9-15-2011

Stochastic Real-time Optimal Control: A Pseudospectral Approach for Bearing-Only Trajectory Optimization

Steven M. Ross

STOCHASTIC REAL-TIME OPTIMAL CONTROL: A PSEUDOSPECTRAL APPROACH FOR BEARING-ONLY TRAJECTORY OPTIMIZATION

DISSERTATION

Steven M. Ross, Lieutenant Colonel, USAF

AFIT/DS/ENY/11-24

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/DS/ENY/11-24

STOCHASTIC REAL-TIME OPTIMAL CONTROL: A PSEUDOSPECTRAL APPROACH FOR BEARING-ONLY TRAJECTORY OPTIMIZATION

DISSERTATION

Presented to the Faculty

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

Steven M. Ross, B.S.A.E., M.S.A.E.

Lieutenant Colonel, USAF

September 2011

AFIT/DS/ENY/11-24

STOCHASTIC REAL-TIME OPTIMAL CONTROL: A PSEUDOSPECTRAL APPROACH FOR BEARING-ONLY TRAJECTORY OPTIMIZATION

Steven M. Ross, B.S.A.E., M.S.A.E.
Lieutenant Colonel, USAF

Approved:

[illegible] — 11 Aug 11

Richard G. Cobb, PhD
Chairman — Date

[illegible] — 11 Aug 11

William P. Baker, PhD
Member — Date

Frederick G. Harmon — 11 Aug 11

LtCol Frederick G. Harmon, PhD
Member — Date

Accepted:

MU Thomas — 31 Aug 2011

M. U. Thomas, PhD
Dean, Graduate School of
Engineering and Management — Date

AFIT/DS/ENY/11-24

Abstract

A method is presented to couple and solve the optimal control and the optimal estimation problems simultaneously, allowing systems with bearing-only sensors to maneuver to obtain observability for relative navigation without unnecessarily detracting from a primary mission. A fundamentally new approach to trajectory optimization and the dual control problem is presented, constraining polynomial approximations of the Fisher Information Matrix to provide an information gradient and allow prescription of the level of future estimation certainty required for mission accomplishment.

Disturbances, modeling deficiencies, and corrupted measurements are addressed recursively using Radau pseudospectral collocation methods and sequential quadratic programming for the optimal path and an Unscented Kalman Filter for the target position estimate. The underlying real-time optimal control (RTOC) algorithm is developed, specifically addressing limitations of current techniques that lose error integration.

The resulting guidance method can be applied to any bearing-only system, such as submarines using passive sonar, anti-radiation missiles, or small UAVs seeking to land on power lines for energy harvesting. System integration, variable timing methods, and discontinuity management techniques are provided for actual hardware implementation. Validation is accomplished with both simulation and flight test, autonomously landing a quadrotor helicopter on a wire.

AFIT/DS/ENY/11-24

To my incredible wife and wonderful children, who make life sweet. You bring joy to everything we do. May God be pleased with our efforts

—Col 3:23.

Acknowledgments

The project would not have been possible without the patient instruction and guidance of my advisor, Dr. Rich Cobb, to whom I am deeply indebted. Thanks for everything! Your succinct assessment of the first full flight test pretty well sums it up. Additional thanks is due to my committee members, Dr. Baker and LtCol Harmon, who were both instrumental in my understanding of optimal control and the pseudospectral method. Without your help, I'd likely still be staring at Fornberg. . . probably upside down.

I also owe a sincere word of gratitude to AFRL/RB for the research funding, as well as the use of the μAVARI indoor flight facility with the associated crew and equipment. In addition, I wish to acknowledge the great help of Mark A. Smearcheck of the AFIT Advanced Navigation Technology (ANT) Center with translating code, writing the dealer function, and braving launches and recoveries of the quadrotor during flight control development—not an entirely safe task. . .

The original idea for this project came to me from Dr. John Raquet, a friend and a mentor. You have been an example to follow, both in the academic arena and without, and I am indebted for the impact your entire family has had on mine during this time, my deep thanks. Any time the cousin of the chauffer for the Assistant Deputy Minister of Interior Decoration in Palau needs a lab tour—I'm there.

Steven M. Ross

Table of Contents

Page

Page

Page

List of Figures

Figure Page

Figure Page

Figure Page

List of Tables

List of Abbreviations

Abbreviation		Page
AFIT	Air Force Institute of Technology	131
AFRL	Air Force Research Laboratory	6
ANT	Advanced Navigation Technology (Center)	131
CRLB	Cramér-Rao Lower Bound	27
cg	Center of Gravity	42
DCM	Direction Cosine Matrix	46
DOP	Dilution of Precision	66
DRA	Defense Research Associates	6
DoD	Department of Defense	6
EKF	Extended Kalman Filter	52
FIM	Fisher Information Matrix	27
FOV	Field-of-View	10
GPM	Gauss Pseudospectral Method	21
GPS	Global Positioning System	8
HARM	High-Speed Anti-Radiation Missile	3
HEFK	Hybrid Extended Kalman Filter	15
INS	Inertial Navigation System	10
IRSTS	Infrared Search and Tracking System	3
KKT	Karush-Kuhn Tucker	22
LGL	Legendre-Gauss-Lobatto	22
LGR	Legendre-Gauss Radau	22
LG	Legendre-Gauss	22

Abbreviation		Page
LOS	Line of Sight	3
LPM	Legendre-Gauss-Lobatto Pseudospectral Method	21
LQE	Linear Quadratic Estimator	18
LQG	Linear Quadratic Gaussian	36
LQR	Linear Quadratic Regulator	18
MIMO	Multiple-Input, Multiple-Ouput	18
MPC	Model Predictive Control	30
NLP	Non-Linear Programming	20
NOC	Neighboring Optimal Control	120
PD	Proportional-Derivative (Control)	137
PI	Proportional-Integral (Control)	141
PLUS	Power Line Urban Sentry	6
PSM	Pseudospectral Methods	12
PWM	Pulse Width Modulation	133
pdf	Probability Density Function	59
RB	Air Vehicles Directorate	147
RPM	Radau Pseudospectral Method	21
RPV	Remotely Piloted Vehicle	8
RTOC	Real-time Optimal Control	12
SPKF	Sigma-Point Kalman Filter	59
SQP	Sequential Quadratic Programming	20
sUAS	Small, Unmanned Aerial System	6
TCP	Transmission Control Protocol	123

Abbreviation		Page
TPBVP	Two-Point Boundary Value Problem	1
UAV	Unmanned Aerial Vehicle	8
UKF	Unscented Kalman Filter	13
UT	Unscented Transformation	12
μAVIARI	Micro Air Vehicle Integration and Application Research Institute	145

List of Symbols

Symbol		Page
$\mathbf{0}_n$	Zero Matrix, $n \times n$	47
$\mathbf{C}$	Path Constraints	82
$\mathbf{C}^{\tilde{\mathbf{x}}}$	State Path Constraints	83
$\mathbf{C}^{\mathbf{u}}$	Control Path Constraints	83
c_i	Basis Function Weighting Constant	21
$\mathbf{D}$	Differentiation Matrix	125
e_p	Proportional Path Error	101
$\mathbf{F}$	Matrix of Dynamic Constraint Functions Evaluated at Collocation Points	126
$\boldsymbol{f}$	State Dynamics	82
$\mathbf{H}$	Measurement Function Jacobian	66
$\mathbf{H}_y$	Polar Measurement Function	48
$\mathcal{H}$	Hamiltonian	25
$\mathbb{H}$	Heaviside Step Function	83
$\mathbf{h}$	Measurement Function	44
$\mathbf{I}_n$	Identity Matrix, $n \times n$	47
$\boldsymbol{\mathcal{I}}$	Fisher Information Matrix	66
I_{xx}, I_{yy}, I_{zz}	Moments of Inertia	136
J	Performance Index	23
$\mathbf{K}$	Kalman Gain	58
$\mathbf{K}_{UKF}$	Unscented Kalman Filter Gain	61
L, M, N	Body Axis Moments	136

Symbol		Page
L_i	Lagrange Polynomial Basis Function	125
$\boldsymbol{\mathcal{L}}$	Lagrange Cost Functional	82
$\mathbf{M}$	Observability Grammian	67
N	Number of Collocation Nodes	124
$\mathbf{P}$	State Covariance Matrix	57
$\mathbf{P_y}$	Polar State Covariance Matrix	58
$\mathrm{P}_{xx_{max}}$	Final Covariance Limit, x-direction	150
P_N	Legendre Polynomial, Degree N	124
p	Likelihood Function	64
$p,\ q,\ r$	Body Axis Angular Velocities	136
Q_{gust}	Simulated Wind Noise Strength	110
$\mathbf{Q}$	Process Noise Strength	47
$\mathbf{R}$	Residual Error Matrix	130
R	Covariance	45
r_{cam_x}	Camera Lever Arm Coordinate, x_b-Axis Direction	46
$\mathbf{S}$	Propagation Jacobian	55
$\mathbf{s}$	Polar Propagation Equation	55
$\mathbf{s}_x$	Transformation Function, Polar to Cartesian	54
$\mathbf{s}_y$	Transformation Function, Cartesian to Polar	54
$t,\ t_0,\ t_f$	Time (Continuous), Initial, Final	22
t_{app}	Expected Time at Approach Point	122
t_{bl}	Length of Blending Segment	122
t_k	Time (Discrete)	45

Symbol		Page
t_{perch}	Expected Time at Perch Point	158
$\bar{t}_i$	Midpoint Time Between Collocation Points	130
$\mathbf{U}_i^{\mathrm{LGR}}$	Matrix of All Controls at Every Collocation Point	126
$\bar{\mathbf{U}}$	Matrix of All Controls at Every Discretization Mid-Point	130
$\mathbb{U}$	Set of Admissible Controls	26
$\mathbf{u}$	Control Vector	46
$\mathbf{u_y}$	Polar Control Vector	48
$\mathbf{u}^*$	Optimal Control Vector	84
$u,\ v,\ w$	Body Frame Velocities	137
u_x	Navigation Frame Control Component, x-Axis Direction	46
V_{ac}	Total Velocity (SAM Avoidance Case Study)	96
$\mathbf{v}$	Navigation Frame Velocity Vector	43
$v_x,\ v_y,\ v_z$	Navigation Frame Velocity Components	43
$v_{x_{min}},\ v_{x_{max}}$	Navigation Frame Velocity Limits, x Component	43
v_{max}	Total Velocity Limit	43
$\mathbf{W}_u$	Control Weighting Matrix	83
$W_c^{(i)}$	Sigma-Point Covariance Weight	60
$W_m^{(i)}$	Sigma-Point Mean Weight	60
$\mathbf{w}_k$	Process Noise Vector	47
w	Wind (SAM Avoidance Case Study)	98
w_i	Quadrature Weight	23
w_{gust}	Simulated Wind Gust	110

Symbol		Page
Z	Measurement Random Variable	64
z_{min}, z_{max}	Altitude Limits	49
$\widehat{\mathbf{z}}_k^-$	Measurement Space Sigma-Point *a priori* Mean	60
α	Sigma-Point Spread Parameter	60
β	Total Measurement Angle	44
β_{image}	Image Measurement Angle	44
$\mathbf{\Gamma}$	Mayer Cost Function	82
$\boldsymbol{\gamma}$	Boundary Conditions	82
Δt	Time Step	47
Δt_{calc}	Inclusive Loop Time (RTOC, Estimation, Delay)	87
Δt_{meas}	Expected Measurement Interval	61
δ	Delta Function	110
δ_{kj}	Kronecker Delta Function	45
ζ	Measurement Sample	64
η_{gust}	Simulated Wind Input Noise	110
η_k	Measurement Noise	45
θ	Pitch Angle	44
κ	Sigma-Point Tuning Parameter	60
$\mathbf{\Lambda}(t, t_0)$	Relative State Propagation Matrix, from $t = t_0$ to t	55
$\boldsymbol{\lambda}$	Costate Vector, Lagrange Multipliers	25
λ	UKF Scaling Parameter	59
μ	Distribution Tuning Parameter	61
$\mathbf{\Xi}_{cor}$	Position Correction Matrix	122

Symbol		Page
$\Xi_{0\to 1}$	Position Correction Cosine Wave Vector (Smooth-Fast)	122
$\Xi_{0\to 1 flare}$	Position Correction Cosine Wave Vector (Smooth-Smooth)	123
ξ_k	Measurement Model	45
ξ_i	Information State	79
ρ	Range	48
ρ_i	Range to SAM i	106
σ_β	Measurement Angle Standard Deviation	67
$\boldsymbol{\varsigma}(t, t_0)$	Cartesian Position Change from t_0 to t	55
$\tau,\ \tau_0,\ \tau_f$	Transformed Time Variable, Initial (-1), Final (1)	22
τ_k	Collocation Point	124
$\mathbf{\Phi}_{k+1,k}$	State Transition Matrix, t_k to t_{k+1}	66
ϕ	Bank Angle	136
ϕ_i	Basis Function	21
ϕ_p	Polynomial Basis Function	125
$\mathcal{X}_k^{(i)-}$	Sigma-Point Vector, ith Column, Time t_k, Prior to Update	59
ψ	Heading Angle	136
ψ_c	Commanded Heading	101
ψ_{fb}	Heading Control Feedback (SAM Avoidance Case Study)	101
ψ_e^*	Heading Error Between True Optimal and Calculated Optimal Solutions (SAM Avoidance Case Study)	100
$\widehat{\psi}^*$	Estimate of Optimal Heading	104
Ω	Finite Time Interval	89

STOCHASTIC REAL-TIME OPTIMAL CONTROL: A PSEUDOSPECTRAL APPROACH FOR BEARING-ONLY TRAJECTORY OPTIMIZATION

I. Introduction

"The difficulty is designing machines that can approximate the remarkable human ability to reason and make decisions in an environment of uncertainty and imprecision." -Lotfi A. Zadeh [120]

This dissertation addresses a problem at the crossroads of the fields of estimation and optimal control. For a basic two-point boundary value problem (TPBVP), optimal control can be thought of most simply as finding the "best" path and control to get from "here" to "there." On the navigation side, optimal estimation can be thought of as finding the best guess of a target location given a set of imperfect measurements. Present levels of technology are excellent at doing both. . . individually. But what is the best path to get to a target with a location that is *not* well known? If the quality of the target estimate can be improved by varying the path taken, what is the optimal path that will accomplish a primary mission, while maneuvering enough to get the estimation quality required for success in that mission? This research seeks an automated method to find that solution quickly, fast enough for real-time guidance, and robust enough for the uncertainties and disturbances of real life.

The human mind is an amazing optimization machine that solves these problems regularly. Every control decision, from the way you drive home from work to the way you hit a baseball, is made in an optimal manner. We continually try to maximize or minimize some performance index of time, effort, power, accuracy, or a myriad of other considerations—often simultaneously. In the world of control, a modern computer can accomplish a task far more precisely, but cannot compare in ability to deal with a wide range of uncertain inputs and incomplete information. In the estimation realm, we have finely-honed computer filtering algorithms and data processing techniques to "squeeze out" every piece of useful sensor information provided, but our machines lack the human's intuitive feel for *how to move* to make that information *better*. The ability to sense what we are missing—and how to get it—causes us to tilt our ear, to lean around a corner, and to slow down before a blind intersection. Improving the information isn't the primary mission, but it is done "enough" to meet the needs of a higher goal.

The goal of this research is to provide this capability to an autonomous controller, capable of being used in real-time, with the recognition that what is optimal in a stochastic environment is not only a function of "What do I want?" but also of "What do I know now, and how well do I know it?" A guidance system should be able to figure out what information is still needed for success, and be able to produce the path and control to get it—taking as little as possible away from the primary mission.

1.1 Motivation

The ability to maneuver in relation to current levels of target knowledge will directly benefit systems in which estimation performance is dependent upon the geometry of the constellation of measurements that has been received. This research focuses on path guidance to land small aircraft on power lines using a single camera for a sensor. Range to the target must be found through maneuver, as is the case for several examples of bearing-only systems that would benefit from the same type of guidance, such as submarines using passive sonar, high-speed anti-radiation missiles (HARM), or systems with infrared search and tracking systems (IRSTS), shown in Figure 1.

(a) Seawolf Submarine (Photo: U.S. Navy)

(b) IRSTS (Photo: MILAVIA)

(c) HARM Missile (Photo: FAS.org)

(d) Aeryon Scout Quadrotor (Photo: Aeryon Labs)

Figure 1. Bearing-only Systems

Each of these systems relies on a bearing-sensor to track targets as part of a greater mission. The HARM system receives bearing information from the electronic emissions of a ground radar site and must determine a path to hit the site with maximum energy while respecting its own sensor limitations [28]. Motion away from the target line of sight (LOS) increases the fidelity of the target position estimate, but can simultaneously decrease the missile's energy.

Submarines (and other Naval vessels) do almost all of their target motion analysis passively [43], with bearing-only sonar tracking algorithms very similar to the optical tracking problem of the IRSTS. In both cases, choosing to use active ranging (via sonar or radar, respectively), while much more accurate, gives away the presence (and interest) of the sensor, and its position. Passive ranging requires maneuver, and current submarine techniques to accomplish it haven't significantly changed since the 1950s—a turn is made orthogonal to the target LOS and that heading is held long enough to produce a bearing fan of measurements suitable for algorithms such as Ekelund or Spiess ranging [104], followed by one extra turn to eliminate ambiguities. A maneuver that would do this while closing to attack range, or while increasing standoff distance is left to the "seat of the pants" intuition of the commander [43]. Ideally, an automated guidance system could integrate the tools of optimal control and bearing-only target analysis to maneuver the submarine in such a manner that it achieves exactly the minimum target certainty required for the fire-control system precisely at the time the submarine reaches maximum torpedo range.

1.1.1 Bearing-only Target Analysis.

Bearing-only target analysis is a classic estimation problem, and exists in applications from basic triangulation in land surveying to missile detection systems [42]. The inability to sense range with each measurement, combined with the inherent non-linearity of the problem, make estimation of a target location, or source, problematic. One common solution is to take measurements from non-collocated sensors, as is done with stereo vision, in Figure 2.

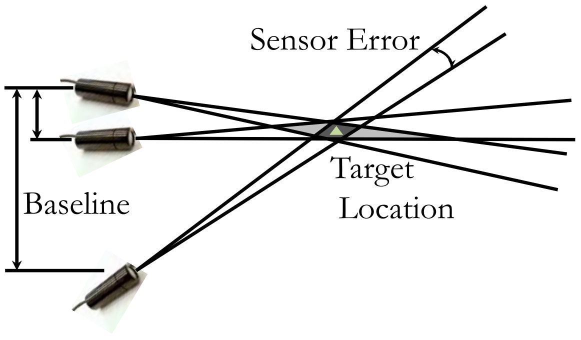


Figure 2. Ranging with Stereo Vision

The limitation to this technique is that the size and shape of the uncertainty "bubble" around the estimate is a function of the baseline between the sensors. The farther the target, the more baseline is required for resolution. Systems such as small aircraft with optical cameras and fighter jets with IRSTS lack the physical dimensions for enough baseline to make stereo vision effective at their respective ranges of interest.

For a monocular system, range estimation is analogous, but the sensor must physically be moved orthogonal to the target LOS (or be in a position to observe orthogonal target motion), artificially creating enough baseline to enable triangulation. This motion comes at the expense of the primary mission, unless the entire purpose is localization of the target. Depending on the accuracy of the sensor, a wide range of aspect angles may be required for a reasonable range estimate. If other, more accurate, ranging sensors are available, such as laser range finders, radar, or active sonar, these would obviously be preferred. However, many systems are limited by stealth considerations, physical dimensions, or payload capacity to a single, passive bearing sensor. One such system is DARPA's 19 gram Nano-Hummingbird with a monocular camera shown in Figure 3.

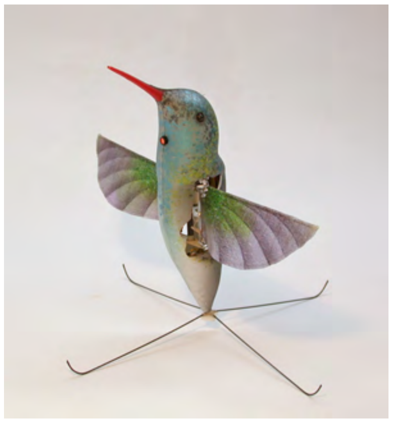


Figure 3. DARPA Nano-Hummingbird (Photo: AeroVironment)

1.1.2 Power Harvesting.

The Department of Defense (DoD) has dedicated an unprecedented amount of time and energy into research of small, unmanned aerial systems (sUAS) in recent years for a variety of purposes [109]. The ability to move a sensor, or other small payload, to a particular site for surveillance and other purposes provides great capabilities, particularly if it can be done undetected [31]. The trend of recent design has been to reduce the size of these vehicles dramatically. The Nano-Hummingbird is a great example, but current tactical systems are on the order of the Wasp and the Raven® B, with approximate wingspans of 72-cm and 140-cm, respectively, shown in Figure 4.

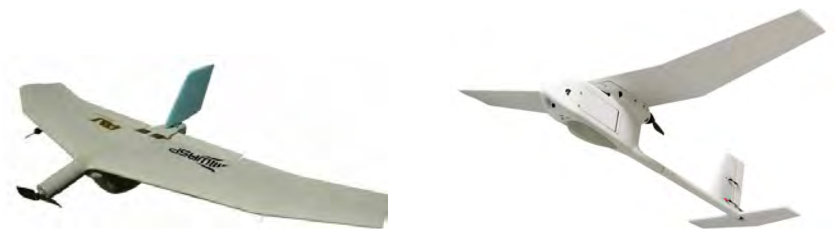
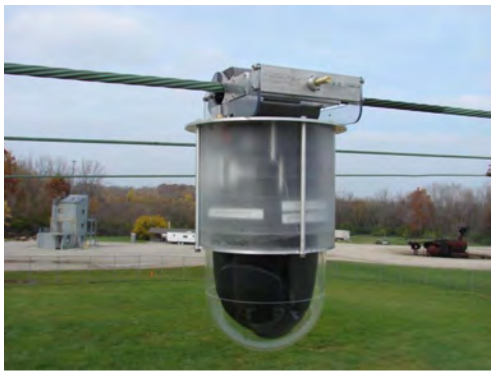

(a) Wasp III (b) Raven® B

Figure 4. Current Tactical sUAS Systems with Monocular Sensors (AeroVironment)

Obviously, sensor quality and availability decrease commensurate with the vehicle's size and weight. In addition, smaller systems have lower flight speeds and greatly decreased range. Compounding the problem is the obvious lack of payload capacity. For electric motors, battery life is severely limited by allowable payload. This translates into short range assets that have limited persistence.

One possible method of significantly extending both range and station time is energy harvesting off of available power lines during a mission [17]. The Power Line Urban Sentry (PLUS) program at the Air Force Research Laboratory (AFRL) with the work of Defense Research Associates (DRA) has been successful with recharging batteries through induction, by clamping around medium-sized power lines, such as you would see in a typical neighborhood [98].

This technique has powered observation sensors with a camera, modem, and server board, allowing the camera to be accessed and controlled by a common iPhone. The concept is to extend this technology to sUASs, hanging them from a power line until recharged as in Figure 5.


(a) Camera Powered by Passive Induction (Photo: DRA)

(b) Conceptual Future Use (Photo: Bob Fornal)

Figure 5. Power Line Harvesting

The observation camera technology is at the early fieldable stages, but currently the weight of an inductor clamp large enough to recharge a sUAS sized battery in a reasonable time is problematic, given the extremely limited available payload capacity of small aircraft. In addition to battery development, future advances in inductive technology, such as recharging pads for cell phones, will almost certainly open harvesting as a viable future option for power regeneration. Even now, the technology exists to design a "home base" power station, attached the same way as the current sensor suites. A sUAS could be used locally from the position of the base, such as flying preprogrammed loops for border security, etc., and could return to the base for power replenishment. Multiple vehicles could cycle off of the power line for continuous coverage.

There is a near-term requirement, therefore, for a control algorithm to find, approach, and perch on power lines. The critical difficulty in this is the measurement of the relative position between the sUAS and the power line itself. AFRL's research has shown that avoiding the issue by merely tracking the angle to the power line and running into it at flight speed (with a hook system designed for that purpose) is overly abrupt and can lead to failure of the vehicle [17]. Morphing of the wings in an attempt to decrease the stall speed has been attempted [116], but the thought of automating this process only accentuates the great need for accurate relative position data between the sUAS and the intended landing point. As this research is extended from perching on power lines to rooftops and window ledges, the price for a "miss" goes up, and the requirement for accurate relative data becomes even greater. Using preset landing coordinates is ineffective and removes too much flexibility from the system. Though we have made great strides in GPS receiver miniaturization and accuracy, mensurated coordinates of every power line out there are simply not available. In theory, we could use space-based assets and extensive mission planning to get an exact point to fly to, but experience from attempts at open-loop control for relative taskings strongly suggests that this is not a feasible solution. Real-time feedback of the relative position must be made available, and in the bearing-only sensor case where range is important, this must be attained through manipulation of the path.

1.2 Important Semantics

Throughout the document, the following notation and definitions are used:

- The Air Force is making an effort to move toward the use of the acronyms sUAS and RPV (Remotely Piloted Vehicle) and away from the more familiar general term UAV (Unmanned Aerial Vehicle). Since the power line scenario is specific to the sUAS, that term will be used, with the understanding that the algorithm itself could be applied to any UAV, obviously on a different scale and with a different final mission.

- The terms **path** and **trajectory** are synonymous.

- The term **observer** is used to describe a system that plans and tracks a trajectory to produce sufficient observability of a target location to accomplish a mission. In the power line landing context, the term observer will be used interchangeably with the term "vehicle," and refers to the entire unmanned system, including the sUAS platform and the associated sensors.

- The **target**, or source, is the object whose location the observer is attempting to estimate. Note that the target is not necessarily the maneuvering goal of the observer (i.e. the submarine needs to ascertain the position of a target contact, but is maneuvering to an attack position relative to it, not to the target itself). For the landing scenario, the term target is synonymous with power line (for the true application) or wire (for the flight test).

- The **approach point** is the desired maneuver end point of the observer during the segment of the mission directed by the **path planner**, which is the algorithm that determines the optimal trajectory for the given conditions. The prescribed level of certainty in the target position must be attained prior to reaching the approach point.

- The term **localization** will be used to denote the specific case where **bearing-only tracking** techniques are applied to a target known to be static.

- Discrete **time steps** of a state $x(t = t_k)$ are abbreviated x_k where it will not cause confusion. Context must be used to determine whether the length of the step is Δt (for the control station and autopilot), Δt_{meas} (for measurement updates $\beta_k,\ \beta_{k+1}$), or Δt_{calc} for epochs. The term **epoch** is reserved for one step of the path planner, which produces a complete time history of the states and controls in each solution. For instance, the phrase $\mathbf{x}_{0_{k+1}} = \mathbf{x}_k(t + \Delta t_{calc})$ means that the initial state to be used in the path planner at epoch $k + 1$ is determined by propagating the state time history received at epoch k forward to time $t+\Delta t_{calc}$. Values at the same time step, but calculated before and after a measurement update has been incorporated are delineated by $x_k^-,\ x_k^+$.

1.3 Assumptions

This project assumes the existence of an observer vehicle with an indigenous navigation capability from a system such as GPS, INS, or some sort of image processing such as optical flow [54, 112], sufficient to determine its inertial position relative to constraining borders, be they terrain, walls, an altitude ceiling, political boundaries, etc. The borders will be observed as position limitations, but the observer is assumed to have control authority to move freely within the borders, respecting velocity and acceleration constraints (obstacle avoidance is not considered). The observer has available processing power for estimation and real-time optimization.

The observer is also assumed to be equipped with a bearing sensor capable of identifying the target and producing an angular measurement to it, and the target is assumed to be initially within the sensor's field-of-view (FOV). The measurements are delayed, but time tagged, and corrupted by uncertainties in the sensor and the vehicle orientation. Specifically for the power line scenario, it is assumed that the vertical angle to the power line can be found with a line detection algorithm operating on sequential images from an optical sensor. It is assumed that the power line is horizontal, and that the angle along the wire is not observable. Future research could certainly expand on this with allowance for utility pole identification, stadiametric ranging, sag analysis, or other considerations. With no observable lateral changes, there is no benefit to flight parallel to the wire, and the optimal trajectory becomes planar, maneuvering in the vertical to increase observability. The submarine variant of the problem can also be considered planar, only horizontal. In this case, there is some observability that could be gained from vertical motion, but the realizable benefit from the restricted ability to maneuver in the vertical is small at the long horizontal distances typical for submarine contacts that are still un-ranged.

1.4 Project Summary

This dissertation proposes a new method of approaching the bearing-only trajectory planning problem that enables simultaneous consideration of the optimal control problem and prescribed final estimation requirements, overcoming the typical limitations of previous approaches. The trajectory planning goal is to provide an optimal path and control for arrival at a point, or set of points, offset relative to a target position, the location of which must be determined to a predefined certainty by varying the engagement geometry while receiving stochastic, delayed angle measurements. In addition to allowing a general cost function, the method treats required final directional covariance in the target estimate as a constraint, optimally considering the observability requirements of the bearing-only sensor, without wasting maneuver effort beyond the minimum necessary for accomplishment of tasks such as landing or weapons employment.

In order to be implemented beyond theory, considerations of noise and flight disturbances mandate the need for the trajectory planning capability to be part of an on-line system with feedback, made fast and simple enough to be applied iteratively in combination with an estimation filter using own-ship position and target bearing measurement data. A real-time optimal control (RTOC) system is designed using an Unscented Transformation (UT) based filter for target estimation coupled with an efficient pseudospectral method (PSM) algorithm designed around the same principles.

System stability and precision are validated through Monte Carlo-style simulation. An existing quadrotor helicopter is then extensively modified to allow application of the RTOC system, and effectiveness and feasibility are verified through flight test, guiding the quadrotor to a wire and landing upon it. Several integration techniques are created and presented that should be considered in an effort to apply a RTOC system to actual hardware.

1.4.1 Contributions.

Several significant contributions to the field of science have been made in the accomplishment of this work:

- The most significant contribution is a fundamentally new method of approaching the bearing-only trajectory optimization problem. A myriad of small variations have been applied to this problem, all of them centering around optimizing some scalar information metric. This work provides the ability to achieve a *predetermined final certainty* in a target estimate, while simultaneously accomplishing a primary mission beyond pure localization. Prescribing a final certainty level as a constraint allows any general cost function to be used for primary guidance of the vehicle, as most appropriate for a given system and its primary mission, while guaranteeing that the physical certainty required for the navigation needs of that mission will also be met. The method does not suffer from the problematic loss of directional information caused by scalar compression of an information metric in other methods. The key enabler for this technology is the ability to estimate the effects of discrete measurement updates with *information states* in a polynomial space, allowing propagation of geometric certainty information in relation to time within the context of the optimal control problem.

- This work also contributes a physically realizable RTOC algorithm for a system with moderate dynamics using pseudospectral methods that are tailored to have a coherent effort with an estimation filter. The same underlying principles of the Unscented Kalman Filter (UKF) designed for this work are incorporated into the optimal control problem. Specific computational issues, such as singularity avoidance, are addressed in order to allow a method for a PSM to be used to control a vehicle to an unknown final boundary condition. Beyond simulation, this work fleshes out all of the details from concepts and theory to hardware implementation.

- The application of pseudospectral methods is new to the field of real-time optimal control, and has seen little, if any, application beyond simulation for systems with moderate dynamics. Current trends in the RTOC community include speeding up an outer trajectory planning loop to the point where control can be applied in a recursive, open-loop manner, re-planning the optimal path fast enough to achieve the equivalent of optimal feedback control. While effective in the simulation environment, research for this project highlighted significant limitations in these techniques. Removal of the classical feedback concepts, while tempting, loses the insights gained from integration of path error, making the system unable to properly respond to non-zero mean disturbances. A return to the classical application of optimal control with an error feedback loop is proposed, with the addition of an error bias feedback loop to the path planner enabling the system to respond to a stochastic environment in an optimal manner. This method should be adopted as the industry standard for application of future RTOC algorithms, independent of the numeric solution method used.

- A significant advantage in RTOC is provided through allowing an unknown calculation time for the optimization cycle, making use of new solutions as soon as they are available. The necessary implementation tools of tip/tail blending and variable-rate loop integration are developed. Though more complex, the optimal path update rate is increased markedly, greatly improving the flexibility and response to uncertainty for any RTOC system.

- Finally, the algorithm produced also provides the community with a planning tool likely to be needed as power line landings become more of a possibility. Recognizing that some very small systems will not have the computational capacity and energy for RTOC, this tool provides a way to find and extract the key characteristics of the optimal path. The general shape and decision points of the solution will vary from system to system by dynamics, scale, and speed limitations. By using the simulation provided herein, the trajectories may be run for the specifics of a particular system, and heuristics can be built which mimic the optimal solution without the computational burden.

Application of this technology to modern systems translates into a first shot opportunity for a submarine, a higher end-game energy for a HARM, or the ability to land a sUAS on a power line for energy harvesting.

1.5 Document Outline

All of the major concepts involved in this research are presented in this document, to a level of detail that should allow reconstruction, if desired. Chapter II presents the relevant current state of the art in the field of optimal control. Limitations of computation time and inadequacy for stochastic problems are discussed, and direct methods of transcription and collocation are detailed as potential techniques to speed up the process enough for real-time implementation. Past efforts in the area of trajectory optimization are covered, as well as attempts to combine trajectory optimization with optimal control in *dual control* methods. Real-time efforts and limitations are discussed throughout the chapter.

Chapter III describes the details of the specific land-on-a-wire problem, and scopes the region of interest. The most relevant coordinate systems used and the dynamics and measurement models are introduced in the Cartesian system, and transformed into the polar coordinate system for use with the hybrid Extended Kalman Filter (HEFK) and the shooting method later developed. Chapter IV addresses the bearing-only estimation problem, and develops the HEKF and an Unscented Kalman Filter as estimation options. The HEKF was used for a large portion of the research and is available for future users who desire the final covariance limitations in the polar format without additional non-linear transformations. The filter that was selected for the final flight tests was based on the Unscented Transformation.

In Chapter V, the question is addressed of how to get the information from the discrete measurement updates, and the geometry from which they were taken from, encapsulated into a form which an optimal solver could use to determine how to adjust an optimal path. Information states are developed that are polynomial approximations of elements in the Fisher Information Matrix. These are used to allow the optimal solver to have an information gradient for how to change the path, and to allow application of the final required covariance as a boundary condition. From this, the optimal control problem is constructed with the information states augmenting the system model.

With a single-shot solution in hand, Chapter VI focuses on the structure of RTOC implementation, specifically addressing a current RTOC practice of equating recursive open-loop solutions with feedback control when the solutions are available fast enough. Case studies are provided as counter examples, and a structure that is more effective in the face of real-world biases and time-correlated disturbances is presented.

Chapter VII implements that structure in the full RTOC algorithm used for the land-on-a-wire problem, addressing hardware considerations such as discontinuities and timing. Use of a variable calculation time is shown to increase system flexibility and responsiveness by increasing the optimal solution update rate, and potential issues with doing this are managed. Radau Pseudospetral Methods are used to transcribe the continuous optimal control problem into a non-linear programming problem, and adaptive grid refinement is used to further increase the solution speed.

Chapter VIII describes the actual quadrotor helicopter system, as well as the flight control modifications required to enable the actual flight test. The results of the flight tests are presented with analysis in Chapter IX, along with results from a Monte Carlo-style simulation that looks at robustness and accuracy. The conclusions drawn from the results, as well as recommendations for future work, are found in Chapter X. For reconstruction, the flight control simulator Simulink model is presented in Appendix A, and selected portions of the MATLAB® code for the RTOC algorithm that may be of particular interest are presented in Appendix B.

II. Related Work

THE primary focus of this dissertation is the design of a real-time optimal control algorithm capable of commanding and updating an optimal path for a sUAS to perch on a wire with bearing-only measurement data, considering current and required uncertainty levels in the definition of optimality. Making the system implementable requires the integration, application, and expansion of existing knowledge from several broad and often overlapping areas, including optimization, non-linear flight dynamics, aircraft control, navigation, and estimation. Many areas where work was required that is not expected to be contributory to the body of knowledge are not highlighted in this chapter.

2.1 Optimal Control

Optimal control and trajectory optimization have been studied for centuries. In essence, it is the search for the set of control signals that will minimize (or maximize) some performance criterion while satisfying some physical constraints [66]. The roots of optimal control rest in the Calculus of Variations, formulated by giants such as Bernoulli, Newton, Leibniz, Euler, and Lagrange [40]. Great strides occurred in the 1800's, when Hamilton and Jacobi formalized the concept of a differential equation governing the partial derivative of an objective function with respect to the parameters of a family of extremals (we'd call them states). Legendre, Clebsh, and Weierstrass followed by refining the necessary conditions for a true optimal solution, and by the early 1900s, Bolza and Bliss had built the structure of the Calculus of Variations to its present form [12].

As is usually the case, technological advancements opened up new needs for engineering solutions, and the space race of the 1950s brought the next jump in optimal control with the work of Soviet Lev Semanovich Pontryagin [89], with his maximum principle, and American Richard Bellman [3], known best for his work in dynamic programming. As control systems became more digital and computers more prolific, Rudolf Kalman hugely expanded the practical applicability of optimal theory when he found an optimal state feedback gain through solving the backward Riccati equation on an infinite time horizon for Multiple-Input, Multiple-Ouput (MIMO) systems [60].

2.1.1 Limitations of Optimal Control.

Kalman's feedback gain process, later dubbed the Linear Quadratic Regulator (LQR), and its sister, the Linear Quadratic Estimator (LQE), were especially significant [2]. For linear systems (or reasonably linearizable systems), a solution for optimal feedback was now practical and realizable, with appropriate attention to robustness [14].

Unfortunately, an optimal feedback solution is often not available for systems with complexities such as non-linear problem spaces, intricate cost functions, time-varying physical constraints, or problems where knowledge of the objective is dependent on the path chosen (such as simultaneous trajectory optimization and localization). In this case, the basic practice is to numerically solve the optimal control problem in an open-loop sense *a priori*, assuming the boundary conditions that will exist when the control is applied. The optimal control is then applied and disturbances are rejected by feeding back the error between the expected optimal path and the current position [62]. Stability and feasibility are maintained if the system remains "close enough" to the nominal path.

For cases where the exact state at the time the control will be needed is not foreknown, and for cases where disturbances, model inaccuracies, or noisy measurements cause large deviations from the optimal path, an ability to recursively solve the problem with new information is intuitively desirable. The drawback, historically, has been the extensive calculation time required to numerically solve an optimal control problem. Methods dubbed shooting, multiple shooting, genetic algorithms, simulated annealing, particle swarm optimization, and others have been used with varying speed and numerical stability. The most promising techniques have included parameterization of the problem into a finite solution space, as is accomplished in direct methods.

2.2 Direct Methods

Optimal control methods can be generally categorized into either direct or indirect [5]. Indirect methods involve determining extremals with the Hamiltonian and first-order optimality conditions [13, 66]. While these methods offer great insight into the problem, they have several drawbacks. First, indirect methods cumulatively evaluate the objective function (and its gradient) over the entire trajectory, as opposed to direct methods which do this only at several points. Direct methods, therefore, have more information on where to apply changes to the initial guess, resulting in larger radii of convergence than with indirect methods, which require a good (and generally nonintuitive) initial guess of both states and costates [7]. In addition, if the problem is constrained, the indirect method requires breaking the problem into constrained and unconstrained arcs, which may not be known *a priori* [5]. In addition, indirect methods are often extremely sensitive to problems with unknown boundary conditions [93].

Direct methods are more robust to errors in the initial guess, more computationally efficient, and apply to a larger range of problems. Euler was the first to create what we now call the direct method of finite differences, though the method lay dormant for quite some time [26]. Direct methods transcribe the optimal control problem into a non-linear programming problem (NLP), which in modern days is then solved numerically [107, 48].

2.2.1 Transcription and Collocation.

The underlying technique for a direct method is collocation, or transcription—terms which are often used interchangeably. The state vector is approximated and represented by a discrete number of variables (e.g., coefficients of a Fourier series). The continuous dynamic constraints for the system are then evaluated at select collocation points, or nodes, producing a discrete number (albeit typically a large number) of *static* equations—one for every state, at every node [86]. These constraints are used to form a new, static optimization problem, seeking a vector of state and control variables at each collocation point to minimize the overall cost while obeying each of the new static constraints. In essence, the problem has been transformed from an infinite-dimensional optimization problem to a finite-dimensional, non-linear programming problem [8]. There is no guarantee that the optimality or the dynamics hold at other than the collocation points [101], but I. M. Ross has shown, for an increasing number of nodes, that "If the optimal solution of the discrete problem converges, it must converge to an optimal solution of the continuous problem [41]." After conversion to an NLP, the problem can be solved with a host of solvers designed for this purpose such as SNOPT [39], SPRNLP [6], or KNITRO [15], most of which use Sequential Quadratic Programming (SQP) as the primary solution method and account for matrix sparseness with a semi-definite reduced-Hessian.

2.2.2 Pseudospectral Methods.

Instead of directly discretizing a state or control history, the number of optimization parameters can be decreased by parameterizing the vector using a series:

$$\mathbf{u}(t) = \sum_{i=1}^{N} c_i \phi_i(t) \tag{1}$$

The constants, c_i, are the parameters solved appropriate for the set of basis functions, $\{\phi_i(t)\}_{i=1}^{N}$. If orthogonal polynomials are used as the basis functions, and the zeros of orthogonal polynomials (or their derivatives) are used for the collocation points, the method is dubbed *pseudospectral* [25, 96]. Using polynomials allows trivial differentiation, which makes enforcement of the dynamic constraints more efficient than other direct methods which rely on integration to approximate the vector field [56].

Pseudospectral methods had their origin in spectral methods, a technique for solving partial differential equations referenced as far back as Reddien in 1979 [94] and used extensively in the realm of fluid dynamics [16]. The ideas migrated into control theory in the field of chemical engineering with the work of Cuthrell (among others) [20]. Within recent years, the application of pseudospectral methods to optimal control has grown quickly, and the frequency of journal articles on the subject has had a sharp rise. At least in simulation, pseudospectral methods have been applied to the control of platforms spanning from cars [71] to hypersonic reentry vehicles [58].

There has been a great deal of development and refinement of PSM, resulting in three primary varieties, the Legendre-Gauss-Lobatto Pseudospectral Method (LPM), the Gauss Pseudospectral Method (GPM), and the Radau Pseudospectral Method (RPM). The fundamental difference stems from the selection of collocation points. Commonly, for problems with a finite final time (may be unknown), the affine transformation:

$$t = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2} \tag{2}$$

is applied to transform the problem from time interval $t \in [t_0, t_f]$ to the interval $\tau \in [-1, 1]$. The infinite horizon problem is mapped from $t \in [t_0, \infty)$ to the finite horizon $\tau \in [-1, 1]$, but states and controls at the final point are intentionally not calculated to avoid a singularity [27, 34]. Transforming the time allows selection of interpolation points from the interval -1 to 1. The distinction is made between state interpolation points, which include the endpoints $\tau = -1$ and $\tau = 1$, and the collocation points, where the dynamic constraints are applied [35]. GPM does not collocate at either endpoint, but only at the interior Legendre-Gauss (LG) points. This style of collocation leads to a set of discrete Karush-Kuhn Tucker (KKT) optimality conditions identical to the discretized form of the first-order optimality conditions of the continuous problem at the LG points, allowing the costates to be accurately estimated using KKT multipliers from the NLP [5]. RPM uses Legendre-Gauss Radau (LGR) points, which include one endpoint or the other (the non-symmetric points can be mirrored about zero). Though the KKT conditions differ, the method includes collocation at an endpoint, reducing the requirement to solve for that point and potentially increasing the accuracy of the solution. Notably, differentiation matrices from both GPM and RPM are both non-square and full rank, allowing the expression as an integration matrix, making the problem reversible. Costate estimates for both GPM and RPM converge exponentially. LPM, which uses Legendre-Gauss-Lobatto (LGL) points for collocation (including both endpoints), has a square, singular differentiation matrix. This directly provides the state and control at both endpoints, as well as ensuring the dynamics are met, but at the cost of a potentially non-convergent costate [35]. The weights, differentiation matrices, and techniques for generation of enough constraints differ for each of the methods.

For each of the techniques, the orthogonal nodes are not equally spaced, but clustered near the endpoints, similar to Chebyshev points. This spacing minimizes the Runge phenomenon, a potentially divergent oscillation that can occur when increasing the order of an interpolating polynomial, as in Figure 6 [65].

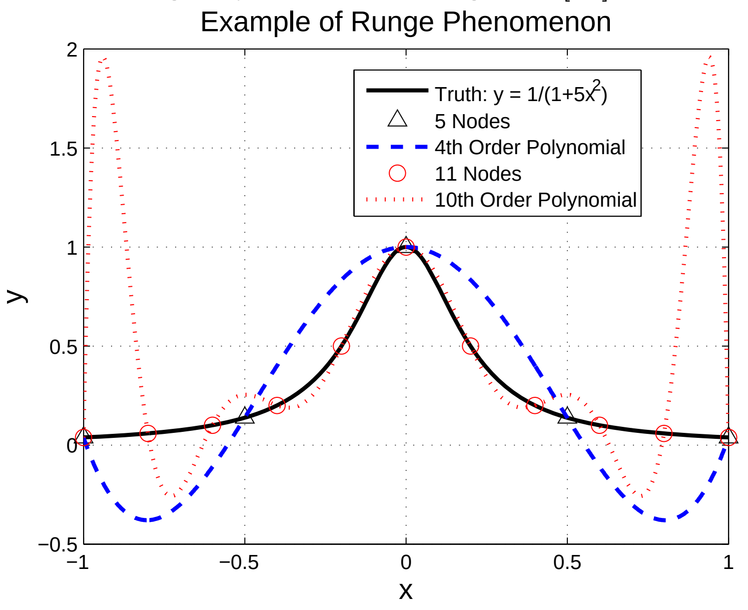


Figure 6. Runge Phenomenon as the Number of Equally Spaced Nodes is Increased

In addition to accurate interpolation, the proper selection of collocation points also aids in the evaluation of the objective function. With the states and control only being evaluated at discrete points, the objective function can be quickly calculated with quadrature, exact to polynomials of degree $2n + 1$, and guaranteed to converge for higher order polynomials to any continuous function by the Weierstrass Approximation Theorem [65]:

$$J = \int_{-1}^{1} f(x, u)\, dx \approx \sum_{i=1}^{n} w_i f(x_i, u_i) \tag{3}$$

where weights, w_i, are selected appropriate to the collocation scheme (e.g., Gauss points, Gauss weights).

Controls or states with discontinuities are problematic, often suffering from Gibb's phenomenon, (a large oscillation prior to a jump in the solution) [32]. If the problem is known to be non-smooth (a change in mass when a rocket drops a stage, for example), it is best dealt with by segmenting at problem areas with "knots" [96], or phases [91]. These can also be used to mark a point in the problem where the dynamics change. Since the nodes are concentrated at the start and end of each phase, the break point will generate the greatest nodal density, and the number of nodes for each phase can be increased until the solution is sufficiently accurate.

Tsuchiya sought to increase the density of nodes in the first portion of a solution in a near-real-time implementation for aircraft guidance. Recursive solutions were provided every 30 seconds. Assuming convergence of the next path, only the first 30 seconds of each provided path was flown. An introductory segment of fixed time was declared, with a higher node density to provide smoother control for the portion of the path that would actually be used [110].

