a problem hard for a genetic algorithm? Some anomalous results and their explanation.* Machine Learning, 13: p. 285-319, 1993.

75. A. S. Fraser. *Simulation of genetic systems by automatic digital computers. 5-linkage, dominance and epistasis.* Biometrical Genetics, ed. O. Kempthorne. New York: Macmillan, 1960.

76. Freewing Aerial Robotics Corp. *Freewing Aerial Robotics official website.* Online publication available at www.freewing.com Accessed 15 Jan. 2006.

77. Freewing Aerial Robotics Corp. *Freewing Tilt-Body™ "Scorpion" applicability to the 1997 U.S. Joint & Maritime Requirements.* Online publication available at http://www.freewing.com/1997rfi.pdf Accessed 12 Nov. 2005.

78. M. Gardner. *The fantastic combinations of John Conway's new solitaire game "Life".* Scientific American, 223 (April): p. 120-123, 1970.

79. O. H. Gerlach, G. van de Moesdijk and J. C. van der Vaart. *Progress in the Mathematical Modeling of Flight in Turbulence*, in *AGARD Conference Proceedings No. 140 on Flight in Turbulence* proceedings: p. 1-38, 1973.

80. D. E. Goldberg. *Genetic Algorithms in search, optimisation and machine learning.* Addison-Wesley, 1989.

81. D. E. Goldberg. *Optimal initial population size for binary-coded genetic algorithms*, Technical Report TCGA Report No. 85001, University of Alabama: Tuscaloosa, AL, 1985.

82. D. E. Goldberg, K. Deb. *A comparative analysis of selection schemes used in genetic algorithms.* ed. G. J. E. Rawlins. Morgan Kaufmann, 1991.

83. D. E. Goldberg, K. Deb and D. Thierens. *Toward a better understanding of mixing in genetic algorithms.* Journal of the Society of Instrument and Control Engineers, 32(1): p. 10-16, 1993.

84. D. Graham, D. McRuler. *Analysis of Nonlinear Control Systems.* Dover, 1971.

85. R. D. Greenhalgh, e. al. *Aircraft recovery methods*, Patent No. 3,980,259, US, Sep. 1976.

86. J. J. Grefenstette. *Evolvability in dynamic fitness landscapes: A genetic algorithm approach*, in *Congress of Evolutionary Computation* proceedings, IEEE Press, 3: p. 2031–2038, 1999.

87. M. S. Grewal, L. R. Weill and A. P. Andrews. *Global Positioning Systems, Inertial Navigation, and Integration.* Wiley-Interscience, 2001.

88. W. M. Haddad, V. Kapila. *Mixed H_2 / L_1 fixed-architecture controller design for multi-input/single-output systems.* Journal of the Franklin Institute, 336(3): p. 435-448, 1999.

89. N. Hansen. *The CMA Evolution Strategy: A Tutorial.* Online publication available at http://www.bionik.tu-berlin.de/user/niko Accessed May 2005.

90. N. Hansen, A. Ostermeier. *Completely derandomized self-adaptation in evolution strategies.* Evolutionary Computation, 9(2): p. 159-195, 2001.

91. J. Healey. *Establishing a database for flight in the wakes of structures.* Journal of Aircraft, 29(4): p. 559-564, 1992.

92. F. Herrera, M. Lozano and J. L. Verdegay. *Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis.* Artificial Intelligence Review, 12(4): p. 265-319, 1998.

93. J. F. Hicklin. *Application of the genetic algorithm to automatic program generation.* M.S. Thesis, Department of Computer Science, University of Idaho, 1986.

94. T. Higuchi, S. Tsutsui and M. Yamamura. *Theoretical analysis of simplex crossover for real-coded genetic algorithms*, in *Parallel Problem Solving from Nature IV*, p. 365-374, 2000.

95. Y.-C. Ho, A. E. Bryson and S. Baron. *Differential games and optimal pursuit-evasion strategies.* IEEE Transactions on Automatic Control, 10(4): p. 385-389, 1965.