2.2.2.1 Adaptive Grid Refinement.

Darby has contributed an hp-Adaptive method that adjusts griding on the fly, even for systems where the shape of the solution is not known [21]. Finite element "hp" methods were adapted, where h refers to the segment width and p denotes the order of the polynomial degree in each segment. Recalling that the dynamics of the states and controls are only enforced at the collocation points, Darby calculates the same collocation constraint (the derivative of the approximating polynomial must match the derivative supplied from the dynamics), but the constraint is evaluated *between* collocation points, forming a matrix of midpoint residuals. Oversimplifying, if a single residual is high, a discontinuity is suspected and a segment break is added for the next iteration. If many residuals are high, a poor polynomial fit is assumed and p is increased.

This method was adopted for the real-time controller in this project. Accomplishing collocation in this manner allows fewer nodes to be used in attaining the initial solution, without fear of missing important characteristics in the optimal path, as differences between nodes will be checked. Fewer nodes translates to a less complex NLP, solved with a greater speed. While more solution iterations are required, each iteration "bootstraps" the guess from its predecessor, greatly aiding convergence.

2.2.3 Real-Time Implementation Methods.

Recent efforts to apply optimal control in real-time are increasing. In cases where a feedback law (LQR, LQG, etc.) cannot be formed, a partial solution can be used for some simple problems. Kalmár-Nagy found that for a simple minimum-time TPBVP, knowing the structure of the solution (bang-bang in this case [75]) can sometimes offer relationships that must be held constant, producing a "near optimal" problem with greatly reduced order that can be solved quickly—either completely open-loop, or partially closed [61].

Benson recognized another potential technique using the Gaussian pseudospectral method for real-time control [4]. His novel idea hinged on the recognition of the availability of an accurate costate from the method, particularly the initial costate, even with a small number of nodes. Assuming the state, $\mathbf{x}$, dynamics, $\mathbf{f}$, time, t, costate, $\boldsymbol{\lambda}$, Hamiltonian, $\mathcal{H}$, control $\mathbf{u}$, and the set of admissible controls, $\mathbb{U}$, the relationships for the state and costate are found through the familiar first-order necessary conditions:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}\left(\mathbf{x}(t), \mathbf{u}(t), t\right) \tag{4}$$

$$\frac{d\boldsymbol{\lambda}}{dt} = -\frac{d\mathcal{H}^T}{d\mathbf{x}}\left(\mathbf{x}(t), \boldsymbol{\lambda}(t), \mathbf{u}(t), t\right) \tag{5}$$

Further, Pontryagin's maximum principle supplies the optimal control:

$$\mathbf{u}(t) = \arg \min_{\mathbf{u} \in \mathbb{U}} \left[\mathcal{H} \left(\mathbf{x}(t), \lambda(t), \mathbf{u}(t), t \right) \right] \tag{6}$$

Benson's unique concept is to use the pseudospectral method to determine the initial value for the costate. This value is combined with a measurement of the actual state to determine the current control with Equation 6. The value for control, along with the state measurement and the pseudospectral approximation for the initial costate, are used to propagate the time derivative of the costate with Equation 5, using a single step numerical technique such as a Runge-Kutta integration scheme [65]. As the costate is propagated, the control is continually updated with Equation 6. As disturbances and modeling errors alter the current state from the optimal trajectory, the optimal problem is re-solved using the current state as the initial condition to find a new estimate for the current (now the new initial) costate. The costate propagation is re-initialized with this value and the recursion continues. Of course, this solution assumes that the cost function is not time-varying.

Gong and I. M. Ross outline a different style of recursive feedback—certainly the most popular, and arguably the simplest. A real-time optimal trajectory planner produces an outer-loop reference trajectory as quickly as possible, while a linear or non-linear controller maintains the reference trajectory until the next update. The concept is that if the outer-loop reference trajectory can be calculated quickly enough, the inner loop can be removed [41, 95]. The comparison is made between simple sample and hold style discrete control and the forward-looking, open-loop solutions, repeatedly applied, referred to as "Carathéodory-π feedback" [95]. The conclusion is reached that with fast enough open-loop solutions, the search for a closed-loop feedback can be abandoned [10] (this conclusion will be challenged in Chapter VI). When the outer loop is not "fast enough," errors will occur in the initial conditions. An assumed initial condition is seeded to the trajectory planner (based on expectations from the last optimal plan), but disturbances, unknowns, and differing computation times will change the actual value of the state when the new optimal solution becomes available. Yan and Strizzi [119, 106] have implemented Bryson's neighboring optimal control law [13, 18] using an indirect method in an effort to correct for small deviations from the assumed conditions. This technique was replaced with cosine wave smoothing for this project.

The intent of this research is to build upon these efforts in the area of real-time optimal control. The developed methods will be applied to solving a classical estimation problem—trajectory optimization for bearing-only target analysis.

2.3 Trajectory Optimization

Trajectory modification for the purpose of localization and bearing-only tracking (BOT) has been implemented in the submarine community, at least at the heuristic level, for at least 60 years [104]. The ability to estimate range with only an angle sensor is intuitively dependent on the geometry from which the measurements are taken, as was shown in Figure 2 on page 4. Most efforts to increase the efficacy of the observer's trajectory on target state estimation have attempted to optimize a path based on control from two general categories—pure localization theory, and *dual control theory*, typically a suboptimal hybrid of estimation and optimal control. Both methods rely on the principles of pure localization, and trajectories are optimized based on some representation of target position information, such as the Fisher Information Matrix (FIM), the Cramér-Rao Lower Bound (CRLB), or an estimated error covariance. One limitation of these techniques is the loss of the full directional quality that should be guiding the trajectory when the information reference is compressed into a scalar performance index. There has been a great amount of effort invested in attempting to find out which cost functional is least affected by this limitation.

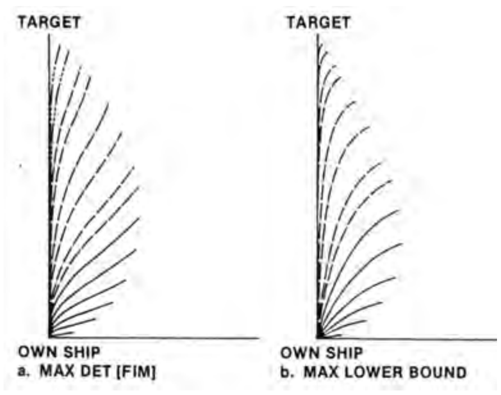
2.3.1 Localization and Bearing-only Tracking.

Lindgren [72], with Nardone and Aidala [82] laid some of BOT problem's foundational groundwork in the submarine context by developing criteria for observability. Efforts to increase observability began with "two leg" options, looking for a "lead-lag" trajectory [76], or fixing the heading for the initial leg and optimizing the heading for the second leg [29].

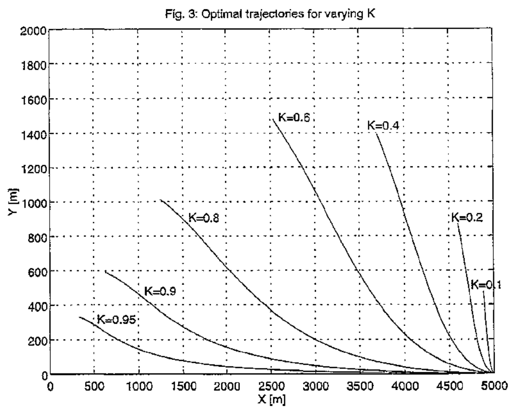
Hammel expanded on this [47, 45], pushed the BOT processing algorithms [46], and investigated the application to trajectory planning by maximizing an analytic approximation of the determinant of the FIM. The FIM provides a measure of the amount of information that is obtained from measurements, and is a function of the geometry of the problem, rather than the estimation method. Maximizing the determinant of the FIM effectively minimizes the volume of the uncertainty ellipsoid around the target position estimate.

Hammel's method for optimal control problem formulation became the standard approach—the continuous problem was parameterized, assuming the observer to have a constant speed and infinite heading-rate ability. A preset number of equal length, constant heading segments was then assumed, reducing the optimal control to a single sequence of headings to apply to the segments. Note that a constant velocity and fixed final time (indirectly assumed through a fixed number of equal duration segments) are common assumptions made in these techniques for tractability. This represents a major shortcoming—in effect, when solving for the optimal path, the sensitive parameters of path length and the number of measurements must be provided as assumed inputs, though they greatly change the nature of the solution. Figure 7 demonstrates this with plots from Hammel [44] and Oshman [84], where both VT/r_0 and K represent a required solution input parameter of the ratio of total path length to initial (unknown) range—essenti

observer. A ratio of one or greater r

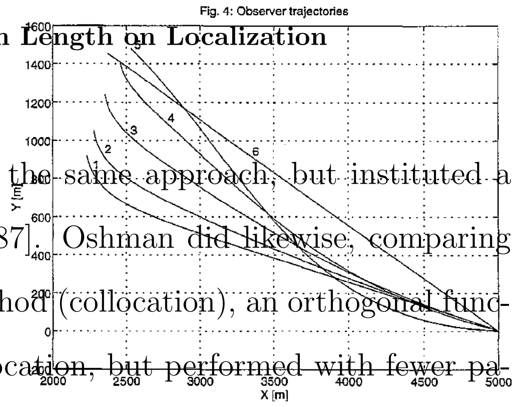


(a) Families of Solutions Varying $VT/$



(b) Families of Solutions Varying K [84]

Figure 7. Effect of Specifying Path Length on Localization



Passerieux followed Hammel with much of the same approach, but instituted a numeric solution for the actual optimization [87]. Oshman did likewise, comparing the optimization of a direct gradient-based method (collocation), an orthogonal function parameterization method (still direct collocation, but performed with fewer parameters by approximating the control vector with orthogonal basis functions), and with a differential inclusion method (removing control by replacing it with a state constraint, such as an equation for constant velocity) [85]. Liu used a suboptimal approach, analytically maximizing a lower bound on the determinant of the FIM, vice the determinant itself [73].

Faced with problems that stem from compression of the information metric into a scalar, Helferty moved away from the determinant of the FIM [50]. Minimizing the scalar uncertainty volume (or the area, for this particular 2-D case) was found to produce solutions that may favor highly eccentric confidence ellipsoids. This is especially problematic for localization problems, where the largest ambiguity axis often corresponds to the unknown range variable, where most of our attention is needed. Minimizing the trace of the CRLB was suggested instead. The CRLB is a lower bound on the error covariance of the estimation problem. It represents the best certainty attainable from measurements *along that path*, not necessarily what could be obtained by some other path, and by definition is the inverse of the FIM for an *efficient* estimator. With each eigenvalue corresponding to the square of one axis of the confidence ellipsoid, the trace (sum of the eigenvalues) yields the sum of the squares of each axis. Therefore, minimizing it penalizes solutions with a large axis of uncertainty resulting in less ambiguity of optimal solutions [49]. Logothetis developed a similar "mutual information metric," the maximization of which was equivalent to the minimization of the CRLB determinant [74].

The trace of the FIM has at times been selected as the metric of choice and efficient to calculate, but has also been shown to be unstable and potentially singular [88]. Le Cadre created an approximation of the FIM that was additively monotonic, and then took the trace of the approximation [67]. He later followed the concept, allowing for maneuver of the target using a hidden Markov model (HMM), and determined the optimal heading sequence with classical dynamic programming [68]. More recently, Per Skoglar used a steepest descent method for the optimization and a particle filter for the estimation. For the Gaussian case, he showed that trajectory planning with the determinant of the FIM was equivalent to using the differential entropy of the posterior target density [102]. In the context of multiple robots using Model Predictive Control (MPC), Leung chose to maximize the minimum eigenvalue of the FIM for localization [70].

Ponda compared solutions using several of the most popular FIM metrics (determinant, maximum eigenvalue of the inverse, negative trace, and trace of the inverse) in the context of the same problem—determining the location of a ground target optically with a sUAS, allowing 100 measurements in a fixed path length [88]. Unsurprisingly, the determinant of the FIM was found to no longer contain information about the angular dependence between the measurements (compression). Maximizing the trace of the FIM was better, and avoided some local minimum problems along a single path, but found to be unstable and have the potential to result in a singularity. The largest eigenvalue of the inverse of the FIM (minimizing the largest axis of the uncertainty ellipsoid), and the trace of the inverse of the FIM (minimizing the average variance of the estimates) yielded similar results, with faster convergence and higher stability in the optimization. The final metric was preferred. In simulation, Ponda found that increasing the allowed number of measurements led to a growing number of local extrema with severe sensitivities to initialization. As must often be done in the world of optimization, impractical results were avoided by initializing the optimization close to the global minimum, which, of course, is problematic for real applications.

Note that a common thread in all of these cases is that a scalar approximation of the information metric is the cost functional that is optimized. Regardless of which particular metric is used, all of them suffer from the loss of some directional information when a scalar is produced from a multi-dimensional information matrix. The effort to minimize this unavoidable effect is one reason for the variety of approaches. Another common theme is that bearing-only tracking and localization techniques select guidance purely for better estimation of the target location. The actual path that is selected is of no consequence, excepting that the path must be restricted from reaching the target, else the optimal information gathering technique becomes collision (information from bearing measurements will be shown to be inversely proportional to the square of range). The UAV scenarios accomplish this by mandating a fixed, planar altitude above the target and optimizing over a receding horizon, and the submarine and robot scenarios typically choose a fixed final time indirectly, short of that required to reach the target of interest. Unfortunately, such solutions are highly dependent on the time horizon selected, making "optimality" more of a mathematical construct than a practical reality.

2.3.2 Dual Control Theory.

The previous references are examples of optimizing a trajectory to increase the quality of estimation, without concern for the actual direction of the path. The converse can be seen in optimal problems that still seek to estimate the target location, but without reference to the geometric effect of the path. The focus may be simply on "camera-on-target" time or homing, as solved with several methods, such as direct collocation [38, 97], neural networks [37], or heuristics [99, 118, 23].

As a real-time example, in [38], Geiger designed a controller to solve for a string of waypoints that would enable a sUAS to maximize time above a target with a known position. The technique was rooted in work by Dickmanns [22], with equally spaced nodes, Hermitian interpolation, and a receding horizon approach. The solution shape was to fly directly to the target, then to perform a maximum rate turn back around, forming a cloverleaf pattern after multiple passes. To achieve the fixed 4 second update interval, only 7 nodes were used with a short 20 second "look ahead" time for the receding horizon (about enough time for one turn). This represents an important step in real-time optimal control, but does not account for the geometric effects of the path on target estimation quality.

The work herein addresses the problem of accomplishing both efforts simultaneously. Localization is critical, but so are the path characteristics—with the path being primary. The submarine example from the introduction concerns tracking a contact, but the primary mission is often moving into position to employ ordnance. Firing a torpedo is the *mission*, target position certainty is a *requirement*. In the same way, the HARM missile seeks not only to localize its target, but to hit it. The sUAS must maneuver to localize a power line, but the real mission is to land.

This type of problem is by definition non-holonomic—achieving the final state is the key, but that state is dependent on the path taken to achieve it. The control and estimation concepts are fundamentally coupled and *inseparable* [79]. The system has two purposes that may directly conflict with each other, but both are necessary—the quality of estimation affects the quality of control and vice versa [64]. This is addressed with so-called *dual control* or *dual effects theory* [30]. Dynamic programming and search-based approaches are the general solution techniques, but are commonly prohibitive even for small problems [64, 66, 13].

Frew addressed a similar problem to this work with exhaustive search. In guiding a robot with an angle sensor, the problem was again parameterized to find a heading sequence, but in this case, a particular final covariance was able to be achieved. This was not done in the optimal control formulation (a contribution of this dissertation), but by considering the outcomes of a generated acceptable set of paths. For tractability, only five turn-and-drive segments were allowed with turns restricted to one of five directions (45° apart initially, see Figure 8b for an example of three steps with 20° spacing). The final covariance in the target estimate was then calculated using a measurement at each step for each of the 3,125 paths.

Four total iterations were performed, the latter three centered around the best path of the previous run, with the space between allowable angles decreasing each time. The number of segments used became the cost function (options being integers 1-5, representing the minimum time solution). The first path calculated that obtained the required final covariance was declared the best path, because any additional paths

Depending on the trajectory-design objective, the maneuver duration is specified in one of two ways. For the case when minimum uncertainty is desired in fixed time, the total trajectory duration and number of maneuvers are specified. In this case the maneuver duration is just $T_{man} = T_{total}/n$ where n is the number of maneuvers. For the fixed-accuracy scenario, the maneuver duration is fixed and the number of

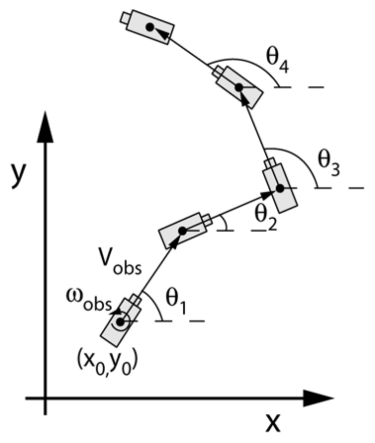


Figure 4.2 Trajectory parameterization

(a) Motion with Five Segments

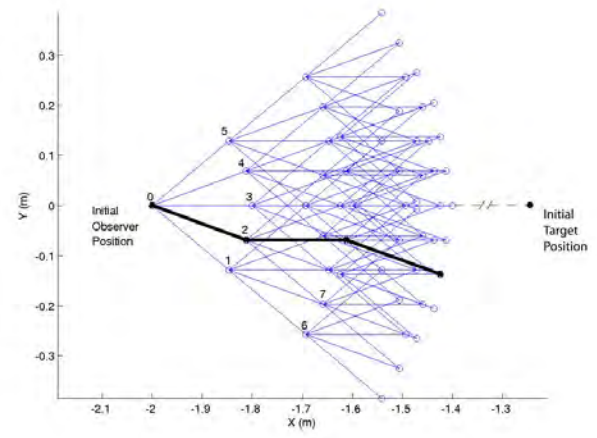


Figure 4.5 Breadth-first expansion from initial observer position using five possible heading values

(b) Possible Paths with 20° Heading Separation

The breadth-first expansion is shown in Figure 4.5. In this example the range of possible heading values is restricted to

$$S^i_{heading} = \{-40°, -20°, 0°, 20°, 40°\} \quad (4.24)$$

where the superscript i indicates the set applies to the ith maneuver.

The expansion is breadth first, so the points labelled 1-5 are all generated from the initial position. Once they are generated, point 1 is selected and points 6 and 7 are generated. This process is continued for all directions from point 1, then from point 2, then 3, etc. The lines that originated at the initial observer position are said to be at level or depth 1 and represent the possible positions of the observer after one maneuver. The lines originating from the endpoints of the level 1 maneuvers are referred to as level 2 and the convention continues for all levels. All maneuvers at a given level occur the same time away from the original node. Because the target motion is predicted based only on the current target estimate, the predicted target location that corresponds with the observer position at a given node is the same for every node at a given level. In other words, the target is predicted to move to the same location regardless of whether the

Figure 8. Optimization of a Robot Path by Exhaustive Search [33]

on that round that also met the final covariance requirement (and many likely would) could at best only tie in terms of the number of segments taken. Obviously, dimensionality fast becomes an issue, and every additional segment allowed increases the required number of expected covariance calculations exponentially.

Other authors, attempting to make the problem tractable, and sometimes analytically solvable, have split the dimensions in which control is optimized for path guidance and estimation improvement [57, 113]. Because the true problem is inseparable, this assumption fundamentally changes the nature of the solution, and the results can only be suboptimal. For the 2-D problem, control in one dimension is typically mandated, most often assuming a constant closure in the direction of the target for a known final time. Motion in an orthogonal direction is then solved for as a one-dimensional pure localization problem.

In [57], Johnson worked towards a solution that could be used in real-time, using simplifications for an analytic solution to guide a formation partner from one position to another, relative to the flight lead, using optical information to better discern the given position of the flight lead. With constant speed and heading, the problem is the same as static localization. For control, each axis was treated independently. Altitude was held constant. Relative velocity in the X-direction was also constant (an approximation of aircraft velocity difference for small heading crossing angles). The initial distance was assumed known, and direct force was used for control. Measurement value was equated with distance from a centerline. With these assumptions, all that remained was a one-dimensional TPBVP with no constraints, a known final time, and a linear system with two states—lateral position and lateral velocity.

An LQR technique was used to solve the problem analytically, with one cost term to penalize distance from the $Y = 0$ centerline, and another term to encourage it for observability. Figure 9 shows the result, with an aircraft being directed from an initial position of $X = 100$, $Y = 5$, to a final position at $(0, 0)$.

JOHNSON, CALISE, WATANABE, HA, AND NEIDHOEFER

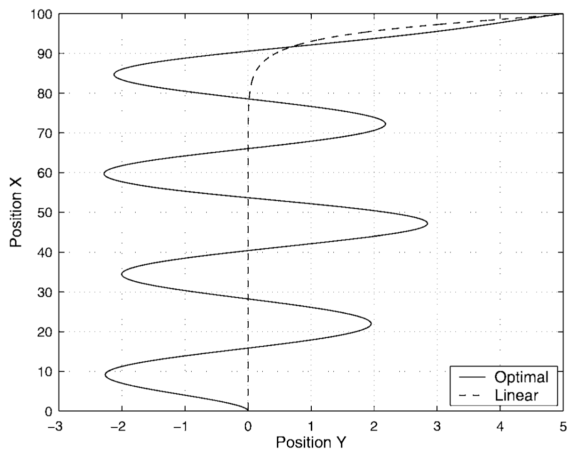


Fig. 7 Vehicle trajectory.

Figure 9. Analytic Dual-Control Solution Achieved by Isolating Each Dimension [57]

In this simulation, the follower aircraft changes its relative position from [$\Delta x = 100$ ft, $\Delta y = 10$ ft, $\Delta z = 0$ ft] to [$\Delta x = 50$ ft, $\Delta y = 0$ ft, $\Delta z = 0$ ft]. In one case (Fig. 11), the relative position command is given as a step command at a time of 20 seconds. In the other case (Fig. 12), the optimal path given by Equation (23) is utilized as the command from 20 seconds to a time of final time 60 seconds. Without the optimal guidance policy, there remains a steady

Bishop had a variation of this dimension-separating concept, shown in Figure 10. A constant decrease in range was assumed for each time step, but it was not tied to a direction. Localization was optimized to find the best location for that step (no future consideration), allowing instant motion to any location in two-dimensions on ever shrinking concentric circles until reaching the target [9].

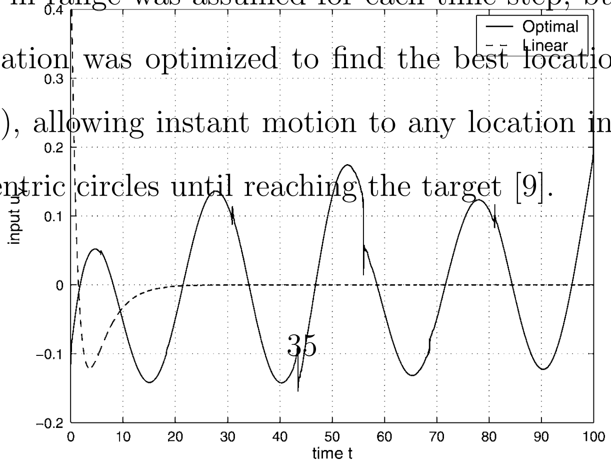


Fig. 8 Control input u_Y.

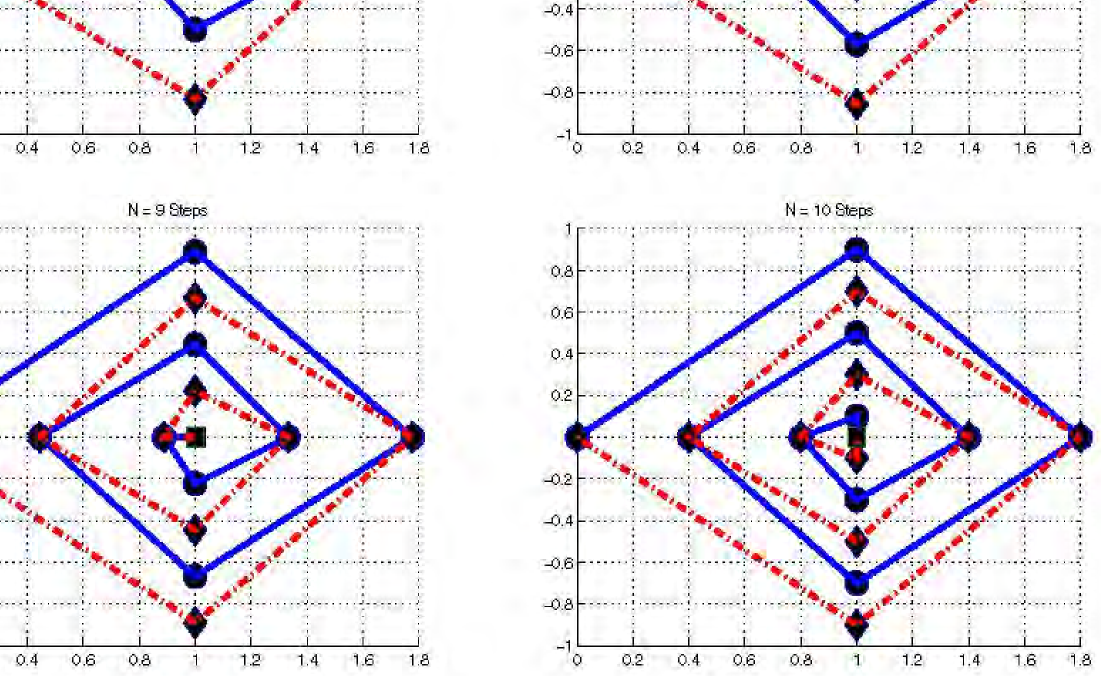


Figure 10. Optimal Pursuer Trajectory with Constant Range Decrease for Each Step [9]

gation with $N \in [5, 10]$. At each step k the pursuer considers the bearing history up to and

a better
tion (in
than the
raint on

KS

erization
and one-
ing-only
ay so as
lost and
ticularly
xamples

explicit
rize op-
und for
interest-
sults on
essence,
efficient
ever, the
e biased
nsidered

Much like Johnson [57], but without treating control and estimation efforts completely independently, Kim assumes a constant velocity toward the target, and then suboptimally adds control and estimation efforts with a weighted feedback [64]:

$$u = K_x\hat{\mathbf{x}} + K_y\mathbf{y} \tag{7}$$

The normal Linear Quadratic Gaussian (LQG) technique is used with an LQR gain, K_x, on the estimated state feedback, driving the system state to zero, while the second term feeds back the current covariance to "nudge" the system away from zero to increase the observability, as also explored in homing missile guidance research [103, 53].

There are other techniques for determining the amount of "nudge" to add to the fixed-final time LQR solution, such as the one-step-ahead method [74, 114], which finds the input of control that would result in the greatest decrease of uncertainty in the target position in the next one step—assuming the next measurement will be the last. This leads to a more optimal next step, but does not translate into achieving the optimal path overall. Watanabe extended this to consider N steps ahead, but

REFERENCES

[1] C.F. Lin. *Modern Navigation, Guidance and Control Processing - Vol. II*. Prentice Hall, Englewood Cliffs, NJ, 1991.

[2] P. Zarchan. *Tactical and Strategic Missile Guidance*. AIAA, Inc., Washington, DC, 1994.

[3] J.Z. Ben-Asher and I. Yaesh. *Advances in Missile Guidance Theory*. AIAA, Inc., Washington, DC, 1998.

[4] B.L. Stevens and F.L. Lewis. *Aircraft Control and Simulation*. John Wiley and Sons, Inc., New York, NY, 1992.

[5] A.V. Savkin, P.N. Pathirana, and F. Faruqi. The problem of precision missile guidance: LQR and H^∞ control frameworks. *IEEE Transactions on Aerospace and Electronic Systems*, 39(3):901–910, 2003.

[6] S.C. Nardone, A.G. Lindgren, and K.F. Gong. Fundamental properties and performance of conventional bearings-only target motion analysis. *IEEE Transactions on Automatic Control*, AC-29(9):775–787, 1984.

[7] M. Gavish and A.J. Weiss. Performance analysis of bearing-only target location algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 28(3):817–827, 1992.

[8] E. Fogel and M. Gavish. N^{th}-order dynamics target observability from angle measurements. *IEEE Transactions on Aerospace and Electronic Systems*, 24(3):305–308, 1988.

[9] C. Jauffret and D. Pillon. Observability in passive target motion analysis. *IEEE Transactions on Aerospace and Electronic Systems*, 32(4):1290–1300, October 1996.

[10] T.L. Song. Observability of target tracking with bearings-only measurement. *IEEE Transactions on Aerospace and Electronic Systems*, 32(4):1468–1472, October 1996.

[11] I. Kadar. Optimum geometry selection for sensor fusion. In *Proceedings of SPIE Conference on Signal Processing, Sensors Fusion and Target Recognition VII*, pages 96–107, Orlando, FL, 1998.

[12] A.G. Dempster. Dilution of precision in angle-of-arrival positioning systems. *Electronics Letters*, 42(5), 2006.

[13] H.L. Van Trees. *Detection, Estimation and Modulation Theory - Part*

concluded that the one or two-steps-ahead solutions would be helpful for an on-line system, but not necessarily close to the optimal solution. Using more than one or two steps ahead was not practically implementable [115].

Hodgson used differential inclusion to solve for a missile path that, for a fixed dwell time, would balance a cross range resolution term from an imaging radar with the determinant of the CRLB [52]. The method still has the limitations of dependence on arbitrary weights and loss of directional information through compression to a scalar, but provides a variation on needing a fixed final time by using a range-to-go as the independent variable. This is appropriate for systems that cannot turn directly orthogonal to the target, and have a small variation in range-to-go rate (else the measurement update interval becomes a function of the path length and direction, as a fixed number of measurements must still be declared).

2.3.3 Trajectory Optimization Shortcomings.

Though an extensive body of work exists in the field of trajectory optimization with a bearing-only sensor, there are areas which still need to be addressed. Regardless of the metric selected, all of the methods suffer from compression when trying to characterize the directional certainty about a point with a scalar. Second, there is no method of dual control that does not at some level depend on an arbitrary weighting balance between control and estimation. Further, these methods require pre-declaration of some variables (final time, path length, number of maneuvers, and/or number of measurements) that the solution is sensitive to.

Perhaps most importantly, there does not exist a practical method, implementable in real-time, for achieving a *particular* final covariance (Frew did this in a pure localization sense without consideration for control, but exhaustive search is not feasible for real-time work at this point). This is a major stumbling block for actual use of these methods, beyond simulation. Current methods provide guidance to get "the best estimate possible in a given time/path length/number of measurements," or provide "more" information based on a weighting scheme with current covariance. This is not feasible for real-world applications which operate, even stochastically, in reference to measurable, physical limitations.

In the submarine attack example, deviations from a direct path to the target will increase the fidelity of a target estimate, but also take precious time and could cost the first shot. Maneuvers should be kept at the minimum necessary for a valid fire-control solution—physical requirements based on the torpedo capabilities and friendly-fire clearance limitations. For the HARM example, the target estimate needs to be of a quality to ensure the desired effects, based on real ranges of circular-error-probable miss distance and effective blast radius. Maneuvers beyond this deplete energy for the critical end-game maneuvering. For the sUAS studied in this work, the physical drivers of the problem are the ability of the aircraft to accurately reach a commanded point, and the physical dimensions of the attachment apparatus used to connect to the wire.

III. Problem Description and Modeling

CONSIDER the autonomous control of a sUAS for surveillance and other missions. Completely autonomous UAS control for surveillance missions is still an on-the-horizon capability, requiring a combination of several technologies, some still relatively immature. Decision making, mission definition and accomplishment, target identification and measurement, obstacle avoidance, and long-range communication of surveillance data are not addressed here. The scope of this dissertation deals with a small part of the overall mission—energy harvesting from a power line. Short range and limited station times are active constraints on the usefulness of our small and micro-UASs. Both could be greatly extended through the ability to recharge batteries. Conceptually, a small group of sUASs could be sent for surveillance of the same target. With two recharging on a nearby power line and a third in the air, continuous coverage could be provided without operator input.

The concept of energy harvesting through induction is not new, and the use of power mats and such for cordless devices is becoming commonplace. The most efficient method is to place a clamp around a source, as done with an an inductive ring around a spark plug wire for an old timing light. Getting an inductive clamp down to a light enough weight realistic for small vehicles, yet effective enough to charge in a reasonable time without arcing problems is a current topic of research at Defense Research Associates (DRA).

3.1 Segmentation of Control Modes

The process for landing on a power line will require several segments, where the goals and methods of control change as milestones are accomplished. The minimum number of control mode switches include an *acquisition* segment, where control is provided to reach a position likely to pick up the power line in a sensor, an *approach* segment, where control is determined by the observability needs to accurately estimate the wire's relative positive while guiding to an offset approach point, and a *flare* segment, where control is provided to perform a maneuver that will safely attach to the wire from known flight conditions and offset. The concept is illustrated in Figure 11.

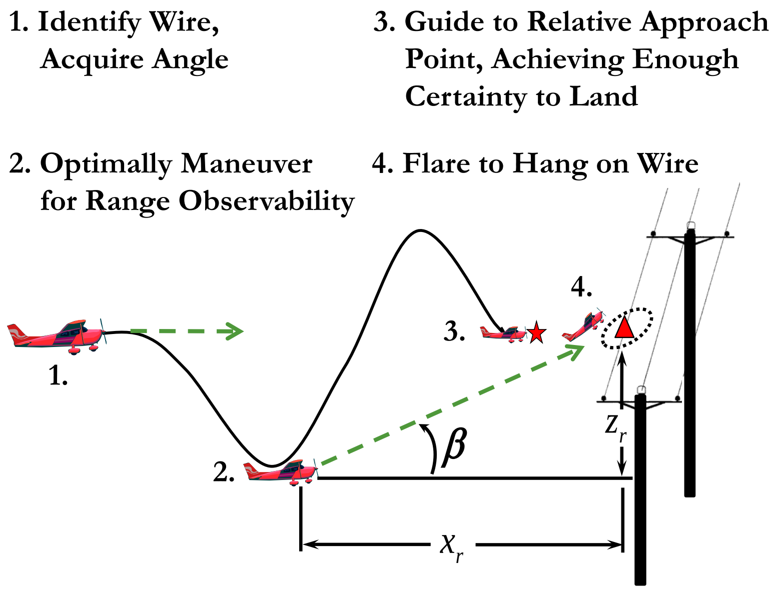

1. Identify Wire, Acquire Angle
3. Guide to Relative Approach Point, Achieving Enough Certainty to Land
2. Optimally Maneuver for Range Observability
4. Flare to Hang on Wire
1.
2.
3.
4.
β
z_r
x_r

Figure 11. Conceptual Approach and Flare Segments

The acquisition segment is within our current capability. It is assumed that the vehicle has navigational awareness through GPS, INS, optical flow, or some other capacity. This includes having a rough knowledge of power line locations, available on local maps or from imagery. While certainly not accurate enough to land with, this is sufficient to find a power line by maneuvering to a position orthogonal to the wire. Identification of the line can be accomplished with a feature extraction algorithm, such as a fast Hough transform, operating on sequential images. The images can be collected from a device such as simple webcam, available on many of the smaller UASs.

This work addresses the approach segment—beginning with an initial measurement of angle to the wire, and ending at an approach point with the prescribed states necessary to begin a flare-to-land maneuver, such as relative distance, relative height, heading, speed, and other requirements for specific systems. Since the approach point is defined relative to the power line's true location, it must be estimated to a quality likely to end in a successful flare prior to arrival.

The actual flare segment is currently being investigated by several institutions for fixed-wing sUASs. In [19], a fixed-wing glider was perched on a wire using an aggressive flare maneuver from both 2.5-m and 1.5-m approach points, using full information about the location of the wire. The approach point in this work, x_{app}, was correspondingly set to 2-m.

Since the test platform for the algorithm was a helicopter vice a fixed-wing UAS, an aggressive flare segment was not required. The final condition in the optimal controller was simply set to slow to a hover by the time it reached the approach point. Once the final conditions are achieved, to include the minimum target position certainty, the RTOC control mode is switched off, and the helicopter flies directly to a perch point underneath the last known location of the wire, continuing to update its position until the wire is no longer in the field-of-view of the camera. As the vehicle approaches the perch point, it slows gently to a stop and descends to engage a hook on top of the vehicle.

3.2 Modeling for the Relative Position Problem

Full modeling for control of the real quadrotor involves 3-axis position and velocities, orientation angles and rates, engine states and lag estimates, control variables, and many other parameters in four reference frames. The necessary portions of the quadrotor and its flight controls are described in Chapter VIII. For consideration of only the relative estimation problem and the optimal control portions of the problem, however, the model can be greatly simplified.

Body frame coordinates, $\mathbf{x}_b = (x_b,\, y_b,\, z_b) \in \mathbb{R}^3$, are defined on the quadrotor with the origin at the center of gravity (cg), the positive x_b-axis direction pointing out of the camera (referred to as the "nose" of the vehicle), y_b-axis positive out of the "right wing", and z_b-axis positive up (non-standard, left-handed system for readability of later plots), as shown in Figure 12.

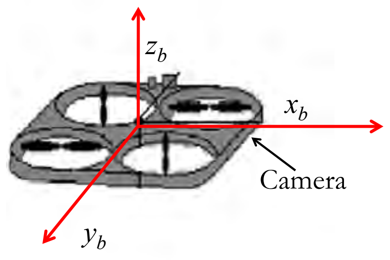


Figure 12. Body Axis Frame

Though the estimation may be performed in purely relative terms, reference to the inertial frame must be maintained to avoid constraints, be they aerodynamic limitations (maximum altitude), physical considerations (terrain, walls), or tactical limits (political borders, assigned airspace). A navigation frame is defined, anchored inertially, with the x-axis parallel to an assumed flat Earth and positive in the shortest direction to the power line from the point at which the first measurement is received. The y-axis is defined orthogonally, parallel to the Earth and positive in the same direction as the y_b-axis at initialization (all Euler angles zero). The z-axis is again defined positive up (non-standard) for convenience. For the actual flight test, the origin of the navigation frame was at the center of the indoor flight test facility.

As detailed in Chapter I, the power line is modeled as horizontal, with the angle along the wire unobservable. This leads to a planar problem, with maneuvering in the vertical to increase observability. For simplicity, all reference to the y-axis is omitted from mention, except when necessary in the discussion of flight control. During flight test, the vehicle is directed to $y = 0$ prior to the first measurement, and is regulated to zero during the run. The inertial position coordinate vector, $\mathbf{x} \in \mathbb{R}^{\mathbf{2}}$, is then defined as:

$$\mathbf{x}(t) \equiv \begin{bmatrix} x(t) & z(t) \end{bmatrix}^T \tag{8}$$

Velocity is likewise defined in the planar navigation frame:

$$\mathbf{v}(t) \equiv \begin{bmatrix} v_x(t) \\ v_z(t) \end{bmatrix} = \frac{d\mathbf{x}(t)}{dt} \tag{9}$$

An upper total velocity limit, $v_x^2 + v_z^2 \leq v_{max}^2$, was imposed (no minimum speed required for a helicopter), but in the manner controls were actually applied to the quadrotor, individual limitations of $|v_x| \leq v_{x_{max}}$ and $|v_z| \leq v_{z_{max}}$ became more restrictive.

The true target coordinates are $(x_t,\ z_t)$, and estimates are denoted with the hat symbol, as in $\hat{x}_t$. For notational convenience, a vector of relative coordinates between the target and the vehicle is defined using the convention shown in Figure 13.

$$\mathbf{x}_r(t) \equiv \begin{bmatrix} x_r(t) \\ z_r(t) \end{bmatrix} = \begin{bmatrix} x_t - x(t) \\ z_t - z(t) \end{bmatrix} \tag{10}$$

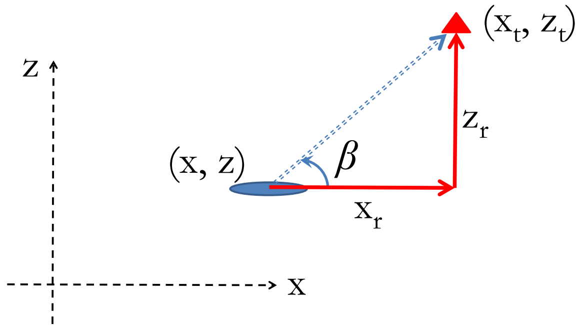


Figure 13. Relative Cartesian Formulation

The measurement angle, β, is received by a camera mounted in the nose of the quadrotor, and defined positive up from the horizon. The camera is fixed in position and orientation relative to the cg of the vehicle. With no required lateral motion, the bank angle, ϕ, and heading angle, ψ are regulated to zero. The measurement angle is then considered to be vertical (or corrected to vertical) from the level inertial frame:

$$\mathbf{h}\left[\mathbf{x}(t)\right] \equiv \beta(t) = \tan^{-1}\left[\frac{z_r(t)}{x_t(t)}\right] \tag{11}$$

The function symbol $\tan^{-1}$ refers to the full quadrant arctangent throughout this dissertation. It should be noted that the measurement angle is a combination of the image angle, β_{image}, produced from a pixel count in a known FOV, with the deck pitch angle, θ, as shown in Figure 14. Because the calculation of the image angle causes some delay, it is critical that the images be time tagged and correlated with a short history of pitch angle measurements.

$$\beta(t) = \beta_{image}(t) + \theta(t) \tag{12}$$

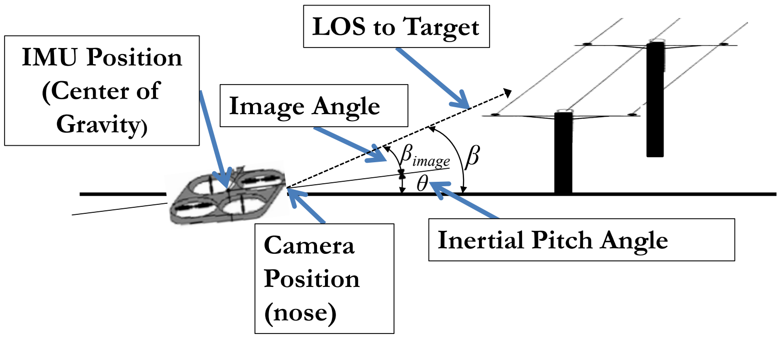


Figure 14. Correction of Deck Pitch Angle for Inertial Measurement

For estimation purposes, β is measured at discrete times, $t_k \in [t_0, t_f]$, and is modeled as an independent random variable:

$$\xi_k \equiv \mathbf{h}(\mathbf{x}_k) + \eta_k \tag{13}$$

where $\{\eta_k\}_{k=1}^{n}$ is a zero-mean, Gaussian, white noise sequence with a constant covariance:

$$\begin{aligned} E[\eta_k] &= 0 \\ E[\eta_k \eta_j^T] &= R\delta_{kj} \end{aligned} \tag{14}$$

with δ_{kj} representing the Kronecker delta function. The added noise models the combined uncertainty in the measurement from errors in the line detection algorithm and errors in the estimate of the current pitch angle.

For actual implementation, it is important to consider the fact that the cg of the vehicle is not likely to be collocated with the bearing sensor. In this case, the optimal trajectory planning is really a *sensor positioning* algorithm and the optimal path solved for is really the optimal path of the *camera*. If significant, the effects of the transformation must be considered on the constraints and the control, with the appropriate transformation. In this case, assuming a fixed camera lever arm in the body frame, $(r_{cam_x},\ r_{cam_z})$, a direction cosine matrix (DCM) is used:

$$\begin{bmatrix} x_{cam_b}(t) \\ z_{cam_b}(t) \end{bmatrix} \equiv \begin{bmatrix} x_b(t) + r_{cam_x} \\ z_b(t) + r_{cam_z} \end{bmatrix} \tag{15}$$

$\Rightarrow$

$$\begin{bmatrix} x_{cam}(t) \\ z_{cam}(t) \end{bmatrix} = \begin{bmatrix} x(t) \\ z(t) \end{bmatrix} + \begin{bmatrix} \cos\theta(t) & -\sin\theta(t) \\ \sin\theta(t) & \cos\theta(t) \end{bmatrix} \begin{bmatrix} r_{cam_x} \\ r_{cam_z} \end{bmatrix} \tag{16}$$

Control is modeled after the actual quadrotor, which uses the advantages of a helicopter to decouple vertical and horizontal components:

$$\mathbf{u}(t) \equiv \begin{bmatrix} u_x(t) \\ u_z(t) \end{bmatrix} = \frac{d\mathbf{v}(t)}{dt} \tag{17}$$

limited by $|u_x| \leq \left(\frac{dv_x}{dt}\right)_{max}$ and $\left(\frac{dv_z}{dt}\right)_{min} \leq u_z \leq \left(\frac{dv_z}{dt}\right)_{max}$, with gravity causing a difference in vertical acceleration capability. This model is limited by two factors. A helicopter near maximum performance cannot accelerate upward and forward at maximum rates simultaneously. In addition, the real equations of motion have more lag caused by additional integration steps in horizontal acceleration. The true control signal is a differential RPM on the motors. The corresponding lift difference changes the pitch or bank angle, which then causes horizontal acceleration. For the slow speeds and very low bank angles of the quadrotor in the indoor flight test facility, however, this model was sufficient for outer loop trajectory planning. For an example of backing out controls down to the servo level from optimal trajectories, see [117].

For actual propagation and use in the own-ship position Kalman Filter, the velocity and acceleration were assumed constant over a time step, and the discrete-time state equation was used:

$$\mathbf{x}_{k+1}^{(KF)} \equiv \begin{bmatrix} x_{k+1} \\ z_{k+1} \\ v_{x_{k+1}} \\ v_{z_{k+1}} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_2 & \mathbf{I}_2\Delta t \\ \mathbf{0}_2 & \mathbf{I}_2 \end{bmatrix} \mathbf{x}_k^{(KF)} + \begin{bmatrix} \mathbf{0}_2 \\ \mathbf{I}_2\Delta t \end{bmatrix} \mathbf{u}_k + \mathbf{w}_k \tag{18}$$

with $E\left[\mathbf{w}_k\right] = \mathbf{0}$ and $E\left[\mathbf{w}_k\mathbf{w}_i^T\right] = \mathbf{Q}_k\delta_{ik}$.

3.3 Transformation to Polar Coordinates

The final flight test version of the software developed in this project was implemented in the Cartesian frame. Much of the research, however, was accomplished using a polar coordinate transformation. This is still recommended for some scenarios, as will be discussed in Chapter IV. In this dissertation, the polar coordinate system is non-standard, with the origin at the estimated target position, as shown in Figure 15, defining $\rho(t)$ positive for the current range, and the polarity of β opposite of the tradi

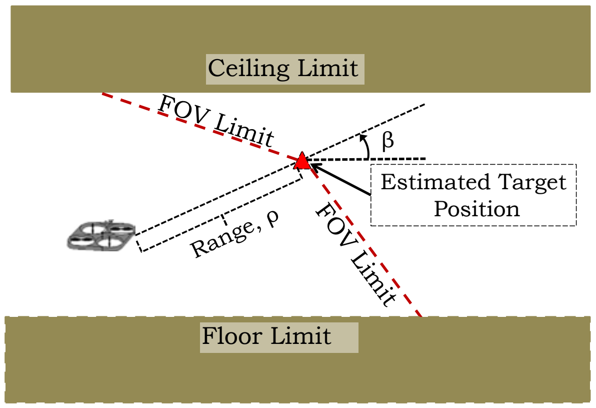


Figure 15. Polar Formulation

This formulation allows several advantages, and is recommended for similar research that has fewer position constraints, such as the submarine problem, and sensor systems capable of using angular rate, $\dot{\beta}$, in the cost function or constraints. The advantage to the Cartesian system is fast propagation of the linear dynamics, at the cost of a non-linear measurement function. The polar system, defined as:

$$\mathbf{y}(t) = \begin{bmatrix} \rho(t) \\ \beta(t) \end{bmatrix} \tag{19}$$

has a linear measurement function:

$$\mathbf{H}_y = \begin{bmatrix} 0 & 1 \end{bmatrix} \mathbf{y}(t) \tag{20}$$

The linear measurement function will aid in accurate measurement updates to the target estimate, but typically at the cost of a non-linear dynamics function. However, if control is applied in the form of radial acceleration, $\ddot{\rho}(t)$, defined as positive away from the target, and tangential acceleration, $\ddot{\beta}(t)$, defined positive clockwise:

$$\mathbf{y}^{(KF)}(t) = \begin{bmatrix} \rho(t) \\ \beta(t) \\ \dot{\rho}(t) \\ \dot{\beta}(t) \end{bmatrix} \qquad \mathbf{u}_{\mathbf{y}}(t) = \begin{bmatrix} \ddot{\rho}(t) \\ \ddot{\beta}(t) \end{bmatrix} \tag{21}$$

The dynamics can then be represented as fully linear and time invariant:

$$\dot{\mathbf{y}}^{(KF)}(t) = \begin{bmatrix} \mathbf{0}_2 & \mathbf{I}_2 \\ \mathbf{0}_2 & \mathbf{0}_2 \end{bmatrix} \mathbf{y}^{(KF)}(t) + \begin{bmatrix} \mathbf{0}_2 \\ \mathbf{I}_2 \end{bmatrix} \mathbf{u}_{\mathbf{y}}(t) \tag{22}$$

Approaching the model in this manner pushes all of the non-linearities into the determination of the initial conditions for every epoch of the path planner, and into the constraints. Through the constraints, the control of the helicopter can be appropriately scheduled. Noting that $\mathbf{v} = -d\mathbf{x}_r/dt$, the initial conditions can be found with:

$$\mathbf{y}^{(KF)}(t_0) = \begin{bmatrix} \sqrt{x_r^2(t_0) + z_r^2(t_0)} & m \\ \tan^{-1}\frac{z_r(t_0)}{x_r(t_0)} & rad \\ \frac{-x_r(t_0)v_x(t_0) - z_r(t_0)v_z(t_0)}{\sqrt{x_r^2(t_0)+z_r^2(t_0)}} & m/s \\ \frac{z_r(t_0)v_x(t_0) - x_r(t_0)v_z(t_0)}{x_r^2(t_0)+z_r^2(t_0)} & rad/s \end{bmatrix} \tag{23}$$

Position constraints (floor, ceiling) can be applied with the simple transformation:

$$(z_{min} - \hat{z}_t) \leq \rho(t)\sin\beta(t) \leq (z_{max} - \hat{z}_t) \tag{24}$$

Speed and acceleration constraints are developed with (dropping time dependency):

$$\begin{aligned} & x_r = \rho\cos\beta & & z_r = \rho\sin\beta \\ \Rightarrow\quad & \dot{x}_r = -\dot{\beta}\rho\sin\beta + \dot{\rho}\cos\beta & \Rightarrow\quad & \dot{z}_r = \dot{\beta}\rho\cos\beta + \dot{\rho}\sin\beta \\ \Rightarrow\quad & v_x = \dot{\beta}\rho\sin\beta - \dot{\rho}\cos\beta & \Rightarrow\quad & v_z = -\dot{\beta}\rho\cos\beta - \dot{\rho}\sin\beta \end{aligned} \tag{25}$$

Making total squared velocity:

$$\begin{aligned} {v_x}^2 + {v_z}^2 &= \dot{\beta}^2\rho^2\sin^2\beta - 2\dot{\beta}\dot{\rho}\rho\sin\beta\cos\beta + \dot{\rho}^2\cos^2\beta + \dot{\beta}^2\rho^2\cos^2\beta \\ &\qquad + 2\dot{\beta}\dot{\rho}\rho\cos\beta\sin\beta + \dot{\rho}^2\sin^2\beta \\ &= \dot{\rho}^2 + (\dot{\beta}\rho)^2 \end{aligned} \tag{26}$$

constrained with:

$$\Rightarrow v_{\min}^2 \leq \dot{\rho}^2 + (\dot{\beta}\rho)^2 \leq v_{max}^2 \tag{27}$$

The acceleration limitations on the vehicle are similarly treated. The horizontal acceleration capability of the quadrotor becomes limited by:

$$\dot{v}_x = \dot{\beta}^2 \rho \cos\beta + \left(\dot{\beta}\dot{\rho} + \rho\ddot{\beta}\right)\sin\beta + \dot{\beta}\dot{\rho}\sin\beta - \ddot{\rho}\cos\beta \tag{28}$$

$$\Rightarrow \left|\left(\rho\ddot{\beta} + 2\dot{\beta}\dot{\rho}\right)\sin\beta + \left(\dot{\beta}^2\rho - \ddot{\rho}\right)\cos\beta\right| \leq \left(\frac{dv_x}{dt}\right)_{max} \tag{29}$$

The vertical limitation is then transformed to:

$$\dot{v}_z = \dot{\beta}^2 \rho \sin\beta - \left(\dot{\beta}\dot{\rho} + \ddot{\beta}\rho\right)\cos\beta - \dot{\beta}\dot{\rho}\cos\beta - \ddot{\rho}\sin\beta \tag{30}$$

$$\Rightarrow \left(\frac{dv_z}{dt}\right)_{min} \leq \left(\dot{\beta}^2\rho - \ddot{\rho}\right)\sin\beta + \left(-\ddot{\beta}\rho - 2\dot{\beta}\dot{\rho}\right)\cos\beta \leq \left(\frac{dv_z}{dt}\right)_{max} \tag{31}$$

It is stressed that this method can be used for trajectory optimization even though the actual range to the target is not known. The current navigation estimate is provided to the path planner as if it were the actual target location. The constraints are valid because they are defined relative to that point in space, whether it ends up being the actual target location or not. In implementation, it was found that the linear measurement function was a strength for the estimation filter, but the significant non-linearities in the path constraints had potential to slow down the optimization (slightly). In an attempt to get the most from both worlds, a Hybrid EKF is developed in Chapter IV that can be used in some scenarios, as well as the UKF that was used in the final power line landing flight tests that validated the complete system.

IV. Bearing-only Estimation

Range estimation is clearly the core of the bearing-only analysis problem in this dissertation, and the concepts of range observability are central to the trajectory optimization problem as well. The fundamental non-linearity of the system has made this a classic relative estimation problem.

Historically, triangulation has been accomplished by solving a system of equations generated from a point-slope equation taken at each measurement position, with the target coordinates as unknowns, or by generating equations with the law of sines, and using the collection of ranges at each measurement as the unknowns. With two measurements, the answer for the estimate is exact (wrong, excepting perfect measurements, but exact). With more than two measurements, the solution is over determined. A matrix of equations is formed, and the estimate error is minimized (in the 2-norm sense) with a pseudo-inverse following the linear least squares method.

For many on-line applications, at least for linear systems, the Kalman Filter has become the industry standard—propagating a system forward based on known controls and modeled dynamics, estimating what the next measurement will be at that state, and applying a portion of the residual difference between the actual and expected measurements based on an optimal Kalman gain. Again, proper selection of the gain minimizes the errors in a least squares sense.

For non-linear systems, as mentioned in Chapter III, the coordinate representation selected can impact the ability to accurately estimate relative position. For the bearing-only estimation problem, sensor measurements can be made linear in a polar coordinate system, but the propagation of the system is linear in the Cartesian frame. Reference to a Cartesian navigation frame must be maintained for considerations such as inertial own-ship position estimation and ground avoidance, but understanding of the polar reference frame is also required to maintain the target within the camera FOV angle limits.

The Cartesian-polar conversion problem is almost ubiquitous in tracking and navigation applications, and many solution options have been created to minimize the effects of the non-linearity. The Extended Kalman Filter (EKF) [11] is a common approach, linearizing the measurement function by evaluating its Jacobian at the current estimated state, allowing the linear Kalman filter equations to be used. The EKF has significant limitations, however. Linearization of the measurement function is only as good as the current estimate, and errors will not only cause the state update to be biased, but will result in over-confidence in the covariance matrix. This ill-conditioning can cause the uncertainty estimates to collapse prematurely around bad state estimates, leading to instability of the filter. This is particularly prevalent when the degree of non-linearity is high, or when the initial estimates for mean and covariance are significantly off [1].

In the submarine context, the Modified Polar form introduced in [1] assists with this, especially at long ranges with low bearing rates. The drawback is that the filter still must deal with conversion to and from Cartesian coordinates to avoid the real-world navigation and dynamic constraints. Other options include least squares filters [105], maximum-likelihood techniques [36], particle filters [63], Gauss methods [45], pseudo-linear trackers [51], and many other debiasing techniques [69].

For this research, two main estimation filters were used—a Hybrid Extended Kalman Filter and an Unscented Kalman Filter. Both filters were found to be effective, with little difference in actual estimation performance. The ability to get an unbiased mean for a small amount of additional complexity, and the required shape of the final covariance ellipsoid were the primary drivers of the design decision to use the unscented filter. The UKF, in the form implemented, provides the expected final covariance estimates in terms of P_{xx} and P_{zz} (the respective diagonal elements of the covariance matrix), without using an undesirable extra non-linear conversion. For the particular implementation of the quadrotor, the shape of the hook used to attach to the wire necessitated that the wire's position be known to a particular level in the z-axis direction to enter the "mouth," and a particular level in the x-axis direction to know when to stop, and when to descend (see Figure 16). This made the UKF more desirable.

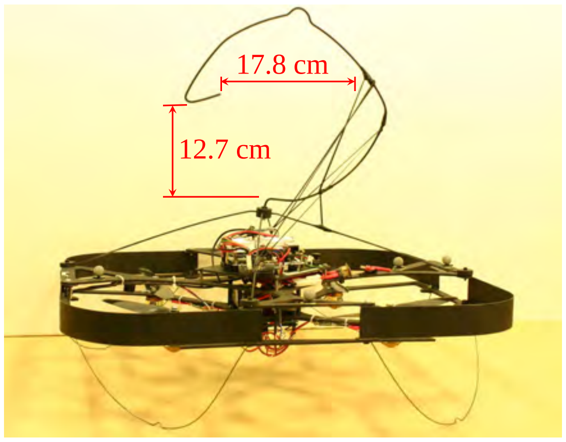


Figure 16. Quadrotor Hook Design

For cases where uncertainty in terms of range and angle is important, such as would be for the likely sUAS design of a device with a conical "mouth" to attach to a power line, the HEKF should be considered. For long range engagements with a slow bearing-rate, the modified polar form is recommended.

4.1 The Hybrid EKF

The HEKF is a variant of the EKF based on [1], but not using the modified polar form. It is a mid-point between using an EKF defined purely in either the Cartesian or polar frames, and variants of it have been used in cases such as this where the motion of the vehicle will be provided in one frame, but the measurement in another. The effect of the non-linearity is not magically bypassed—a transformation between the forms will still be made based on a faulty angle estimate. However, by applying the propagations that we know in inertial space, the measurements we know in polar space, and by tracking state error from a nominal condition vice the actual states, we can minimize the filter errors.

As the nominal trajectory will be provided by the path planner, there is no need to track the velocity or acceleration in the filter. The uncertainty in the position errors are assumed to have reached a steady-state (additive covariance), and the size of the errors are assumed to be small in relation to the errors in the angle measurements. Only the two relative states are then required, as defined in Equations 19 and 20, on page 48, and related by the one-to-one transformations:

$$\mathbf{x}_r(t) = \mathbf{s}_x\left[\mathbf{y}(t)\right] = \begin{bmatrix} y_1(t)\cos y_2(t) \\ y_1(t)\sin y_2(t) \end{bmatrix} \tag{32}$$

$$\mathbf{y}(t) = \mathbf{s}_y\left[\mathbf{x}_r(t)\right] = \begin{bmatrix} \sqrt{x_1^2(t) + x_2^2(t)} \\ \tan^{-1}\left[\frac{x_2(t)}{x_1(t)}\right] \end{bmatrix} \tag{33}$$

Assuming that the vehicle will move along a nominal path, we can add changes in Cartesian position over a time step, $\varsigma(t, t_0)$, for a discrete standard form of the dynamics:

$$\begin{aligned} \mathbf{x}_r(t) &= \mathbf{\Lambda}(t, t_0)\mathbf{x}_r(t_0) + \boldsymbol{\varsigma}(t, t_0) \\ &= \mathbf{\Lambda}(t, t_0)\mathbf{s}_x\left[\mathbf{y}(t_0)\right] + \boldsymbol{\varsigma}(t, t_0) \end{aligned} \tag{34}$$

For the linear, Cartesian case, $\mathbf{\Lambda}(t, t_0)$ is reduced to identity. Substituting Equation 34 into Equation 33 yields the propagation equation for our states in polar form:

$$\begin{aligned}\mathbf{y}(t) &= \mathbf{s}_y \left[\mathbf{\Lambda}(t, t_0)\mathbf{s}_x \left[\mathbf{y}(t_0)\right] + \boldsymbol{\varsigma}(t, t_0)\right] \\ &\equiv \mathbf{s}\left[\mathbf{y}(t_0); t, t_0\right] \end{aligned} \tag{35}$$

Following the Extended Kalman Filter derivation [77], and using the $x_{1k|k-1}$ to indicate the first element of $\mathbf{x}_r$ at time t_k using all available measurements through time t_{k-1}, we may express this in a discrete propagation:

$$\begin{aligned}\mathbf{y}_{k|k-1} &= \mathbf{s}_y \left[\mathbf{\Lambda}_{k,k-1}\mathbf{s}_x \left[\mathbf{y}_{k-1|k-1}\right] + \varsigma_{k,k-1}\right] \\ &= \begin{bmatrix} \sqrt{\left(s_{x1}\left[\mathbf{y}_{k-1|k-1}\right] + \varsigma_{1k,k-1}\right)^2 + \left(s_{x2}\left[\mathbf{y}_{k-1|k-1}\right] + \varsigma_{2k,k-1}\right)^2} \\ \tan^{-1}\left[\frac{s_{x2}[\mathbf{y}(k-1|k-1)] + \varsigma_2(k,k-1)}{s_{x1}[\mathbf{y}(k-1|k-1)] + \varsigma_1(k,k-1)}\right] \end{bmatrix} \\ &= \begin{bmatrix} \sqrt{x_{1k|k-1}^2 + x_{2k|k-1}^2} \\ \tan^{-1}\left[\frac{x_{2k|k-1}}{x_{1k|k-1}}\right] \end{bmatrix} \end{aligned} \tag{36}$$

and may formulate a matrix of partial derivatives evaluated along the nominal trajectory:

$$\mathbf{S}_{k,k-1} = \frac{\partial \mathbf{s}\left[\mathbf{y}_{k-1|k-1}; t_k, t_{k-1}\right]}{\partial \mathbf{y}_{k-1|k-1}} \tag{37}$$

For ease of calculation, observe that:

$$\begin{aligned}\mathbf{x}_{rk|k-1} &= \mathbf{\Lambda}_{k,k-1}\mathbf{x}_{rk-1|k-1} + \boldsymbol{\varsigma}_{k,k-1} \\ &= \begin{bmatrix} y_{1k-1|k-1}\cos y_{2k-1|k-1} + \varsigma_{1k,k-1} \\ y_{1k-1|k-1}\sin y_{2k-1|k-1} + \varsigma_{2k,k-1} \end{bmatrix} \end{aligned} \tag{38}$$

$\Rightarrow$

$$\frac{\partial \mathbf{x}_{r_{k|k-1}}}{\partial \mathbf{y}_{k-1|k-1}} = \begin{bmatrix} \cos y_{2k-1|k-1} & -y_{1k-1|k-1} \sin y_{2k-1|k-1} \\ \sin y_{2k-1|k-1} & y_{1k-1|k-1} \cos y_{2k-1|k-1} \end{bmatrix} = \begin{bmatrix} \cos y_{2k-1|k-1} & -x_{2k-1|k-1} \\ \sin y_{2k-1|k-1} & x_{1k-1|k-1} \end{bmatrix} \tag{39}$$

Allowing the partials of the transformation to be formed:

$$\frac{\partial \mathrm{s}_1 \left[\mathbf{y}_{k-1|k-1}; t_k, t_{k-1}\right]}{\partial y_{1k-1|k-1}} = \frac{1}{2}\left(\frac{1}{\sqrt{x_{1k|k-1}^2 + x_{2k|k-1}^2}}\right)\left[2x_{1k|k-1}\left(\frac{\partial x_{1k|k-1}}{\partial y_{1k-1|k-1}}\right) + 2x_{2k|k-1}\left(\frac{\partial x_{2k|k-1}}{\partial y_{1k-1|k-1}}\right)\right] = \frac{x_{1k|k-1}\cos y_{2k-1|k-1} + x_{2k|k-1}\sin y_{2k-1|k-1}}{\sqrt{x_{1k|k-1}^2 + x_{2k|k-1}^2}} \tag{40}$$

$$\frac{\partial \mathrm{s}_1 \left[\mathbf{y}_{k-1|k-1}; t_k, t_{k-1}\right]}{\partial y_{2k-1|k-1}} = \frac{1}{2}\left(\frac{1}{\sqrt{x_{1k|k-1}^2 + x_{2k|k-1}^2}}\right)\left[2x_{1k|k-1}\left(\frac{\partial x_{1k|k-1}}{\partial y_{2k-1|k-1}}\right) + 2x_{2k|k-1}\left(\frac{\partial x_{2k|k-1}}{\partial y_{2k-1|k-1}}\right)\right] = \frac{x_{2k|k-1}x_{1k-1|k-1} - x_{1k|k-1}x_{2k-1|k-1}}{\sqrt{x_{1k|k-1}^2 + x_{2k|k-1}^2}} \tag{41}$$

$$\frac{\partial \mathrm{s}_2 \left[\mathbf{y}_{k-1|k-1}; t_k, t_{k-1}\right]}{\partial y_{1k-1|k-1}} = \frac{1}{1 + \left[\frac{x_{2k|k-1}}{x_{1k|k-1}}\right]^2}\left[\frac{x_{1k|k-1}\frac{\partial x_{2k|k-1}}{\partial y_{1k-1|k-1}} - x_{2k|k-1}\frac{\partial x_{1k|k-1}}{\partial y_{1k-1|k-1}}}{x_{1k|k-1}^2}\right] = \frac{x_{1k|k-1}\sin y_{2k-1|k-1} - x_{2k|k-1}\cos y_{2k-1|k-1}}{x_{1k|k-1}^2 + x_{2k|k-1}^2} \tag{42}$$

$$\frac{\partial \mathrm{s}_2\left[\mathbf{y}_{k-1|k-1}; t_k, t_{k-1}\right]}{\partial y_{2k-1|k-1}} = \frac{1}{1+\left[\frac{x_{2k|k-1}}{x_{1k|k-1}}\right]^2}\left[\frac{x_{1k|k-1}\frac{\partial x_{2k|k-1}}{\partial y_{2k-1|k-1}} - x_{2k|k-1}\frac{\partial x_{1k|k-1}}{\partial y_{2k-1|k-1}}}{x_{1k|k-1}^2}\right]$$
$$= \frac{x_{1k|k-1}x_{1k-1|k-1} + x_{2k|k-1}x_{2k-1|k-1}}{x_{1k|k-1}^2 + x_{2k|k-1}^2} \tag{43}$$

Making the complete partial derivative matrix in a form that will be used for discrete propagation of the covariance:

$$\mathbf{S}_{k,k-1} = \begin{bmatrix} \frac{x_{1k|k-1}\cos y_{2k-1|k-1} + x_{2k|k-1}\sin y_{2k-1|k-1}}{\sqrt{x_{1k|k-1}^2 + x_{2k|k-1}^2}} & \frac{x_{2k|k-1}x_{1k-1|k-1} - x_{1k|k-1}x_{2k-1|k-1}}{\sqrt{x_{1k|k-1}^2 + x_{2k|k-1}^2}} \\ \frac{x_{1k|k-1}\sin y_{2k-1|k-1} - x_{2k|k-1}\cos y_{2k-1|k-1}}{x_{1k|k-1}^2 + x_{2k|k-1}^2} & \frac{x_{1k|k-1}x_{1k-1|k-1} + x_{2k|k-1}x_{2k-1|k-1}}{x_{1k|k-1}^2 + x_{2k|k-1}^2} \end{bmatrix} \tag{44}$$

4.1.1 Hybrid Filter Algorithm.

To assemble the filter, the typical assumption of a Gaussian distribution of measurement noise that is uncorrelated in time is accepted, and is reasonable for this scenario. The filter must be initialized with an initial mean, $\widehat{\mathbf{y}}_0$, and covariance, $\mathbf{P}_{y_0}$, using the most likely pickup bearing (the mostly likely initial angle based on the acquisition segment profile), and the most likely pickup range, based on analysis of the true sensor performance. The typical EKF non-linear integration for the propagation of the state estimate is replaced by simply applying the inertial change in state for one time step from the semi-discrete optimal path of the trajectory planner, $\mathbf{x}_i^*$:

$$\varsigma(t_k, t_{k-1}) = \mathbf{x}_i^*(t_k) - \mathbf{x}_i^*(t_{k-1}) \tag{45}$$

The state is then advanced with Equation 34, and converted back to polar coordinates with Equation 33. The covariance is propagated in polar form as well, with:

$$\mathbf{P}_{k|k-1} = \mathbf{S}_{k,k-1}\mathbf{P}_{k-1|k-1}\mathbf{S}_{k,k-1}^T \tag{46}$$

Note that no process noise was added to the state or covariance propagation equations, based on the assumption that the inertial vehicle position estimate had reached steady-state. This means that, with a static target, the target position estimate uncertainty does not grow between measurement updates.

Measurements are modeled as in Equation 13 on page 45, but replacing the measurement function with the polar form from Equation 20:

$$\xi_k = \mathbf{H_y} \mathbf{y}_k + \eta_k \tag{47}$$

When measurements become available, the system estimate and error uncertainty can now be updated with the common linear Kalman Filter equations:

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P_y}_{k|k-1} \mathbf{H_y}^T \left[\mathbf{H_y} \mathbf{P_y}_{k|k-1} \mathbf{H_y}^T + R \right]^{-1} \\ \widehat{\mathbf{y}}_{k|k} &= \widehat{\mathbf{y}}_{k|k-1} + \mathbf{K}_k \left[\xi_k - \mathbf{H_y} \widehat{\mathbf{y}}_{k|k-1} \right] \\ \mathbf{P_y}_{k|k} &= \mathbf{P_y}_{k|k-1} - \mathbf{K}_k \mathbf{H_y} \mathbf{P_y}_{k|k-1} \end{aligned} \tag{48}$$

This form of the HEKF was used for much of the build-up research prior to the final flight test with the actual wire, and it remains a potential option for others following with similar scenarios. In addition, in Chapter V, the information states and their dynamics are developed from the Fisher Information Matrix using the same fundamental principles used here for the EKF. For the final flight test however, the desire to represent the final error covariance in the Cartesian frame, and the desire to avoid a potential bias from the estimated mean drove the decision to use the Unscented Transformation.

4.2 Unscented Kalman Filter

The UKF was used for target estimation for the final quadrotor flight test results presented in Chaper IX [59]. As in an EKF, the target position estimate and the measurements are characterized with probability density functions (pdfs), in this case Gaussian, and represented by the first two moments of the state (mean and covariance). The propagation and update steps are again considered time invariant Markovian processes, allowing recursive calculations at each time step to perform the non-linear transformations of the pdf. These calculations are referred to as "Unscented Transformations," and when implemented with the propagation and measurement steps to perform estimation, the algorithm is dubbed the Unscented Kalman Filter, or sometimes the sigma-point Kalman Filter (SPKF). The UT is based on the fact that it is easier to approximate a probability distribution than an arbitrary non-linear transformation. There are several variants that can be optimized for different applications, varying such factors as the selection of the sigma-points within the necessary conditions, choosing the regression weights, and performing the transformation to different orders of accuracy. For this research, propagation was performed again with the linear transformation using Equations 34 and 45. The measurement update was performed with the following algorithm, taken from [59]:

1. Sigma-points, $\mathcal{X} \in \mathbb{R}^{n_{\mathbf{x}} \times (2n_{\mathbf{x}}+1)}$, are selected for the $n_{\mathbf{x}}$ states in a manner that maintains, for the set, the mean and covariance of the current distribution prior to the measurement update, $\widehat{\mathbf{x}}_{r_k}^-$ and $\mathbf{P}_k^-$:

$$\begin{aligned}
\mathcal{X}_k^{(0)-} &= \widehat{\mathbf{x}}_{r_k}^- \\
\mathcal{X}_k^{(i)-} &= \mathcal{X}_k^{(0)-} + \left(\sqrt[c]{(n_{\mathbf{x}}+\lambda)\,\mathbf{P}_k^-}\right)_i \qquad && i = 1,\ldots,n_{\mathbf{x}} \\
\mathcal{X}_k^{(j)-} &= \mathcal{X}_k^{(0)-} - \left(\sqrt[c]{(n_{\mathbf{x}}+\lambda)\,\mathbf{P}_k^-}\right)_j \qquad && j = n_{\mathbf{x}}+1,\ldots,2n_{\mathbf{x}}
\end{aligned} \tag{49}$$

The symbol $\left(\sqrt[c]{(\cdot)}\right)_i$ represents the ith column of the matrix square root, obtained with the Cholesky decomposition.

2. Each state vector sigma-point is transformed into the measurement space through the observation function, Equation 11, with the appropriate element substitutions for x_r and z_r:

$$\mathcal{Z}_k^{(i)-} = \mathbf{h}\left[\mathcal{X}_k^{(i)-}\right] \qquad i = 0, \ldots, 2n_x \tag{50}$$

3. The statistics of the projected sigma-points are calculated for the estimated measurements. The mean is found with a weighted sum:

$$\widehat{\mathbf{z}}_k^- = \sum\nolimits_{i=0}^{2n_x} W_m^{(i)} \mathcal{Z}_k^{(i)-} \tag{51}$$

where the weights are determined with:

$$\begin{aligned} W_m^{(0)} &= \lambda/(n_x + \lambda) \\ W_m^{(i)} &= 1/[2\,(n_x + \lambda)] \qquad i = 1, \ldots, 2n_x \end{aligned} \tag{52}$$

with scaling parameter $\lambda = \alpha^2\,(n_x + \kappa) - n_x$ to meet the necessary condition for an unbiased mean, $\sum_{i=0}^{2n_x} W_m^{(i)} = 1$. The gain on the sigma-point spread, loosely speaking, was set at $\alpha = 0.001$, and the tuning parameter was set at $\kappa = 0$. As the noise on the variables is independent, the variance may be additively applied, calculating the covariance and cross correlation with:

$$\begin{aligned} \mathbf{P}_{zz_k} &= \sum\nolimits_{i=0}^{2n_x} W_c^{(i)} \left(\mathcal{Z}_k^{(i)-} - \widehat{\mathbf{z}}_k^-\right)\left(\mathcal{Z}_k^{(i)-} - \widehat{\mathbf{z}}_k^-\right)^T + R_k \\ \mathbf{P}_{xz_k} &= \sum\nolimits_{i=0}^{2n_x} W_c^{(i)} \left(\mathcal{X}_k^{(i)-} - \widehat{\mathbf{x}}_k^-\right)\left(\mathcal{Z}_k^{(i)-} - \widehat{\mathbf{z}}_k^-\right)^T \end{aligned} \tag{53}$$

using the covariance weights:

$$\begin{aligned} W_c^{(0)} &= \lambda/(n_x+\lambda) + (1-\alpha^2+\mu) \\ W_c^{(i)} &= 1/[2\,(n_x+\lambda)] \qquad i = 1,\ldots,2n_x \end{aligned} \tag{54}$$

For this Gaussian distribution, the tuning parameter was set optimally at $\mu = 2$.

4. A weighting gain is then calculated:

$$\mathbf{K}_{UKF_k} = \mathbf{P}_{xz_k}(\mathbf{P}_{zz_k})^{-1} \tag{55}$$

and applied to project the appropriate residual error onto the mean prediction and update the covariance:

$$\begin{aligned} \widehat{\mathbf{x}}_{r_k}^{+} &= \widehat{\mathbf{x}}_{r_k}^{-} + \mathbf{K}_{UKF_k}\left(\mathbf{z}_k - \widehat{\mathbf{z}}_k^{-}\right) \\ \mathbf{P}_k^{+} &= \mathbf{P}_k^{-} - \mathbf{K}_{UKF_k}\mathbf{P}_{zz_k}\mathbf{K}_{UKF_k}^{T} \end{aligned} \tag{56}$$

During the flight test, the UKF and the trajectory planner were run consecutively. As multiple measurements often became available while the trajectory planner was calculating ($\Delta t_{meas} = 0.33$), all new measurements were processed in batch at each iteration.

V. Simultaneous Solution of the Optimal Control and Estimation Problems

SIMULTANEOUS solution of the optimal control problem and the optimal estimation problem requires breaking down the fundamental observability requirements of an estimator into elements that can inform an optimal control solver *how* to move to make the estimate *better*, in the midst of some control task. Examples of initial work in this direction were provided in Chapter II, under the umbrella of localization. Localization techniques find a scalar metric to assess estimate quality, and seek the path that will optimize that metric in a given time or number of measurements. As has been shown, most of the work in localization is in the area of finding the most desirable performance index, since compressing the necessarily multi-dimensional knowledge of a target's position into a scalar results in a loss of directional information that can be problematic.

Dual control concepts were also introduced that broaden this effort. The fundamental localization techniques remain unchanged, but a control desire is added to the performance index with the information metric. Basic methods include separating the efforts into different dimensions and treating them independently. More comprehensive efforts typically pit the contending desires of control and estimation against one another—applying a cost functional element on control that regulates the states *to* a particular path, and another cost related to estimation quality that pushes the states *away* from that path in an effort to increase observability.

Many of the same limitations from localization exist for dual control. The directional information is still compressed to a scalar cost function, and researchers have still relied on a fixed final time (at times indirectly). Of further concern for dual control is the question of how to determine the set of weights that balance how much effort should go to each desire, typically dealt with by using an arbitrary function of the current estimate uncertainty. In the end, the values set for the weights will determine the overall level of certainty that the system has at the end of the path, which may or may not meet the physical needs of the system.

Each of these limitations of current methods needs to be addressed in turn, starting with the scalar cost function (determinant, trace, etc.). Regardless of the metric selected, all of them attempt to encapsulate directional information contained within the Fisher Information Matrix.

5.1 Development of the Fisher Information Matrix from the Cramér-Rao Lower Bound

The concept of Fisher Information is a byproduct of the development of the Cramér-Rao Lower Bound, commonly used in estimation, the derivation of which is taken from [111]. Fisher Information is fundamentally tied to the concept of observability in the framework of estimation theory. If a measurement is treated as a random variable, Z, with ζ being a sample of that variable, and the measurement is dependent on the state, $\mathbf{x}$, treated as an unknown but deterministic parameter, then a likelihood function, $p(Z;\mathbf{x})$ would describe the probability of receiving a particular ζ given a known $\mathbf{x}$. Plotting $p(Z;\mathbf{x})$ gives insight into the observability of $\mathbf{x}$ through the measurement ζ. If the plot showed a low variance (a tight peak), then there is a strong ability to estimate $\mathbf{x}$ with the measurement ζ. It could be said that ζ relates a good deal of information about $\mathbf{x}$, or that $\mathbf{x}$ is highly observable. Note that the ability to estimate $\mathbf{x}$ is dependent on the collection of measurements. This is characterized by the FIM, which is developed from the definition of an unbiased observer, which states that the error of an estimate conditioned on a particular state will be zero:

$$E\left[\widehat{\mathbf{x}}(Z) - \mathbf{x} \middle| \, \mathbf{x}\right] = \int_{-\infty}^{\infty} \left[\widehat{\mathbf{x}}(\zeta) - \mathbf{x}\right] p(\zeta; \mathbf{x}) \, d\zeta = 0 \tag{57}$$

This must be true for all values of $\mathbf{x}$, therefore:

$$\frac{\partial}{\partial \mathbf{x}} \int_{-\infty}^{\infty} [\widehat{\mathbf{x}}(\zeta) - \mathbf{x}] p(\zeta; \mathbf{x}) \, d\zeta = 0 \tag{58}$$

Assuming that $\partial p(\zeta; \mathbf{x})/\partial \mathbf{x}$ exists and is absolutely integrable, the partial is taken inside the integral and the chain rule is applied:

$$-\int_{-\infty}^{\infty} p(\zeta; \mathbf{x}) \, d\zeta + \int_{-\infty}^{\infty} \left(\widehat{\mathbf{x}}(\zeta) - \mathbf{x}\right) \frac{\partial p(\zeta; \mathbf{x})}{\partial \mathbf{x}} \, d\zeta = 0 \tag{59}$$

Note that the measurement is assumed to be Gaussian, with the associated exponential distribution. Therefore:

$$\frac{\partial p(\zeta; \mathbf{x})}{\partial \mathbf{x}} = p(\zeta; \mathbf{x}) \frac{\partial \ln p(\zeta; \mathbf{x})}{\partial \mathbf{x}} \tag{60}$$

The furthest right partial derivative is referred to as the score. Also note that by definition of a pdf:

$$\int_{-\infty}^{\infty} p(\zeta; \mathbf{x}) \, d\zeta = 1 \tag{61}$$

Equation 59 then reduces to:

$$\int_{-\infty}^{\infty} \left(\frac{\partial \ln p(\zeta; \mathbf{x})}{\partial \mathbf{x}} p(\zeta; \mathbf{x}) \left[\widehat{\mathbf{x}}(\zeta) - \mathbf{x}\right] \right) \, d\zeta = 1 \tag{62}$$

$$\Rightarrow$$

$$\int_{-\infty}^{\infty} \left(\frac{\partial \ln p(\zeta; \mathbf{x})}{\partial \mathbf{x}} [p(\zeta; \mathbf{x})]^{1/2} \right) \left([p(\zeta; \mathbf{x})]^{1/2} \left[\widehat{\mathbf{x}}(\zeta) - \mathbf{x}\right] \right) \, d\zeta = 1 \tag{63}$$

The CRLB is then found by squaring both sides and splitting the integral with the Cauchy-Schwarz inequality:

$$\int_{-\infty}^{\infty} (\widehat{\mathbf{x}}(\zeta) - \mathbf{x})^2 p(\zeta; \mathbf{x}) \, d\zeta \cdot \int_{-\infty}^{\infty} \left(\frac{\partial \ln p(\zeta; \mathbf{x})}{\partial \mathbf{x}} \right)^2 p(\zeta; \mathbf{x}) \, d\zeta \geq 1 \tag{64}$$

The left argument is recognized as the expected mean-squared error of the estimator, and the right argument is defined as the Fisher Information, the variance of the score (the mean of the score can be shown to be zero). For an unbiased estimator, then, the CRLB tells us that the certainty with which we know our estimate is limited by the Fisher Information of the likelihood function:

$$\mathrm{Var}\left[\widehat{\mathbf{x}}\right] \geq \mathcal{I}^{-1}\left(\mathbf{x}\right) \tag{65}$$

where:

$$\mathcal{I}\left(\mathbf{x}\right) = E\left[\left(\frac{\partial \ln p(Z; \mathbf{x})}{\partial \mathbf{x}} \right)^2 \middle| \mathbf{x} \right] \tag{66}$$

A more useful formulation is found with Equation 60 and the assumption made for Equation 59, differentiating the likelihood function with respect to $\mathbf{x}$:

$$0 = \frac{\partial}{\partial \mathbf{x}} \int_{-\infty}^{\infty} p(\zeta; \mathbf{x}) \, d\zeta = \int_{-\infty}^{\infty} \frac{\partial p(\zeta; \mathbf{x})}{\partial \mathbf{x}} \, d\zeta = \int_{-\infty}^{\infty} \frac{\partial \ln p(\zeta; \mathbf{x})}{\partial \mathbf{x}} p(\zeta; \mathbf{x}) \, d\zeta \tag{67}$$

Assuming the second partial exists and is integrable, the equation is differentiated again:

$$\int_{-\infty}^{\infty} \frac{\partial^2 \ln p(\zeta; \mathbf{x})}{\partial \mathbf{x}^2} p(\zeta; \mathbf{x}) \, d\zeta + \int_{-\infty}^{\infty} \left(\frac{\partial \ln p(\zeta; \mathbf{x})}{\partial \mathbf{x}} \right)^2 p(\zeta; \mathbf{x}) \, d\zeta = 0 \tag{68}$$

which leads us to the familiar form of the FIM:

$$\mathcal{I}\left(\mathbf{x}\right) = E\left[\left(\frac{\partial \ln p(Z; \mathbf{x})}{\partial \mathbf{x}} \right)^2 \middle| \mathbf{x} \right] = -E\left[\frac{\partial^2 \ln p(Z; \mathbf{x})}{\partial \mathbf{x}^2} \middle| \mathbf{x} \right] \tag{69}$$

Taylor applied this form of the FIM to a dynamical system with a non-linear, time-varying state vector under deterministic inputs with time-varying measurements corrupted by additive, Gaussian white noise sequences [108]. Because the measurements are assumed to be independent, the likelihood function is a product of the individual Gaussian exponential distributions. Under the logarithm, this becomes a sum, allowing a recursive form of the FIM to be found by taking the expectation of the second partials:

$$\boldsymbol{\mathcal{I}}_{k+1|k} = \left[\boldsymbol{\Phi}_{k+1,k}^{T}\right]^{-1}\boldsymbol{\mathcal{I}}_{k}\boldsymbol{\Phi}_{k+1,k}^{-1} + \mathbf{H}_{k+1}^{T}\mathbf{R}_{k+1}^{-1}\mathbf{H}_{k+1} \tag{70}$$

where $\boldsymbol{\Phi}_{k+1,k}$ is the state transition matrix from $\mathbf{x}_{t_k}$ to $\mathbf{x}_{t_{k+1}}$, and $\mathbf{H}_{k+1}$ is the Jacobian of the observation function from Equation 11:

$$\begin{aligned}\mathbf{H}_{k+1} &\equiv \frac{\partial \mathbf{h}\left[\mathbf{x}_{k+1}\right]}{\partial \mathbf{x}_{k+1}} \\ &= \begin{bmatrix} \frac{\partial}{\partial x}\tan^{-1}\frac{z_{r_{k+1}}}{x_{r_{k+1}}} & \frac{\partial}{\partial z}\tan^{-1}\frac{z_{r_{k+1}}}{x_{r_{k+1}}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{1+\left(\frac{z_{r_{k+1}}}{x_{r_{k+1}}}\right)^2}\left(\frac{z_{r_{k+1}}}{x_{r_{k+1}}^2}\right) & \frac{1}{1+\left(\frac{z_{r_{k+1}}}{x_{r_{k+1}}}\right)^2}\left(\frac{-1}{x_{r_{k+1}}}\right) \end{bmatrix} \\ &= \begin{bmatrix} \frac{z_{r_{k+1}}}{\rho_{k+1}^2} & \frac{-x_{r_{k+1}}}{\rho_{k+1}^2} \end{bmatrix}\end{aligned} \tag{71}$$

Equation 70 shows that the amount of information that each measurement provides is encapsulated in the term $\mathbf{H}_{k+1}^{T}\mathbf{R}_{k+1}^{-1}\mathbf{H}_{k+1}$. Assuming a constant uncertainty for each measurement, σ_{β}, the directional information is contained within $\mathbf{H}_{k+1}^{T}\mathbf{H}_{k+1}$. The same conclusion is reached if the problem is addressed with a least squares approach on a Taylor series expansion expanded around a nominal state, as is done in the GPS dilution of precision (DOP) analysis ($\mathbf{H}^{T}\mathbf{H}$ is known as the DOP matrix [81]), flipping the problem to use measurement uncertainty in angle vice the GPS uncertainty in range. Unsurprisingly, this also can be seen in the observability Grammian as well [78]:

$$\mathbf{M} \equiv \int_{t_0}^{t_1} \mathbf{\Phi}^T(\tau, t_0)\, \mathbf{H}^T(\tau)\, \mathbf{H}(\tau)\, \mathbf{\Phi}(\tau, t_0)\ d\tau \tag{72}$$

In the polar formulation, the transition matrix rotates the information matrix to the new orientation of ρ and β as the observer moves in relation to the target. In the Cartesian formulation, however, the estimate of the target state is anchored to the navigation frame, which does not change as the relative coordinates vary. The state transition matrix is identity. If the certainty in the observer's position has reached a steady-state, then the only time the information certainty in the target estimate changes is when there is a measurement update, removing process noise from consideration. Using Equation 70 and the appropriate trigonometric identities, the Fisher Information Matrix becomes:

$$\boldsymbol{\mathcal{I}}_k = \boldsymbol{\mathcal{I}}_0 + \frac{1}{\sigma_\beta^2} \begin{bmatrix} \sum_{i=1}^{k} \frac{\sin^2\beta_i}{\rho_i^2} & -\sum_{i=1}^{k} \frac{\sin\beta_i \cos\beta_i}{\rho_i^2} \\ -\sum_{i=1}^{k} \frac{\sin\beta_i \cos\beta_i}{\rho_i^2} & \sum_{i=1}^{k} \frac{\cos^2\beta_i}{\rho_i^2} \end{bmatrix} \tag{73}$$

5.1.1 Directional Compression and One-Step Ahead Analysis.

Many of the trajectory planning techniques currently in use compress metrics similar to Equation 73 into a scalar to determine the optimal path. This can be instructive. Applying a one-step ahead approach and using the determinant of the FIM as the metric of choice, the question becomes how to maximize the information in the next step. Adopting the abbreviations $S_k = \sin\beta_k$ and $C_k = \cos\beta_k$, the determinant becomes:

$$\det\left(\mathbf{H}_k^T\mathbf{H}_k + \mathbf{H}_{k+1}^T\mathbf{H}_{k+1}\right) = \begin{vmatrix} \frac{1}{\rho_k^2}S_k^2 + \frac{1}{\rho_{k+1}^2}S_{k+1}^2 & -\frac{1}{\rho_k^2}S_kC_k - \frac{1}{\rho_{k+1}^2}S_{k+1}C_{k+1} \\ -\frac{1}{\rho_k^2}S_kC_k - \frac{1}{\rho_{k+1}^2}S_{k+1}C_{k+1} & \frac{1}{\rho_k^2}C_k^2 + \frac{1}{\rho_{k+1}^2}C_{k+1}^2 \end{vmatrix} \tag{74}$$

$$\begin{aligned}
&= \frac{1}{\rho_k^4}S_k^2C_k^2 + \frac{1}{\rho_k^2\rho_{k+1}^2}S_k^2C_{k+1}^2 + \frac{1}{\rho_k^2\rho_{k+1}^2}S_{k+1}^2C_k^2 + \frac{1}{\rho_k^4}S_{k+1}^2C_{k+1}^2 \\
&\qquad - \frac{1}{\rho_k^4}S_k^2C_k^2 - \frac{2}{\rho_k^2\rho_{k+1}^2}S_kS_{k+1}C_kC_{k+1} - \frac{1}{\rho_k^4}S_{k+1}^2C_{k+1}^2 \\
&= \frac{1}{\rho_k^2\rho_{k+1}^2}\left(S_k^2C_{k+1}^2 - 2S_kS_{k+1}C_kC_{k+1} + S_{k+1}^2C_k^2\right) \\
&= \frac{1}{\rho_k^2\rho_{k+1}^2}\left(S_kC_{k+1} - S_{k+1}C_k\right)^2 \\
&= \frac{1}{\rho_k^2\rho_{k+1}^2}\left(\frac{1}{2}\sin\left(\beta_k+\beta_{k+1}\right) + \frac{1}{2}\sin\left(\beta_k-\beta_{k+1}\right)\right. \\
&\qquad \left. -\frac{1}{2}\sin\left(\beta_{k+1}+\beta_k\right) - \frac{1}{2}\sin\left(\beta_{k+1}-\beta_k\right)\right)^2 \\
&= \frac{1}{\rho_k^2\rho_{k+1}^2}\left(\sin\left(\beta_k-\beta_{k+1}\right)\right)^2
\end{aligned} \tag{75}$$

This equation gives insight to the geometry of the problem, and supports natural intuition. Subsequent measurements from the same angle yield no new information—neither do measurements from an opposing angle across the target (difference of π). To accomplish the goal of minimizing the area of uncertainty around the target location estimate (from any fixed ρ_k and β_k), the observer should move in such a manner to decrease the range and increase the orthogonality of the next measurement. Note, however, that the information about the shape of the uncertainty ellipse has been lost in the compression. Initial efforts for this research treated this one-step ahead approach as an optimal problem, analytically solving for a control policy analogous to [114]. The result was a spiral toward the target very similar to that of [88].

Though implementable in real-time, this localization approach was not optimal over the entire trajectory, and failed to address a significant number of the limitations of previous research. Most glaringly, the future value for the covariance at the end of the path is unknown, and the shape of that uncertainty ellipsoid is lost in the compression. Real systems require a trajectory that will allow them to achieve a particular certainty magnitude and shape determined by physical realities. A method was sought to reach a particular final certainty, based on the true system requirements.

5.2 A New Approach

Localization and dual control methods compress an information metric to a scalar performance index and seek a control that will maximize the amount of information. This may or may not meet the certainty requirements of a system based on physical realities—the required fire control solution to launch a torpedo for the submarine, the size and shape of the hook used for a sUAS to land on a wire, etc. If a path received from a trajectory planner balances a weighted effort on localization and control, the ending certainty level is unknown. If requirements are not met, the mission results in failure. If requirements are over-met, the solution may have met optimality conditions, but the cost function did not match the true needs. In that case, the trajectory planner produced the right solution to the wrong problem.

The intent of this dissertation is to fundamentally change the way the dual control problem is approached. For systems where a level of information is a necessary, but secondary tool required to perform a primary mission, effort and energy should not be wasted on information-gathering maneuvers that are unneeded. The path planning algorithm should not seek to maximize certainty, nor to nebulously balance the amount of effort based on the certainty you have now, especially if the geometry of the problem is such that the certainty level will greatly change soon. The amount of certainty expected at the *final* condition, the point of mission accomplishment, should be what drives maneuvers—not the current state and estimate. The final expected uncertainty is a particular amount of information in each direction, dependent on the system and the mission.