96. F. M. Hoblit. *Gust Loads on Aircraft: Concepts and Applications.* AIAA Education Series, 1988.

97. J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

98. R. B. Hollstien. *Artificial genetic adaptation in computer control systems*. Doctoral dissertation, University of Michigan, 1971.

99. K. Hope. *Angular Motions of an Australian FFG in Rough Seas SS6-7 - Statistics and Spectra.* Australian Defence Force Academy, 1996.

100. H. Y. Hu, D. P. Jin. *Non-linear dynamics of a suspended travelling cable subject to transverse fluid excitation.* Journal of Sound and Vibration, 239(3): p. 515-529, 2001.

101. R. Hyde. *The Aplication of Robust Control to VSTOL Aircraft*. PhD thesis, Cambridge, 1991.

102. R. Hyde. *Flight control system design for VSTOL aircraft using switched H_∞ controllers.* Transactions of the Institute of Measurement and Control, 13(3): p. 140-143, 1991.

103. R. Hyde, K. Glover. *Taking H_∞ controller into flight*, in *IEEE Conference on Decision and Control* proceedings, 2, 1993.

104. R. Hyde, K. Glover and G. Shanks. *VSTOL first flight of an H_∞ control law.* Computing & Control Engineering Journal, 6(1): p. 11-16, 1995.

105. A. Isidori. *Nonlinear Control Systems.* Berlin: Springer-Verlag, 1989.

106. Javad Navigation Systems. *A GPS Tutorial: Basics of High-Precision Global Positioning Systems.* Online publication available at http://www.javadgps.com/jns/gpstutorial/index.html Accessed 25 Apr. 2006.

107. A. H. Jazwinski. *Stochastic Processes and Filtering Theory.* Academic Press, 1970.

108. S. J. Julier, J. K. Uhlmann. *A new extension of the Kalman filter to nonlinear systems*, in *SPIE AeroSense Symposium* proceedings. Orlando, FL, USA, 3068: p. 182-193, 21-24 Apr. 1997. Available online at http://www.cs.unc.edu/~welch/kalman/media/pdf/Julier1997_SPIE_KF.pdf

109. N. Kaise, Y. Fujimoto. *Applying the evolutionary neural networks with genetic algorithms to control a rolling inverted pendulum*, ed. McKay, et al, in *2nd Asia-Pacific Conference on Simulated Evolution and Learning (SEAL-1998)* proceedings, 1585 of LNAI: p. 223-230, 1999.

110. R. E. Kalman. *A new approach to linear filtering and prediction problems.* Transaction of the ASME—Journal of Basic Engineering: p. 35-45, Mar. 1960.

111. I. Kaminer, P. P. Khargonekar and G. Robel. *Design of a localizer capture and track modes for a lateral autopilot using H_∞ synthesis.* IEEE Control Systems Magazine, 10(4): p. 13-21, 1990.

112. M. Khammash, L. Zou. *Robust aircraft pitch-axis control under weight and center of gravity uncertainty*, in *IEEE Conference on Design and Control* proceedings, 1, 1999.

113. S. Khantsis. *Development of a semi-automatic aircraft control system for carrier landing task and the software for experimental research (in Russian).* Final Thesis, Department of Flight Dynamics and Control, MAI University: Moscow, 2001.

114. S. Khantsis, A. Bourmistrova. *UAV controller design using evolutionary algorithms*, ed. S. Zhang, R. Jarvis, in *AI 2005: Advances in Artificial Intelligence: 18th Australian Joint Conference on Artificial Intelligence* proceedings. Sydney, Australia, Springer-Verlag: Lecture Notes in Computer Science, 3809: p. 1025-1030, 5-9 Dec. 2005.

115. B. S. Kim, A. J. Calise. *Nonlinear flight control using neural networks.* Journal of Guidance, Control and Dynamics, 20(1): p. 26-33, 1997.