5.2.1 Suboptimal Final Covariance Shooting Method.

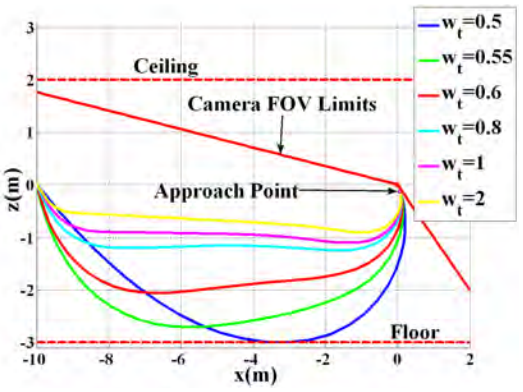
To begin the process of evaluating a path based on the final estimate error uncertainty, a shooting method was developed. This method used ideas similar to some of the dual control methods that selected weights to balance control and estimation efforts. Instead of basing the weights on current certainty levels, the shooting method uses the future certainty expected at the critical moment. As a circular argument, an iteration was introduced to optimize on the correct set of weights, analogous to indirect optimal shooting approaches. The weights are adjusted until the optimal path contains the desired characteristics of the prescribed final uncertainty levels in each direction, assuming that measurements will continue to be received along the route. Using the orthogonality lessons from Equation 75 and the polar formulation, the cost function was proposed:

$$J_{subopt} = w_t t_f + \int_{t0}^{tf} -w_x \sin^2\beta - w_z \cos^2\beta + \mathbf{u_y}^T \mathbf{W_{u_y}} \mathbf{u_y} \; dt \tag{76}$$

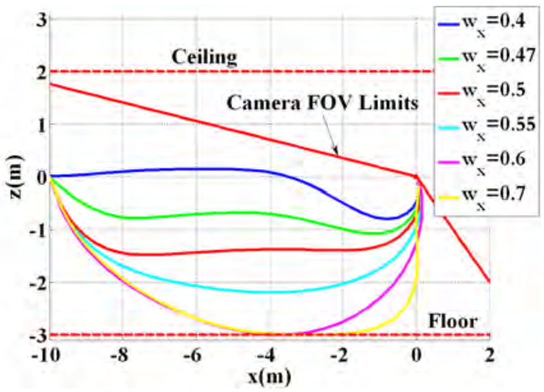
The final time requirement ensured that unnecessary maneuvers were not accomplished, and the sine and cosine terms ensured that information from both orthogonal directions was gained. A weakness of this method (and many of the dual methods) is the failure to account for the fact that measurements at a close range provide a higher level of information. Initial weights were selected based on simulation of the path that will be solved with the initial guess, since that is known *a priori*. As measurements are received and the estimate of the target location begins to move, an inner iteration loop is accomplished. First, the optimal path is solved for based on the initial weights. The measurement function is then linearized about each anticipated measurement along that path, and the EKF update equations are used to propagate the certainty for all expected measurements, as done in Equation 48 on page 58. The result provides the entire expected covariance matrix at the final time, allowing decisions to be made directionally, vice only being able to work with a scalar approximation. The weighting is then adjusted based on the *future* expected uncertainty, and the loop is continued until tolerances are met.

A heuristic function is required to adjust the weights. The weight on the controls is held fixed, and the weighting on the directional information is determined by a ratio, resulting in two "knobs" to adjust the path—one on direction ratio, w_x, and one on the final time, w_t. If the final expected covariance in the x-direction, P_{xx}, does not meet the requirements, its weight is increased in relation to that on P_{zz} ($w_z = 1 - w_x$). If the certainty in both directions exceeds the required standard, the path can be made shorter, and the relative weight on the final time is increased. Families of solutions can be produced by tuning the two "knobs," w_t and w_x, as shown in Figure 17.

This method overcomes some of the major limitations of previous dual control approaches. Besides being able to provide the requirement of a final uncertainty, the system no longer has final time as a fixed entity. This is critical, as the path may need to be shortened or lengthened for more measurements in response to physical certainty requirements. Two paths are shown in Figure 18. In both cases, the initial geometry of the problem makes getting information in the z-axis direction easy, while the x-axis direction is initially unobservable and requires maneuver to achieve the necessary observability. The first profile represents a solution where a low amount of

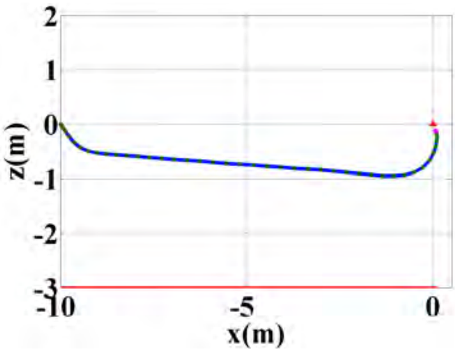


(a) Varying Final Time Weight, w_t

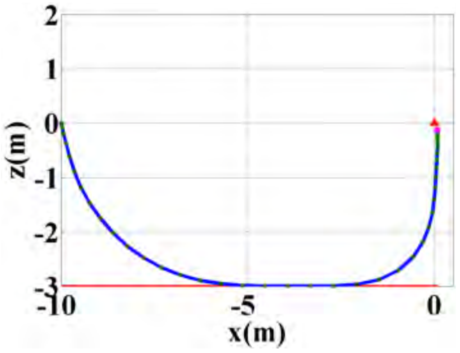


(b) Varying Direction Ratio Weight ($w_z = 1 - w_x$)

Figure 17. Iterative Method of Shooting for Final Covariance



(a) Profile 1, $w_t = 2$, $w_x = 0.55$, $w_z = 0.45$



(b) Profile 2, $w_t = 0.5$, $w_x = 0.55$, $w_z = 0.45$

Figure 18. Flightpaths with Different Levels of Required Final Covariance

additional information is required over the current levels of certainty, and the second path is representative of a path with much greater need for information gain. Note the speed profile differences in the details of Figure 19 and Figure 20. In the first

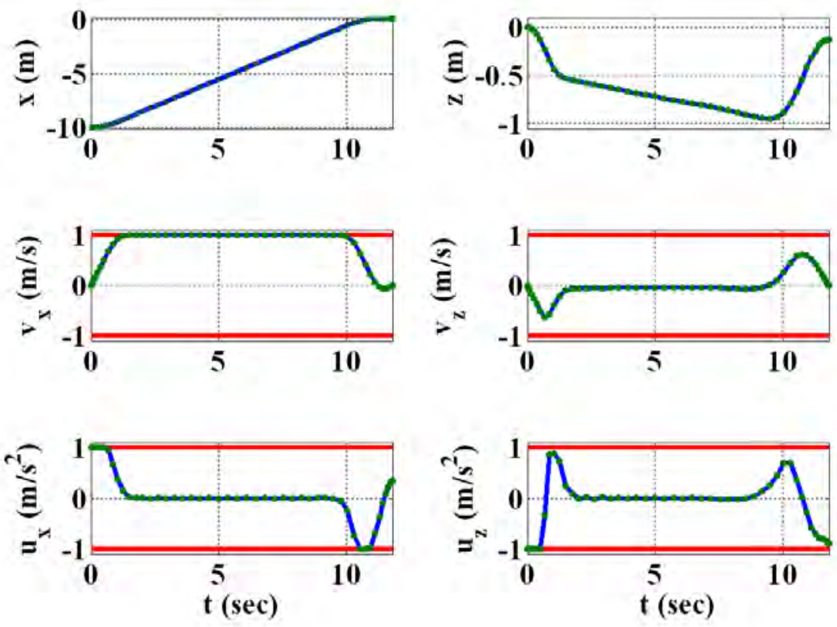


Figure 19. Profile 1, High Total Speed for Entire Flight

case, a maximum speed profile is optimum, while in the second, the optimum profile is to move at maximum speed to an angle nearly orthogonal to the x-axis, and then to dwell at a very low speed—collecting additional measurements to increase the certainty in the x-direction.

This ability to change speed and path length far exceeds the current methods of dual control, which rely on fixed numbers of measurements (fixed final time) and fixed velocities in the solution. The dual control solutions are optimal in a mathematical sense, but unless you happen to pick the optimal number of measurements for your needs and the optimal speed, the solution isn't really what you are looking for. This deficiency can clearly been seen in Figure 7 on page 29.

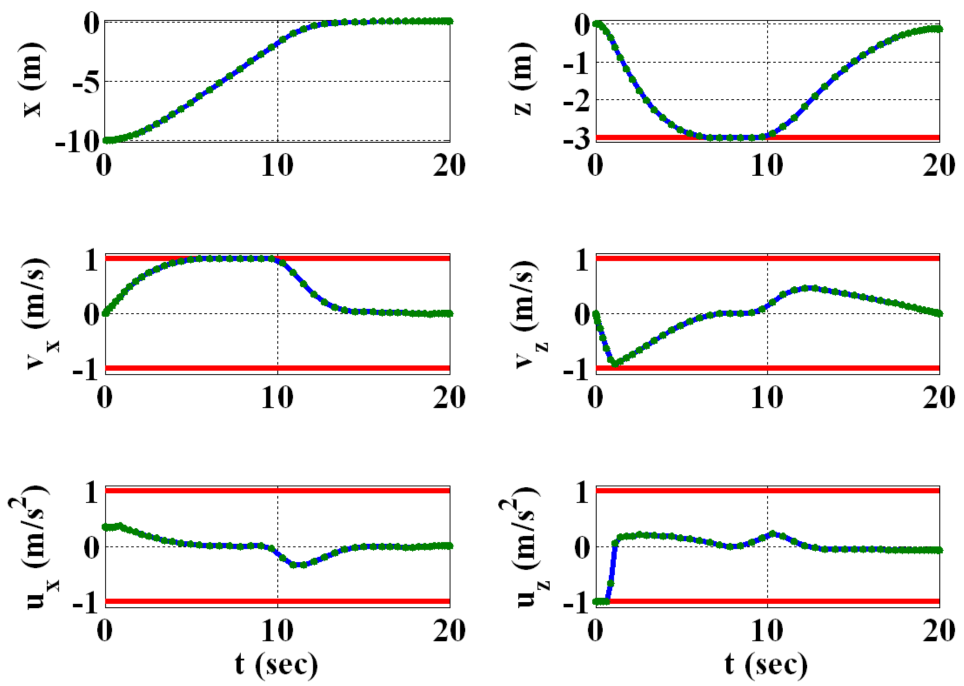


Figure 20. Profile 2, High Speed to Good Observation Point, Followed by a Dwell to Collect Extra Measurements

5.2.1.1 Shooting Method Limitations.

There are several drawbacks to this shooting method, with two that particularly stand out. The first is the requirement for a heuristic program to search for the weighting combination that will result in the right final characteristics. There are many potential local minimums in this choice, as there are potentially any number of weighting combinations that may be sufficient given two “knobs” to adjust. Mathematically, a global minimum could be attained by assuming a weight ratio to prescribe the balance between time and direction efforts, thereby reducing the scope of the problem to only one tuning parameter, but making that assumption would further limit the optimality of the solution.

The obvious second drawback is the inefficiency involved with having two optimization loops. Not only does the system have to iterate to find the optimal solution for each set of weights, but it must iterate to find the optimal weights to supply the required certainty levels—for every planning epoch. Each time the system receives a new measurement, the estimated target location moves, invalidating the previous solution and the process must begin again. In theory, these updates will increase in speed as the target estimate becomes more certain with many measurements, and using the previous solution as a "bootstrap" guess will speed up computation time, but the process is too inefficient for a real-time program. A smooth, efficient, single-shot solution was desired—one that incorporates the shooting method's gains of a determined final covariance and a flexible number of measurements, but that solves the optimal control and the optimal estimation problems in a single epoch.

5.2.2 Single-Shot Simultaneous Control and Estimation.

In order to overcome the limitations of all of the localization and dual control methods addressed in this dissertation, the basic approach to the formulation of the optimal control problem must be fundamentally altered. Instead of optimizing on a particular information metric, or balancing control and estimation desires (based on that information metric), a general cost function should be allowed that encapsulates the control desires for *mission accomplishment* for any given system. In the absence of a need for additional information, this cost function should result in a solution that follows the most desired path, be it minimum time, minimum energy, or any other function. The final error covariance requirements must be removed from the performance index and be addressed as they really are—a constraint. If the path requires more maneuvering to achieve a better final target estimate, the path planner should determine how much and in what directions, deviating from the intent of the general cost function as little as possible. If the mission can be accomplished in the optimal manner without additional information, the solution should be found as if observability was not considered.

Though straightforward in theory, this concept is problematic. The final error covariance cannot simply be applied as a final state combination constraint, since the problem is non-holonomic. There is no way to calculate the final certainty based only on the final point—the entire path must be considered. One possible solution would be to solve the entire path first, and then propagate the Kalman filter equations forward to see if the path met observability requirements (this is the essence of the shooting method in Section 5.2.1). This method, however, does not provide the optimal control solver with any path gradient information for how to change the path in order to improve the characteristics (hence the weight iteration scheme of the shooting method).

To get the information of *how* to change the path for observability requirements into the context of the optimal control solver, the uncertainty information must be contained within the states, or be contained within additional appended states. Only in this manner will the constraint Jacobian contain the gradient information necessary to correctly move the path. To do this requires a method that will quantify how the level of information changes with respect to time, in relation to a particular system state vector.

Attaining an appropriate dynamical equation is problematic for a continuous formulation, as the information changes are characterized by steps at discrete times when measurements are received. A fixed time step could potentially be assumed and the optimal control problem attempted with equally spaced nodes in a parameterized system, but sacrificing the pseudospectral node spacing of modern direct methods means giving up speed and accuracy desired for an on-line system.

Maybeck presents a continuous equation for propagation of uncertainty matrices within the context of the linear Kalman filter [78]:

$$\dot{\mathbf{P}}(t) = \mathbf{F}(t)\mathbf{P}(t) + \mathbf{P}(t)\mathbf{F}^T(t) + \mathbf{G}(t)\mathbf{Q}(t)\mathbf{G}(t) - \mathbf{P}(t)\mathbf{H}^T(t)\mathbf{R}_c^{-1}\mathbf{H}(t)\mathbf{P}(t) \qquad (77)$$

In this equation, $\mathbf{F}$ contains the state propagation information. For the polar formulation, this rotates the covariance matrix to align with the changing states of ρ and β. For the Cartesian formulation, which is tied to the inertial frame, $\mathbf{F} = \mathbf{0}$. In the third term, $\mathbf{G}$ encapsulates the input-output transfer functions, which regulate the influence of the assumed dynamics noise, described by $\mathbf{Q}$. This adds the increasing covariance trait between measurements. For this problem, since the target is static and the own-ship position estimate is assumed to have achieved steady-state, the error covariance does not change between measurements, so $\mathbf{G} = \mathbf{0}$ as well. It would seem then, that the dynamics of $\mathbf{P}$ could be estimated by $\dot{\mathbf{P}}(t) = -\mathbf{P}(t)\mathbf{H}^T(t)\mathbf{R}_c^{-1}\mathbf{H}(t)\mathbf{P}(t)$. In that case, the elements of the covariance matrix could be appended to the state vector, and limited to the desired final required covariance size and shape with appropriate boundary conditions. To achieve Equation 77, however, a simplifying assumption of continuously available measurements was made. For the sUAS scenario using line detection algorithms on sequential images for measurements, the expected update rate was between 2 and 3 Hz. Allowing a continuous measurement assumption, and allowing the accompanying linearization of the system, the resulting covariance estimate is not responsive enough, particularly to the first measurement, and the error is slow to correct, as shown in Figure 21.

To incorporate the measurement sample time into an approximation for the covariance dynamics—again assuming that the only change happens at the measurement update—a single update equation can be used:

$$\begin{aligned}\mathbf{P}(t_{i+1}^-) &= \mathbf{P}(t_i^+) \\ &= \mathbf{P}(t_i^-) - \mathbf{P}(t_i^-)\mathbf{H}^T(t_i)\left[\mathbf{H}(t_i)\mathbf{P}(t_i^-)\mathbf{H}^T(t_i) + \mathbf{R}(t_i)\right]^{-1}\mathbf{H}(t_i)\mathbf{P}(t_i^-)\end{aligned} \tag{78}$$

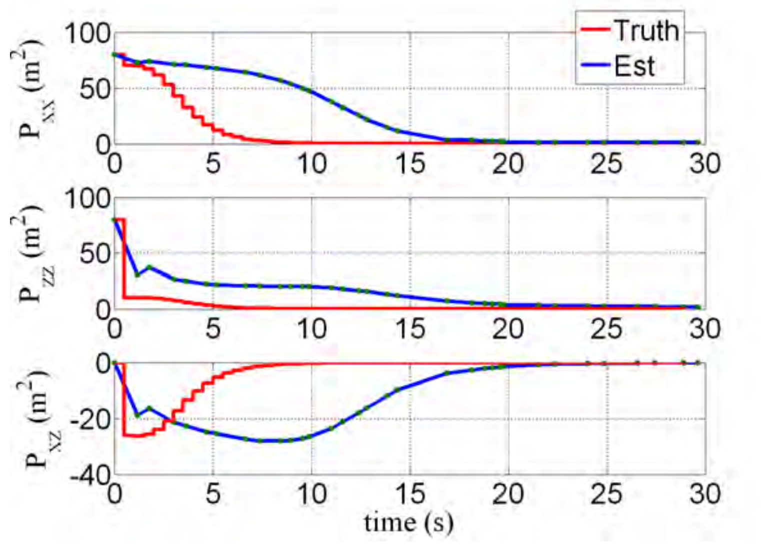


Figure 21. Inadequacy of Continuous Measurement Assumption for Covariance Propagation

$\Rightarrow$

$$\begin{aligned}\frac{\mathbf{P}(t_{i+1}^-) - \mathbf{P}(t_i^-)}{\Delta t_{meas}} &= \frac{-\mathbf{P}(t_i^-)\mathbf{H}^T(t_i)\left[\mathbf{H}(t_i)\mathbf{P}(t_i^-)\mathbf{H}^T(t_i) + \mathbf{R}(t_i)\right]^{-1}\mathbf{H}(t_i)\mathbf{P}(t_i^-)}{\Delta t_{meas}} \\ &\approx \dot{\mathbf{P}}(t) \end{aligned} \tag{79}$$

Clearly, this first-order approximation is only accurate for small values of Δt_{meas}, and is questionable at best for this application. Even if accurate, however, attempting to iterate within the context of an optimal control solver when determination of the state dynamics at every step of every iteration includes multiple matrix multiplications and an inverse can result in poor performance and numeric instability.

5.2.3 Information States and Associated Dynamics.

The principles of the FIM can be used to address this problem. The FIM contains all of the required directional information necessary to direct the optimal path planner, so a method for inserting that information into the context of the optimal control problem was developed as follows. Equation 73 defining the FIM for this application is repeated here for convenience:

$$\boldsymbol{\mathcal{I}}_k = \boldsymbol{\mathcal{I}}_0 + \frac{1}{\sigma_\beta^2}\begin{bmatrix} \sum_{i=1}^{k} \frac{\sin^2\beta_i}{\rho_i^2} & -\sum_{i=1}^{k} \frac{\sin\beta_i \cos\beta_i}{\rho_i^2} \\ -\sum_{i=1}^{k} \frac{\sin\beta_i \cos\beta_i}{\rho_i^2} & \sum_{i=1}^{k} \frac{\cos^2\beta_i}{\rho_i^2} \end{bmatrix} \tag{80}$$

Allowing the assumptions that measurements will continue to be received every Δt_{meas} seconds, and that the standard deviation of each measurement, σ_β, is constant for all measurements, an integral may be used to approximate the discrete steps of the measurement updates, similar to the Euler-Maclaurin formula:

$$\boldsymbol{\mathcal{I}}_k \approx \boldsymbol{\mathcal{I}}\left(t\right)\big|_{t=t_k} \equiv \boldsymbol{\mathcal{I}}_0 + \begin{bmatrix} \int_{t_0}^{t_k} \frac{\sin^2\beta(t)}{\Delta t_{meas}\sigma_\beta^2\rho^2(t)}\, dt & -\int_{t_0}^{t_k} \frac{\sin\beta(t)\cos\beta(t)}{\Delta t_{meas}\sigma_\beta^2\rho^2(t)}\, dt \\ -\int_{t_0}^{t_k} \frac{\sin\beta(t)\cos\beta(t)}{\Delta t_{meas}\sigma_\beta^2\rho^2(t)}\, dt & \int_{t_0}^{t_k} \frac{\cos^2\beta(t)}{\Delta t_{meas}\sigma_\beta^2\rho^2(t)}\, dt \end{bmatrix} \tag{81}$$

Recalling that $\boldsymbol{\mathcal{I}}_k = \mathbf{P}_k^{-1}$ for an efficient estimator, continuous information states, $\xi_i(t)$, are defined based on the elements of this FIM approximation such that:

$$\begin{aligned} \xi_1(t) &\equiv \left[\mathbf{P}^{-1}(t_0)\right]_{11} + \int_{t_0}^{t} \frac{\sin^2\beta(t)}{\Delta t_{meas}\sigma_\beta^2\rho^2(t)}\, dt \\ \xi_2(t) &\equiv \left[\mathbf{P}^{-1}(t_0)\right]_{12} + \int_{t_0}^{t} \frac{\cos^2\beta(t)}{\Delta t_{meas}\sigma_\beta^2\rho^2(t)}\, dt \\ \xi_3(t) &\equiv \left[\mathbf{P}^{-1}(t_0)\right]_{22} - \int_{t_0}^{t} \frac{\sin\beta(t)\cos\beta(t)}{\Delta t_{meas}\sigma_\beta^2\rho^2(t)}\, dt \end{aligned} \tag{82}$$

where $\left[\mathbf{P}^{-1}(t_0)\right]_{ij}$ refers to the ijth component of the matrix at time t_0. Clearly:

$$\boldsymbol{\mathcal{I}}(t) = \boldsymbol{\mathcal{I}}_0 + \begin{bmatrix} \int_{t_0}^{t} \dot{\xi}_1\left(t\right)\ dt & \int_{t_0}^{t} \dot{\xi}_3\left(t\right)\ dt \\ \int_{t_0}^{t} \dot{\xi}_3\left(t\right)\ dt & \int_{t_0}^{t} \dot{\xi}_2\left(t\right)\ dt \end{bmatrix} \tag{83}$$

The dynamics of the information states can then be found from the derivative of the FIM approximation:

$$\frac{d\boldsymbol{\mathcal{I}}(t)}{dt} = \begin{bmatrix} \dot{\xi}_1(t) & \dot{\xi}_3(t) \\ \dot{\xi}_3(t) & \dot{\xi}_2(t) \end{bmatrix} \tag{84}$$

With the dynamics available, and the initial conditions found from the inverse of the initial covariance matrix, the information states may be appended onto the state vector in the optimal control problem. The approximate FIM may then be formulated at any point in time, the inverse of which should yield a close approximation to the covariance at that time. Looking forward using results of the actual flight tests, Figure 22 shows a qualitative example of the accuracy of the approximation by post processing flight test data from Run #1, the first run with an actual wire. The approximation data are covariance elements calculated with the inverse of the

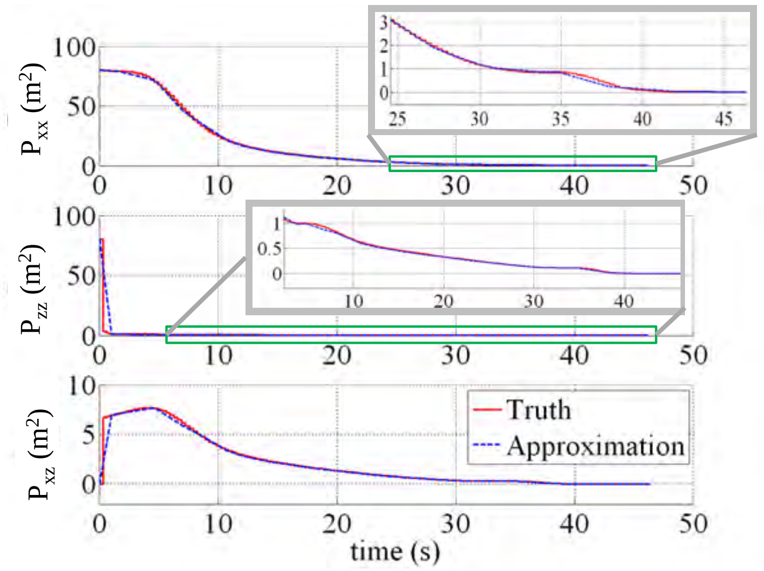


Figure 22. Ability to Accurately Approximate Covariance with Information States, Flight Test Run #1

approximate FIM, which was assembled from the information states. The truth data are generated by applying the Extended Kalman Filter measurement equations in the Cartesian formulation at the measurement update times:

$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_{k|k-1}\mathbf{H}_k^T\left[\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + R\right]^{-1}\\ \mathbf{P}_{k|k} &= \mathbf{P}_{k|k-1} - \mathbf{K}_k\mathbf{H}_k\mathbf{P}_{k|k-1}\end{aligned} \tag{85}$$

By constructing the FIM from the information states and taking its inverse, this method provides a way to bring the information contained in the error covariance matrix into the context of the optimal problem in a manner that provides a gradient for how to change the path to affect certainty directionally. In this manner, with some considerations that are addressed in Section 5.3, the error uncertainty at the final time—the true mission requirement for the bearing-only systems addressed in this dissertation—can now by explicitly prescribed through a multi-state boundary condition.

5.3 Optimal Control Problem Formulation

The optimal control problem for each epoch of the real-time trajectory planner can now be formulated using an augmented state vector:

$$\tilde{\mathbf{x}} = \begin{bmatrix} x & z & v_x & v_z & \xi_1 & \xi_2 & \xi_3 \end{bmatrix}^T \tag{86}$$

Control is as defined in Equation 17 on page 46 with the limitations described there. In Bolza form, the optimal control problem is to determine the state-control function pair, $\{\tilde{\mathbf{x}}(t), \mathbf{u}(t)\}$, and final time, t_f (in this case t_0 is known for each epoch), which minimize the cost functional:

$$J = \mathbf{\Gamma}\left(\tilde{\mathbf{x}}\left(t_0\right), t_0, \tilde{\mathbf{x}}\left(t_f\right), t_f\right) + \int_{t_0}^{t_f} \boldsymbol{\mathcal{L}}\left(\tilde{\mathbf{x}}\left(t\right), \mathbf{u}\left(t\right), t\right)\; dt \tag{87}$$

subject to the dynamic constraints:

$$d\tilde{\mathbf{x}}/dt = \boldsymbol{f}\left(\tilde{\mathbf{x}}\left(t\right), \mathbf{u}\left(t\right), t\right) \tag{88}$$

the path constraints:

$$\mathbf{C}\left(\tilde{\mathbf{x}}\left(t\right), \mathbf{u}\left(t\right), t_0, t_f\right) \geq \mathbf{0} \tag{89}$$

and the boundary conditions:

$$\boldsymbol{\gamma}\left(\tilde{\mathbf{x}}\left(t_0\right), t_0, \tilde{\mathbf{x}}\left(t_f\right), t_f\right) \geq \mathbf{0} \tag{90}$$

with equality constraints imposed via a second constraint on the additive inverse.

The advantage to this new method of incorporating final covariance as an event constraint (a multi-state boundary condition) in the optimal control problem is that a general performance index can be used to best fit the situation. Note that the final time should remain free. Previous methods have defined a fixed-final-time horizon, or have implicitly done so by fixing the number of measurements. A free final time allows alteration of the number of measurements received, which can greatly impact the solution. The vehicle must have the ability to slow down in an area (or lengthen the portion of the path that is in a certain direction for fixed velocity problems) if more measurements are required from that aspect angle.

For the sUAS landing-on-a-wire scenario, the final time was selected to be minimized with:

$$\mathbf{\Gamma} = t_f \tag{91}$$

The states are free to move without penalty within the limitations of $\mathbf{C}$, but weighting could easily be added in other applications for best possible tracking or avoidance of areas while still gaining the required certainty for mission accomplishment. A small penalty was added on control:

$$\boldsymbol{\mathcal{L}}(t) = \mathbf{u}^T(t)\mathbf{W}_u\mathbf{u}(t) \tag{92}$$

with the weights in $\mathbf{W}_u$ set to 0.1 on each diagonal element. The addition of a small weight on control is an effective method of avoiding numerical instabilities associated with optimal problems posed on a *singular arc*.

5.3.1 Avoidance of the Singular Arc.

A brief analytical look at the problem sheds light on the singular arc issue, a recurring issue for many numerical problems. Constraints will be detailed in the next section, but for now, none of the constraints in this particular formulation include a combination of states and controls, and they are not functions of the initial or final time, allowing them to be split into constraints on the state vector and constraints on the controls, respectively:

$$\mathbf{C}\left(\tilde{\mathbf{x}}\left(t\right), \mathbf{u}\left(t\right), t_0, t_f\right) = \left\{\mathbf{C}^{\tilde{\mathbf{x}}}\left(\tilde{\mathbf{x}}(t)\right), \mathbf{C}^{\mathbf{u}}\left(\mathbf{u}(t)\right)\right\} \tag{93}$$

Defining Lagrange multipliers, $\lambda_i(t)$, and the unit Heaviside step function:

$$\mathbb{H}\left(-C_i\right) = \begin{cases} 0, & \text{for } \mathrm{C}_i\left(\tilde{\mathbf{x}}\left(t\right)\right) \geq 0 \\ 1, & \text{for } \mathrm{C}_i\left(\tilde{\mathbf{x}}\left(t\right)\right) < 0 \end{cases} \tag{94}$$

The Hamiltonian, $\mathcal{H}$, can then be defined using the variational approach for problems with state variable inequality constraints in [66] by defining a new state variable:

$$\dot{x}_8(t) \equiv \left[C_1^{\tilde{\mathbf{x}}}\left(\tilde{\mathbf{x}}\left(t\right)\right)\right]^2 \mathbb{H}(-C_1^{\tilde{\mathbf{x}}}) + \left[C_2^{\tilde{\mathbf{x}}}\left(\tilde{\mathbf{x}}\left(t\right)\right)\right]^2 \mathbb{H}(-C_2^{\tilde{\mathbf{x}}}) + \cdots + \left[C_{n_{c\tilde{\mathbf{x}}}}^{\tilde{\mathbf{x}}}\left(\tilde{\mathbf{x}}\left(t\right)\right)\right]^2 \mathbb{H}(-C_{n_{c\tilde{\mathbf{x}}}}^{\tilde{\mathbf{x}}}) \tag{95}$$

for the $n_{c\tilde{\mathbf{x}}}$ constraints in $\mathbf{C}^{\tilde{\mathbf{x}}}$. The derivative of $x_8(t)$ is always positive, and the value for the state is kept at zero by enforcing boundary conditions of $x_8(t_0) = 0$ and $x_8(t_f) = 0$, thereby enforcing the state inequality constraints for all time.

The Hamiltonian for the now $n + 1$ states can then be expressed as:

$$\begin{aligned}
\mathcal{H}(\tilde{\mathbf{x}}(t), &\mathbf{u}(t), \boldsymbol{\lambda}(t), t) = \boldsymbol{\mathcal{L}}\left(\tilde{\boldsymbol{x}}\left(t\right), \mathbf{u}(t), t\right) + \lambda_1(t) f_1\left(\tilde{\mathbf{x}}(t), \mathbf{u}(t), t\right) \\
&+ \cdots + \lambda_n(t) f_n\left(\tilde{\mathbf{x}}(t), \mathbf{u}(t), t\right) \\
&+ \lambda_{n+1}(t)\left[C_1^{\tilde{\mathbf{x}}}\left(\tilde{\mathbf{x}}\left(t\right)\right)\right]^2 \mathbb{H}(-C_1^{\tilde{\mathbf{x}}}) + \cdots + \left[C_{n_{c\tilde{\mathbf{x}}}}^{\tilde{\mathbf{x}}}\left(\tilde{\mathbf{x}}\left(t\right)\right)\right]^2 \mathbb{H}(-C_{n_{c\tilde{\mathbf{x}}}}^{\tilde{\mathbf{x}}}) \\
&\equiv \boldsymbol{\mathcal{L}}\left(\tilde{\boldsymbol{x}}\left(t\right), \mathbf{u}\left(t\right), t\right) + \boldsymbol{\lambda}^T(t) \boldsymbol{f}\left(\tilde{\mathbf{x}}\left(t\right), \mathbf{u}\left(t\right), t\right)
\end{aligned} \tag{96}$$

In a case where control was unconstrained, the sufficient optimality condition for control would yield:

$$\frac{\partial \mathcal{H}(\tilde{\mathbf{x}}^*(t), \mathbf{u}^*(t), \boldsymbol{\lambda}^*(t), t)}{\partial \mathbf{u}} = \mathbf{0} \tag{97}$$

As the controls are constrained for this problem by $\mathbf{C}^{\mathbf{u}}$, which defines the admissible controls $\mathbf{u} \in \mathbf{U}$, Pontryagin's maximum principle (or minimum in this case) must be applied:

$$\mathbf{u}^* = \underset{\mathbf{u}^* \in \mathbf{U}}{\arg\min}\, \mathcal{H} \tag{98}$$

Without the addition of the quadratic term that was added in $\boldsymbol{\mathcal{L}}$, none of the control terms in the Hamiltonian are higher than first-order, meaning that:

$$\frac{\partial^2 \mathcal{H}}{\partial \mathbf{u}^2} = 0 \tag{99}$$

Because $\partial^2\mathcal{H}/\partial\mathbf{u}^2$ is singular, $\mathbf{u}$ is not uniquely defined by the optimality condition—this is the definition of a singular arc [7]. There are several methods of dealing with singular arcs, but most of them involve substantial insight into the shape of the optimal solution, taking time derivatives of $\partial\mathcal{H}/\partial\mathbf{u}$ until the control does show up, or reformulating the problem into one without a singular arc. Many numeric methods rely on the Hessian for direction and step size information. Adding a very light control cost, as in Equation 92, can eliminate much of the volatility that can be associated with numeric optimal solutions on a singular arc. If there is no noticeable change to the optimal trajectory, or if the changes are acceptable for the system in question, this technique provides a simple method for smoothing the control solution provided by numeric solvers.

5.3.2 Constraints.

The Dynamic constraints, $\boldsymbol{f}$, of Equation 88 were defined by Equations 9, 17, and 84. Path constraints were applied to scale the problem within the physical limitations of the available indoor flight test facility in order to make use of the Vicon motion capture system, described in Chapter VIII. In practice, the optimization software used required inequality constraints on all states and controls. Variables not intended to be constrained had constraint values set well out of a realistic range, but not at infinity to keep gradients meaningful. The potentially active constraints of $\mathbf{C}$ are shown in a consolidated notation:

$$\begin{bmatrix} -9\,m \\ 0.8\,m \\ -0.5\,m/s \\ -0.5\,m/s^2 \\ -30^\circ \end{bmatrix} \leq \begin{bmatrix} x \\ z \\ v_x, v_z \\ u_x, u_z \\ \hat{\beta} \end{bmatrix} \leq \begin{bmatrix} x_{\text{app\_offset}} + \hat{x}_t \\ 5.5\,m \\ 0.5\,m/s \\ 0.5\,m/s^2 \\ 40^\circ \end{bmatrix} \tag{100}$$

The forward horizontal component, x, was limited to stay within the boundaries of the indoor flight test facility and the expected approach point defined relative to the wire position estimate, $x_{\text{app\_offset}} + \hat{x}_t$. The approach point itself is only an estimate, changing each epoch, but it does provide some safety buffer until the desired target certainty is reached. Vertical limits were set so that the landing gear would clear the floor, and the "ceiling" limit ensured that the vehicle would stay low enough to remain visible by a sufficient number of Vicon cameras. For the true sUAS scenario, the upper "ceiling" limit could be removed in the absence of airspace limitations, simplifying the problem for the optimal solver. The vertical "ground" limit could be replaced with a terrain model or a min-safe altitude for terrain avoidance, as appropriate.

The vehicle speed was also limited, increasing the total engagement time to make it representative of an actual approach. This allowed for a realistic test of the ability of the RTOC system to control in real-time despite the inherent computational delays. The path constraint on $\hat{\beta}$ was intended to keep the vehicle in a position for the fixed camera to maintain the wire within the FOV. The hat notation is kept to denote that the value is calculated using the current target estimate, as the true FOV limits are not known. No measurements are received when outside the true FOV. For the quadrotor, this is always the case as the vehicle transitions to land-mode and flies underneath the wire, but could potentially happen during the flight due to disturbances or a bad target estimate. If possible for a full system, it is recommended that the camera and hooking method be designed to keep the wire within the sensor FOV until connected, to allow the ability to correct for swinging wires, wind gusts, and other endgame disturbances.

5.3.3 Boundary Conditions and Formulation of Final Covariance Constraints.

The solution of the optimal control problem is iterative. Initial conditions for each epoch are not the current conditions, but the *expected* conditions at the time the next solution is expected to become available. Based on experience with current hardware, a complete loop time, $\Delta t_{calc} = 0.9$ seconds, is assumed inclusively for the optimization problem, the estimation problem, and all transport delays. The very first solution is seeded with zeros. After one solution exists, position and velocity initial conditions are taken from the time history of the previous epoch's solution, $\tilde{\mathbf{x}}_k(t)$, propagated forward by Δt_{calc}:

$$\begin{bmatrix} \mathbf{x}_{0_{k+1}} & \mathbf{v}_{0_{k+1}} \end{bmatrix}^T = \begin{bmatrix} \mathbf{x}_k(t + \Delta t_{calc}) & \mathbf{v}_k(t + \Delta t_{calc}) \end{bmatrix}^T \tag{101}$$

Care must be exercised when initializing the information states, as they are only estimates of the true FIM components, based on the assumption that measurements will be consistently received with a fixed time interval. The realities of processing delays, poor image backgrounds, and hardware issues in general may lead to slow or skipped measurements. This information must be incorporated, or the accuracy of the information state estimates will drift over time. As a result, the initial conditions for the information states are reset each epoch based on the actual covariance from the estimation filter, $\mathbf{P}_k$, propagated forward by Δt_{calc}. To do so, an expected measurement time vector is created based on the actual reception time of the last measurement, t_{last_meas}:

$$t_{meas} = [t_{last\_meas} + \Delta t_{meas}, t_{last\_meas} + 2\Delta t_{meas}, \ldots, t_{last\_meas} + n\Delta t_{meas}] \tag{102}$$

where n represents the maximum number of measurements that can be incorporated such that:

$$t_{last\_meas} + n\Delta t_{meas} \leq t_{0_{k+1}} \tag{103}$$

The expected relative states at each of these times for epoch k are then found, $\mathbf{x}_r(t_{meas_i})$, $i = 1 \ldots n$. At each point, the Jacobian is produced with Equation 71, and the EKF update is recursively performed with Equation 85. The result is $\mathbf{P}_{0_{k+1}} = \boldsymbol{\mathcal{I}}_{0_{k+1}}^{-1}$, and the elements of the inverse are used as the initial conditions for each of the information states.

For the terminal conditions in Equation 90, the first four constraints of $\boldsymbol{\gamma}$ take the system to a hover at the approach point:

$$\begin{bmatrix} x_{k+1}(t_f) \\ z_{k+1}(t_f) \\ v_{x_{k+1}}(t_f) \\ v_{z_{k+1}}(t_f) \end{bmatrix} = \begin{bmatrix} x_{\text{app\_offset}} + \hat{x}_k \\ z_{\text{app\_offset}} + \hat{z}_k \\ 0 \\ 0 \end{bmatrix} \tag{104}$$

For the information states, the physical considerations of hook size, in addition to the steady-state uncertainty of the vehicle's own-ship position estimate determine the final certainty needs. For this particular hook design, uncertainty was best described by setting $\mathrm{P}_{xx_{max}}$ and $\mathrm{P}_{zz_{max}}$ at the final time, but any shape covariance ellipsoid could be specified based on system requirements.

To apply the terminal covariance conditions in terms of the information states, the elements were found with inverse relationships:

$$\mathrm{P}_{xx}(t_f) = \xi_2(t_f) \,/\, \left[\xi_1(t_f)\xi_2(t_f) - \xi_3^2(t_f)\right] \leq \mathrm{P}_{xx_{max}} \tag{105}$$

$$\mathrm{P}_{zz}(t_f) = \xi_1(t_f) \,/\, \left[\xi_1(t_f)\xi_2(t_f) - \xi_3^2(t_f)\right] \leq \mathrm{P}_{zz_{max}} \tag{106}$$

Care must be taken in the application of these boundary conditions, as the denominator can be near singular. This makes taking the gradients of the constraint problematic for the numerical solution and can lead to instability. To avoid this, the constraint can be re-written by noting that the denominator is positive. *Proof*: For t_f, $\Delta t_{meas}\,, \sigma_\beta \in \mathbb{R}^1\,(0,\infty)$, incorporating the assumption that the vehicle has not hit the target, $x_r, z_r \in \mathbb{R}^1\,(-\infty,\infty) : |x_r| + |z_r| \neq 0$, and assuming the system is initialized with some estimate of initial information with no initial cross-correlation, $\xi_1, \xi_2 \in \mathbb{R}^1\,[0,\infty) : \xi_1(0) = \xi_{1_0},\ \xi_2(0) = \xi_{2_0}$, and $\xi_3 \in \mathbb{R}^1\,(-\infty,\infty) : \xi_3(0) = 0$, then for the finite time span $\Omega = [0, t_f]$, the information states are defined everywhere on Ω:

$$\begin{aligned} \xi_1(t) &= \xi_{1_0} + \int_\Omega \frac{\sin^2\beta(t)}{\rho^2(t)}\,dt < \infty \\ \xi_2(t) &= \xi_{2_0} + \int_\Omega \frac{\cos^2\beta(t)}{\rho^2(t)}\,dt < \infty \\ &\Rightarrow\ \xi_1, \xi_2 \in L_2^\Omega \end{aligned} \tag{107}$$

therefore,the Cauchy-Schwarz inequality may be used to show:

$$\xi_1(t)\xi_2(t) \geq \int_\Omega \left(\frac{\sin\beta(t)}{\rho(t)}\right)^2 dt \cdot \int_\Omega \left(\frac{\cos\beta(t)}{\rho(t)}\right)^2 dt \geq \left(\int_\Omega \frac{\sin\beta(t)\cos\beta(t)}{\rho^2(t)}\,dt\right)^2 \tag{108}$$

$$\Rightarrow\ \xi_1(t)\xi_2(t) - \xi_3^2(t) \geq 0 \qquad \forall t \in \Omega$$

A singular denominator would mean an infinite uncertainty, a condition that cannot be returned to after the finite initialization, implying a strict inequality:

$$\xi_1(t)\xi_2(t) - \xi_3^2(t) > 0 \qquad \forall t \in \Omega \qquad\qquad \square \tag{109}$$

With a positive denominator, the constraints may be rewritten in $\boldsymbol{\gamma}$ to avoid any numeric instability as:

$$\begin{aligned} \mathrm{P}_{xx_{max}} \left[\xi_1(t_f)\xi_2(t_f) - \xi_3^2(t_f)\right] - \xi_2(t_f) \geq 0 \\ \mathrm{P}_{zz_{max}} \left[\xi_1(t_f)\xi_2(t_f) - \xi_3^2(t_f)\right] - \xi_1(t_f) \geq 0 \end{aligned} \tag{110}$$

VI. RTOC Structure—Requirement for Integrated Error Feedback

"In theory, there is no difference between theory and practice. . . In practice, there is."

THIS chapter develops the feedback structure that should be used in real-time optimal controllers, particularly focusing on the area of adding error integration into the recursive formulation—a technique that has been declared unnecessary in much of the current research in this relatively new field. The shortcomings of this approach are shown through two case studies. The first case study is made simple enough to allow an analytical expression of the error caused by choosing to use only a fast open-loop recursive structure, the common approach in recent studies. Two more appropriate RTOC structures are suggested, and the second case study implements one of them in a scenario likely to benefit from RTOC—aircraft attack planning in the context of pop-up surface threats and stochastic disturbances.

6.1 Fast Recursive Open-Loop Control vs. Closed-Loop Feedback

The concept of RTOC is simple. Optimal solutions are desired for control, but the solutions are only optimal for deterministic problems. If any of the assumed parameters in the problem are inaccurate (target position, wind, etc.), the solution provided is most likely not optimal, and may no longer be valid for mission accomplishment. If a new solution could be provided fast enough, however, the optimal trajectory could be updated recursively with the most current parameter estimates. This method typically includes "bootstrapping" the previous optimal solution as the guess for the next epoch, significantly decreasing computation time. The idea of a recursive open-loop solution is compelling—once perturbed from the initial optimal path, why waste control effort with a feedback loop working back toward the original reference trajectory? Why not find the most optimal control *now*, and apply that? Figure 23 illustrates the situation.

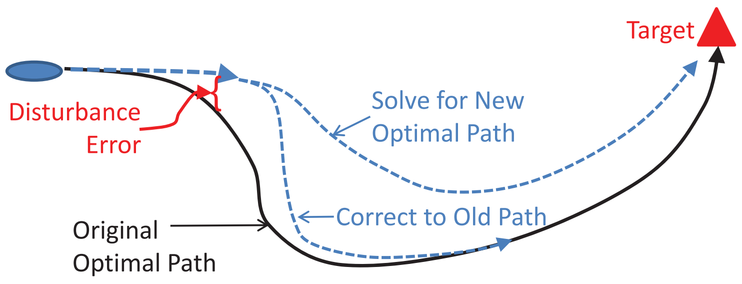


Figure 23. Decision to Follow Initial Optimal Trajectory, or to Re-solve the Optimal Path from the Current Condition

Once the state is perturbed from the expected optimal path, correcting back to that trajectory is likely not optimal from the disturbed position, and a new path originating from the current state should be introduced. Re-solving the optimal control problem as often as possible, and maintaining that reference path between optimal path updates with a faster, inner control loop results in a two degree-of-freedom design, such as the one shown in Figure 24, which can be found in similar forms in [80] and [106].

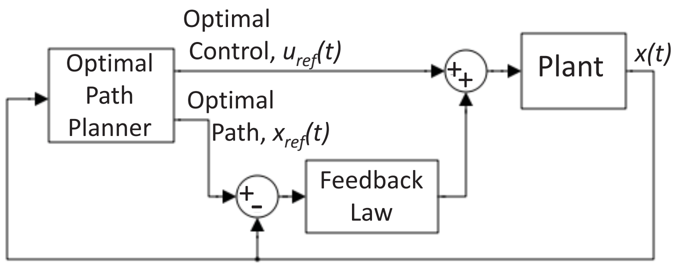


Figure 24. Two Degree-of-Freedom Control Scheme

Recently, several authors have taken the speed advantages of efficient optimization techniques and increased processing power to move the control concept one step further—eliminating the inner loop altogether and controlling in a purely recursive open-loop manner. Conceptually, if you have control at any point that you have defined as optimal, why would you add anything to it? It is tempting to draw the conclusion that if the recursive open-loop optimal control can just be solved fast enough, there is no need for feedback, or that recursive open-loop control can be equated to feedback control. This proposition is a current trend in the literature for RTOC structure design. Consider the comparison of open-loop recursion with closed-loop control in some of those pushing the state of the art in the field of RTOC:

> "The feedback law is not analytically explicit; rather, closed-loop control is obtained by a rapid re-computation of the open-loop time-optimal control at each update instant." [55]

In simulated satellite guidance, again suggesting that rapid open-loop control would provide optimal disturbance rejection of closed-loop feedback:

> "A conceptually simple approach to controlling such non-linear systems is by solving the problems online. If such problems can be solved online, there is no need for an off-line design of closed-form feedback laws as, by definition, the control system would have acquired this intelligence....Rather than tracking a pre-computed solution, the control scheme proposed in this paper re-solves the optimal control problem and updates the control command as soon as a new solution is obtained. This results in a sampled-data feedback law which provides optimality in the presence of various types of disturbances." [100]

For simulated re-entry vehicle control:

"The key for successful implementation of these feedback principles relies on a sufficiently fast generation of open-loop controls. Thus, if open-loop controls can be generated as demanded by [a given speed requirement for his problem], closed-loop is achieved quite simply." [10]

In a foundational work on RTOC:

"Suppose optimal open-loop controls could be computed in real time. This implies optimal feedback control." [95]

and elsewhere:

"It has been known since the birth of optimal control that if open-loop controls can be generated in real-time, they are basically equivalent to feedback controls." [106]

The concept that fast open-loop solutions equate to closed-loop feedback controls, with the elimination of the inner loop of Figure 24, has become pervasive. While there certainly is a level of feedback that is implicitly achieved with a recursive open-loop structure, it falls far short of "optimal feedback control" in an environment with any true stochastic inputs, as will be shown below.