116. C. G. Koh, Y. Rong. *Dynamic analysis of large displacement cable motion with experimental verification.* Journal of Sound and Vibration, 272(1-2): p. 187-206, Apr. 2004.

117. J. R. Koza. *Genetic Programming II: Automatic discovery of reusable programs.* Cambridge, Massachusetts: MIT Press, 1994.

118. J. R. Koza. *Genetic Programming: On the programming of computers by means of natural selection.* Cambridge, Massachusetts: MIT Press, 1992 (6th ed. 1998).

119. J. R. Koza. *Genetic Programming: On the programming of computers by means of natural selection.* Cambridge, Massachusetts: MIT Press, 1992 (6th ed. 1998).

120. J. R. Koza, F. H. Bennett III, D. Andre and M. A. Keane. *Four problems for which a computer program evolved by genetic programming is competitive with human performance*, in *IEEE International Conference on Evolutionary Computation* proceedings. Nagoya, Japan, 1996.

121. J. R. Koza, F. H. Bennett III, D. Andre and M. A. Keane. *Genetic Programming III: Darwinian Invention and Problem Solving.* Kluwer Academic Publishers, 1999.

122. J. R. Koza, M. A. Keane, F. H. Bennett III, J. Yu, W. Mydlowec and O. Stiffelman. *Automatic creation of both the topology and parameters for a robust controller by means of genetic programming*, in *IEEE International Symposium on Intelligent Control / Intelligent Systems and Semiotics* proceedings: p. 344-352, 1999.

123. J. R. Koza, M. A. Keane, M. J. Streeter, W. Mydlowec, J. Yu and G. Lanza. *Genetic Programming IV: Routine Human-Competitive Machine Intelligence.* Kluwer Academic Publishers, 2003.

124. J. R. Koza, M. A. Keane, J. Yu, F. H. Bennett III and W. Mydlowec. *Automatic creation of human-competitive programs and controllers by means of genetic programming.* Genetic Programming and Evolvable Machines, 1(1): p. 121-164, 2000.

125. J. B. Kuipers. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality.* Princeton University Press, 1999.

126. H. J. Kushner. *Dynamical equations for optimum non-linear filtering.* Journal of Differential Equations, 3: p. 179-190, 1967.

127. H. Kwakernaak. *Robust control and H_∞ optimization - tutorial paper.* Automatica, 29(2): p. 255-273, 1993.

128. H. Kwakernaak, R. Sivan. *Linear Optimal Control Systems.* John Wiley & Sons, 1972.

129. S. H. Lane, R. F. Stengel. *Flight control design using nonlinear inverse dynamics.* Automatica, 24: p. 471-484, 1988.

130. W. B. Langdon. *Data structures and genetic programming: Genetic Programming + Data Structures = Automatic Programming!*, in *Genetic Programming*. Vol. 1, Kluwer Academic Publishers, 1998.

131. W. B. Langdon, et al. *The evolution of size and shape*, in *Advances in genetic programming 3*, ed. L. Spector, W. B. Langdon and et al, MIT Press, p. 163-190, 1999.

132. W. B. Langdon, R. Poli. *Foundations of genetic programming.* Berlin: Springer-Verlag, 2002.

133. S. Lee, C. Ha and B. S. Kim. *Adaptive nonlinear control system design for helicopter robust command augmentation.* Aerospace Science and Technology, 9(3): p. 241-251, 2005.

134. J. S. Lingard. *The performance and design of ram-air gliding parachutes*, Technical Report TR 81103, RAE, 1981.

135. M.-S. Liu. *Three-dimensional numerical simulation of turbulent flow around two high-rise buildings in proximity.* Wind & Structures, An International Journal, 1(3): p. 271-284, 1998.

136. M.-S. Liu, N. L. Long. *High order accurate ship airwake predictions for the helicopter/ship interface problem.* Annual Forum Proceedings - American Helicopter Society, 1, 1998.

137. A. S. Locke. *Guidance.* Princeton, NJ: Van Nostrand, 1956.

138. C. L. Logan. *A comparison between H_∞ / μ-synthesis control and sliding-mode control for robust control of a small autonomous underwater vehicle*, in *IEEE Symposium on Autonomous Underwater Vehicle Technology* proceedings: p. 399-416, 1994.

139. J. Maciejowski. *Multivariable Feedback Design.* Addison Wesley, 1989.

140. S. W. Mahfoud, G. Mani. *Financial forecasting using genetic algorithms.* Applied Artificial Intelligence, 10: p. 543-565, 1996.

141. MathWorks Inc. *MATLAB Aerospace Blockset: 6DoF (Quaternion).* Online publication available at http://www.mathworks.com/access/helpdesk/help/toolbox/aeroblks/6dofquaternion.html Accessed 7 Nov. 2005.

142. MathWorks Inc. *MATLAB Aerospace Blockset: von Karman Wind Turbulence Model (Continuous).* Online publication available at http://www.mathworks.com/access/helpdesk/help/toolbox/aeroblks/vonkarmanwindturbulencemodelcontinuous.shtml Accessed 25 Sep. 2004.

143. MathWorks Inc. *Robust Control Toolbox User's Guide.* 2001. Available online at http://www.mathworks.com/access/helpdesk/help/pdf_doc/robust/robust.pdf

144. D. Mavris, J. Prasad and D. Scharge. *A simulation methodology for modeling ship airwake turbulence.* 25th Annual international symposium of the society of flight test engineers, Aug. 1994.

145. R. E. McFarland, K. Duisenberg. *Simulation of rotor blade element turbulence*, Technical Memorandum 108862, Acc. No. NASA TM-108862, NASA, Jan. 1995.

146. M. W. McKee. *VTOL UAVs Come of Age: US Navy Begins Development of VTUAV.* Online publication available at http://www.vtol.org/uavpaper/NavyUAV.htm Accessed 7 Jan. 2006.

147. D. McLean, H. Matsuda. *Helicopter station-keeping: comparing LQR, fuzzy-logic and neural-net controllers.* Engineering Applications of Artificial Intelligence, 11: p. 411-418, 1998.

148. D. McRuler, I. Ashkenas and D. Graham. *Aircraft Dynamics and Automatic Control.* Princeton University Press, 1973.

149. B. L. Miller, D. E. Goldberg. *Genetic algorithms, tournament selection and the effects of noise*, IlliGAL Report No. 95006, University of Illinois: Urbana, 1995. Available online at http://www.assuredigit.com/tech_doc/more/Goldberg_GA_tournament_seletion_and_the_effects_of_noise.pdf

150. P. Montegomery, B. W. Parkinson. *Carrier differential GPS for takeoff to landing of an autonomous aircraft*, in *National Technical Meeting* proceedings, Institute of Navigation, 1996.

151. D. Morton, D. F. Enns and B. Zhang. *Stability of dynamic inversion control laws applied to nonlinear aircraft pitch axis models.* International Journal of Control, 63(1): p. 1-25, 1996.

152. National Air and Space Museum. Smithsonian Institution. *Pioneer RQ-2A*. Online publication available at http://www.nasm.si.edu/research/aero/aircraft/pioneer.htm Accessed 4 Jan. 2006.

153. D. M. Newman, K. C. Wong. *Six Degree of Freedom Flight Dynamic and Performance Simulation of a Remotely-Piloted Vehicle.* The University of Sydney, 1993.

154. A. E. Nix, M. D. Vose. *Modelling genetic algorithms with Markov chains*, in *Annals of Mathematics and Artificial Intelligence*. Vol. 5, J. C. Baltzer A. G. Scientific Publishing Company: Switzerland, p. 79-88, 1992.

155. P. Nordin, W. Banzhaf. *Complexity compression and evolution*, ed. L. J. Eshelman, in *6th International Conference on Genetic Algorithms* proceedings. Pittsburgh, PA, Morgan Kaufmann: p. 310-317, 15-19 July 1995.