6.2 Lack of Error Integration in Instantaneous Optimal Solutions

The purely recursive open-loop structure has shown success in simulations for the above problems, but lessons from classical control theory suggest significant limitations of this approach. The single degree-of-freedom design—removing the inner feedback loop—recursively provides an instantaneous optimal solution (future time history) for the control and state (the faster, the better, in theory). While valuable, if the designer of a RTOC system makes the assumption that rapid open-loop solutions yield the same performance as traditional feedback control, the resulting design will fail to leverage the information that can be gleaned from comparing the historical efforts to outcomes.

The whole question of RTOC implies that there are disturbances or unmodeled effects to be rejected, else the optimal solution would only need to be found once, rather than in real-time. Recursively solving the problem gives freedom to respond to stochastic or unanticipated effects. Especially for cases where these disturbances end up not falling into the classic categories of Gaussian, white, and zero-mean, integration of the error between the expected and actual state and control history can supply either additional compensation, or a more accurate model of the true system dynamics through estimation of the disturbance. If the likely errors for the system are indeed non-zero mean, or at least time correlated (and thus likely non-zero mean over some time interval), these effects should be accounted for in selection of the control. This requires one of many methods of feedback control that are not achieved with a purely recursive open-loop design. Two non-linear optimal control problems are posed to demonstrate this principle. The first is an overly-simplified course guidance problem to allow analytic proof of the error. The second case study will address corrective implementation in a realistic scenario.

6.3 Case Study A: Simplified Aircraft Course Planning

Consider an aircraft simply modeled as a point mass system with rectilinear position components:

$$\mathbf{x}_{ac}(t) = \begin{bmatrix} x_{ac}(t) \\ y_{ac}(t) \end{bmatrix} \tag{111}$$

The aircraft is flying at a constant altitude, with a constant velocity, V_{ac}. The pilot has been cleared direct to a waypoint, or fix, $(x_{ac_f},\ y_{ac_f})$, and is using the autopilot to provide course guidance. The system dynamics are simply:

$$\dot{\mathbf{x}}_{ac}(t) = \begin{bmatrix} V_{ac}\cos\psi(t) \\ V_{ac}\sin\psi(t) \end{bmatrix} \tag{112}$$

where aircraft heading, $\psi(t)$, is the control variable. Turn dynamics are ignored for simplicity.

The optimal control problem is a two-point boundary value problem, with a minimum time performance index presented in Mayer formulation:

$$J_{ac} = t_f \tag{113}$$

Assigning $\boldsymbol{\lambda}(t) \in \mathbb{R}^2$ as a vector of Lagrange multipliers, the Hamiltonian is defined as:

$$\mathcal{H}(\mathbf{x}_{ac}(t), \psi(t), \boldsymbol{\lambda}(t), t) = \lambda_1(t)V_{ac}\cos\psi(t) + \lambda_2(t)V_{ac}\sin\psi(t) \tag{114}$$

The first-order necessary conditions provide the costate equations:

$$\begin{aligned} -\frac{d\mathcal{H}}{dx_{ac}} &= \dot{\lambda}_1^*(t) = 0 \\ -\frac{d\mathcal{H}}{dy_{ac}} &= \dot{\lambda}_2^*(t) = 0 \end{aligned} \tag{115}$$

The optimality condition for the unconstrained control provides:

$$\frac{d\mathcal{H}}{d\psi} = 0 = -\lambda_1^*(t)V_{ac}\sin\psi^*(t) + \lambda_2^*(t)V_{ac}\cos\psi^*(t) \tag{116}$$

$$\Rightarrow \frac{\lambda_2^*(t)}{\lambda_1^*(t)} = \tan\psi^*(t) \tag{117}$$

The optimal control is therefore constant, implying for this case that the state dynamics are constant, which allows a solution for the optimal control through simple integration of both states from the initial state conditions $\mathbf{x}_{ac}(0) = [x_{ac_0} \;\; y_{ac_0}]^T$:

$$\begin{aligned}\mathbf{x}^*_{ac_f} &= \mathbf{x}_{ac_0} + \int_0^{t_f} \dot{\mathbf{x}}^*_{ac}(t)\; dt \\ &= \mathbf{x}_{ac_0} + t_f \begin{bmatrix} V_{ac}\cos\psi^* \\ V_{ac}\sin\psi^* \end{bmatrix}\end{aligned} \tag{118}$$

The unknown final time is removed by solving both equations for t_f and equating them, leaving the optimal control:

$$\psi^*(t) = \tan^{-1}\left(\frac{y_{ac_f} - y_{ac_0}}{x_{ac_f} - x_{ac_0}}\right) \tag{119}$$

Note that for recursive open-loop control, the initial values in Equation 119 are simply the current position for each iteration, and the optimal control solved for by any method will simply be a function of the relative position ratio. Absent disturbances, the optimal path, and the actual path, will unsurprisingly be direct to the target as shown in Figure 25.

6.3.1 Addition of Stochastic Disturbances.

As they are unknown beforehand, the addition of the typical zero-mean, white, Gaussian, stochastic elements in the forms of model deficiencies or disturbances does not change the predicted solution for the optimal control. The effects of disturbances can be countered, somewhat, by re-solving for a new optimal path at various time steps, as was illustrated in Figure 23. However, the production of a new, instantaneous solution does not provide anticipation of future disturbance effects, or any correction for past errors or modeling discrepancies. For unmodeled effects which are

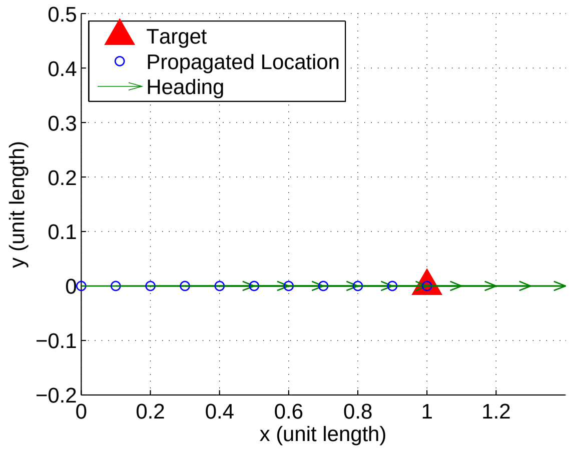


Figure 25. Recursive Optimal Control Solution with No Disturbances, Δt=0.1 Units

more time correlated—or those that can be characterized by an unknown, non-zero mean—effective control requires some level of feedback, such as integration of the error between the expected and actual state path for each step, or estimation of the unknown parameter(s) causing the disturbance.

To illustrate this, a constant bias, w, is added to the system in one axis. This bias represents some of the effects of a wind component parallel to that unit direction. Smaller stochastic effects of the wind are not modeled for this case study in order to more clearly show the predominant impact and to provide the opportunity for an analytical solution. The effects of a time-correlated noise source can be seen by simply replacing the experiment with a correlated function, $w(t)$. Even a time-correlated function that is zero-mean overall can be cut into segments of time where the mean is biased in one direction or the other, so the general effects of the noise contribution will be the same as demonstrated here, on smaller time scales.

The dynamics of Equation 112 become:

$$\dot{\mathbf{x}}_{ac}(t) = \begin{bmatrix} V_{ac}\cos\psi(t) \\ V_{ac}\sin\psi(t) + w \end{bmatrix} \tag{120}$$

and the Hamiltonian is updated to be:

$$\mathcal{H}(\mathbf{x}_{ac}(t), \psi(t), \boldsymbol{\lambda}(t), t) = \lambda_1(t) V_{ac} \cos\psi(t) + \lambda_2(t) \left(V_{ac} \sin\psi(t) + w \right) \tag{121}$$

The costate equations do not change, and the Lagrange multipliers are still found to be constant. The optimality condition shows that the optimal control is constant as well, allowing integration of the states and removal of the unknown final time, leaving the relationship for the true optimal control, ψ_t^*:

$$\frac{y_{ac_f} - y_{ac_0}}{x_{ac_f} - x_{ac_0}} = \frac{V_{ac} \sin\psi_t^* + w}{V_{ac} \cos\psi_t^*} \tag{122}$$

The true optimal path is shown in Figure 26, with an arbitrary constant wind bias of -4 (unit length)/(unit time).

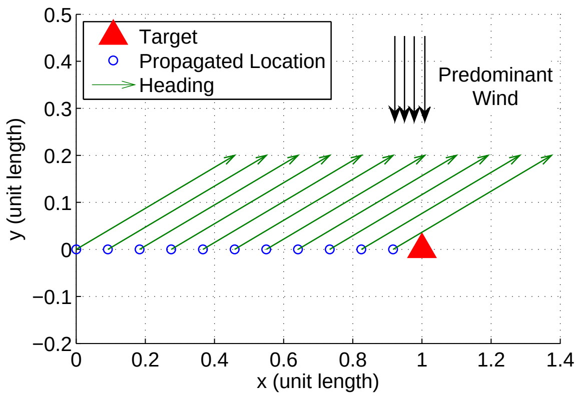


Figure 26. True Optimal Solution, with Non-Zero-Mean Disturbance, Δt=0.1 Units

If, however, the optimal steering is calculated without knowledge of the bias for each step of the digital controller, there obviously is error between the calculated optimal steering, ψ^*, and the true optimal steering, ψ_t^*. With the appropriate trigonometric identities, the instantaneous steering error from any point may be found by re-solving the optimal problem from the new initial location and defining:

$$\psi_e^* \equiv \psi^* - \psi_t^* = \sin^{-1}\left[w \bigg/ \left(V_{ac}\sqrt{1 + \frac{y_{ac_f} - y_{ac_0}}{x_{ac_f} - x_{ac_0}}}\right)\right] \tag{123}$$

This steering error results in a "homing" trajectory instead of a direct flight path, as shown in Figure 27a. The key point to emphasize is that this steering error will always exist (excepting a displacement in the direction of a pure head or tail wind). Note that Equation 123 is not dependent on sample time, or the speed of the recursive solution update, but only on the geometry of the problem at the time of the update and the intensity of the wind. A recursive open-loop solution will *always* produce a flawed steering solution, without the use of some sort of feedback to allow accounting for the wind bias. Attempts to increase the recursion rate may decrease the total path error, but never overcome the bias (analytically proven for this problem in Equation 123). Figure 27b shows a recursion rate of 0.01 time units.

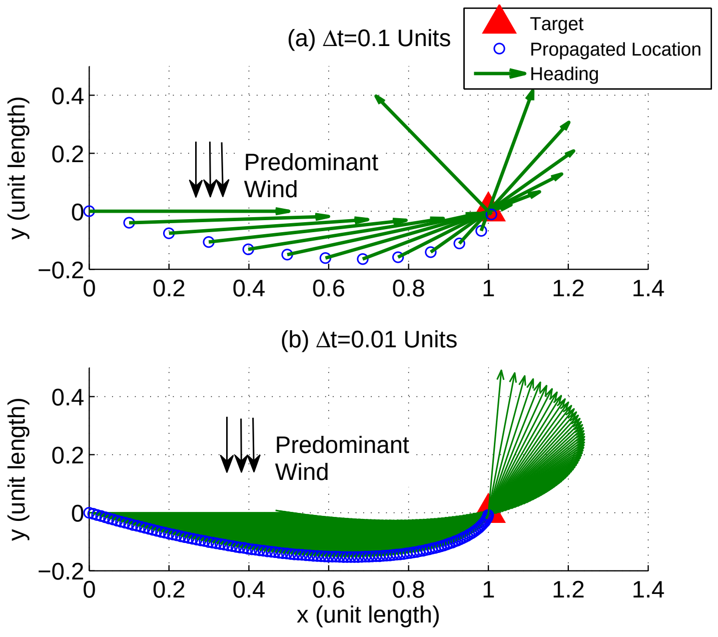


Figure 27. Recursive Optimal Solution with Non-Zero-Mean Disturbance (Homing)

These results highlight the main lesson of this chapter—the pitfall of assuming that a high recursion rate on an open-loop optimal solution is equivalent to optimal feedback control. In the face of non-zero mean disturbance, the resultant path in Figure 27 is clearly short of what would be considered "optimal." A simple feedback scheme demonstrates that the control solved for through rapid recursive open-loop planning requires additional input. Figures 28 and 29 show the effects of adding proportional-integral (PI) control in the form:

$$\psi_{fb}(t) = k_p e_p(t) + k_i \int_{t_0}^{t} e_p(\xi)\ d\xi \tag{124}$$

where $e_p(t)$ represents the orthogonal component between the current position and the intended direct path. The gains were arbitrarily selected as $k_p = -20$ (unit length)/radian and $k_i = -60$ (unit length)/radian. The command, ψ_c, then becomes:

$$\psi_c(t) = \psi^*(t) + \psi_{fb}(t) \tag{125}$$

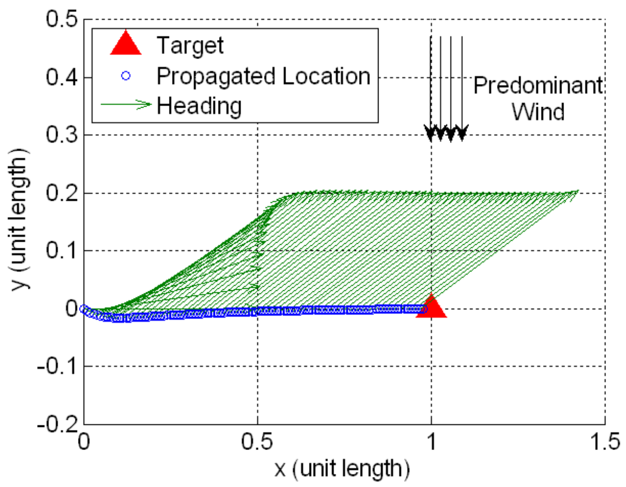


Figure 28. Optimal Recursion with the Addition of PI Feedback, Δt=0.01 Units

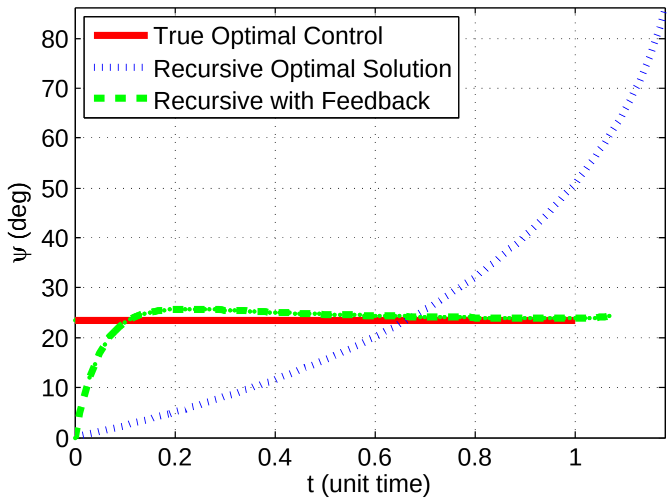


Figure 29. Control Requirements with and without Feedback

The addition of feedback to the recursive optimal solution causes the resultant path and control to more clearly follow the true optimal solution, regardless of the recursive update rate. In terms of total time-to-target (the objective), the analytical solution for this particular example took 1.092 time units to complete the route, very near to the 1.094 units for the recursive open-loop system with feedback, as compared to the 1.19 units with recursive open-loop updates only.

Beyond just the timing differences and the associated increase in fuel requirements, the arced path of the route found without an inner control loop has real-world navigation implications. On a regular basis under both instrument and visual flight rules, aircraft are assigned to "proceed direct" to a certain fix, or are filed to proceed along a route corridor by means of navigation aids (e.g. TACAN, VOR, etc.) which provide a bearing angle to a fix. In either case, separation from other aircraft, clearance of terrain, and line-of-sight for reception of the navigation aid signals is only protected for a narrow corridor width. The clearance to "proceed direct" implies correcting against the winds to fly a direct ground path, not merely homing to the target as you would with a recursive open-loop controller, which would result in the large lateral excursions illustrated in Figure 27. Could the RTOC approach be changed to minimize error from a direct path? Certainly, but again, this implies implementing some sort of explicit error feedback. The intent of this demonstration was to provide a counter-example to the concept that speeding up the open-loop recursion rate was equivalent to achieving optimal feedback control.

Therefore, in the design of control schemes to implement RTOC with a fast open-loop structure, consideration of the expected character of anticipated disturbances becomes critical. For systems that can anticipate time-correlated (at least relative to the system dynamics), or non-zero-mean disturbances, some sort of integral control is required to achieve near-optimum performance.

6.3.2 Error Integration through the Addition of Noise Estimates into the System Dynamics.

For this simple case study, adding feedback was straightforward, and an inner PI error loop around the planned and actual state paths was included. For more complex, highly non-linear systems, this technique may not be feasible. This is especially the case for systems with large deviations from the planned path as a result of a high ratio of disturbances to control authority, or systems with severe non-linearities that would require, for example, an inordinate amount of gain scheduling. A better method is to recognize that if you applied the optimal control and did not follow the expected optimal path, the dynamics of the model are not correct. Allowance for estimated error parameters found through path error integration can be added into the dynamics for the next epoch, in an effort to answer the "right" question.

For this application, this would involve first using the path error to form an estimate for the wind bias, $\widehat{w}(t)$, and then updating the dynamics equation for each recursive solution to include the current estimate for the wind. For more complicated scenarios, the effects of the disturbances on the dynamics could be estimated through both proportional and integrated error elements. For this simple case study, however, the optimal control estimate, $\widehat{\psi}^*$, can be solved analytically:

$$\widehat{\psi}^*(t_k) = \tan^{-1}\left(\frac{y_{ac\_f} - y_{ac}(t_k)}{x_{ac\_f} - x_{ac}(t_k)}\right) - \sin^{-1}\left[\widehat{w}(t_k) \Big/ \left(V_{ac}\sqrt{1 + \frac{y_{ac\_f} - y_{ac}(t_k)}{x_{ac\_f} - x_{ac}(t_k)}}\right)\right] \tag{126}$$

Again, this is an instantaneous solution at any time, t_k, used by substituting the current state into the original problem as new initial conditions. The effects are shown in Figure 30. Note that no attempt is made to return to any previous reference

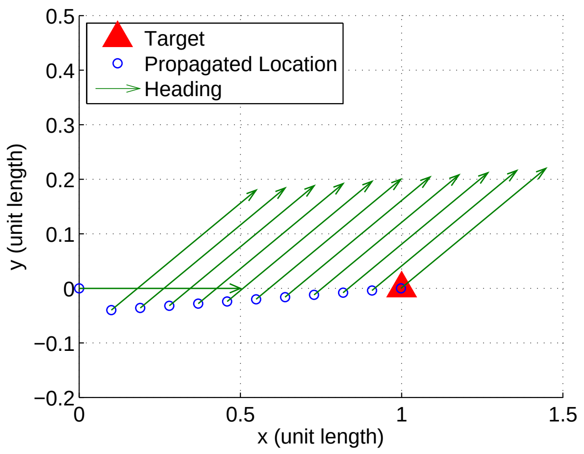


Figure 30. Recursive Optimal Control using Feedback to Update Dynamics

solution, but instead the system follows the optimal path that was calculated from each current position. Since the only disturbance that was added was constant, linear, and no measurement noise was considered, the estimate is correct after only one time step. Beyond that, the calculated solution matches the true optimal solution from that point, since all of the information about the disturbance is completely known.

Even in a realistic environment where the disturbances are changing, this final control structure represents the best of all worlds, combining the positive aspects of classical control with the emerging benefits of real-time optimal control. The control to be applied is completely generated by the numerical optimization scheme, but implicitly contains the integrated error feedback, which is used to update the system dynamics and change the optimal control problem for each iteration, overcoming the inability of the purely recursive open-loop structure to handle time-correlated or non-zero mean unmodeled effects.

6.4 Case Study B: Real-Time Aircraft Attack Planning

A more robust and realistic example quickly shows the potential impacts of a failure to consider error integration in recursive real-time optimal control. One of the most obvious applications for optimal path planning is for threat avoidance. Stealth considerations of radar cross section, threat radar detection capability, and effective surface-to-air missile (SAM) engagement ranges must be considered in attack planning. For maximum effectiveness, the plan should be accomplished in real time. Pop-up threats, by definition unanticipated, cannot be avoided using mission planning that was accomplished prior to take-off. In addition, without the ability to change the plan enroute, a pilot cannot immediately exploit weaknesses such as a defense system that has been removed or reduced in operational capability in some manner by another strike package. All of this is possible with RTOC.

Consider a strike planned on a soft target, defended with a perimeter of SAM threats along the planned route. For specific applications, the performance index would be designed to consider the specific capabilities of each threat and the advantages of the attacking aircraft, but as a generic illustration, a 15 nm (nautical mile) or 30 nm ring is assigned to each SAM location, and the run is constrained to a constant altitude with a constant 350 knots true air speed (TAS). The SAM ring represents a weapon employment zone, outside of which the aircraft can safely operate in the absence of air-to-air threats.

Admittedly avoiding minimum exposure and stealth issues (which could be incorporated with the appropriate modeling), the basis for the performance index for this attack scenario remains a Mayer cost function of final time, as it was for the case study in Equation 113. This equates to a minimum fuel consumption index for a constant altitude and airspeed run (if throttle increases during turns are considered negligible). Other options could include a penalty for proximity to threats, if flight was allowed within the threat rings. In practice, non-stealth aircraft pilots determine a safe distance from SAMs and stick to it, unless threat ring penetration is required.

The no-wind dynamics remain unchanged from Equation 112, and the control is still a commanded heading, which would be the input to a standard heading-hold autopilot with a feedback-based bank angle control law to drive the physical actuators. RTOC provides the flexibility of avoiding additional pop-up threats simply by adding new path constraints, ensuring flight outside of the SAM threat rings:

$$[x(t) - x_i]^2 + [y(t) - y_i]^2 \geq \rho_i^2 \qquad i = 1 \ldots n_{SAM} \tag{127}$$

where n_{SAM} is the number of currently known SAMs.

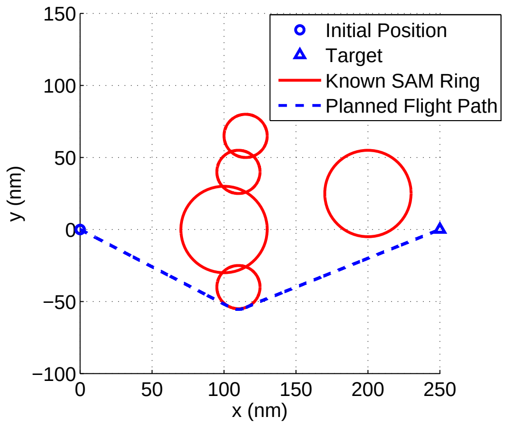
Control is accomplished by recursively solving the optimal control problem, with no explicit feedback (as before, some implicit feedback is available through the re-initializing of the optimal control problem at the current measured position). As previously stated, this mirrors the structure of RTOC becoming popular in the literature, eliminating the inner feedback loop around the optimal path.

The optimal control solution is found using a direct technique of the class of pseudospectral methods known as the Gaussian Pseudospectral Method, which differs slightly from the Radau method that was used for the quadrotor flights. The method and software used for this simulation are described in [90, 92]. With this efficient method, computation time for each epoch took an average of 0.18 seconds using 30 nodes (conservative for such an application) on a standard desktop 2.49 GHz processor with Microsoft Windows® XP running a MATLAB® environment. Increases in speed could be expected if the software were tailored for this specific application and the algorithm translated into a faster programming language such as C++™. Considering the scenario, however, this is more than adequate for real-time control. Furthermore, in order to clearly refute the point about open-loop recursion equating closed-loop feedback if done "fast enough," the simulation was artificially accomplished with *zero* computation time. Though this is unrealistic, it puts the recursive solution in the best possible light, showing the limitations of what could be accomplished even as the optimal control problem approaches being solved in real-time. Any limitations remaining, therefore, are deficiencies in the technique, and not complications from computational delay between the request and the receipt of the optimal solution.

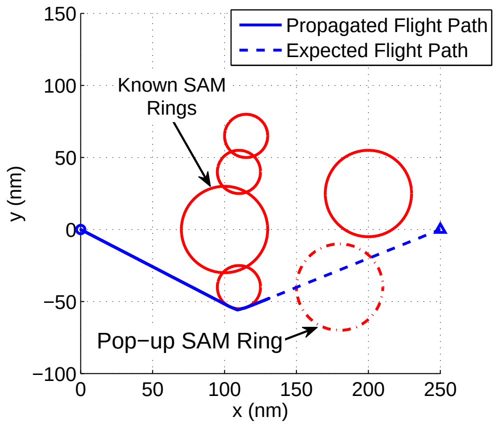
6.4.1 Pop-up SAM Avoidance Results, No Wind Condition.

Figure 31a shows the initial optimal path, planned by the subject aircraft as it starts an attack run, avoiding the known SAM rings and proceeding to the target. In a deterministic system, with the absence of disturbances such as wind or any further threat information, this route would be flown perfectly, and according to Bellman's principle of optimality, even as the optimal problem is recalculated along the route of flight, the resultant course will never change [66].

Introduction of new information is incorporated and adjusted by adding the appropriate constraints. Figure 31b shows the position of the aircraft along the optimal path as the aircraft's systems become aware of a new emitter and the path must be altered. Solutions are continually being reproduced in the path planner, only this time the constraints will have changed.



(a) Initial Flight Path

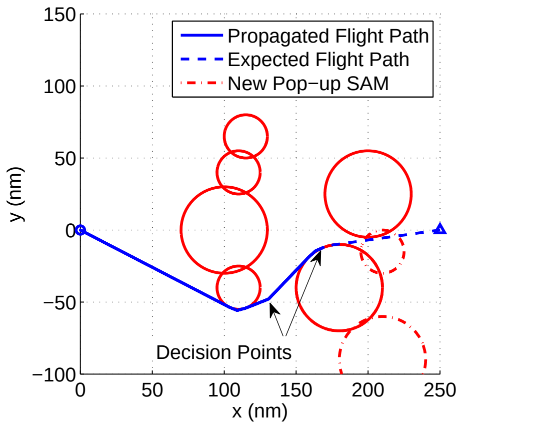


(b) Time=25 min, as a Pop-Up Threat Emerges

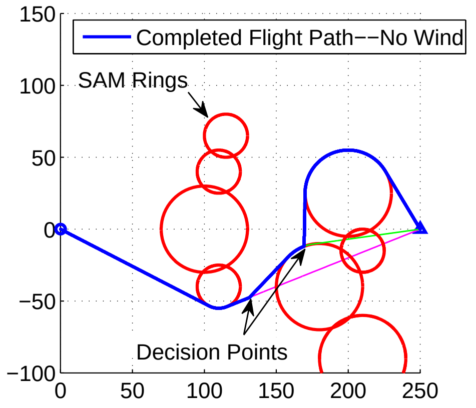
Figure 31. Recursive Optimal Path Planning Around Surface-to-Air Threats—No Wind

Though direct methods are relatively insensitive to initial guesses, optimization via any gradient method is only guaranteed to find local minimums. For this scenario, any path around each side of every "wall" of contiguous threats will produce a local minimum, resulting in a non-convex space of convex channels. Several guess generating algorithms can be designed to determine the possible channels for investigation for the global minimum, such as the branch-and-bound technique found in [24]. Intelligent planning can be also be used to decrease the number of options. Potential methods include dynamic programming concepts, starting from the end of the solution and working backwards—once a global solution has been found to completion from any point, there is no need to search that portion of the path again. Another, simpler, solution for the minimum time problem is just to sort the channels by distance, checking the shortest channel for feasibility of the optimal solution (with respect to turn rate and other constraints). When the final optimal time of the first feasible channel is less than the minimum possible time in the remaining channels, the search is complete. This brute force method is neither elegant nor efficient, but a better solution is beyond the intent of this case study.

Figure 32a shows the result of two new emitters being sensed by the aircraft. The guess-generation algorithm provides two routes to investigate, and after running the optimization routine on the shorter, the longer route is discarded since the minimum possible time is greater than the time of the feasible solution, resulting in the completed flight path of Figure 32b.



(a) Time=30 min, with New Pop-Up Threats. Major Adjustment Required



(b) Completed Flight Path

Figure 32. Completion of Recursive Optimal Path Planning Around Pop-up Surface-to-Air Threats—No Wind

These results support the efficacy of using an optimal control solution in defining flight paths which include changing parameters or constraints, and are on the level with the kind of RTOC simulations solved by the authors quoted in Section 6.1. The difficulty arises when stochastic inputs in the form of disturbances and measurement noises are considered. As demonstrated in the simple case of Case Study A, the controller will still achieve the primary goal, however, the path taken may be far less than the best that could be accomplished in the same circumstances, and may still result in mission failure. For the design of an RTOC system, additional control may likely be desired to overcome the lack of path-error integration in the recursive-only structure, especially in the presence of possible non-zero mean or time-correlated disturbances or measurement errors.

6.4.2 Effect of Non-Zero Mean or Time Correlated Stochastic Disturbances.

A first-order Gauss-Markov process is used to simulate potential wind gust intensity:

$$\dot{w}_{gust}(t) = -\frac{1}{T} w_{gust}(t) + \eta_{gust}(t) \tag{128}$$

where T is a time constant for the system, and η_{gust} is zero-mean, white, Gaussian noise with $E[\eta_{gust}(t)] = 0$ and $E[\eta_{gust}(t)\eta_{gust}(t+\tau)] = Q_{gust}\delta(\tau)$, using the standard definition for the delta function. Similar Gauss-Markov processes were used to determine a lower frequency variation in wind intensity, w_{pred_wind}, and for determining variance in the wind direction. Measuring wind velocity in knots, and direction in degrees, the time constants for the two wind components were 200 hrs and 40 hrs, with respective input strengths of 0.25 and 0.2 knots^2, and wind direction was determined with a time constant of 50 hours and unit intensity noise. The resulting wind intensity and direction were added to a predominant wind and predominant direction biases, respectively, resulting in the disturbance input shown in Figure 33. This is representative of a weather forecast for winds 220° variable 230° at 30 gusting 35 knots (or an average summer day at altitude).

As the final time was unknown, a longer time history of wind was generated than was actually used. The wind disturbance causes the same steering difficulty shown in Figure 27 for Case Study A. No matter how fast a recursive optimal solution is calculated, without a position feedback loop to directly compensate, or feedback in

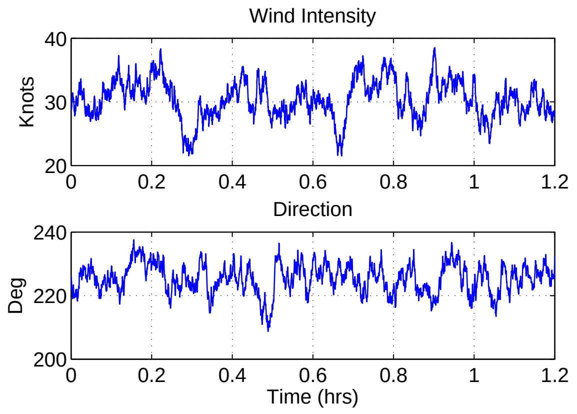


Figure 33. Wind Disturbance Added to the System

the form of a wind estimate term from integration of path error being fed to the optimal control problem, the unmodeled wind will always result in a steering error between the calculated solution and that which would be truly optimal. Once near the SAM rings, the errors in steering become more critical and a constraint is violated, as shown in the inset of Figure 34. The magnitude of the constraint violation is a

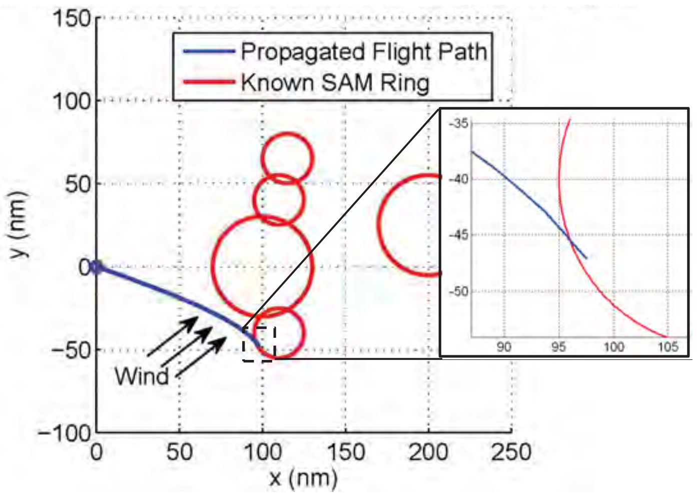


Figure 34. Steering Failure with Recursive RTOC Control Structure in the Presence of Wind

function of the size of the wind disturbance, the recursive solution update timing, and any applied turn rate limit. If the aircraft is allowed an infinite turn rate, the recursive system will always meet the constraint as the update interval approaches zero (this assumes the vehicle is riding the "outside" of a constraint that is curved away and does not necessarily hold for attempts to ride the inside of a curve).

For this scenario, minor deviations will likely not mean the difference between life and death, but there certainly are systems with hard limits (physical terrain, structures, etc.), and optimal solutions often ride as close as allowable to those limits. If the ability of the system to change course is limited (i.e. a slow maximum turn rate), then late steering corrections approaching a constraint can cause large violations.

Besides potential violations, the main point of the exercise is to show that the path itself is clearly not optimal. Recall from Case Study A that there will *always* be steering error in the case of a time-correlated or non-zero mean disturbance. This can be seen in the bending of the optimal path of Figure 34, just as was the case for the homing solution of Figure 27. For Case Study A, the analytic solution in Equation 123 showed that the steering error was not a function of the update timing, but of the problem geometry and the magnitude of the disturbance. This is why faster updates did not remove the problem, as illustrated in Figure 27b. Increasing the update rate does decrease the amount of time that you follow the erroneous heading, but there will only be small changes in the erroneous heading command for the next step until there is significant deviation from the optimal path, when it is too late.

6.4.3 Integration of Path Error.

To correct the non-optimal bending of the path due to the disturbance bias, the bias is estimated and included in the optimal control formulation for the next epoch. Note that, though helpful, it is not required that the source of the bias even be known. Path deviations may come from poor sensors, wind, poorly rigged flight controls, or other sources. As in adaptive control, applying the open-loop control and comparing the resulting trajectory to the expected trajectory provides the opportunity to estimate parameters which may be used to update the model for each epoch.

For this implementation, estimates of the wind direction and velocity are required, broken down into components in the x and y directions. In the absence of a direct measurement source, this can be produced from the difference between the expected and actual position in each axis divided by the time step (or an averaged position over several time steps). A simple estimation filter is used for the demonstration, with the initial condition determined by the first measurement:

$$\widehat{w}_x\left(t_{k+1}\right) = \widehat{w}_x\left(t_k\right) + k_{wind}\left[w_{x_{meas}}\left(t_k\right) - \widehat{w}_x\left(t_k\right)\right] \tag{129}$$

An identical formulation is used for the y-axis component. For simplicity, one tenth of the residual error is applied at each time step ($k_{wind} = 0.1$), but the Kalman filter equations could easily be implemented for a more optimal choice for k_{wind}.

With an available wind estimate generated from the closed-loop feedback of the vehicle state, the assumed system dynamics are updated by adding the appropriate components into each channel and the recursion is allowed to proceed. Using decision points similar to those from Figure 32b, where the aircraft is made aware of pop-up SAM threats, the completed flight path can be seen in Figure 35, and is almost indistinguishable from the no-wind optimal path. The mission is accomplished in the presence of changing threats and non-zero mean disturbances.

6.5 Recommended RTOC Structure

Both case studies have shown the detrimental effects of implementing RTOC in a purely fast open-loop recursive scheme. The current trend in RTOC algorithms has been to use recent computational speed increases to implement a purely feed-

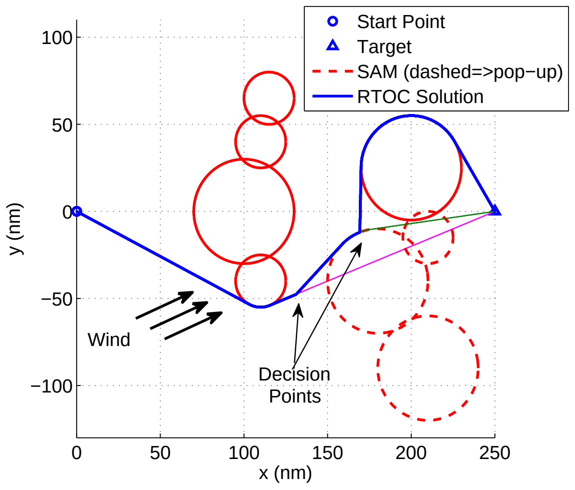


Figure 35. Complete Flight Path, Wind Compensated for through Estimation from Position Feedback

forward system with instantaneous optimal solutions only. This eliminates the use of a traditional inner-loop to maintain the optimal path in the presence of disturbances in favor of merely replacing the optimal path entirely. Though this can be effective in simulation, this method is by no means optimal, and it suffers greatly in the presence of stochastic inputs—particularly those which are non-zero mean or time-correlated.

Individual control problems will always require a designer's eye for the best control structure for a particular purpose, but no matter what method of control is selected, the integration of past error between the expected and actual trajectories must be included in the determination of future control. For systems guided with RTOC to handle changing environments (such as pop-up SAMs), a classical inner-feedback loop is still required for steady-state performance. The inner-loop error signal is added to the optimal control to maintain the optimal trajectory in the presence of unmodeled effects and non-zero mean or time-correlated disturbances. When possible, an additional method includes both this inner loop, and feeding back disturbance estimates into the optimal control problem, changing the dynamics equations in each epoch to make the model best match reality.

VII. RTOC Algorithm and Implementation Tools

This chapter addresses the algorithm employed for the real-time optimal control portions of the research, detailing both the framework of the RTOC implementation, and the optimal control solution algorithm itself. Completion of the design process through actual hardware implementation and subsystem integration brought out several key implementation lessons that will be useful to future RTOC designers.

7.1 RTOC Algorithm

Figure 36 provides the essential decision outline for three control segments required to land the quadrotor on a power line. For more specifics, the top level shell of the Matlab® code to execute this loop is provided in Appendix B. The acquisition

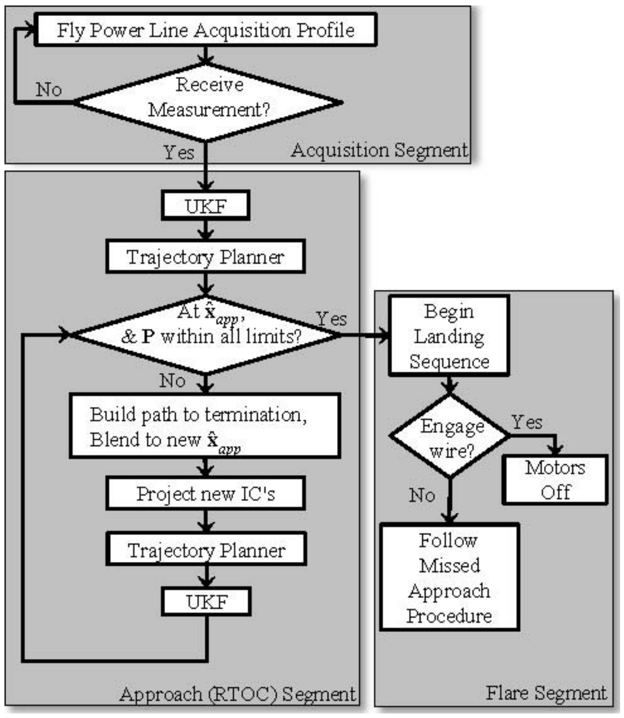

Fly Power Line Acquisition Profile
No
Receive Measurement?
Yes
Acquisition Segment
UKF
Trajectory Planner
At $\hat{\mathbf{x}}_{app}$, & $\mathbf{P}$ within all limits?
Yes
No
Build path to termination, Blend to new $\hat{\mathbf{x}}_{app}$
Project new IC's
Trajectory Planner
UKF
Approach (RTOC) Segment
Begin Landing Sequence
Engage wire?
Yes
Motors Off
No
Follow Missed Approach Procedure
Flare Segment

Figure 36. RTOC Algorithm Structure

segment is completed when the power line is identified by the sensor, and an initial target estimate and trajectory are initiated. For the flight test, a "shell" was created with a list of commands to the quadrotor to takeoff, stabilize, and move to a hover position until the first measurement was received or a timeout occurred, at which time the aircraft landed. For both the quadrotor and the full power line scenario, since the initial target estimate and covariance are provided as a guess to the UKF (based on likely height of the power line and likely sensor acquisition range), an initial trajectory can also be pre-calculated off-line, and used to seed the trajectory planner's initial guess. This is not required, since direct methods are tolerant of poor initial guesses, but it sets up the system for a fast first solution. After the first pass of the trajectory solver, the previous epoch's solution is always used for the initial guess, trimming off the initial portion that should have already been flown. Once the approach segment's main loop is entered, it is executed until the vehicle reaches the approach point with the required certainty in the target location, at which point the aircraft enters the flare segment to land.

The heart of the approach segment is the iterative RTOC algorithm. As the recursive estimation filter provides updated target coordinates, the estimate for the required approach point, $\widehat{\mathbf{x}}_{app}$, is updated, and the trajectory planner then calculates an update to the optimal path. Each solution is a control state pair, $\{\mathbf{x}_k^*(t), \mathbf{u}_k^*(t)\}$, $t \in [t_k, t_f]$, that is semi-discrete—every epoch contains the complete state and control time history for the remainder of the flight. Non-optimal portions of the path are spliced onto the path as well. These commands give the vehicle a "missed approach" plan for what to do if the exit criteria for the approach segment are not achieved. This would be especially significant for times when the measurement data is lost for a significant length of time. For short periods with no new data, the plan will simply be updated to initialize with a higher than expected covariance than was planned for in the previous epoch. The quadrotor's missed approach plan consisted of a simple landing profile. For the full sUAS, it would likely include circling back to the location of the last known good measurement, with a further contingency plan after that.

Once the main RTOC loop of the approach segment is entered, note that the call to the UKF counter-intuitively happens *after* the trajectory planner. The trajectory planner consumes most of the loop time, Δt_{calc}. With a slow sensor update rate, it is not likely that measurements will arrive between the time the UKF provides an estimate and the time the trajectory planner begins calculations on the next epoch. During the trajectory planner calculations, however, multiple measurements will likely be received, and the target estimate—and thus $\widehat{\mathbf{x}}_{app}$—should incorporate the new measurement data prior to checking to see if the approach point has truly been achieved and the required certainty has been met.

7.1.1 Initial Condition Validity.

An easily overlooked, but critical, consideration must be taken with respect to initial conditions. It was outlined in Section 5.3.3 that the initial conditions for each trajectory planning epoch are set based on the expected future conditions at the time the solution is planned to be available. The initial condition $x_{0_{k+1}} = x(t_k + \Delta t_{calc})$ is based on the optimal time history $\mathbf{x}_k$, which was solved relative to the target estimate $\widehat{\mathbf{x}}_{t_k}$. Note also that many of the constraints on the optimal solution are also set relative to the target estimate, such as the constraint to stay within an area where the target will be seen in the fixed camera FOV. Since it is derived from the optimal solution, $x_{0_{k+1}}$ will always reside within the constraints, at least as well as they were known to be for epoch k. However, as the algorithm of Figure 36 progresses, the condition can occur (and often does, since optimal solutions tend to "ride" on constraints), that when the target estimate is updated to $\widehat{\mathbf{x}}_{t_{k+1}}$ and the relative boundaries move, the initial condition may rest outside of the boundaries for that epoch.

A processing step must be made at every epoch to check all of the initial conditions for validity, else the trajectory planner will never converge to a feasible solution. For the quadrotor algorithm, invalid initial conditions were moved to the closest point in the most current valid flight envelope. This may result in a discontinuity between the present position and the next commanded position. A smoothing function can be applied as will be developed in Section 7.1.3 to mitigate difficulties caused by using a variable calculation time.

7.1.2 Variable Calculation Time.

For simplicity of process integration, researchers working in RTOC typically choose to update the optimal solution at a fixed loop time, Δt_{calc}. This allows the flight control algorithm to look for a new optimal solution at a set time in the flight control loop. The downside to this approach is that the trajectory planner must be finished prior to that time, and the calculation time can vary greatly. A very conservative Δt_{calc} must be selected, and efficiency is sacrificed as *every* iteration, by design, takes the maximum allowable iteration time. Allowing the loop time to be variable increases the rate of receiving optimal path updates. The downsides are coding complexity for timing transitions, and the fact that the projected initial conditions may not match the current commanded conditions at the new epoch.

A variable calculation time was used for this research, and Δt_{calc} was set as the *expected* calculation time, vice the maximum. The efforts of the flight control autopilot and the optimization software were processed independently, but threaded together to allow the optimal solution to be applied as soon as it was available. As an engineering safety valve, maximum iteration limits were still set for the optimization software, but they were not triggered in the tests conducted once the problem and constraint formulations were finalized. The concept was that if the optimal solver was unable to converge on a particular instantiation of the optimal problem, it would be reset with the current conditions and target estimate, throwing out the previous solution as its initial guess.

Using a variable calculation time method could potentially impact the application of optimal solutions that are not available until after the expected amount of calculation time. For solutions that are available (t_{k+1}) earlier than expected ($t_{0_{k+1}}$), the new portion of the optimal solution is simply appended to the discrete path and the effects are transparent:

$$\begin{aligned}&\text{if } \; t_{k+1} < t_{0_{k+1}} = t_k + \Delta t_{calc}, \\ &\qquad \mathbf{x}^*_{k+1}(t) = \{\mathbf{x}^*_k[t_k, t_k + \Delta t_{calc} - \Delta t],\ \mathbf{x}^*_{k+1}[t_k + \Delta t_{calc}, t_f]\}\end{aligned} \tag{130}$$

For solutions that are available later than expected, the implication is that a discontinuity is possible in the state and control at time t_{k+1}. The error between the actual state and the planned state as each old solution is replaced is now a factor not only of how close the vehicle tracks the planned state, but also depends on the distance and direction the vehicle has traveled in the amount of time the calculation took beyond that which was expected. If the calculation took significantly longer than expected, this discontinuity could be significant.

A similar discontinuity can occur at the approach point. While the trajectory planner is calculating, new measurements are still being received. Considering this information, the estimate of the target location will likely have moved while the trajectory is being applied. The unfortunate cumulative effect is that by the time an optimal solution becomes available, it travels from a place the vehicle is no longer at, to a place the target estimate is no longer at. This cannot be solely controlled by increasing the recursion timing, as the target estimate moves in instantaneous steps as measurements come in. A blending strategy ensures smooth, continuous control and adds corrections to the path ends.

7.1.3 Correction Blending of Path Ends.

There are optimal methods for resolving the differences in initial and final conditions, most notably those of neighboring optimal control (NOC) [13, 119]. For systems where these differences are critical, NOC is recommended. Experimentation with this system suggested that the differences in initial conditions were very small (as it will be for systems where the calculation time is fairly predictable). During flight test, the longest calculation time was only 0.11 seconds beyond what was anticipated, leading to very small initial discontinuities. Changes at the “tail” of the path can be substantial, depending on how far the target estimate moves during each measurement update.