156. M. Norgaard. *The KALMTOOL toolbox.* Online publication available at http://www.iau.dtu.dk/research/control/kalmtool.html.

157. M. Norgaard, N. K. Poulsen and O. Ravn. *New developments in state estimation for nonlinear systems.* Automatica, 36(11): p. 1627-1638, 2000.

158. U.-M. O'Reilly, F. Oppacher. *The troubling aspects of a building block hypothesis for genetic programming*, in *Foundations of Genetic Algorithms 3*, Morgan Kaufmann: San Francisco, CA, p. 73-88, 1995.

159. M. Olhofer, Y. Jin and B. Sendhoff. *Adaptive encoding for aerodynamic shape optimization using Evolution Strategies*, in *Congress on Evolutionary Computation* proceedings. Seoul, South Korea, 1: p. 576-583, 2001.

160. P. J. Oliviera, B. A. Younis. *On the prediction of turbulent flows around full-scale buildings.* Journal of Wind Engineering and Industrial Aerodynamics, 86, 2000.

161. C. Onnen, R. Babuska, U. Kaymak, J. M. Sousa, H. B. Verbruggen and R. Isermann. *Genetic algorithms for optimization in predictive control.* Control Engineering Practice, 5(10): p. 1363-1372, 1997.

162. I. Ono, S. Kobayashi. *A real-coded genetic algorithm for function optimisation using unimodal normal distribution crossover*, ed. T. Bäck, in *7th International Conference on Genetic Algorithms* proceedings. San Mateo, CA, Morgan Kauffman: p. 246-253, 1997.

163. A. Ostermeier, A. Gawelczyk and N. Hansen. *A derandomized approach to self-adaptation of evolution strategies.* Evolutionary Computation, 2(4): p. 369-380, 1994.

164. A. I. Oyman, H.-G. Beyer and H.-P. Schwefel. *Where elitists start limping: evolution strategies at ridge functions*, in *Parallel Problem Solving from Nature V*, Springer-Verlag: Amsterdam, The Netherlands, p. 34-43, 1998.

165. B. W. Parkinson, J. J. Spilker, eds. *Global Positioning System: Theory & Applications*. Vol. 1: Progress in Astronautics and Aeronautics, AIAA, 1996.

166. A. A. Pashilkar, N. Sundararajan and P. Saratchandran. *A fault-tolerant neural aided controller for aircraft auto-landing.* Aerospace Science and Technology, 10(1): p. 49-61, 2006.

167. T. Perez, M. Blanke. *Simulation of ship motion in seaway*, Technical Report EE02037, The University of Newcastle, Australia, 2002. Available online at http://www.iau.dtu.dk/secretary/pdf/wavesII.pdf

168. T. Perkins. *Stack-based genetic programming*, in *IEEE World Congress on Computational Intelligence* proceedings. Orlando, Florida, IEEE Press, 1: p. 148-153, 27-29 June 1994.

169. H. Piet-Lahanier, S. Le Menec. *Comparisons of guidance law performances*, in *12th International Symposium on Dynamic Games and Applications* proceedings. Sophia Antipolis, French Riviera, 2006.

170. W. J. G. Pinsker. *Glide-path stability of an aircraft under speed constraint*, Reports and Memoranda No. 3705, United Kingdom Ministry of Defence: London, 1971. Available online at http://naca.central.cranfield.ac.uk/reports/arc/rm/3705.pdf

171. Pioneer UAV Inc. *Pioneer UAV official website.* Online publication available at www.puav.com Accessed 4 Jan. 2006.

172. I. Postlethwaite, D. Bates. *Robust integrated flight and propulsion controller for the Harrier aircraft.* Journal of Guidance, Control and Dynamics, 22(2): p. 286-290, 1999.

173. K. Rawson, E. Tupper. *Basic Ship Theory.* Vol. 1 and 2. Longman, 1983.

174. E. Regis. *Spratt, Schmittle and Freewing*, in *Air & Space/Smithsonian Magazine*, Dec. 1994.

175. F. Riegels. *Aerofoil Sections.* Butterworth, 1961.

176. J. P. Rosca. *Analysis of complexity drift in genetic programming*, ed. J. R. Koza, et al, in *Second Annual Conference on Genetic Programming* proceedings. Stanford University, CA, Morgan Kaufmann: p. 286-294, 13-16 July 1997.

177. R. E. Rosenthal, W. J. Walsh. *Optimizing flight operations for an aircraft carrier in transit.* Operations Research, 44(2): p. 305-312, 1996.

178. J. Roskam. *Airplane Flight Dynamics and Automatic Flight Controls.* Roskam Aviation and Engineering Company, 1993.

179. Royal Australian Navy. *ANZAC Class Frigates.* Online publication available at http://www.navy.gov.au/fleet/anzac_frigate.html Accessed 11 Apr. 2006.

180. W. J. Rugh. *Analytical framework for gain scheduling.* IEEE Control Systems Magazine, 11(1): p. 79-84, 1991.

181. W. J. Rugh, J. S. Shamma. *Research on gain scheduling.* Automatica, 36(10): p. 1401-1425, 2000.

182. T. P. Runarsson, X. Yao. *Continuous Selection and Self-Adaptive Evolution Strategies*, in *Congress on Evolutionary Computation* proceedings, 1: p. 279-284, 12-17 May 2002.

183. I. F. Sbalzarini, S. Mueller and P. Koumoutsakos. *Multiobjective optimization using evolutionary algorithms*, in *Summer Program 2000* proceedings, Center for Turbulence Research, NASA Ames/Stanford University, 2000.

184. J. D. Schaffer, L. J. Eshelman. *On crossover as an evolutionarily viable strategy*, ed. R. Belew, L. Booker, in *4th International Conference on Genetic Algorithms* proceedings. San Mateo, CA, Morgan Kaufmann: p. 61-68, 1991.

185. C. Schumacher, P. P. Khargonekar. *Missile autopilot designs using H_∞ control with gain scheduling and dynamic inversion.* Journal of Guidance, Control and Dynamics, 21(2): p. 234-243, 1998.

186. H.-P. Schwefel. *Numerical optimisation of computer models.* New York: Wiley, 1981.

187. B. Sendhoff, M. Kreuz. *Variable encoding of modular neural networks for time series prediction*, ed. V. W. Porto, in *Congress on Evolutionary Computation* proceedings, IEEE Press: p. 259-266, 1999.

188. B. Sendhoff, M. Kreuz and W. von Seelen. *A condition for the genotype-phenotype mapping: Casualty*, ed. T. Bäck, in *7th International Conference on Genetic Algorithms* proceedings, Morgan Kaufmann: p. 73-80, 1997.

189. J. S. Shamma, M. Athans. *Gain scheduling: potential hazards and possible remedies.* IEEE Control Systems Magazine, 12(3): p. 101-107, 1992.

190. S. Shue, R. Agarwal. *Design of automatic landing system using mixed H_2 / H_∞ control.* Journal of Guidance, Control and Dynamics, 22(1): p. 103-114, 1999.

191. U. S. Shukla, P. R. Mahapatra. *The proportional navigation dilemma - pure or true?* IEEE Transactions on Aerospace and Electronic Systems, 26(2): p. 382-392, 1990.

192. M. M. Sidar, B. F. Doolin. *On the Feasibility of Real-Time Prediction of Aircraft Carrier Motion at Sea.* IEEE Transactions on Automatic Control, 28(3): p. 350-356, 1983.

193. Sierra Nevada Corporation. *UAV Common Automatic Recovery System (UCARS).* Online publication available at http://www.sncorp.com/uav1.html Accessed 25 Apr. 2006.