Stability for the quadrotor system in the face of a discontinuity in commanded trajectory was never a question, as the autopilot was designed with velocity limits to be stable for any size command step. The tail of the path was certainly more sensitive to measurement updates, but until the end-game, the tail portion of the path will be replaced each epoch before it is actually flown. As a result, the computational expense of NOC was forgone for a simple and efficient strategy that ensured the path would always end at the most current target estimate, but without discernible delay. This correction is critical for the last seconds of the flight, but is a nice feature for robustness as well, as the path “in hand” is always based on the best information at the time, and is the best plan to follow in the case of mechanical failure of the optimal solver, or a delay caused by an inability to converge on a solution.

The initial condition discontinuities can occur when the optimal solution for epoch $k+1$, available at t_{k+1}, arrives later than the expected time of $t_{0_{k+1}}$. Until t_{k+1}, the system continues to fly the solution that was produced for epoch k. The final condition discontinuities occur when the trajectory planner delivers a path for epoch $k+1$ to the assumed target, $\widehat{\mathbf{x}}^{-}_{t_{k+1}}$, that has been updated by the estimation algorithm to $\widehat{\mathbf{x}}^{+}_{t_{k+1}}$ during the calculation time of the path planner. Sample results of the blending can be seen in Figure 37, where the dark black line indicates the path that is sent to the vehicle at the actual update time t_{k+1}. The path sent is a composite of the solid optimal solution at $\mathbf{x}^{*}_{k}$, the dashed solution at $\mathbf{x}^{*}_{k+1}$, and the blending correction as a result of updating the target to $\widehat{\mathbf{x}}^{+}_{t_{k+1}}$during calculation time.

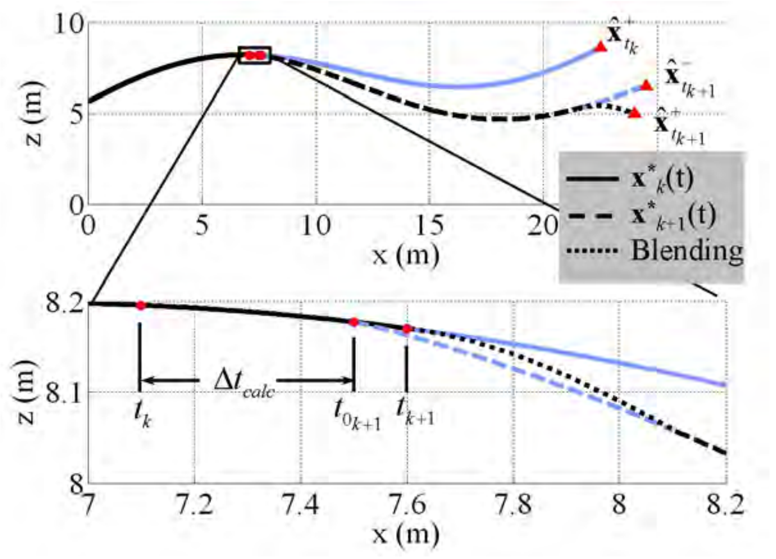


Figure 37. Cosine Blending Corrections

To produce the blending without generating the sharp changes of trajectory with a linear blending method, a cosine wave was used to "round the corners" and smoothly transition the head or tail of the path to the corrected point. The calculations are described here for the tail of the path with a discrete time series vector, as it is actually applied in the physical system. The length of the blending segment, $t_{bl_{max}}$, is selected by the desired segment time, t_{seg}, limited to the amount of time remaining if the path is already within the final window:

$$t_{bl_{max}} = \min\left[t_{seg}, t_f - t_k - \mathrm{rem}\left(\mathrm{t_f} - \mathrm{t_k}, \Delta \mathrm{t}\right)\right] \tag{131}$$

The remainder function (rem) is used to ensure an even division by Δt (t_{seg} is chosen this way as well). In practice, it was found that blending initial differences (if they exist) over one second, and final differences over 5 seconds was efficient and effective. A time vector is then produced:

$$t_{bl} = \left[0, \Delta t, 2\Delta t, \ldots, t_{bl_{max}}\right]^T \tag{132}$$

For corrections at the tail of the path, the point at which the new path departs from the old should be smooth. A correction wave vector from zero to one with a slow initial transition is created using one-quarter of a cosine wave period, and is directionally scaled by the amount the target estimate was moved in each state at the last batch update to create a correction matrix:

$$\begin{aligned} \Xi_{0\to1} &= 1 - \cos\left(\pi t_{bl}/2t_{bl_{max}}\right) \\ \boldsymbol{\Xi}_{cor_k} &= \Xi_{0\to1}\left(\widehat{\mathbf{x}}_{t_k}^{+} - \widehat{\mathbf{x}}_{t_k}^{-}\right)^T \end{aligned} \tag{133}$$

The correction matrix is then used to update the path segment:

$$\mathbf{x}_k^{+}\left[t_{app} - t_{bl_{max}}, t_{app}\right] = \mathbf{x}_k^{-}\left[t_{app} - t_{bl_{max}}, t_{app}\right] + \boldsymbol{\Xi}_{cor_k} \tag{134}$$

When this technique is used to correct discontinuities in initial conditions, the error to be rectified is measured from beyond the moment when the new path becomes available, at $t_{k+1} + t_{bl_{max}}$. During the flare segment of the flight, a similar technique is also used to generate a horizontal profile, providing a smooth path from the approach point to the point where the vehicle actually hooks the wire. In both of these cases, both ends of the blend are desired to be smooth, so a correction wave from zero to one such as the one in Equation 133 is used with a higher frequency (one-half of a full cosine wave). For the initial condition blending, this allows both the initial move from the old path and the blend into the new path to both have smooth transitions:

$$\begin{aligned} \Xi_{0\to1_{flare}} &= 1/2 - 1/2\cos\left(\pi t_b / t_{bl_{max}}\right) \\ \boldsymbol{\Xi}_{cor_{flare}} &= \Xi_{0\to1_{flare}}\left(\hat{x}_t - x_{app}\right) \end{aligned} \tag{135}$$

7.1.3.1 Process Threading.

The last noteworthy implementation lesson came from timing synchronization problems stemming from using a flexible calculation time for the optimal path planner on actual hardware. The UKF, trajectory planner, communication paths, autopilot processes, and speed control servos are each running iterative loops, but all at different rates. Threading loops with known rates is not difficult, but the trajectory planner has a variable cycle time (just over 1 Hz for this application). Working at a much faster rate (50 Hz), the autopilot must have a buffer of future commands to process, and a “dealer” function was implemented as a solution to run between the programs as a storage place for each epoch’s optimal path time history. This allowed the flight control and optimization algorithms to be carried out on separate processors, and handled the asynchronous timing between them without resorting to slowing the process by saving the path to a file. A dealer function can be run at high speeds, checking for a complete path update (the “deck” if you will) without ceasing to provide a list of commanded positions and heading at each time step of the autopilot.

When a new optimal path is formed, it is sent via TCP packets using a blocking protocol to stop processing on the low, variable rate processor until the deck is picked up. This makes any delay less than one time step of the higher rate function (the dealer runs at 100 Hz), and ensures the new path can be used as soon as it is ready. A similar method was used in the other direction to get measurements, limits, and initial conditions into the RTOC process, stopping processing after sending a “ready” poll, checked for during each loop of the dealer. With this technique, slowing down the trajectory planner to allow a fixed calculation time is not necessary.

7.2 Radau Pseudospectral Method

The final area of RTOC implementation to address is the actual solution method for the optimal control problem. Pseudospectral methods have the most advantageous calculation speed, and are appropriate given the knowledge that a flight trajectory will be smooth and differentiable. Adaptive grid refinement techniques were applied to allow segmentation of the problem in the face of potential discontinuities. An open-source software algorithm known as $\mathcal{GPOPS}$ v3.3 was used with the Radau Pseudospectral Method (traditional Radau points, including the initial point) to formulate the continuous problem into an NLP, and the industry standard SNOPT v7 was used to solve it. Using open-source software allowed minor modifications for speed when implemented in real-time. The algorithm used is collected from [90, 35, 34, 39, 5]. The general concepts of transcription were introduced in Chapter II, including transformation of time to the interval $\tau \in [-1, 1]$ to make use of Gaussian quadrature. On that interval, collocation is performed at the Legendre-Gauss Radau points, which may be obtained by first producing the Legendre polynomial, expressed with Rodrigues' formula as:

$$P_N(\tau) = \frac{1}{2^N N!} \frac{d^N}{d\tau^N} \left[\left(\tau^2 - 1 \right)^N \right] \tag{136}$$

where N is the number of nodes desired to collocate at.

The actual collocation points, τ_k, are the roots of $P_N(\tau) + P_{N-1}(\tau)$, which will always contain the initial point, $\tau_1 = -1$, and where $\tau_N < 1$. The quadrature weights associated with these points are solved for off-line with an algorithm based on the LGR Vandermonde matrix, and saved for rapid use during the real-time application. Note that for these weights, w_i, and polynomials, ϕ_p, of degree at most $2N - 2$:

$$\int_{-1}^{1} \phi_p(\tau)\, d\tau = \sum_{i=1}^{N} w_i \phi_p(\tau_i) \tag{137}$$

The discretization points include all of the collocation points and the end point, $\tau_{N+1} = 1$. Using $L_i,\ i = 1, \ldots, N+1$ as a basis, accurate approximation of each of the n_x states, x_j, can be performed with a polynomial of at most degree N:

$$x_j(\tau) \approx \sum_{i=1}^{N+1} x_{ij} L_i(\tau) \qquad j = 1, \ldots, n_x \tag{138}$$

The basis elements are found using the standard Lagrange interpolating polynomial definition:

$$L_i(\tau) = \prod_{j=1, j\neq i}^{N+1} \frac{\tau - \tau_j}{\tau_i - \tau_j} \tag{139}$$

Collocation will require comparing the known derivative for each state from the system dynamics equations with the derivative of the approximating polynomial for each state at each collocation point. Differentiating each state component, x_j, at each collocation point, τ_k, gives:

$$\dot{x}_j(\tau_k) \approx \sum_{i=1}^{N+1} x_{ij} \dot{L}_i(\tau_k) = \sum_{i=1}^{N+1} D_{ki} x_{ij}, \qquad D_{ki} = \dot{L}_i(\tau_k) \tag{140}$$

The components are assembled into the differentiation matrix, $\mathbf{D} \in \mathbb{R}^{N \times N+1}$, with a row for each collocation point and a column for the derivatives of each of the $N+1$ Lagrange polynomials evaluated there. Note that this matrix may be calculated entirely off-line with only the knowledge of the number of nodes to be used in the solution, allowing for extremely efficient calculation of the derivative of each state at every collocation point in an $N \times n_x$ matrix that can be written:

$$\dot{x}_j(\tau_i) \approx (\mathbf{DX})_{ij} \qquad i = 1, \ldots, N \quad j = 1, \ldots, n_x \tag{141}$$

Since the polynomials for each state are at most degree N, the derivative approximation is exact. The matrix $\mathbf{X} \in \mathbb{R}^{N+1 \times n_x}$ is made of the coefficients of Equation 138 and includes row vectors of the state components at every discretization point:

$$\mathbf{X}_i \equiv \mathbf{X}(\tau_i) = \begin{bmatrix} x_{i1} & \cdots & x_{in_x} \end{bmatrix} \qquad i = 1, \ldots, N+1 \tag{142}$$

The n_u dimensional controls can also be expressed as row vectors of all the control elements at a particular time, but this is only necessary at the collocation points:

$$\mathbf{U}_i^{\text{LGR}} = \begin{bmatrix} u_{i1} & \cdots & u_{in_u} \end{bmatrix} \qquad i = 1, \ldots, N \tag{143}$$

To complete the conversion from the continuous optimal control problem into the static NLP, the dynamic constraints, $\boldsymbol{f}$, from Equation 88 on page 82 are expressed as a matrix formed from the state values at each of the collocation points, $\mathbf{F}\left(\mathbf{X}^{\text{LGR}}, \mathbf{U}^{\text{LGR}}\right) \in \mathbb{R}^{N \times n_x}$ such that:

$$F_{ij}\left(\mathbf{X}^{\text{LGR}}, \mathbf{U}^{\text{LGR}}\right) = f_j\left(\mathbf{X}_i^{\text{LGR}}, \mathbf{U}_i^{\text{LGR}}\right) \qquad i = 1, \ldots, N \qquad j = 1, \ldots, n_x \tag{144}$$

where:

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}^{\text{LGR}} \\ \mathbf{X}_{N+1} \end{bmatrix} \tag{145}$$

The NLP is then defined as minimizing the approximation of the continuous cost function:

$$J^{\mathrm{LGR}} = \mathbf{\Gamma}\left(\mathbf{X}_0, t_0, \mathbf{X}_f, t_f\right) + \frac{t_f - t_0}{2} \sum_{k=1}^{N} w_k \boldsymbol{\mathcal{L}}(\mathbf{X}_k, \mathbf{U}_k, \tau_k; t_0, t_f) \tag{146}$$

The original dynamic constraints are now a series of static constraints for every state at every node:

$$\mathbf{DX} - \frac{t_f - t_0}{2} \mathbf{F}\left(\mathbf{X}^{\mathrm{LGR}}, \mathbf{U}^{\mathrm{LGR}}\right) = \mathbf{0} \tag{147}$$

with the original constraints and boundary conditions now evaluated discretely as:

$$\boldsymbol{\gamma}\left(\mathbf{X}_0, t_0, \mathbf{X}_{N+1}, t_f\right) \geq \mathbf{0} \tag{148}$$

$$\mathbf{C}\left(\mathbf{X}_i, \mathbf{U}_i, \tau_i; t_0, t_f\right) \geq \mathbf{0} \qquad i = 1, \ldots, N \tag{149}$$

7.2.1 Solving the NLP.

The solver SNOPT introduces slack variables to convert all constraints to equality conditions. A modified Lagrangian is formulated by augmenting the cost function with Lagrange multipliers applied to each constraint, and the optimality conditions are found by taking the partials of the Lagrangian with respect to the states, controls, and multipliers and setting them to zero. Though $\mathcal{GPOPS}$ provides a very effective automatic differentiation package, analytic derivatives were used as the most accurate and efficient method of gradient determination. The trivial boundary conditions are omitted, but the remaining analytical derivatives are summarized in Table 1.

With the modified Lagrangian, SNOPT uses a two-tier iteration. Simplifying, major iterations linearize all constraints with a truncated Taylor series, evaluating the Jacobian for the constraints at the iterate point and formulating a new subproblem with a quadratic approximation of the modified Langrangian and the linearized constraints. Minor iterations solve each subproblem with a reduced Hessian active-set method. This method seeks to reduce the computational expense of calculating the Hessian by freezing some of the variables, and moving along the feasible curve in the direction of the reduced gradient to minimize the cost function. Reaching a minimum, more of the variables are allowed to move. Upon reaching a solution, a Lagrangian merit function is formed, and a line search along that function is made from the subproblem solution point to a new point, where the constraints are re-linearized and the process continues until tolerances of the major iterations are met.

Table 1. Non-Zero Analytic Derivatives

Equation	Non-Zero Partial Derivatives
$\boldsymbol{\Gamma} = t_f$	$\frac{\partial \boldsymbol{\Gamma}}{\partial t_f} = 1$
$\boldsymbol{\mathcal{L}} = \mathbf{u}^T \mathbf{W}_u \mathbf{u}$	$\frac{\partial \boldsymbol{\mathcal{L}}}{\partial u_x} = 2w_{u_x} u_x$
	$\frac{\partial \boldsymbol{\mathcal{L}}}{\partial u_z} = 2w_{u_z} u_z$
$\boldsymbol{f} = \begin{bmatrix} \dot{x} \\ \dot{z} \\ u_x \\ u_z \\ \frac{\sin^2\beta}{\Delta t_{meas}\sigma_\beta^2\rho^2} \\ \frac{\cos^2\beta}{\Delta t_{meas}\sigma_\beta^2\rho^2} \\ -\frac{\sin\beta\cos\beta}{\Delta t_{meas}\sigma_\beta^2\rho^2} \end{bmatrix}$	$\frac{\partial f_1}{\partial \dot{x}} = \frac{\partial f_2}{\partial \dot{z}} = \frac{\partial f_3}{\partial u_x} = \frac{\partial f_4}{\partial u_z} = 1$
	$\frac{\partial f_5}{\partial x} = \frac{4x_r z_r}{\Delta t_{meas}\sigma_\beta^2\rho^6}$
	$\frac{\partial f_5}{\partial z} = \frac{-2z_r\left(x_r^2 - z_r^2\right)}{\Delta t_{meas}\sigma_\beta^2\rho^6}$
	$\frac{\partial f_6}{\partial x} = \frac{2x_r\left(x_r^2 - z_r^2\right)}{\Delta t_{meas}\sigma_\beta^2\rho^6}$
	$\frac{\partial f_6}{\partial z} = \frac{4z_r x_r^2}{\Delta t_{meas}\sigma_\beta^2\rho^6}$
	$\frac{\partial f_7}{\partial x} = \frac{z_r\left(z_r^2 - 3x_r^2\right)}{\Delta t_{meas}\sigma_\beta^2\rho^6}$
	$\frac{\partial f_7}{\partial z} = \frac{x_r\left(x_r^2 - 3z_r^2\right)}{\Delta t_{meas}\sigma_\beta^2\rho^6}$
$C_1 = \tan^{-1}\left(\frac{z_t - z}{x_t - x}\right)$	$\frac{\partial C_1}{\partial x} = \frac{z_r}{\rho^2}$
	$\frac{\partial C_1}{\partial z} = \frac{-x_r}{\rho^2}$
$\gamma_1 = \mathrm{P}_{xx_{max}}\left[\xi_1(t_f)\xi_2(t_f) - \xi_3^2(t_f)\right] - \xi_2(t_f)$	$\frac{\partial \gamma_1}{\partial \xi_1(t_f)} = \mathrm{P}_{xx_{max}}\xi_2(t_f)$
	$\frac{\partial \gamma_1}{\partial \xi_2(t_f)} = \mathrm{P}_{xx_{max}}\xi_1(t_f) - 1$
	$\frac{\partial \gamma_1}{\partial \xi_3(t_f)} = -2\mathrm{P}_{xx_{max}}\xi_3(t_f)$
$\gamma_2 = \mathrm{P}_{zz_{max}}\left[\xi_1(t_f)\xi_2(t_f) - \xi_3^2(t_f)\right] - \xi_1(t_f)$	$\frac{\partial \gamma_2}{\partial \xi_1(t_f)} = \mathrm{P}_{zz_{max}}\xi_2(t_f) - 1$
	$\frac{\partial \gamma_2}{\partial \xi_2(t_f)} = \mathrm{P}_{zz_{max}}\xi_1(t_f)$
	$\frac{\partial \gamma_2}{\partial \xi_3(t_f)} = -2\mathrm{P}_{zz_{max}}\xi_3(t_f)$

7.2.2 Adaptive Grid Refinement.

Clearly, with constraints defined for the derivative of every state at every node in addition to the typical constraints of an optimal control problem (boundary, path, event, etc.), the dimensionality of the NLP increases greatly with the number of nodes. Using a small number of nodes provides a fast solution, but potentially at the cost of accuracy. For this dissertation, Darby's adaptive griding, introduced in Section 2.2.2.1, is incorporated [21]. The total number of nodes is divided into s segments with N_s nodes in the respective segment:

$$N = \sum_{s=1}^{S} N_s \tag{150}$$

Path constraints and boundary constraints do not change, but the collocated dynamic constraints must be modified to reflect transforming each segment of times $t \in [t_{s-1}, t_s]$ to $\tau \in [-1, 1]$:

$$\begin{bmatrix} \mathbf{D}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2 & \cdots & \mathbf{0} \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{D}_S \end{bmatrix} \mathbf{X} - \begin{bmatrix} \frac{t_1 - t_0}{2}\mathbf{I}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \frac{t_2 - t_1}{2}\mathbf{I}_2 & \cdots & \mathbf{0} \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \frac{t_S - t_{S-1}}{2}\mathbf{I}_S \end{bmatrix} \mathbf{F} = \mathbf{0} \tag{151}$$

Total cost now becomes a sum of the segment costs, and continuity is ensured by forcing each state to start a segment with the value it had at the completion of the prior segment. Formulating the problem in this manner actually increases the sparsity of the NLP Jacobian, resulting in less computational time.

The risk of the method is loss of spectral accuracy with fewer nodes in each segment. To check for this, the collocation constraints in Equation 147, which are mandated to be zero at all of the collocation points, are evaluated in between the collocation points (ideally, the equations should be zero everywhere, but they are only constrained at the nodes). Midpoints between collocation points are found:

$$\bar{t}_i = \frac{t_i + t_{i+1}}{2} \qquad i = 1, \ldots, N_S - 1 \tag{152}$$

The states are evaluated at the midpoints using the Lagrange polynomial approximations, and the controls are approximated with cubic interpolation at the midpoints, resulting in: $\bar{\mathbf{X}},\ \bar{\mathbf{U}} \in \mathbb{R}^{N_S-1 \times n_x}$. The differentiation matrix is the square Lobatto matrix, $\bar{\mathbf{D}} \in \mathbb{R}^{N_S-1 \times N_S-1}$, allowing a midpoint residual matrix to be formed:

$$\mathbf{R} = \left| \bar{\mathbf{D}}\bar{\mathbf{X}} - \frac{t_s + t_{s-1}}{2}\mathbf{F}\left(\bar{\mathbf{X}}, \bar{\mathbf{U}}, \tau; t_{s-1}, t_s\right) \right| \in \mathbb{R}^{N_{S-1} \times n_x} \tag{153}$$

Note that $|\cdot|$ indicates the element-wise absolute value. Ideally, the residuals would all be zero and the dynamic constraints would perfectly match the derivatives of the state approximations between collocation points. If this is not the case, the largest value in each row is collected into a vector, representing the greatest error with respect to the dynamics for each segment. The arithmetic mean of these maximum errors is taken, and the errors are scaled by the arithmetic mean. This allows easy comparison of the errors. For the case where one error is significantly higher than the rest, a problem at a specific *time* is assumed, most likely a result of a discontinuity. The number of segments is therefore increased and another iteration is performed, with a segment break at the problematic time to increase nodal density there. Uniform-type errors exist when all error values are relatively equal. If this is below tolerances, the solution is complete. If not, a poor approximation is assumed and the total number of nodes is increased for the next iteration, resulting in a higher order state approximation.

VIII. Quadrotor Vehicle Description and Flight Control Development

"You go to war with the army you have, not the army you want"

—Former Secretary of Defense Donald H. Rumsfeld

Verification of the effectiveness of the RTOC algorithm beyond simulation was performed with an in-house, custom built quadrotor helicopter (Figure 38), designed at the Air Force Institute of Technology's (AFIT) Advanced Navigation Technology (ANT) Center. The flight control system for the aircraft was

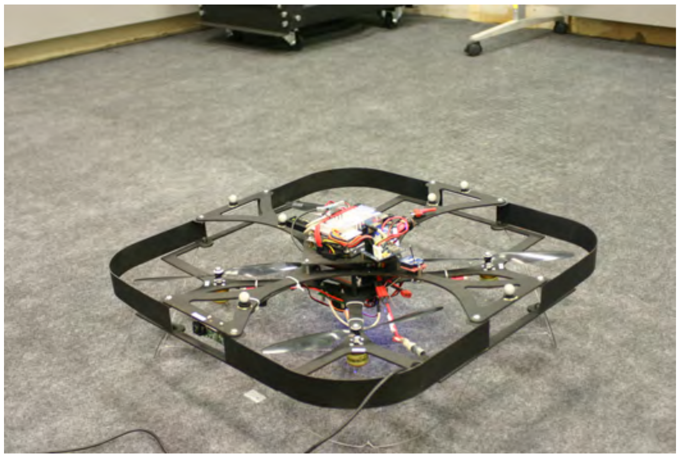


Figure 38. Quadrotor Helicopter

designed with a much simpler purpose in mind, and significant modifications had to be made in order to make the power line landing possible. This chapter details the description of the vehicle, as well as some of the flight control challenges and solutions used for the flight test.

8.1 Vehicle Description

The quadrotor consisted of a 0.607-m square frame with four 22.86-cm blades driven by Goldline AXI 2212/20 brushless motors. The motors were regulated with Phoenix 25 speed controllers and powered by two Li-Polymer 2200 mAh, 11.1V, 3-cell batteries. A Pico-ITX (Linux Ubuntu) with a VIA C7 1-GHz processor with 1-GB of RAM on top of the aircraft was used for data collection and processing of images from a Logitech Quickcam Pro 9000 webcam. As the line detection algorithm was not complete, the bearing measurements for the flight test were provided by the Vicon system and corrupted by noise, vice using the camera. Accelerations were measured with an Analog Devices ADIS 16355AMLZ MEMS-IMU, and inner-loop flight control processing was performed on a custom PIC-24 microcontroller circuit board. Outer-loop RTOC guidance was provided by an algorithm running in MATLAB® R2009a (Microsoft XP), passed to a ground station (Linux Ubuntu) via a dealer function. Mid-loop control commands were generated within the ground station custom C code using a GTK graphics package, and communication to the vehicle was across a 2.4-GHz XBee Pro serial modem. Both computers were Dell 360 2.0-GHz laptops with 2-GB of RAM. Position feedback and flight test data was provided with a Vicon Tracker motion capture system using 60 near IR ($\sim$750-nm) cameras. A schematic of the overall system is shown in Figure 39.

Thrust for the quadrotor is supplied by four independently controlled, fixed-pitch propellers. The propellers in opposing corners spin the same direction, as shown in Figure 40. Altitude is controlled by varying the thrust from all four motors simultaneously. Pitch and roll are controlled by increasing the thrust of both motors on one side of the applicable axis, and decreasing the thrust on the other side. The total thrust remains near constant, maintaining altitude at small angles. Since both sides have one propeller turning clockwise and one turning counter-clockwise, the total

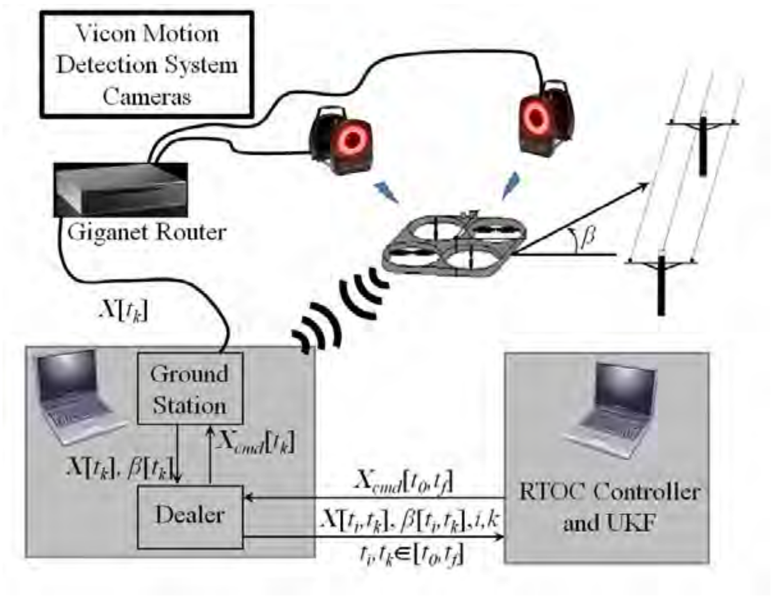


Figure 39. Quadrotor System Schematic

torque also remains the same, maintaining heading. Heading is controlled with an increase and decrease of opposing pairs, maintaining total thrust while changing the total torque.

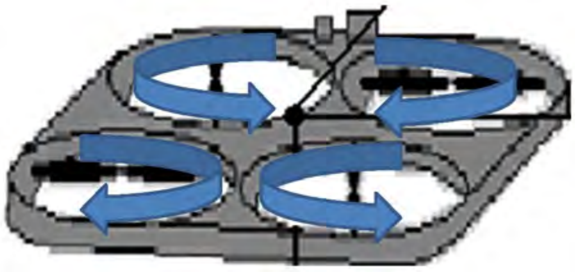


Figure 40. Quadrotor Opposing Pitch Propellers

8.1.1 Autopilot Overview.

Based on the RTOC structure developed in Chapter VI, the autopilot architecture was designed with three main loops. The inner stabilization loop produces the actual Pulse Width Modulation (PWM) signals that drive the motors. Inputs are the body axis angular rate measurements from the on-board IMU, approximations of angular accelerations based on a discrete, first-order lag model, and error commands in the appropriate channels from the portion of the controller in the ground station. The specific structure, gain placements and values, vehicle moments of inertia, and such can be found in the Simulink diagrams and initialization file in Appendix A, but a simplified control flow diagram is shown in Figure 41. The inner feedback loop

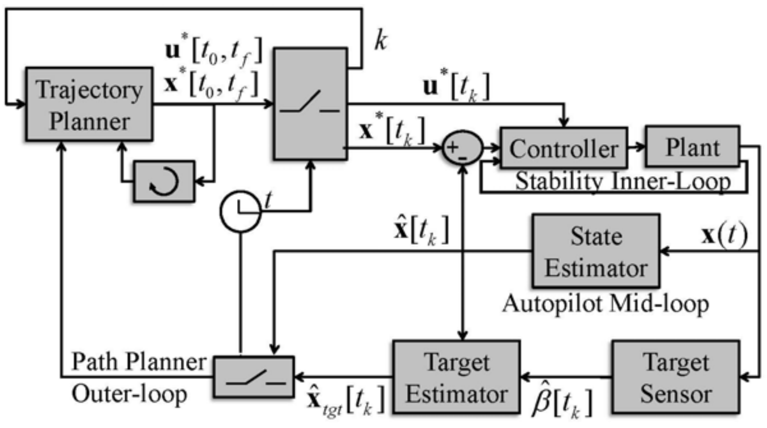


Figure 41. Primary Autopilot Loops

regulates the angular rates and accelerations to zero, while accepting the autopilot commands of the mid-loop, which compares the current position and heading with state vector that is commanded at that time from the most current trajectory time history of the path planner. The path planner takes the measurements from the bearing sensor and plans a new optimal path, using the last optimal solution as an initial guess.

8.2 Flight Control Modifications

The quadrotor flight control system was originally designed to hover at a point. The point could be moved with hand-controlled inputs. Actual steady-state tracking of that point was extremely poor, but immaterial, as the aircraft was flown visually. If the vehicle was low, the commanded point was moved up—how close the vehicle actually was to the commanded hover point was unknown.

Several flight control modifications were required to enable automated path control of the aircraft and the ability to fly to an exact point with no steady-state offset. Some suboptimal decisions had to be made that led to a design that was functional, but incomplete. The actual quadrotor used was the "spare," as the primary aircraft suffered a catastrophic crash just prior to commencing this research due to an error in a line of code. With a fragile, naturally unstable aircraft and no spare parts, a minimalist approach was taken to control development, changing the original design and code as little as possible. The decision to limit the desired flight control work was validated somewhat by an irreplaceable IMU on the custom servo-sensor board failing several times prior to takeoff (luckily) during testing, and eventually burning out the entire board a few sorties after the last of the flight tests presented in Chapter IX. Due to the necessary caution, the ideal course of changing the control scheme to feed-forward control based on the optimal solution was not attempted, choosing instead to simply schedule the motion of the hover point in accordance with the optimal path. Clearly, this will result in late turns and overshoots during more aggressive maneuvers as the commands to turn are not applied until an error already exists between the vehicle and the path. This is most noticeable in the horizontal channels, as the control of the engines does not directly apply force in that axis, but must first generate and integrate angular rate.

8.2.1 Simulation.

All flight control work was developed in simulation to minimize risk. Without an aircraft model or any documentation of the flight controls, the Simulink model in

Appendix A was created by backing out the ground station C code and the code from the servo-sensor board on the vehicle, and applying kinematic and dynamic equations from first principles as detailed in [83]. These equations were modified slightly based on comparison of expected flight profiles to flight test data, adding a drag term in each axis. The drag component was more pronounced in the vertical axis at the slow speeds of the quadrotor, reflecting both propeller drag and a heaving derivative effect common in helicopter models. The heaving derivative reflects the fact that as vertical velocity increases, the angle of attack on the blades decreases, resulting in less lift. The resulting moment equations took body axis moments, $[L\ M\ N]^T$, which were known from the engine model and respective locations of each motor, and integrated them to find the body axis angular velocities, $[p\ q\ r]^T$. The equations were simplified for the quadrotor, which can be considered symmetric in both the $x_b z_b$ and the $y_b z_b$ planes:

$$\dot{p} = \frac{1}{I_{xx}} \left[L - qr\left(I_{zz} - I_{yy}\right)\right] \tag{154}$$

$$\dot{q} = \frac{1}{I_{yy}} \left[M - rp\left(I_{xx} - I_{zz}\right)\right] \tag{155}$$

$$\dot{r} = \frac{1}{I_{zz}} \left[N - pq\left(I_{yy} - I_{xx}\right)\right] \tag{156}$$

The body axis angular velocities were then integrated to find the Euler angles:

$$\dot{\theta} = q\cos\phi - r\sin\phi \tag{157}$$

$$\dot{\phi} = p + q\sin\phi\tan\theta + r\cos\phi\tan\theta \tag{158}$$

$$\dot{\psi} = (q\sin\phi + r cos\phi)\sec\theta \tag{159}$$

With the only forces being the thrust from the propellers, F_z, and a first-order drag force approximation, the force equations were integrated to find the body frame velocities $[u\ v\ w]^T$:

$$\dot{u} = rv - qw - g\sin\theta - k_{D_x}u \tag{160}$$

$$\dot{v} = pw - ru + g\cos\theta\sin\phi - k_{D_y}v \tag{161}$$

$$\dot{w} = qu - pv + g\cos\theta cos\phi + F_z/m - k_{D_z}w \tag{162}$$

where m is the mass and g the gravitational constant. With the body frame velocities and Euler angles, the final simulator step is to rotate to the navigation frame and integrate for position:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} C_\theta C_\psi & S_\phi S_\theta C_\psi - C_\phi S_\psi & C_\phi S_\theta C_\psi + S_\phi S_\psi \\ C_\theta S_\psi & S_\phi S_\theta S_\psi + C_\phi C_\psi & C_\phi S_\theta S_\psi - S_\phi C_\psi \\ -S_\theta & S_\phi C_\theta & C_\phi C_\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{163}$$

where $C_\psi = \cos\psi$, etc.

With the simulator established, the flight control structure was added. A few of the interesting features are summarized below. For full detail on the flight controls and specifics such as gains and moments of inertia, see Appendix A.

8.2.2 Vertical Control Channel.

The system was initially controlled with a proportional-derivative (PD) scheme, using a nominal throttle trim setting to offset the weight, and adjusting all four engines around this setting to correct vertical position error. Though effective for hand flying, significant steady-state error existed between the commanded and actual vertical positions, as show in Figure 42. The nominal throttle setting was clearly too low to account for the weight of the vehicle. To avoid unnecessary tuning flights, a force test stand was built to model the non-linear relationship between thrust and PWM

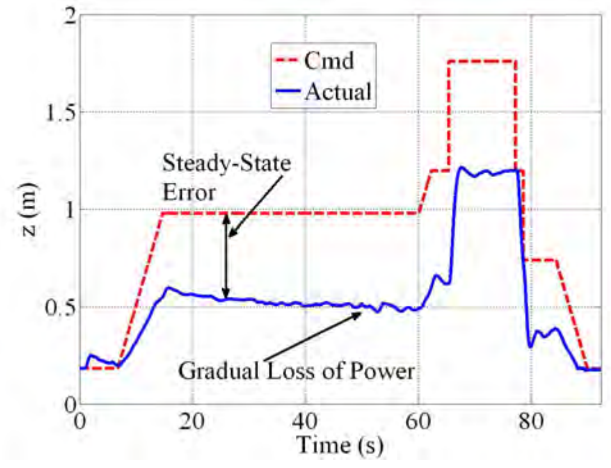


Figure 42. Need for Vertical Error Integration

command to more accurately predict the motor performance. This was helpful, but still insufficient for a precision landing system without active error integration, as the true nominal throttle trim will vary based on loss of battery strength. Furthermore, if the nominal throttle trim is set correctly for flight, the aircraft will "leap" during takeoff, when ground effect makes the propellers much more efficient.

The nominal thrust was set low to match the performance in ground effect for a good takeoff, and a discrete error integrator was added to correct it during flight. Integration is by nature destabilizing, so a very conservative gain level was selected from a root locus plot of the Simulink model linearized about a hover condition. For the full land-on-a-wire test flights, the aircraft was flown to a specific hover position before the run, which started at the 30 second point, so there was ample time to find the correct nominal trim. With additional test sorties, this could be improved.

To avoid integrator windup prior to takeoff while the ground station controller is running, "reset" logic was added to re-zero the integrated error value when the motors were not engaged. As will be discussed in the horizontal channel, saturation limits were applied on both the integrated error value and the amount of proportional vertical error visible to the system in order to limit the maximum vertical speed.

8.2.3 Horizontal Control Channels.

The horizontal channels are controlled by differential power to produce either pitch or bank rate. Position errors in the navigation frame are rotated by heading into the body frame with a simple direction cosine matrix, with the assumption that bank and pitch angles are small:

$$\begin{bmatrix} \Delta x_b \\ \Delta y_b \\ \Delta z_b \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} \tag{164}$$

Velocity in the navigation frame is calculated considering the position change in a single Vicon capture frame over the frame interval, likewise rotated into the body axis. Originally, PD control on position error was applied with angle feedback of ϕ or θ, respectively. This resulted in acceptable performance for hand flying, but in terms of inertial precision, the aircraft was ±0.5-m from the commanded hover position at any point in time (for comparison, it is desired that the vehicle be within ±0.038-m of commanded position to pass through the vertical “mouth” of the hook, assuming zero uncertainty in the estimated position of the wire). Integral control with saturation and reset logic were added, and gains were tuned for performance.

The original design did not have a hard limit bank or pitch angle, as it was designed to be hovered in a small flight area with a hand controller that could only make small changes in commanded position. With a large flight area and the prospect of a (newly designed) automated command system, the potential existed to command a large change in position, which could flip the vehicle over. Saturation limits on the amount of position error entering the command channel were added. Because each channel was dampened with velocity feedback, the desire to move the aircraft to a point was counter-balanced by the desire to keep velocity at zero. Saturating the position error input effectively set a maximum velocity limit. With the system balanced at steady-state, the desired maximum velocity value for both axes could be adjusted for the velocity and proportional feedback gains to determine the required level of saturation (the level of saturation required to produce the right maximum velocity—Figure 67 on page 178 may make this more clear):

$$x_{err_{sat}} = V_{max_x} * k_{v_x} / k_{p_x} \tag{165}$$

Prescribing a maximum horizontal velocity value indirectly limits the bank and pitch angles. The quadrotor needs only a small horizontal thrust component to begin moving from a hover, as it is delicately balanced. Once moving to a steady-state velocity, the bank is removed as the aircraft "coasts," much like a puck on an air hockey table. At the low speeds used for this research, a very minor amount of bank or pitch (approximately 10°) was required to reach steady-state velocity, and only a negligible amount is required to overcome drag and sustain it, as can be seen in the step commands of Figure 43. With bank and pitch angles limited by the position error saturation, the aircraft cannot flip over, regardless of the size of an erroneous input. As an additional benefit, the low pitch and bank requirements work out well for a vehicle with a fixed camera that needs to keep the target in the field of view.

Another feature installed was a variant of anti-windup tracking dubbed "sneak-back" logic. Essentially, when the aircraft is flying to a point beyond the position error saturation distance, the integrator will saturate in the same direction. When the aircraft reaches the point, a large overshoot will occur. To reduce this effect, the integrator is smoothly pulled back toward zero whenever the position error is on or near its limit. With this implementation, a design decision must be made with regard to the maximum velocity limit. If the gain balance to achieve the correct steady-state velocity is set considering a saturated integrator input, then the steady-state veloc-

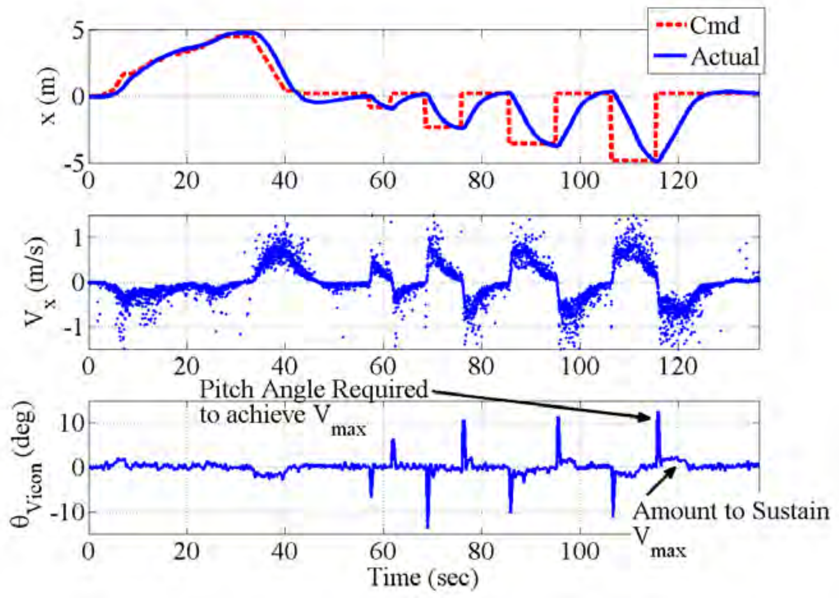


Figure 43. Maximum Velocity Step Commands

ity will be below the maximum as the integrator gets pulled back to zero. If the gain balance is set considering the integrator output to be zero, the vehicle will have the desired maximum velocity at steady-state, but may overspeed briefly while the integrator settles. The latter option was considered acceptable and implemented.

8.2.4 Heading Control Channel.

The least amount of work was accomplished in the heading channel, and it is the least well controlled. Figure 44 shows representative heading error on an early flight. Error integration was necessary and added, with the associated reset logic and saturation. Due to the natural damping when controlling with torque, derivative action was not required, resulting in a proportional-integral (PI) channel. The feedback gains were increased based on gain margin in the linearized Simulink model and fine tuned in flight test. A clear difficulty, in all channels, is the fact that control is modeled as decoupled, but isn't in reality—especially with respect to errors, which are typically

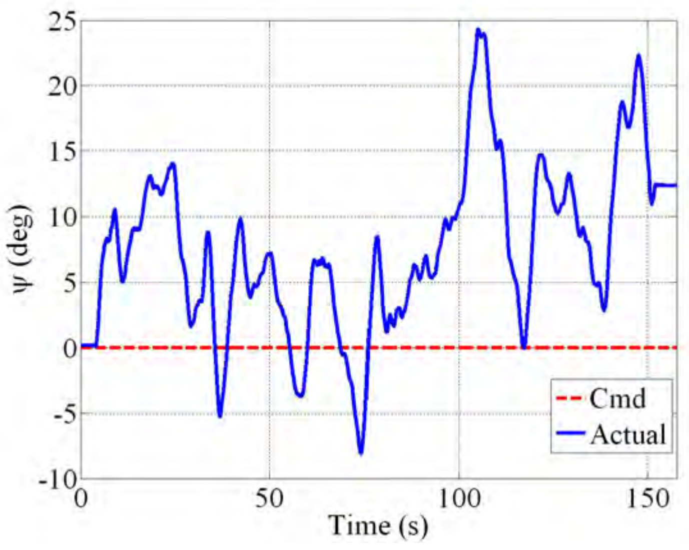


Figure 44. Poor Heading Control

not aligned with an axis. Bending of the aircraft body, imbalanced power output, and misaligned propellers contribute to the steady-state errors that are seen in Figure 44. Integration can balance out the errors in a hover, but it's just that—perfectly balanced, and susceptible to the slightest disturbance. Every time the power settings are altered for any maneuver or change of orientation, the error balance is changed.

8.2.5 Automated Flight.

In order be able to command the system automatically, an automatic flight mode was added that accepted command inputs from the dealer function, shown in Figure 39, as if they had come from the hand controller. The dealer received the optimal path time history from the path planner, and determined the current location in the series. Line numbers with a constant update rate were used vice true time, to avoid clock synchronization errors between the computers. The correct commands for position and heading were then sent to the ground station, while the measurements of the wire position and the vehicle's current measured location were returned to the path planner.

Figures 45 and 46 show one of the early paths, flying in circles with varying altitude followed by level circles to test the vehicle's ability to follow a constantly arcing path. At this point, the tracking had been markedly improved, but the heading channel

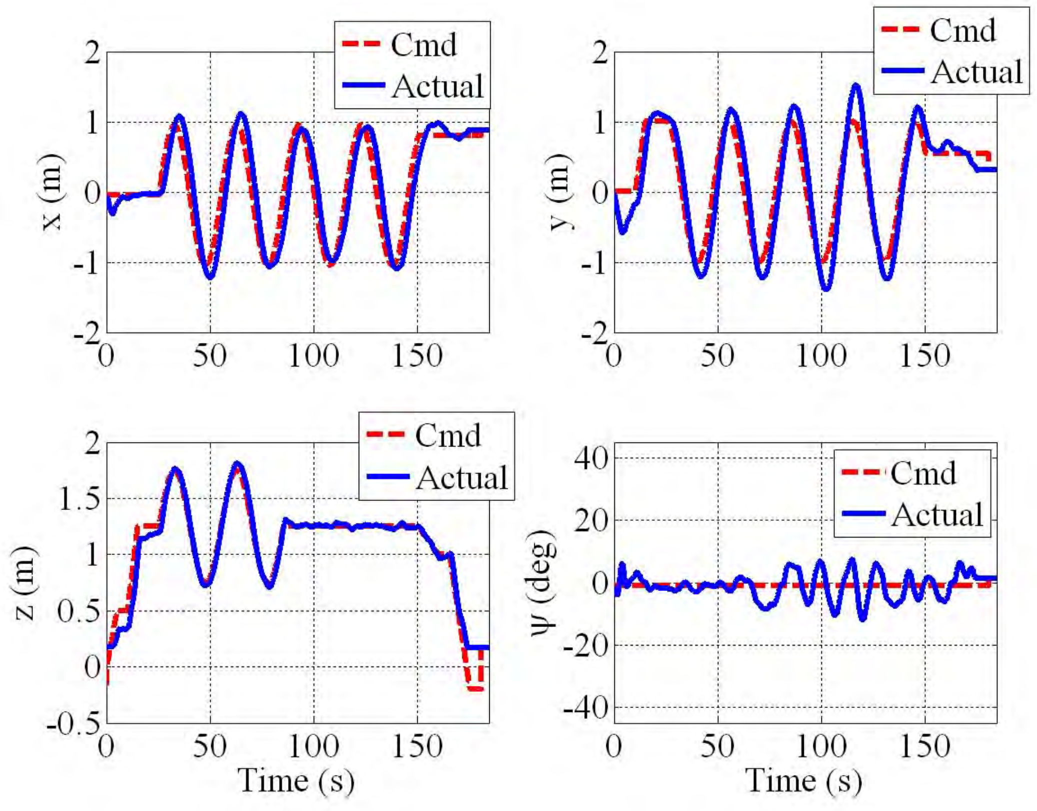
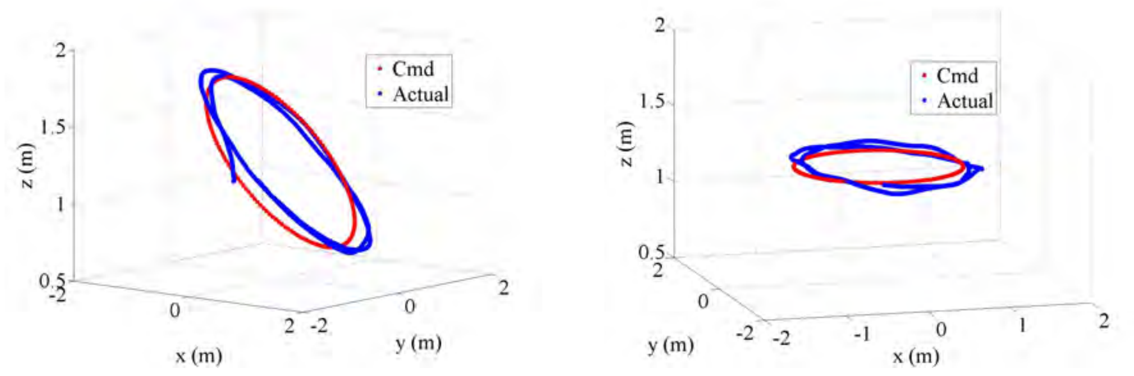


Figure 45. Automatic Flight Control Development Test

appeared to have had much more difficultly with the level circles than the slanted ones, though the reason is unclear. The larger heading errors actually begin 18 seconds prior to the level-off, so it may not be altitude related.

For safety, a "hover" mode was added that allows an observer pilot to switch back to hand controller commands, with a second actuation zeroing out the integrators, in case they were the cause of the needed takeover. At the moment the hover mode was



(a) Varying Altitude, t=25 sec to t=88 sec (b) Level Circles, t=88 sec to t=154 sec

Figure 46. Flight Path for Automatic Flight Control Development Test

activated, the hand controller commands were zeroed out, and the current position and orientation were captured. The captured positions were then continually added as an offset to the commands from the hand controller, resulting in a hover that could be landed manually.

8.2.6 System Identification.

The derived simulator had far less damping than the real aircraft, making use of the simulator for gain selection of little value. A data capture algorithm was written and installed on the quadrotor, allowing flight data to be run through the simulation for comparison of expected and actual performance. Figure 47 shows a safety flight flown in to a simulated wire position compared to the uncompensated simulation output for the same commanded path.

System identification techniques were used to add a compensator for the errors, and the aforementioned drag terms were added. In hindsight, the model deficiencies are most likely due to the fact that the dampening torque effect from the spinning propellers was not accounted for. Additional terms should have been added for this in Equations 157-159. As performed, however, the compensated system did a much

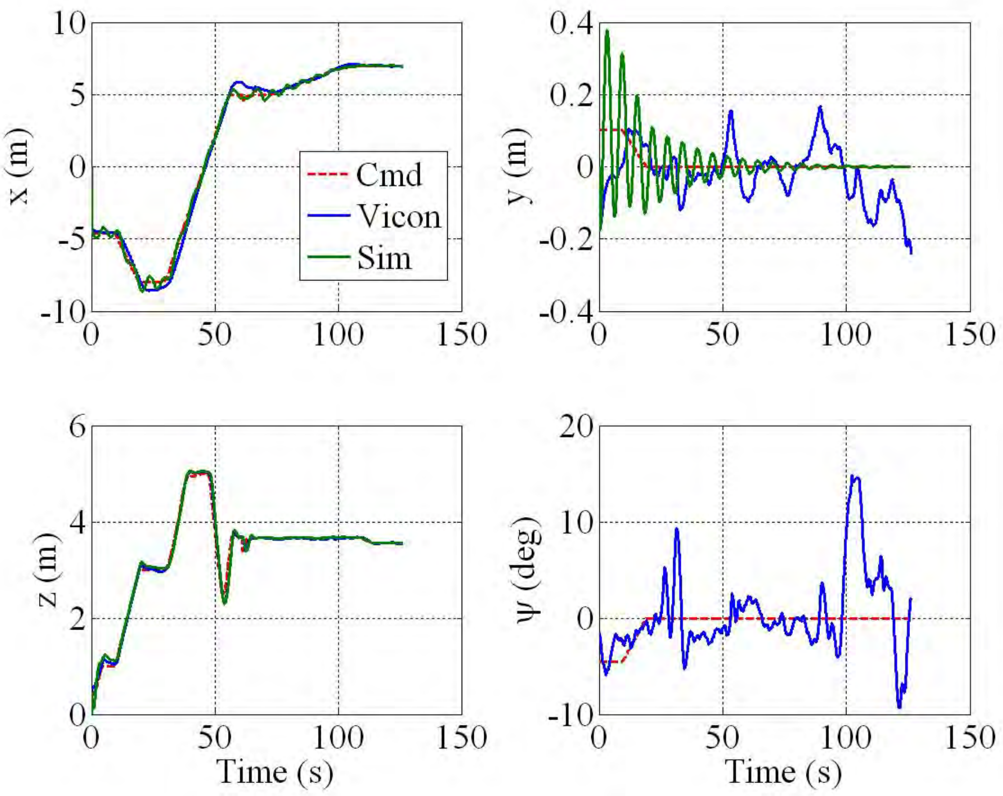


Figure 47. Simulator without Modification to Match Flight Test Data

better job at matching the true system, as shown in Figure 48. This made both gain selection and design of the flare mode much more effective.

8.2.6.1 Flight in the μAVIARI.

Full-scale flight tests were accomplished in the Micro Air Vehicle Integration and Application Research Institute (μAVIARI) indoor flight test facility. Several considerations had to be addressed to enable flight in new facility, most of which had to do with networking with different computers. The Vicon software was also different between the ANT Center and the μAVIARI, but the actual data stream is consistent, so only minor software changes were required, along with DCM changes for the different reference frames. The hand controller code also had to be modified, as the original

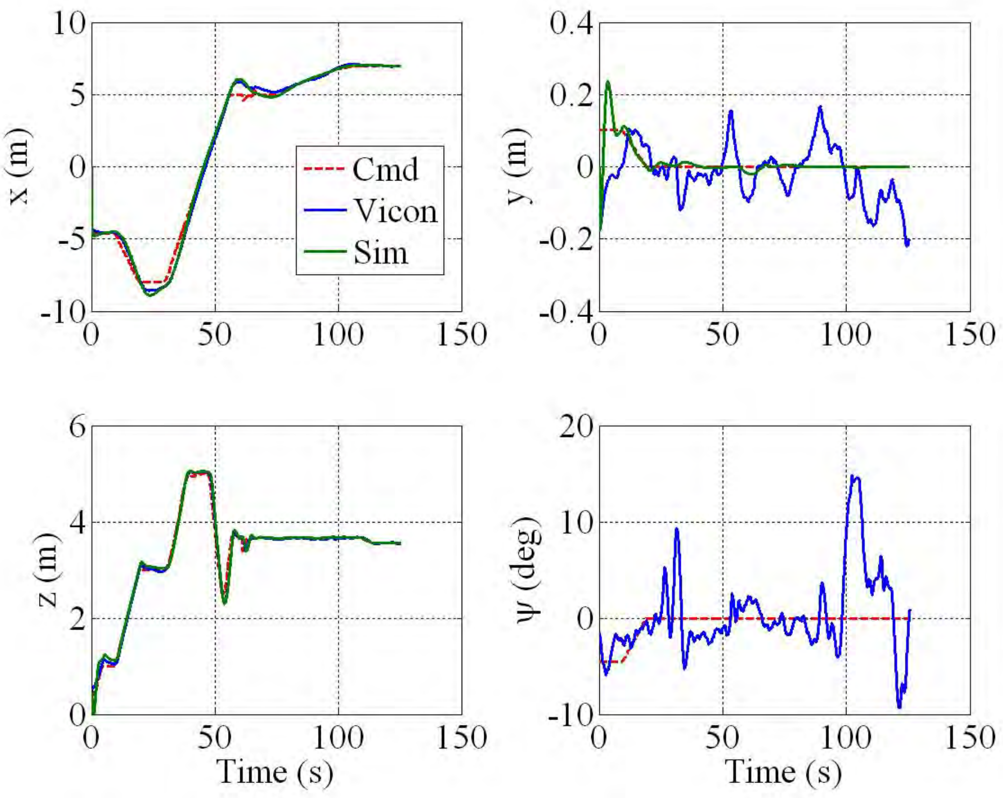


Figure 48. Simulator Modified to Match Flight Test Data

method used each press of a button to move the hover point a percentage of the flight arena size (the trim accumulators on the hand controller only moved between -1 and 1). As the arena got 10 times larger, the hand controller became 10 times as sensitive. Logic was added with a transformation to scale each button actuation to an actual distance.

Lastly, a hook was fashioned from welding rod as a simple method of attaching the quadrotor to the wire. The very flexible nature of the hook in concert with the vibration of the quadrotor excited a large harmonic oscillation. Several iterations of dampening lines were added to the hook until the flight characteristics were satisfactory. These lines can be seen in Figure 16 on page 53.

IX. Results and Analysis

"In science, you can lie and fudge the data because you don't have to make anything work. In engineering, the product is the proof of your honesty."

—Pepper White

THE functionality of the RTOC system was validated through extensive simulation during development, and verification of the algorithm's ability to accomplish the mission of landing on a wire was accomplished through flight test. The flight test profiles were performed in the μAVIARI, operated by the Air Vehicles Directorate of the Air Force Research Laboratory (AFRL/RB), and shown in Figure 49. The in-

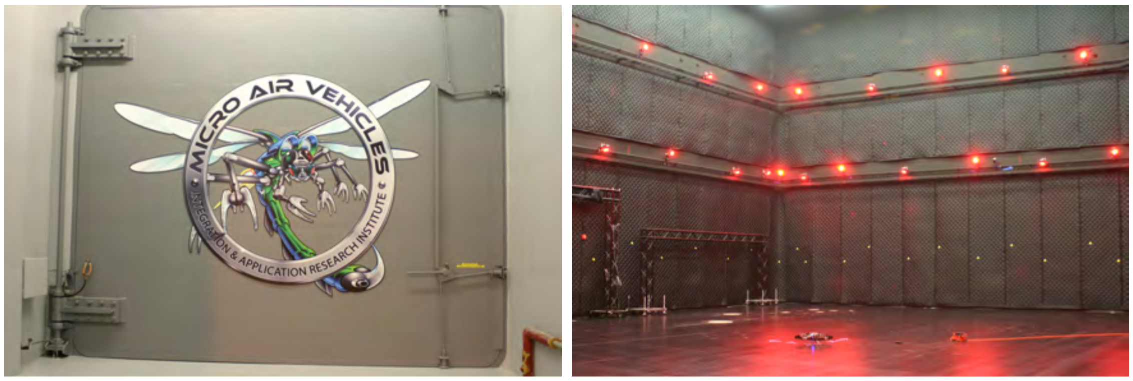


Figure 49. AFRL/RB μAVIARI Indoor Flight Test Facility

door flight test lab allowed use of the Vicon camera system, a flight requirement of the available research vehicle. Though the μAVIARI is very large for an indoor flight test lab, the true power line landing scenario is larger, and the geometry was scaled to fit within physical limitations. An average medium-voltage (distribution) utility pole is approximately 10-m high, but the safe maximum height to maintain visibility by a sufficient number of Vicon tracking cameras in the indoor flight facility was 5.5-m. The walls of the facility dictated a maximum range of approximately 18-m, well inside of the expected range at which a power line could be confidently identified with a webcam type sensor. Correspondingly, the flight test was scaled down in size to fit the μAVIARI, and the vehicle speed was reduced to produce a likely approach segment time of about 30 seconds.

9.1 Simulation Results

In order to test the robustness of the system and the reliability of both the estimation and optimization algorithms, a Monte Carlo-style simulation of 1000 runs was performed on the same scale as the flight tests to maintain comparability. The run number was pre-selected, and the resulting solution parameters of average loop time, mean error, and final directional covariance were confirmed to have converged to within 10^{-3} of their respective units. The problem geometry was varied by moving the actual target location from the initial estimate:

$$x_t \sim \mathcal{N}(\hat{x}_{t_0}, 49\ m^2) \tag{166}$$

$$z_t \sim \mathcal{N}(\hat{z}_{t_0}, 4\ m^2) \tag{167}$$

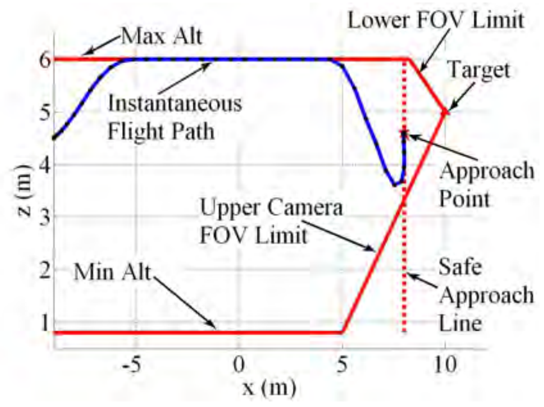
Outliers were limited to stay within the allowable flight space vertically and high enough to maintain the approach point above the allowable floor. The initial pickup range was also limited to a minimum of 12-m to provide some room to maneuver (without some limit, the power line may unrealistically initialize behind the vehicle). Real-world considerations must include a contingency plan for a “go-around” for exceptionally late or missed sensor pickups. The difference in vertical and horizontal certainty reflects the fact that more knowledge will exist concerning the height of the power line than of the initial sensor pickup range. For a full-scale system, actual sensor capability, engineering judgment about likely power line height variance, and the amount of confidence in the mapped power line locations should be included in the selection of the initial covariance, $\mathbf{P}_0$. Disturbances from effects such as wind gusts were added with a bivariate Gaussian distribution, adding a random variance in the vehicle location sampled at the time of each measurement, and measured by the own-ship navigation system:

$$[\ x_{meas} \quad z_{meas}\ ]^T \sim \mathcal{N}_2\left([\ x \quad z\ ]^T, 0.25\ m \right) \tag{168}$$

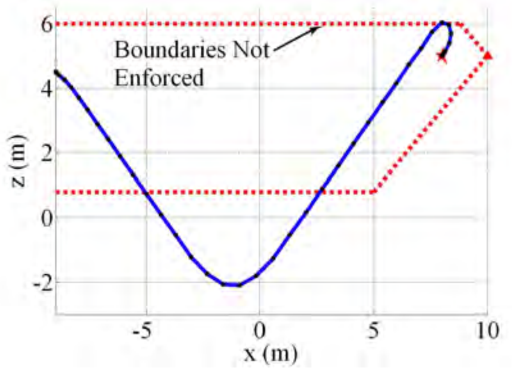
The simulation was initiated with the conditions found in Table 2. The shape of each instantaneous optimal trajectory varies based on the information available to the system at the time. Figure 50 shows typical solutions, with specific problem parameters varied to highlight key features. The results show complete trajectories for the remainder of the flight, as are provided at every epoch by the path planner. The characteristics shown are helpful in creation of heuristics to mimic the optimal solution, potentially a requirement for sUASs without the processing capacity for RTOC. All maneuvering in the simulation is restricted to the vertical plane. Generally, the length of the run (note the asymmetric axes lengths) allows a greater amount of information to be collected about the vertical position of the target, requiring the trajectory planner to move away from the initial LOS angle.

Table 2. Simulation Limitations and Initial Parameters

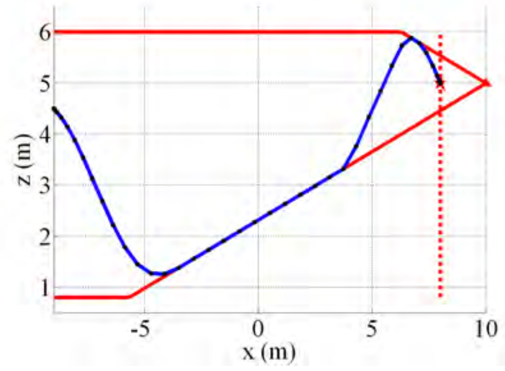
Parameter	Value	Parameter	Value
Est. Loop Time	0.9 s	$\mathrm{P}_{xx_{max}}, \mathrm{P}_{zz_{max}}$	0.02 m^2
$\beta_{min},\ \beta_{max}$	$-30°,\ 40°$	App. Offset (m)	$\begin{bmatrix} -2 & 0.4 \end{bmatrix}^T$
σ_β	0.071 rad^2	$\widehat{\mathbf{x}}_t$(m)	$\begin{bmatrix} 9 & 4 \end{bmatrix}^T$
$\lvert v_x\rvert_{max}, \lvert v_z\rvert_{max}$	0.5 m/s	$\tilde{\mathbf{x}}_0$	$\begin{bmatrix} -8 & 3 & 0 & 0 & \mathrm{P}_{xx_0}^{-1} & \mathrm{P}_{zz_0}^{-1} & 0 \end{bmatrix}^T$
$\lvert u_x\rvert_{max}, \lvert u_z\rvert_{max}$	0.5 $\mathrm{m/s}^2$	$\mathbf{P}_0(\mathrm{m}^2)$	$\begin{bmatrix} 80 & 0 \\ 0 & 80 \end{bmatrix}$
$z_{min},\ z_{max}$	0.8, 6 m		



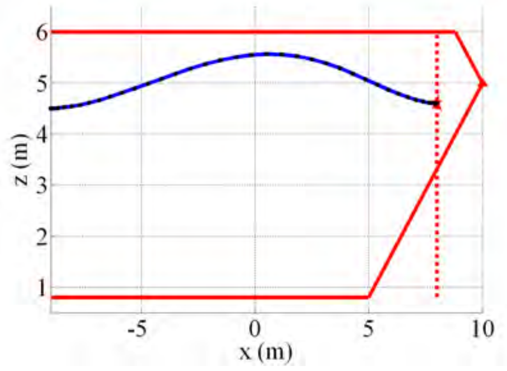
(a) Typical Solution Shape



(b) Speed and Accel Constraints only



(c) 30° Camera FOV



(d) Relaxed Final Covariance Requirements ($P_{xx_{max}} = P_{zz_{max}} = 0.2$ m^2)

Figure 50. Instantaneous Trajectory Shape Sensitivity to Constraints

Figure 50a shows the characteristic shape for the typical initial conditions, where the system is directed to climb to the maximum allowable altitude, or ceiling, for a "high look," moving to obtain a "low look" at the end-game where the measurements are more effective due to the close range. This path visually increases the information elements seen in Equation 73 on page 67. FOV limitations keep the vehicle in a position where the target will be visible to the fixed camera, and a "safe approach" line is enforced to keep the aircraft from flying past the desired approach point and backing up. Though certainly within the capabilities of the quadrotor, it was deemed unsafe to intentionally proceed inside the approach point until the certainty in the actual power line position was within the required limits.

The second panel, Figure 50b, shows an instantaneous flight trajectory with the allowable flight envelope limitations removed, maintaining the speed and acceleration limits in the dynamic constraints. The final required target position certainty was increased, to highlight the fact that the optimal solution may intentionally include transients inside of the safe approach line if not enforced. For the submarine formulation of the problem, this trajectory shape represents an optimal horizontal solution that would be encountered on a larger scale, with the exception of the last transient, which would be avoided by implementing a circular path constraint to stay beyond the opponent's maximum torpedo range until the target position is resolved. All of the characteristics of the vehicle are simply parameters that can be set as appropriate for an individual system. The third panel, Figure 50c, shows the path solution for a camera with a more narrow FOV (30°). The most extreme target approach angle is held as long as necessary. For times when the final certainty requirements do not differ greatly from the current covariance estimate, only a small excursion is necessary to gain the needed amount of information, as shown in Figure 50d.

Obviously, every active parameter in the optimal control problem contributes to the final shape of the trajectory, but the sensitivity of a few dominant parameters found during the research was considered noteworthy. The immediate move away from the initial LOS angle is predictable. The "hook" at the end of the path shown in Figures 50a-50c is also dominant, taking advantage of the wide angular spectrum at close range for the greatest increase in information. For a workable heuristic, the initial move away from the first LOS angle provides the range observability necessary to determine when to make the "hook," which could be initiated at the 6-m remaining point at this speed. Though the characteristic shape is the same, the range at which the "hook point" is executed is non-linear and not necessarily directly scalable to a larger/faster problem. To find it for a particular system, the algorithm should be run in simulation with system specific limitations and expected conditions.

9.1.1 Local Minima.

The geometry of the problem creates a bimodal solution space. With $\beta_0 = 0$, symmetric boundary constraints, no initial vertical velocity, and the approach point level with the target, an optimal path that initially moved up would have a mirrored path with an initial move down and the same total cost. Global favorability of a "high road" versus a "low road" local minimum is dependent on the initial state when the first measurement becomes available. For heuristics, the overall direction tends to be high if the initial position of the vehicle is low relative to the target, and vice versa. Stronger factors are initial vertical velocity (tends to continue in the initial direction), and the vertical difference between the approach point and the actual height of the target (if the approach point is low, the initial move is typically high and vice versa—note the final approach point difference between Figure 50c and Figure 50d). The amount of maneuvering room between the altitude limits and the estimated target position estimate also impacts this decision.

For this system, experience has shown that the "high road" is the global minimum for the given target height, with the greatest sensitivity being to the approach point being set below the target height to account for the height of the hook above the vehicle. A biased guess is not necessary to find the global minimum, and only the initial and final points are used to initialize the system. As a practical method for a system with less certain characteristics, the global minimum for the initial target position guess can be found by checking both initial directions *a priori* through simulation with the planned initial target estimate and covariance, using initial path guesses biased in each direction (the algorithm can, at times, be fooled into a local minimum in this manner). The global solution found should be the seed for the initial calculation of the real-time path planner, which will actually begin to fly this solution between the time the first measurement is received and the time the first optimal trajectory is produced, which will have been solved for using the initial velocity in the correct direction expected at time t_{2_0}. Once the initial trajectory has been begun, switching to the other minimum becomes costly due to the control required to overcome the initial vertical velocity, and the increased percentage of the path that is left near the "middle" of the flight envelope, near the target altitude. Flight in this area contributes little information about the range to the target, which is the "long pole in the tent" in terms of the optimization.

For the second flight test, the initial target estimate was intentionally fabricated to make the "low road" the initial global minimum. This was done by intentionally setting the initial guess too close to the ceiling limit to allow the vehicle room to maneuver above it. The "low road" scenario was demonstrated because it could possibly be encountered with a significantly erroneous target position estimate, though this is unlikely. Note that in the lab, the initial relative position of the power line was fixed physically, and the initial target position estimate was varied to cause differences in path selection. The simulations were set up to mimic this, producing several unlikely, but possible, scenarios where the target was very near the floor, or in the upper "corner" of the flight envelope, such as in Flight Test #2. This was a good test of robustness to potentially poor target estimates, but in a true sUAS landing scenario, the initial relative target estimate will be fixed (based on the expected parameters of the sUAS at initial sensor pickup). The initial solution will therefore be the same for every run (both initial directions having been checked *a priori*), and the vehicle will have already committed to the global minimum direction during the first calculation epoch.

9.1.2 Timing and Accuracy.

For the 1000 simulation runs, the average loop time, including optimization calculation, communication, UKF calculations, and all delays was 0.82 seconds, with a standard deviation of 0.022 seconds. Figure 51 highlights the advantage of using variable calculation timing. If a fixed timing update were selected based on this data,

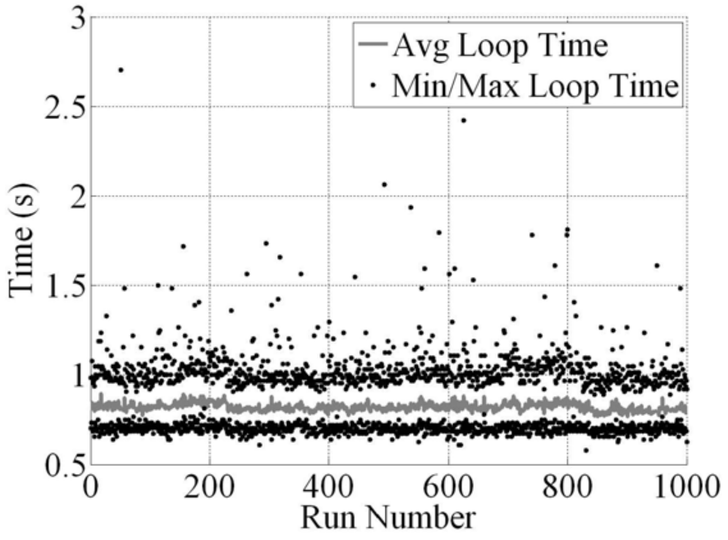


Figure 51. Average Loop Times for Simulation Runs

it would be about $\Delta t_{calc} = 1.3$ seconds, and no trajectory updates would have been available until that time for each epoch. Additional complexity in the creation of the dealer function was required to be able to accept updates as soon as they were available, but 59% more path updates were received, greatly increasing the system's flexibility and ability to deal with uncertainty.

The system's final error upon reaching the approach point is shown in Figure 52. As expected, the performance in the vertical component was better than required,

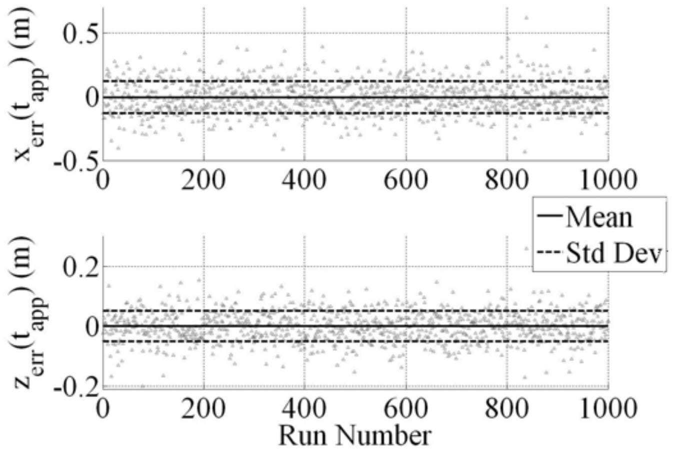


Figure 52. Final Target Estimate Error for Simulation Runs

due to the number of highly orthogonal measurements for the entire flight (variance of the series of final vertical error estimates from the 1000 simulation runs was 0.0026-m^2). For this geometry, the certainty in the horizontal target estimate is the critical parameter that the path planner must meet. The average of the final horizontal covariance estimates from the simulation runs was 0.017-m^2, which closely matched the actual variance of the final horizontal error of 0.016-m^2. The estimated covariance requirement was to be below the limit of 0.02-m^2, but was slightly better than expected due to the fact that typically 2-3 measurements come in during each planning cycle. If the first measurement is the one that put the variance under the limit, the effect of all three is still recorded, as they are processed in batch. This certainty is acceptable for landing considering the size and shape of the quadrotor's arresting hook, and the estimate will be improved with the additional measurements that will come during the flare segment until the camera exits the true FOV limits. More importantly, however, the result validates the algorithm's effectiveness at accomplishing the primary purpose of the research—to create a path in real time that can achieve a required amount of target position certainty in a stochastic environment.

The time series results of the simulation runs can be seen in Figure 53 and Figure 54. The extended times for some runs were due to a more distant target location

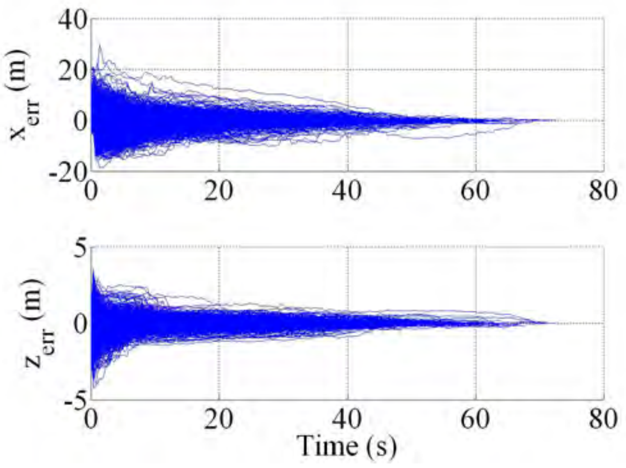


Figure 53. Target Position Estimation Error During 1000 Simulation Runs

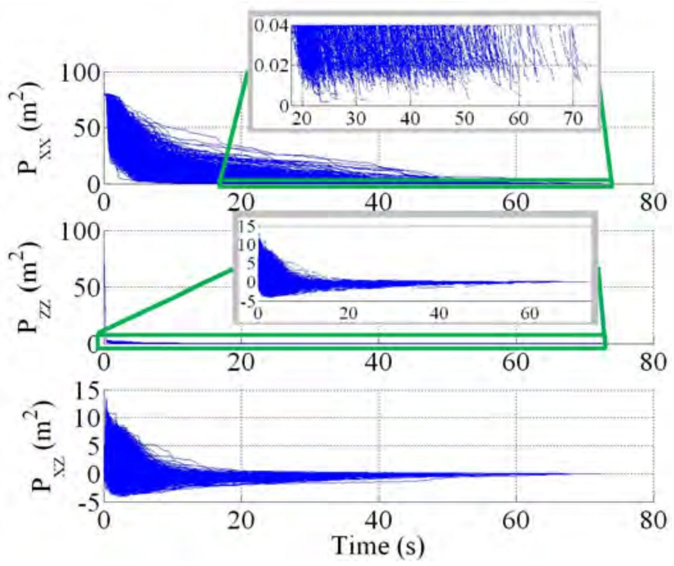


Figure 54. Target Covariance During 1000 Simulation Runs

(the longest run was 38-m). These results show the stability and predictability of the UFK algorithm, and the ability to achieve the final required covariance estimate.

9.2 Flight Test Results

The flight test approach for the system included a build-up series of flights initially working with the stability of the system, followed by the path tracking capability. Most of these flights were accomplished in the small (4-m square) flight facility in the AFIT ANT Center, shown in Figure 55. For tracking, the dealer program was

Figure 55. Flight Control Work Accomplished in the ANT Center (Photo: New York Times)

incorporated to command simple flight profiles, eventually adding the path planning system. Further flights were accomplished to test flying qualities with the arresting hook, which were found to be unacceptable due to a large vibration mode induced by the flexible hook. The hook was dampened with a series of support lines, and scaled down profiles were flown to test the flare segment profile and to test engagement of an actual wire.

Full-size profile flights were first accomplished with a simulated wire in the μAVIARI, followed by the final two end-to-end tests conducted with a real wire to demonstrate the complete system from takeoff to perching on the power line. The only human input to the system for the full profile flights was consent to turn the motors on and off.

The runs were initialized in the same manner as the Monte Carlo-style simulation, with the exceptions noted in Table 3.

Table 3. Flight Test Parameters and Results

Parameters	Run 1	Run 2
$\mathbf{x}_0$ (m)	$\begin{bmatrix} -8 & 3 \end{bmatrix}^T$	$\begin{bmatrix} -8 & 3 \end{bmatrix}^T$
$\mathbf{x}_t$ (m)	$\begin{bmatrix} 8.54 & 4.17 \end{bmatrix}^T$	$\begin{bmatrix} 8.54 & 4.17 \end{bmatrix}^T$
$\widehat{\mathbf{x}}_{t_0}$ (m)	$\begin{bmatrix} 4 & 3 \end{bmatrix}^T$	$\begin{bmatrix} 15 & 5 \end{bmatrix}^T$
Results		
Avg Loop Time	0.83 s	0.85 s
Min Loop Time	0.77 s	0.77 s
Max Loop Time	0.92 s	0.97 s
RTOC Segment	31.53 s	32.33 s
$\mathbf{x}_{error}(t_{perch})$ (m)	$\begin{bmatrix} 0.0117 & 0.0144 \end{bmatrix}^T$	$\begin{bmatrix} 0.0247 & 0.0298 \end{bmatrix}^T$
$\mathbf{P}(t_{app})$ (m)^2	$\begin{bmatrix} 0.0195 & 0.0025 \\ 0.0025 & 0.0024 \end{bmatrix}$	$\begin{bmatrix} 0.0185 & -0.0024 \\ -0.0024 & 0.0024 \end{bmatrix}$

9.2.1 Flight Test Run #1.

It should be noted that Flight Test Run #1 and Run #2 were the actual first and second flights with an installed wire. The complete flight path for Run #1 can be seen in Figure 56, with the flight progressing from the negative x-axis side of the facility with the origin placed near the center of the room. Tracking was acceptable, with the exception of the space at the approach point, which can be more readily seen in Figure 57.

The cause of the deviation is the slow integrators on position error, and the fact that the system was tracking the optimal path with feedback vice using feed-forward of the optimal control. The run starts at $t = 30$ seconds, when the first measurement is accepted and the RTOC system is engaged. Path error in the x-direction increases as the command "leaves" the hovering vehicle and it begins to catch up. While the

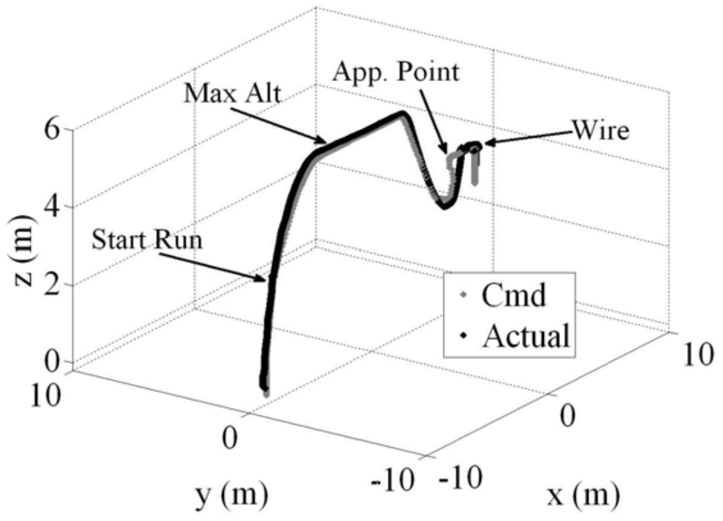


Figure 56. Flight Path, Flight Test Run #1

vehicle travels across the room, the integrator adds up the difference to remove the steady-state error, only to overshoot as the vehicle nears the approach point and the horizontal speed command abruptly stops. For future systems that have more of an ability to flight test the control system, the speed of the integrators should be increased to lessen the amount of time that steady-state errors are present within reason for strong stability. In addition, to anticipate the “corners” in the flight profile, a feed-forward element should be added to the feedback error loop guided by the actual optimal control time history. This will change the control prior to “corners” for better (perfect, in theory) tracking of the path. As is, the system is guided by the error between the current and optimal paths, which will always result in late control inputs, as nothing happens until the paths have already begun to diverge.

The shell profile for the flight is most easily seen in the z-direction (vertical). The aircraft takes off and is directed to hold at 1-m to check stability, taking a moment to integrate the vertical steady-state control requirement as it begins to leave ground effect. After a 5 second hold, the aircraft is directed to a hover at the start run point.

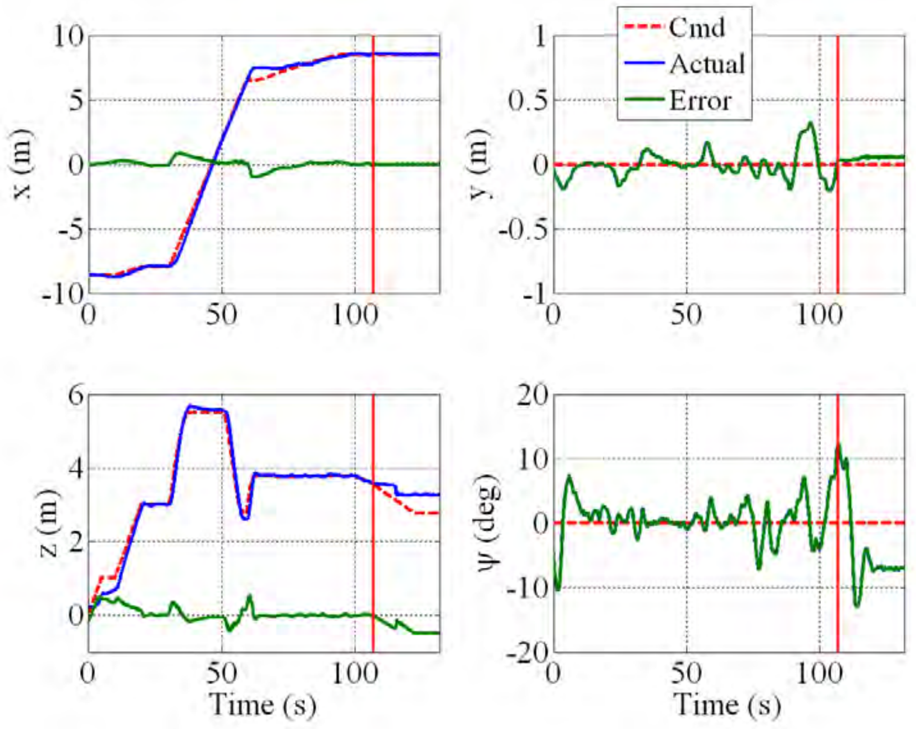


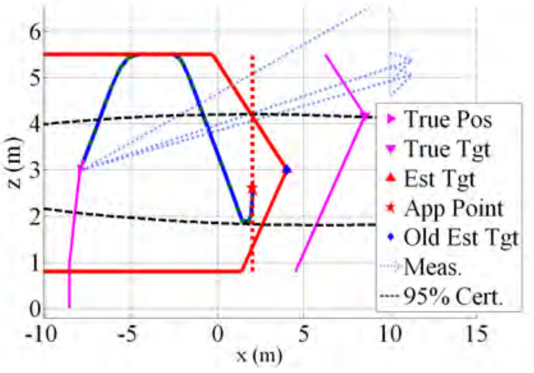
Figure 57. Commanded vs Actual Flight Path, Flight Test Run #1

The RTOC portion of the run is from $t = 30$ seconds until $t = 61.5$ seconds, at which point the vehicle is held level and slowly moves forward to the wire. There is no actual sensor on the vehicle to detect the wire—the vehicle stops based on the last known relative position, as the wire is no longer in the FOV. These are obvious difficulties that should be remedied in a full system. Tracking is good, and the separation of the vertical paths is seen at the point where the wire is actually in contact with the hook, denoted by the vertical red line in each of the plots. At this point, the vertical command continues to descend, but the vehicle stops as soon as the slack is taken out of the wire. The engines are turned off at 115 seconds, as noted by the quick vertical drop as the wire stretches slightly with the remaining vehicle weight.

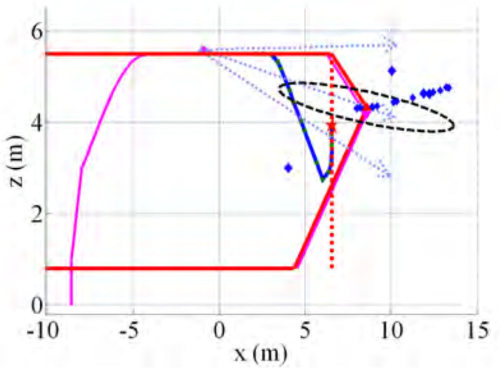
Heading (ψ) error is minimal, once stabilized, with some difficulty during the slower flare portion of the path. The horizontal position command during the flare portion is not linear, but a continually slowing path as the wire is approached. With the heading controlled by the torque balance of the engines, the constant small changes in pitch angle required to track the path and its constant speed changes resulted in some difficulty maintaining heading, and y-axis error, which occurred right at the wire and may have also been induced somewhat by propwash from the nearby wall.

9.2.1.1 RTOC Performance.

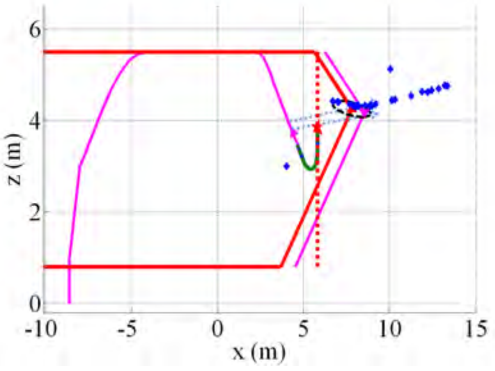
The RTOC system performed exactly as designed. The actions of the path planning system as it converges to the optimal path are difficult to characterize without a string of all of the system updates, but Figure 58 summarizes this with a progression of instantaneous solutions in the vertical plane at separate sample times. The arrows from the vehicle denote the actual bearing measurements received by the system, and the directions give a sense of the magnitude of the measurement errors (they should point through the true target). Both the estimated and actual target location can be seen. The covariance ellipsoid shows a 95% likely confidence ring, and the error in the initial seconds exceeds this slightly as the estimate settles with the first few measurements. The diamonds denote target estimate histories, showing a trend toward the true target with an unsurprising difficulty in resolving range. The range ambiguity can also be seen in the orientation of the covariance ellipse, which has the greatest uncertainty in the direction of the LOS from the vehicle. The reason for the "hook" at the end of the paths is clearly seen, as the path planner moves the vehicle to a position orthogonal to the greatest axis of uncertainty remaining. The last measurements in the profile are critical, both in terms of the value of close range measurement and the value of measurements from that direction.



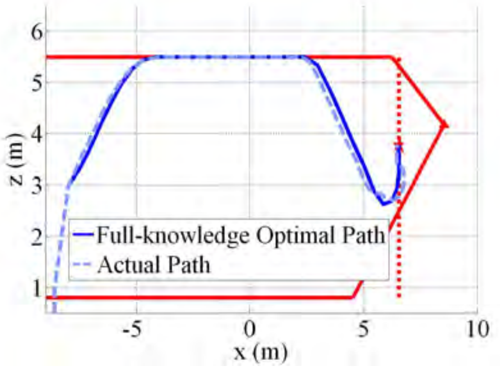
(a) $t_0 = 0.9$ s



(b) $t_0 = 15.5$ s



(c) $t_0 = 25.9$ s



(d) Deterministic vs Stochastic Path

Figure 58. Flight Test Run #1 Snapshots, and Comparison to Full-Knowledge Path

The final panel of Figure 58 shows the comparison of the actual path that was flown by the vehicle with the path that would have been commanded had the target position estimate always been perfectly accurate. This demonstrates the true power of stochastic real-time optimal control. Even with the initial error in the target position, and with the errors in each measurement, the actual path that the vehicle flew was very close to the perfect-information solution.

9.2.2 Flight Test Run #2.

As previously mentioned, the initial target estimate for the second test flight profile was set unrealistically high, making the "low road" the global minimum due to the insufficient observability of the horizontal axis while near the maximum allowable altitude. The flight path is shown in Figure 59. The comparison of commanded vs.

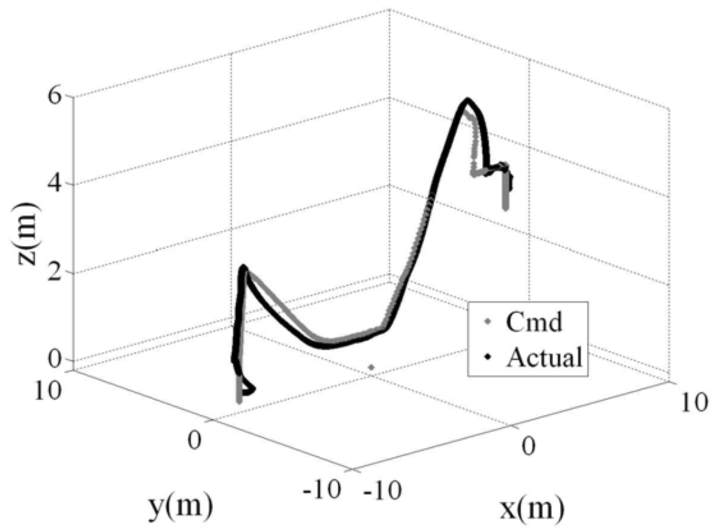


Figure 59. Flight Path, Flight Test Run #2

actual position can be seen in Figure 60. As can be seen, Run #1 and Run #2 exhibited many of the same characteristics.

The RTOC controller performance is shown in the snapshot progression of Figure 61. Note the position of the target estimate in Figure 61a in relation to the maximum allowed altitude. This is what forced the “low road” to be the optimal path with the initial information. Even though the estimate had moved down significantly by Figure 61b, once the vehicle has committed to a certain direction, switching to the local minimum on the other side becomes too costly. In terms of mission accomplishment, the only loss from the perfect-information solution in this contingency case is a small increase in flight time, 0.8 seconds over that of Run #1.

The final panel, Figure 61d, shows the path of the flare mode, which proceeds level from the approach point to the perch point before commanding a descent to engage the hook, as shown in Figure 62.