194. S. Singh, M. Pachter, P. Chandler, S. Banda, S. Rasmussen and C. Schumacher. *Input-output invertibility and sliding mode control for close formation flying of multiple UAVs.* International Journal of Robust and Nonlinear Control, 10, 2000.

195. S. Singh, M. Steinberg and R. DiGirolamo. *Variable structure robust flight control system for the F-14.* IEEE Transactions on Aerospace and Electronic Systems, 33(1): p. 77-84, 1997.

196. S. Singh, M. Steinberg and A. Page. *Variable structure and nonlinear adaptive flight path control*, in *American Control Conference* proceedings, 3: p. 1785-1790, 2002.

197. G. M. Siouris, P. Leros. *Minimum-time intercept guidance for tactical missiles.* Control Theory and Advanced Technology, 4(2): p. 251-263, 1988.

198. S. Skogestad, I. Postlethwaite. *Multivariable Feedback Control.* John Wiley and Sons, 1997.

199. S. F. Smith. *A learning system based on genetic adaptive algorithms*. PhD dissertation, University of Pittsburgh, 1980.

200. S. A. Snell, W. L. Garrard and D. F. Enns. *Nonlinear inversion flight control for a supermaneuverable aircraft*, in *AIAA Guidance, Navigation and Control Conference* proceedings. Portland: p. 808-825, 1990.

201. H. W. Sorenson, ed. *Kalman filtering: theory and application*. IEEE Press, 1985.

202. H. W. Sorenson, A. R. Stubberud. *Non-linear filtering by approximation of the a posteriori density.* International Journal of Control, 8(1): p. 33-51, 1968.

203. T. Soule, J. Foster. *Code size and depth flows in genetic programming*, ed. J. R. Koza, et al, in *Second Annual Conference on Genetic Programming* proceedings. Stanford University, CA, Morgan Kaufmann: p. 313-320, 13-16 July 1997.

204. W. M. Spears. *Adapting crossover in evolutionary algorithms*, ed. J. R. McDonnell, R. G. Reynolds and D. B. Fogel, in *4th Annual Conference on Evolutionary Computing* proceedings, The MIT Press: p. 367-384, 1995.

205. W. M. Spears. *Crossover or Mutation?*, in *Foundations of Genetic Algorithms 2*, ed. L. D. Whitley, Morgan Kaufmann: San Mateo, CA, 1992.

206. W. M. Spears. *On the virtues of uniform crossover*, in *4th International Conference on Genetic Algorithms* proceedings. La Jolla, California, July 1991.

207. W. M. Spears, V. Anand. *A study of crossover operators in genetic programming*, ed. Z. W. Ras, M. Zemankova, in *International Symposium on Methodologies for Intelligent Systems* proceedings, Berlin: Springer-Verlag, 542: p. 409-418, 1991.

208. J. T. Spooner, M. Maggiore, R. Ordonez and K. M. Passino. *Stable Adaptive Control and Estimation for Nonlinear Systems: Neural and Fuzzy Approximator Techniques.* John Wiley and Sons, 2002.

209. M. Steinberg. *Development and simulation of an F/A-18 fuzzy logic automatic carrier landing system*, in *IEEE International Conference on Fuzzy Systems* proceedings, 2: p. 797-802, 1993.

210. M. Strass. *Navy begins first deployment of Sea-ALL UAV for ship protection.* Defence Daily, 212(25), 5 Nov. 2001.

211. T. Tai. *Effect of ship motion on DD-963 ship airwake simulated by multizone Navier-Stokes solution.* 21st Symposium on Naval Hydrodynamics: p. 1007-1017, 1997.

212. The Insitu Group Inc. *The Insitu Group official website.* Online publication available at www.insitugroup.com Accessed 12 Jan. 2006.

213. P. Trivailo, C. Blanksby and et al. *Development of advanced cable dynamic model based on Kane's dynamic equations*, Technical Report TR ERC-DCTSG 2002-01, RMIT University: Melbourne, 2002.

214. R. K. Ursem. *Models for evolutionary algorithms and their applications in system identification and control optimization*. PhD dissertation, Department of Computer Science, University of Aarhus, Denmark, 2003.

215. V. I. Utkin. *Sliding Modes and Their Applications to Variable Structure Systems.* Moscow: Mir, 1978.

216. P. J. M. van Laarhoven, E. H. L. Aarts. *Simulated annealing: theory and applications.* Dordrecht, The Netherlands: D. Reidel, 1987.

217. V. K. Vassilev, J. F. Miller. *The advantages of landscape neutrality in digital circuit evolution*, ed. J. F. Miller, et al, in *3rd International Conference on Evolvable Systems: From Biology to Hardware* proceedings, Springer-Verlag, Lecture Notes in Computer Science 1801: p. 252-263, 2000.

218. H.-M. Voigt, H. Muehlenbein and D. Cvetkovic. *Fuzzy recombination for the Breeder Genetic Algorithm*, ed. L. J. Eshelman, in *6th International Conference on Genetic Algorithms* proceedings. Pittsburgh, PA, Morgan Kaufmann: p. 104-111, 15-19 July 1995.

219. P. Voulgaris, L. Valavani. *High performance linear quadratic and H_∞ designs for a supermaneuverable aircraft.* Journal of Guidance, Control and Dynamics, 14(1): p. 157-165, 1991.

220. Q. Wang, A. M. S. Zalzala. *Genetic control of near time-optimal motion for an industrial robot arm*, in *IEEE International Conference on Robotics and Automation* proceedings, 3: p. 2592-2597, 1996.

221. R. A. Watson, J. B. Pollack. *Combination and recombination in Genetic Algorithms*, Technical report CS-00-209, Dept. Computer Science, Brandeis University, 2000.

222. R. A. Watson, J. B. Pollack. *Symbiotic combination as an alternative to sexual recombination in Genetic Algorithms*, in *Parallel Problem Solving from Nature VI*, ed. M. Schoenauer, et al., Springer-Verlag, p. 425-434, 2000.

223. K. Wise, S. J. *Nonlinear H_∞ optimal control for agile missiles.* Journal of Guidance, Control and Dynamics, 19(1): p. 157-165, 1996.

224. D. H. Wolpert, W. G. Macready. *No free-lunch theorems for optimization.* IEEE Transactions on Evolutionary Computation, 1(1): p. 67-82, 1997.

225. T. A. Wyllie. *The Observer UAV system*, in *NATO RTA Symposium on Unmanned Vehicles* proceedings. Ankara, Turkey, 2000.

226. T. A. Wyllie. *Parachute Recovery for UAV Systems.* Aircraft Engineering and Aerospace Technology, 73(6): p. 542-551, 2001. Available online at

http://www.emeraldinsight.com/Insight/ViewContentServlet?Filename=Published/EmeraldFullTextArticle/Articles/1270730602.html

227. C. Yang, H. Chen. *Nonlinear H_∞ robust guidance law for homing missiles.* Journal of Guidance, Control and Dynamics, 21(6): p. 882-890, 1998.

228. C. Yang, C. Kun. *Nonlinear H_∞ flight control of general six-degree-of-freedom motions.* Journal of Guidance, Control and Dynamics, 23(2): p. 278-288, 2000.

229. I. Yumori. *Real Time Prediction of Ship Response to Ocean Waves Using Time Series Analysis.* Oceans, 13: p. 1082 -1089, 1981.

230. S. Zein-Sabatto, Y. Zheng. *Intelligent flight controllers for helicopter control*, in *International Conference on Neural Networks* proceedings, 2: p. 617-621, 1997.

231. K. Zhou, J. C. Doyle. *Essentials of Robust Control.* Prentice Hall, 1998.

232. E. Zitzler, L. Thiele. *Multiobjective optimization using evolutionary algorithms - A comparative case study*, in *Parallel Problem Solving from Nature V*, Springer-Verlag: Amsterdam, The Netherlands, p. 292-301, 1998.