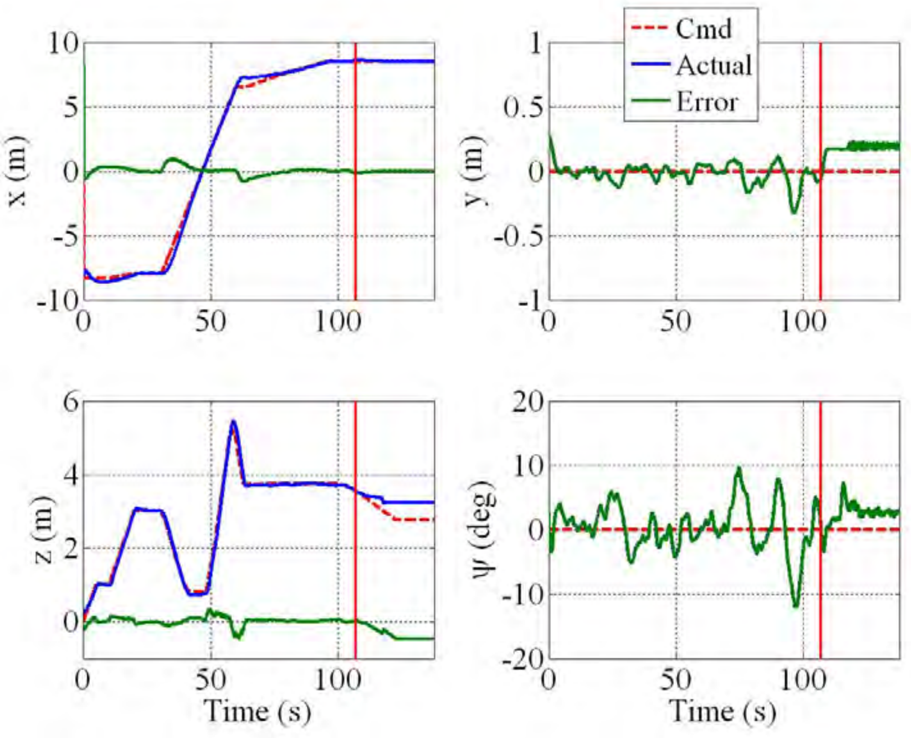
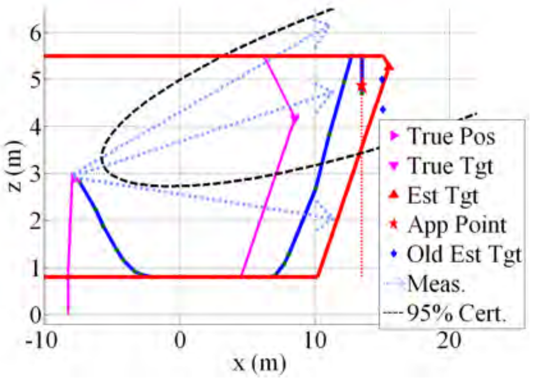
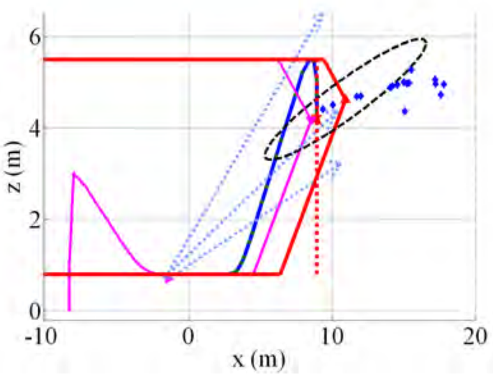


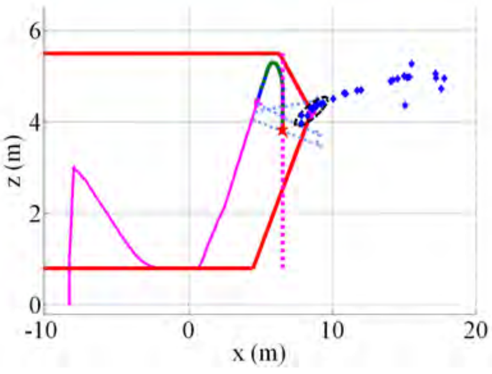
Figure 60. Commanded vs Actual Flight Path, Flight Test Run #2



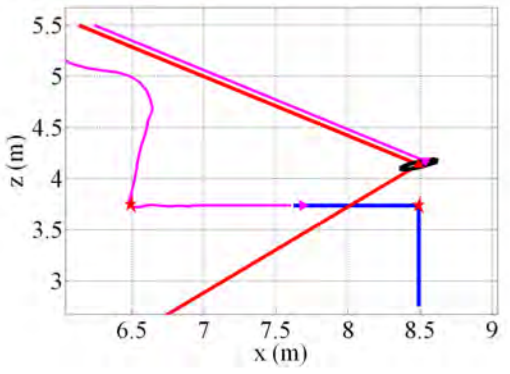
(a) $t_0 = 1.9$ s



(b) $t_0 = 14.5$ s



(c) $t_0 = 26.5$ s



(d) Flare Mode

Figure 61. Snapshot Progression of Flight Test Run #2

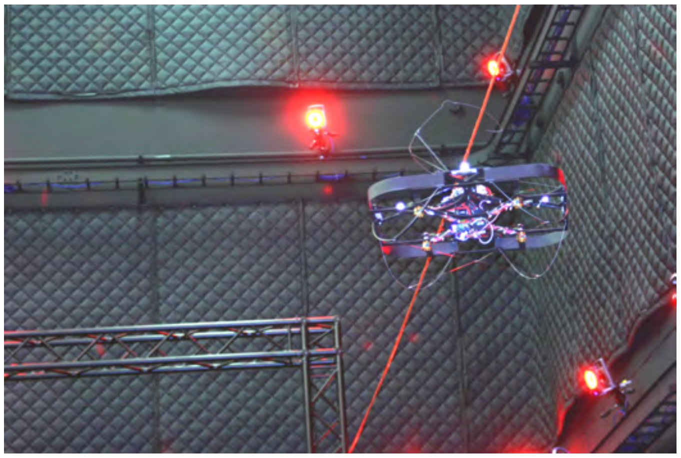


Figure 62. Quadrotor Just Prior to Hook Engagement

X. Conclusions and Future Work

THIS research successfully developed a method to simultaneously solve the optimal control and the optimal estimation problems. A recursive algorithm was designed to implement the method in real-time for disturbance rejection and treatment of uncertainties in the model and measurements. The solution is comprehensive, and was verified in flight test—autonomously landing a quadrotor helicopter on a wire as an enabler for the future capability of energy harvesting. This method may be applied to any system with a bearing-only sensor that requires relative position information about a source in order to perform its primary mission.

10.1 Conclusions

The most obvious conclusion from this research is that a vehicle may be guided to—and landed upon—a wire using stochastic, delayed, bearing-only measurements. Future systems that may benefit from energy harvesting are sensor limited, and most systems of such size have only a monocular camera. Additional sensors may be desirable for landing on power lines, but are not required. Furthermore, this study provides support that the Unscented Kalman Filter is a suitable estimation tool for such applications, and that real-time optimal control may be applied to direct a path that will acquire the level of target position certainty necessary to commit to a landing maneuver.

In the realm of trajectory optimization, several conclusions can be drawn from these efforts. The first is that there is a fundamentally different way to approach the localization and dual control problems that is more suitable and effective than traditional methods. The ubiquitous technique of optimizing a cost functional comprised of a scalar approximation of a multi-dimensional certainty metric has several disadvantages that are overcome with the methods developed in this work.

In the new approach, a user retains the directional information that was formerly approximated by a scalar, dispensing with difficulties of randomly odd-shaped uncertainty ellipsoids and other effects of information compression. This allows shaping of the uncertainty to match the physical requirements of the system, such as the actual shape of an arresting hook on a sUAS. Furthermore, previous methods minimized current uncertainty as much as possible, vice to a specific level. This research has now provided a way to prescribe the *final* uncertainty, which is the true requirement for mission accomplishment. Without this ability, a vehicle will maneuver as much as it can until an arbitrary time, or perhaps will balance the amount of maneuvering based on some arbitrary weight on the current certainty. Either way, it will not know whether it will achieve the necessary amount of target information, or whether it has wasted effort collecting too much information until the vehicle reaches the point where the information is required, when it is too late.

Early efforts provided a shooting solution which would allow a user to prescribe a final covariance. Trial solutions would be checked for the expected final covariance, iterating the weighed cost functional until the path produced yielded the right size and shape final certainty. This method was eclipsed by an elegant, single-shot solution that simultaneously handles the optimal control desires without weighting adjustments while meeting the physical information needs of the system. The single-shot solution was made possible by augmentation of the system state vector with states that contain an estimation in the polynomial space of the knowledge gained by the constellation of discrete measurements normally expressed by the Fisher Information Matrix. Dynamics were developed for these information states, and with care to avoid singularities, boundary conditions were enforced to ensure that by the time the system arrives at the desired final state, it will have collected the appropriate measurements, from the necessary angles, to finish the flight with the desired certainty in the target location estimate.

A further conclusion drawn is that the requirement of fixing a final time in the dual control problem can now be changed to a free final time. This was previously done either explicitly, or implicitly, through methods such as fixing a final distance with a given closure, fixing the total number of measurements with a given update rate, or by fixing an allowable travel proportion of the estimated distance to the target (with a constant speed). A fixed final time is a significant limitation for application to real systems beyond simulation. In reality, the time that will elapse during maneuvers not yet solved for is unknown, as is the number of measurements that will be required to meet the final mission requirements. The proportion of distance relative to the initial unknown distance is obviously also unknown. Choosing any of these, or more directly just choosing the fixed final time ends up being a primary driver of the characteristics of the solution trajectory. A solution that is truly optimal must be able to vary the problem geometry to get the required number of measurements from the necessary angles to accomplish the mission without limiting the set of possible solutions to those paths which end at a particular final time.

Several conclusions can also be drawn in the area of RTOC. The successful application of a recursive algorithm with pseudospectral methods as the engine working sequentially with a UFK receiving measurements from a bearing-only source is of great benefit. It validated the theory of disturbance rejection and the ability to use the speed of the pseudospectral methods to produce solutions that can guide in real-time. Applying the theory to real hardware produced several tools that were not required in previous PSM RTOC simulations, such as an intermediate function to address the asynchronous timing loops between a control system and an unpredictable optimal solver.

This work clearly demonstrated that allowing the calculation time of the optimal solver to vary has great value, increasing the flexibility and responsiveness of the system by increasing the rate of available optimal solutions. The structure necessary to address the potential discontinuities that result from achieving this benefit was also designed and implemented, using a blending solution to ensure smooth and accurate control with the most current data from both the path planner and the estimation filter.

From a systematic perspective of basic RTOC implementation, this research showed that the trend in the RTOC community of equating closed-loop feedback control with a fast, recursive optimal solution is insufficient. This conclusion has developed over time as a byproduct of most of the RTOC applications being limited to simulation. Non-zero mean and time-correlated biases will cause steady-state errors that will be unaccounted for by a purely feed-forward solution. Though such a system will reach the final condition, the optimality of the path it takes is more of a mathematical construct than an operational reality. A more comprehensive and effective method must account for the anticipated future effects of disturbances and model inaccuracies. This can be done through classical integration of the error between the expected and actual paths, and through feedback of disturbance estimates into the dynamical model for each optimal solver epoch. The ideal RTOC structure is to accomplish both, updating the model with estimates, and applying a total control solution that is a combination of the open-loop optimal solution and an integrated error feedback component.

Finally, this research provides a planning tool that may be used to develop heuristics for a suboptimal approach to landing on a power line that may be sufficient for systems with significant computational limitations, as may be the case for many sUAS platforms. Re-creating the single-shot solution with the particular system dynamics and limitations would provide the characteristics common to optimal solution paths.

10.2 Future Work

There are many directions of research that can be pursued from this point. The likely fielded implementation of a full RTOC path planner is for submarine guidance. To modify the problem, the axes must first be simply rotated into the horizontal. A study should be made to determine whether adding a third dimension would be beneficial or not, based on the ratio of relative pickup ranges to vertical maneuvering ability. Previous research has decided that it is not necessary, but if it is added, the information states will need to be increased to 6 elements, and if the final covariance is still the required parameter of choice, a differentiable method for solving or approximating the 3-dimensional FIM inverse will be required.

To incorporate the likelihood of a moving target, the estimation filter must be expanded to include states for target velocity and target heading. The RTOC problem can accept these as constants, and plan the path based on the assumption that the target will not maneuver. If future maneuvers do occur, the path planner will recursively solve the problem with the best information it has at the time. There are open questions along this direction, such as observability requirements (much like the power line problem, with both range and speed unknown, you can receive the same bearing measurements for infinite paths unless the observer maneuvers). In addition, adding a second measurement source for a towed array, and a velocity input from Doppler measurements would make the solution fieldable.

For the sUAS problem, the optical requirements remain unaddressed. An optical line detection algorithm should be implemented, either new, or with existing technologies. This could also be expanded to include stadiametric ranging, since the approximate height-above-ground may be known for the power line or the utility poles. Identification of utility poles in the image would also be of benefit. If the problem can detect lateral motion in relation to the line, then lateral motion will improve observability, and the 3-dimensional FIM should be incorporated as discussed. To incorporate the ability to land upon a ledge or other perch, only the optical requirements for determining an appropriate landing site change (and the flare segment, obviously). The approach segment method used in this work can be used interchangeably, with the safety stand-off distance used herein to avoid hitting a window or other structure by commanding no flight past a safe limit until the range to the ledge is sufficiently certain. The size of the safety limit can shrink in accordance with the current certainty level for that epoch.

The flight test can also be expanded for realism. Adapting the system to a fixed-wing asset and accomplishing the flight test outdoors on a full-sized power line would obviously be ideal, and would drive solutions for more significant disturbance rejection, especially in the landing phase. If the same quadrotor is used, the hook should be redesigned with an "open mouth" that will allow it some vertical error, and a way to detect line engagement, so that it may "drive through" the line estimate, and not need to know it so precisely. The flight control system must also be improved, using feed-forward optimal control inputs with feedback of trajectory error. Lastly, the power-to-weight problem of the device used to recharge the battery inductively needs to be addressed, as current solutions are too heavy for very light platforms.

10.3 Summary

In summary, this work provided a method that can be applied to a system with any given dynamics, and with any cost function, that will allow it to be guided in relation to an estimated target location to accomplish a potentially unrelated primary control mission. Deviations from the optimal path will be made to collect bearing-only measurements in sufficient quantity and with a sufficient angular orthogonality to identify the target location, without wasting maneuver effort beyond the minimum necessary to provide the level of certainty in the target location estimate that is required for mission accomplishment. This method can be modified to apply it towards guidance of submarines using passive sonar, HARM missiles, or other bearing-only systems. For the future capability of energy harvesting, the system can guide a sUAS from a point with an initial bearing measurement to a power line to an approach point from which a flare maneuver can be commenced for landing. With the current capability to autonomously guide a system to a location where a power line can be found, and the current research in the area of the actual flare maneuver, this research makes full-scale landing on a power line a near-term technology.

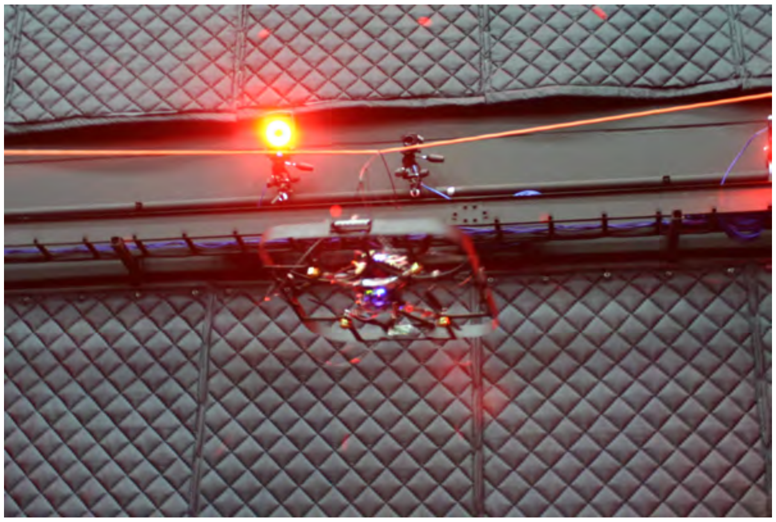


Figure 63. Engine Shutdown

Appendix A. Quadrotor Flight Control Model

THE simulator developed for the quadrotor helicopter is preserved in the Simulink diagrams of this appendix. Direction cosine matrices, equations of motion, and some other features that are either clear by context or covered in the main body of the dissertation are omitted.

A.1 Simulink Model

The main flow of the simulator can be seen in the top-tier diagram of Figure 64. A commanded path is generated either to judge performance and stability with step functions and the like, or to input a commanded profile from the path planner to test tracking performance. Testing of the hover mode was performed with the next block (Figure 65) to ensure that the system would lock a current commanded position when a button on the hand controller was pressed and released (initiation happens on the "release frame" vice the "press frame" to avoid multiple actuations). Initial conditions also must be compensated for in the hover block for use with the automated flight mode. For the hand control mode, if no command is made, the aircraft should not move from the place the engines are started (else the aircraft would jump to the navigation frame origin). Initial conditions are therefore added as offsets to the hand control commands. Since this happens "downstream" in the code, the additive inverse is added during automated flight to cancel the effect out and ensure that the aircraft flies to the actual navigation frame input. The automated commands for the aircraft are derived in real-time to ensure the shell always starts from the vehicle's true initial position.

The zero-order-hold blocks in Figure 64 discretize the model. Commands are discrete in order to use the same transfer functions as are required in the true controller,

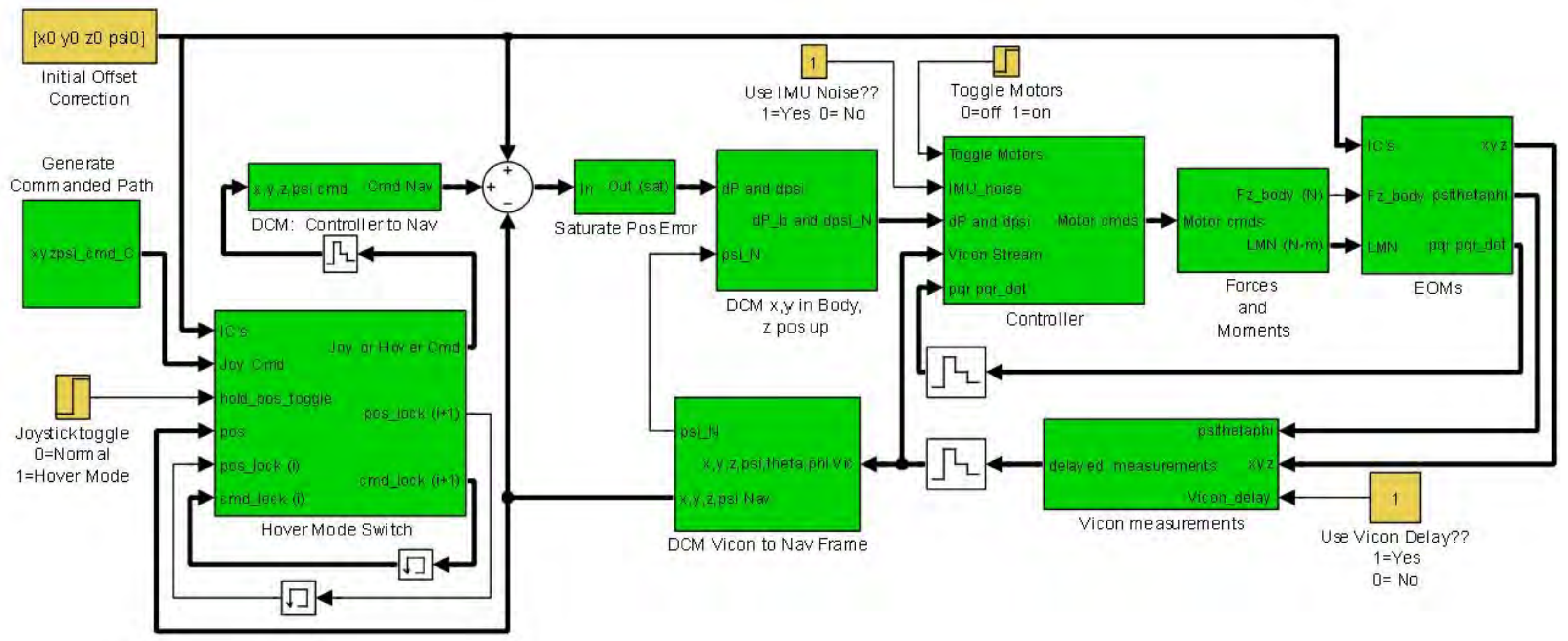


Figure 64. Quadrotor Simulator Top-Tier

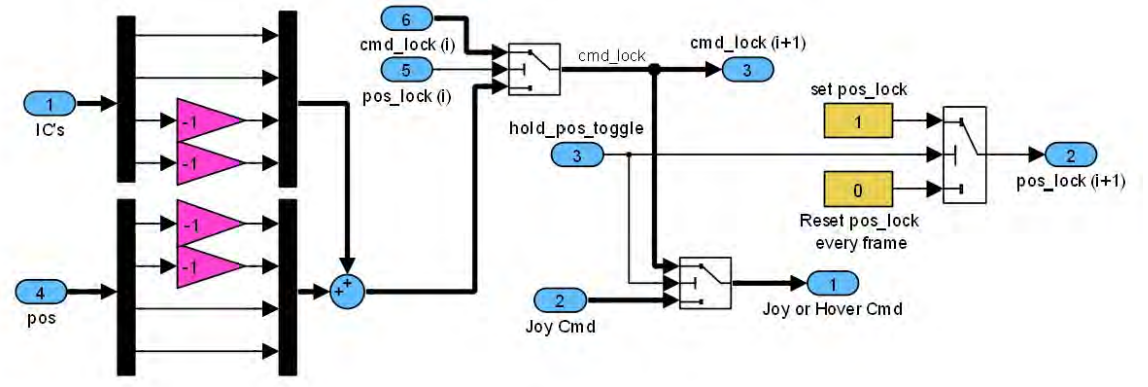


Figure 65. Logic to Initiate Hover Mode, Lock in Current Position, and Compensate for Initial Conditions

but equations of motion are all treated continuously. The first DCM transforms the commanded coordinates from the hand controller axes to the navigation frame, based on the position the observer pilot expects to stand in relation to the room. Commanded position is then compared to expected, and the error signal is saturated, with different levels in each axis, to control the maximum velocity as discussed in Chapter VIII. The resultant error signal is then transformed into the body frame and sent to the ground station controller, which generates control signals for the inner loop controller on board the aircraft, as shown in Figure 66. Ground station control laws for each axis are shown in Figures 67-69. Integration, sneakback, and anti-chatter logic are shown for the x-direction in Figures 70-72. The logic is the same for the y-axis, and is similar in the z-axis, which includes integration and reset logic, but does not require sneakback or anti-chatter. The control signal from the ground station is sent to the servo-sensor board on the aircraft, which is modeled in Figure 73. This is the inner stabilization loop, and it contains an input to simulate IMU noise, as well as the discrete lag filters used to estimate angular accelerations based on the angular rate commands. The commands for each axis are combined in a mixer to determine the actual motor commands. The mixer is easier to understand in equation form:

$$\begin{bmatrix} Motor0_{cmd} \\ Motor1_{cmd} \\ Motor2_{cmd} \\ Motor3_{cmd} \end{bmatrix} = \begin{bmatrix} 1 & 1 & -1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 \\ 1 & -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \dot{p}_{cmd} \\ \dot{q}_{cmd} \\ \dot{r}_{cmd} \\ throttle_{cmd} \end{bmatrix} \tag{169}$$

where the motors are numbered clockwise, starting with zero in the front left corner when facing in the positive x direction.

The forces and moments are determined in Figure 74, using an assumed (not measured) first-order model for motor spin-up delay with the engine thrust and torque models:

$$Torque\ (\mathrm{N-m}) = 4.16029e^{-5}(\mathrm{PWM})^2 - 0.09592(\mathrm{PWM}) + 55.49559 \tag{170}$$

$$Thrust\ (\mathrm{N}) = 6.78e^{-6}(\mathrm{PWM})^2 - 0.009868(\mathrm{PWM}) + 2.90352 \tag{171}$$

Each motor acts 0.15-m from the centerline for purposes of calculating the actual moments. With the forces and moments, the position and orientation of the aircraft is solved for using Equations 156-163 in Chapter VIII. Some delay can be added to the position and orientation to provide the discrete Vicon measurements, but in practice this was found to be negligible. Finally, the measurements are rotated into the navigation frame to complete the top-tier loop. Gain values, constants, and other specific details can be found in the initialization file in Table 4.

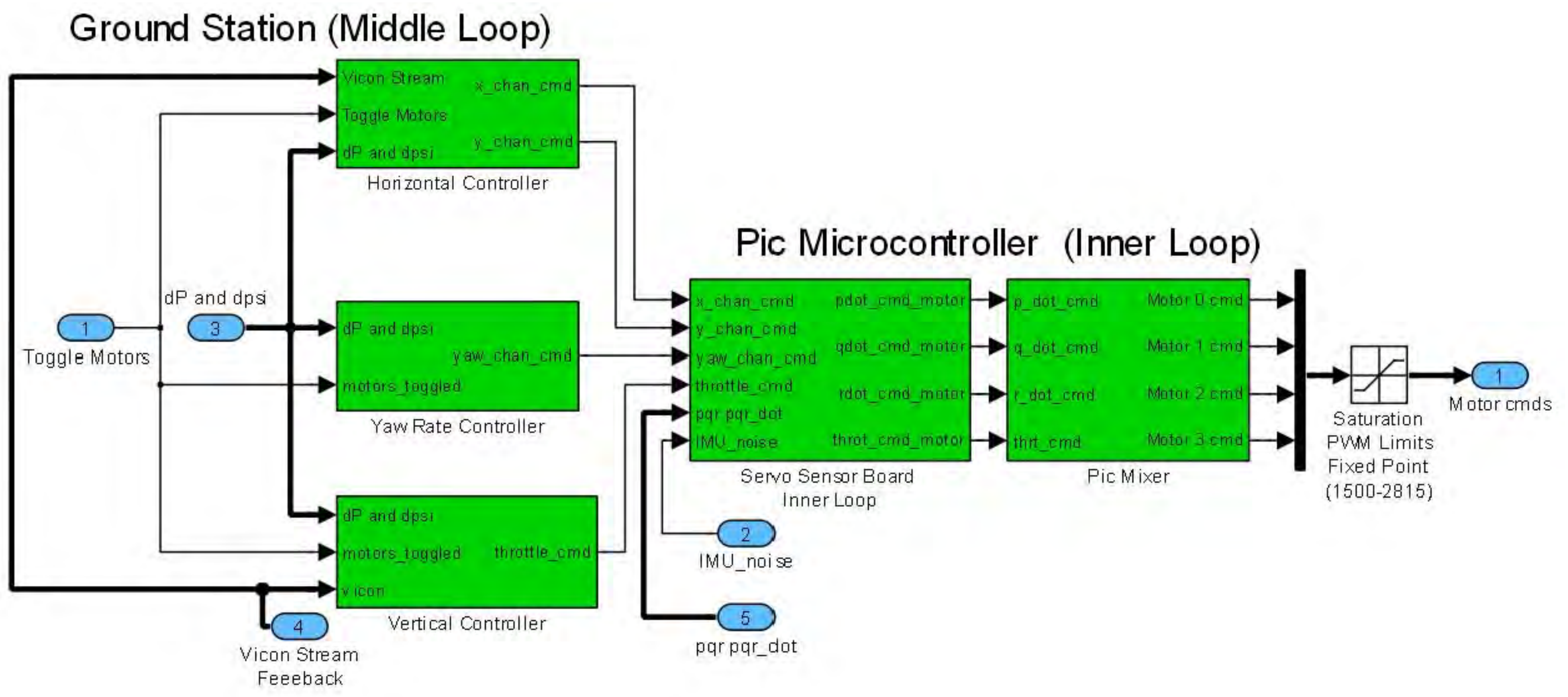


Figure 66. Ground Station and PIC Controllers

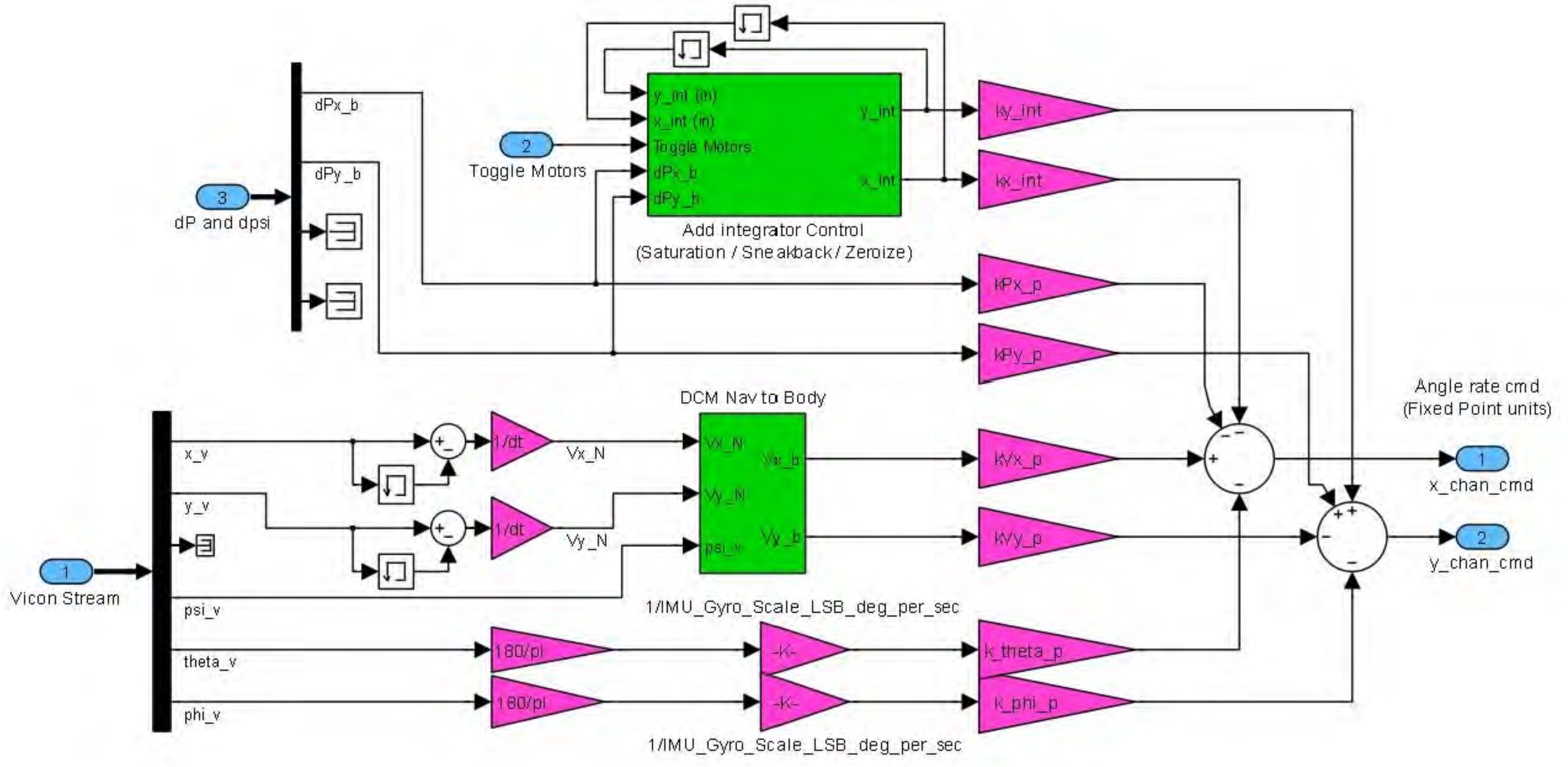


Figure 67. Horizontal Control Laws

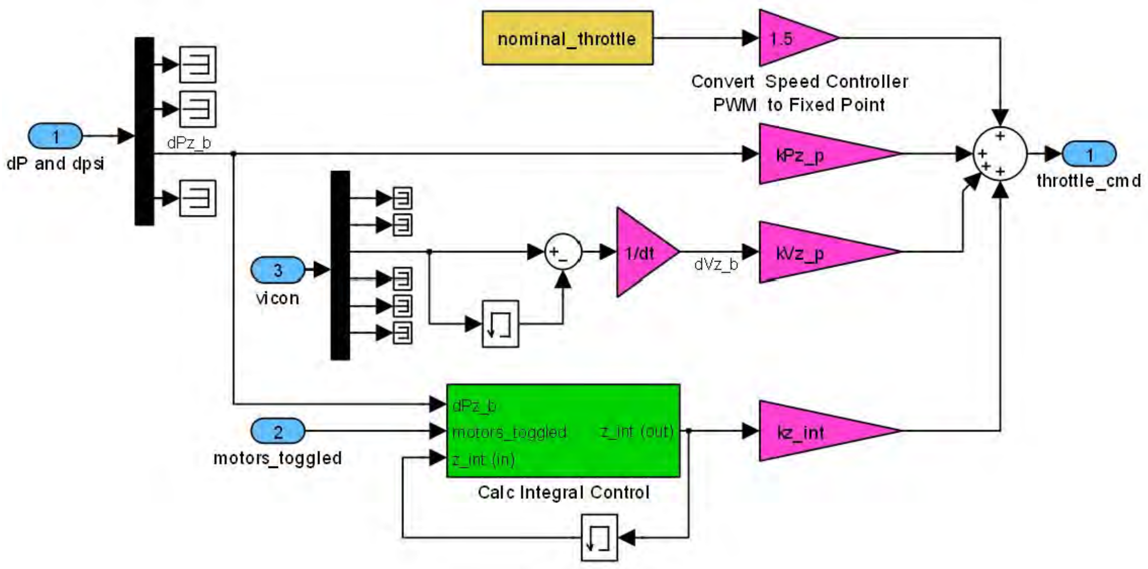


Figure 68. Vertical Control Law

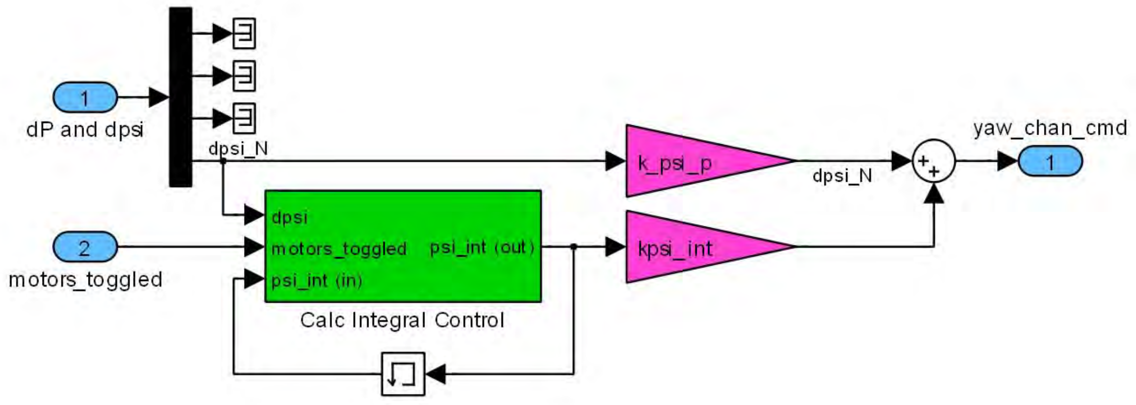


Figure 69. Heading Control Law

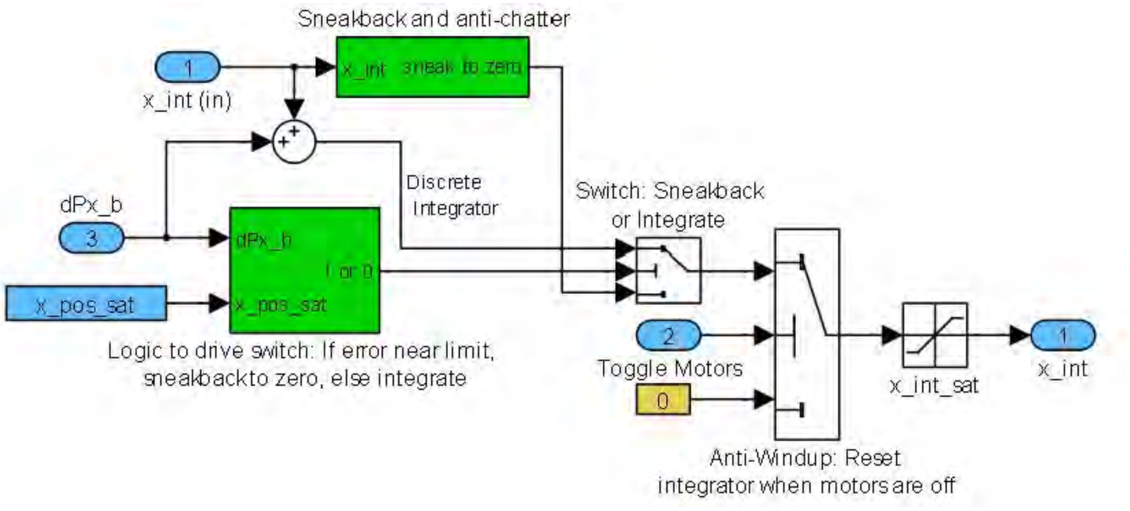

Sneakback and anti-chatter
1
x_int (in)
x_int sneak to zero
Discrete Integrator
dPx_b
3
dPx_b
1 or 0
x_pos_sat
x_pos_sat
Logic to drive switch: If error near limit, sneakback to zero, else integrate
Switch: Sneakback or Integrate
2
Toggle Motors
0
x_int_sat
1
x_int
Anti-Windup: Reset integrator when motors are off

Figure 70. Integrator Management Logic—Integrate, Reset, Anti-windup, Anti-Chatter (Also Used for y-Axis)

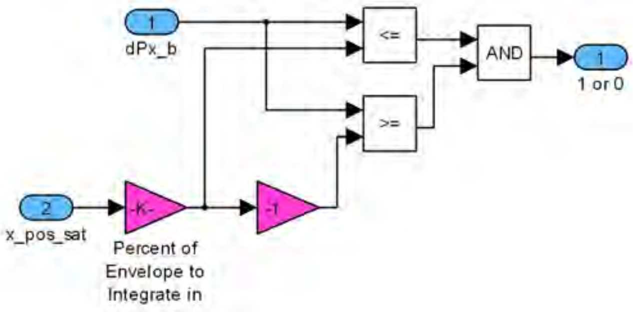


Figure 71. Subsystem for Switch Logic—Integrate if within Limits, Else Pull Integrator Back

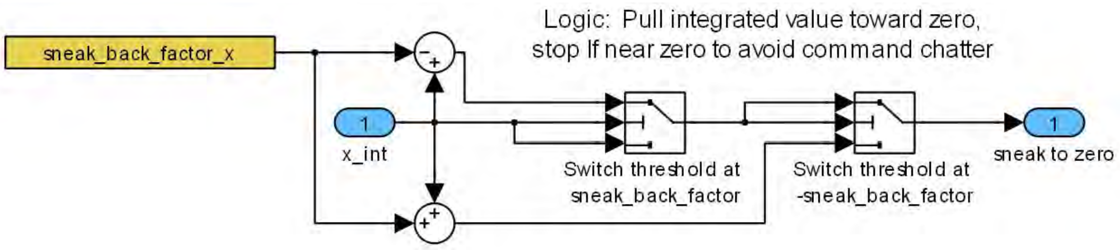

sneak_back_factor_x
Logic: Pull integrated value toward zero, stop If near zero to avoid command chatter
1
x_int
Switch threshold at sneak_back_factor
Switch threshold at -sneak_back_factor
1
sneak to zero

Figure 72. "Sneakback" and Chatter Avoidance Subsystem

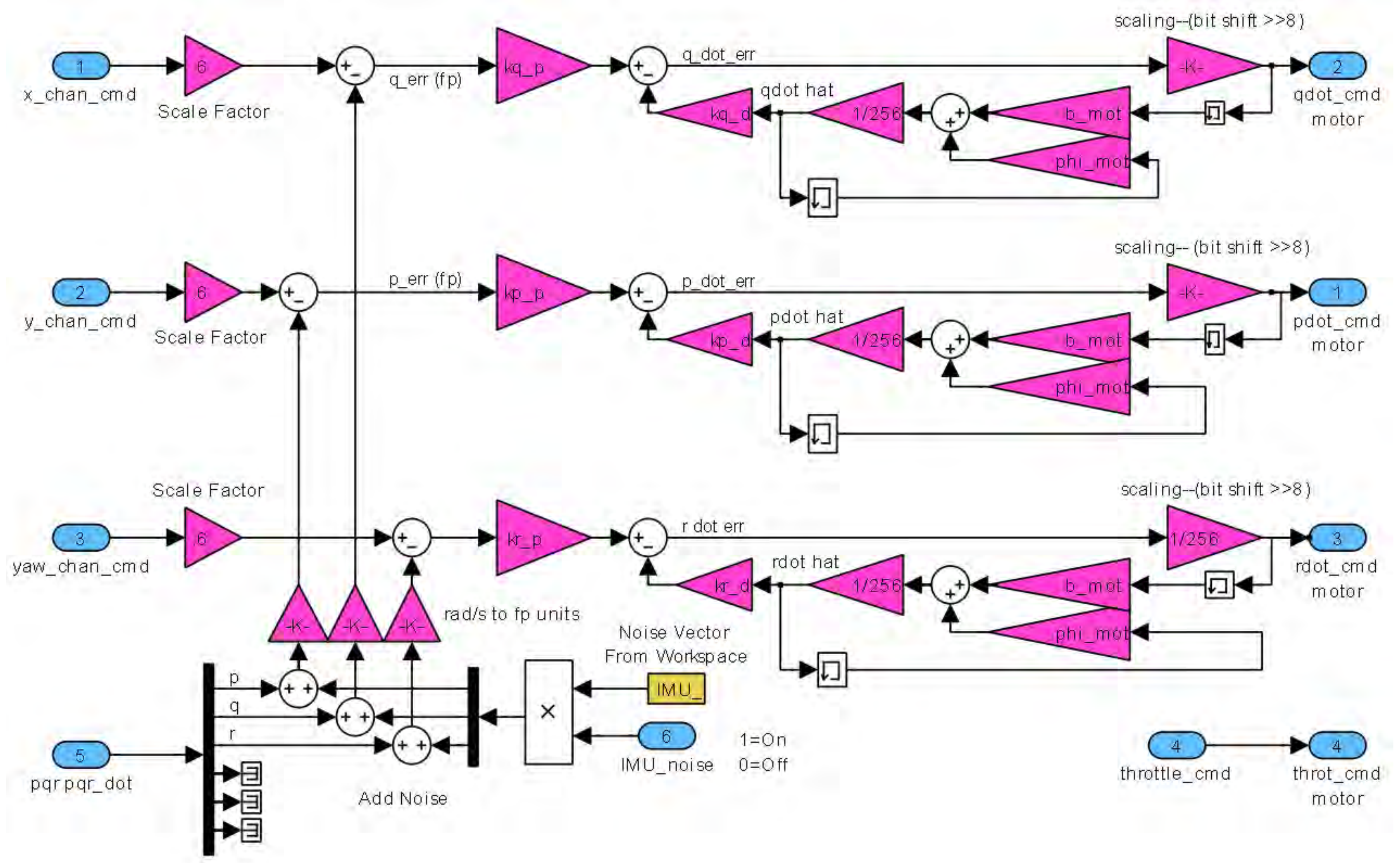


Figure 73. Inner Control Loop (Servo Sensor Board Model with Simulated Noise Input)

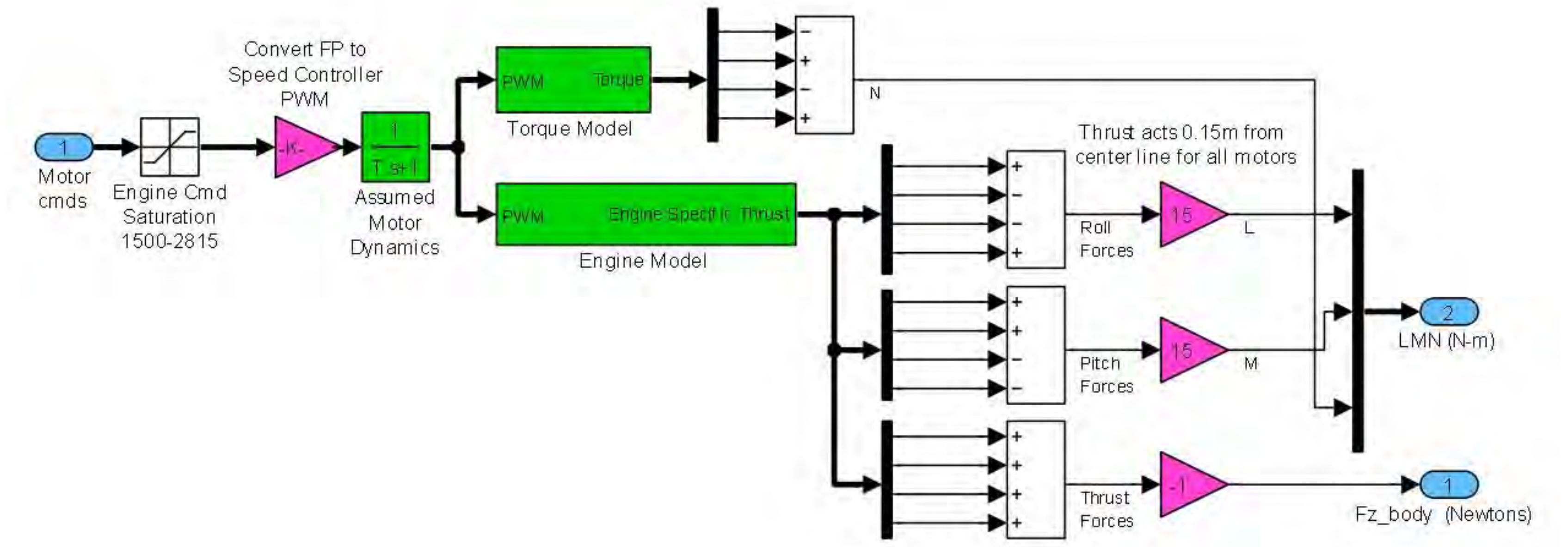


Figure 74. Forces and Moments: Motor Dynamics, Thrust, and Torque Models

Table 4. Simulator Initialization and Constants

Parameter	Value[1]	Comments
g	9.81 m/s^2	
m	1.8 kg	Mass, Batteries Installed
dt	0.01 s	Ground Station Time Step
Vicon_delay	0.1 s	Conservative Delay Estimate
I_{xx}	0.0203 kg-m^2	Mass Model
I_{yy}	0.0205 kg-m^2	
I_{zz}	0.04 kg-m^2	
k_psi_p	4000	Angle Feedback
k_theta_p	6	
k_phi_p	6	
k_Px_p	4000	Proportional Error Feedback
k_Py_p	4000	
k_Pz_p	150	
k_Vx_p	1750	Velocity Damping
k_Vy_p	1750	
k_Vz_p	100	
kp_p	12	Inner Loop (SSB) Angle Rate Feedback
kq_p	12	
kr_p	12	
kp_d	180	SSB Angular Acceleration Feedback
kq_d	180	
kr_d	180	
max_x_vel	1 m/s	Speed Limits
max_y_vel	1 m/s	
max_z_vel_up	1 m/s	
max_z_vel_down	1 m/s	
nominal_throt	1680	
T_{motor}	0.03 s	Lag Constant (Estimated)
IMU_GYRO_SCALE	0.0091575	Least Significant Bit to Fixed Point
b_mot	46	Input Matrix for Angular Accel Estimate
phi_mot	210	Transition Matrix for Angular Accel Est.

Continued on next page...

(continued)

Parameter	Value	Comments
recovery_time	10 s	Controls to Tune Integrator Speed
recovery_time_z	7.5 s	
recovery_time_yaw	10 s	
Integrator Saturation and Gain Values are Solved for Based on the Recovery Time to Produce the Desired Maximum Velocities		
kx_int = kPx_p/recovery_time/50		Adjusted for 50-Hz Controller
ky_int = kPy_p/recovery_time/50		
kz_int = kPz_p/recovery_time_z/50		
kpsi_int = k_psi_p/recovery_time_yaw/50		
x_pos_sat = 1.5 * max_x_vel * kVx_p / kPx_p		
y_pos_sat = 1.5 * max_y_vel * kVy_p / kPy_p		
z_pos_sat_up = max_z_vel_up * kVz_p / kPz_p		
z_pos_sat_dn = max_z_vel_dn * kVz_p / kPz_p		
x_int_sat = x_pos_sat * kPx_p / kx_int		
y_int_sat = y_pos_sat * kPy_p / ky_int		
z_int_sat_up = z_pos_sat_up * kPz_p / kz_int		
z_int_sat_dn = z_pos_sat_dn * kPz_p / kz_int		
psi_int_sat = $\pi/2$ * k_psi_p / kpsi_int		
sneak_back_x = 0.5 * x_int_sat / 50 / recovery_time		
sneak_back_y = 0.5 * y_int_sat / 50 / recovery_time		
sneak_back_z = 0.5 * z_int_sat_dn / 50 / recovery_time		

[1]Gains will convert to Fixed Point Units (balance may look wrong)

Appendix B. Selected Matlab® Code

For length considerations, the complete code required for the system cannot be presented here, but a few items are which may be of particular interest. The main path planning loop is shown to aid in understanding of the flow, and the initialization section walks a user through the lengthy connection process to get the path planning computer networked and synchronized with the ground station computer, Vicon, and the aircraft. This may be useful to those who would like to apply any similar external control system to the ground station for automated flight control in the ANT Center or the μAVIARI, regardless of the specific vehicle. The functions required to interface with $\mathcal{GPOPS}$ are also presented, as the format may be of particular use to future researchers that may require the software for other applications. All of the further path planner subroutines are omitted, as they are relatively application specific. The current version of the quadrotor flight code is also not presented, as it is a work in progress and will shortly be obsolete.

B.1 Main Path Planner Loop

The main path planner loop guides the processes of calculation and communication between the flight control software and the optimal path planning software. Future control time histories are passed to the dealer function, which parses the history and feeds the correct heading and position commands to the flight controller. In return, the dealer function provides the current position of the vehicle, as well a list of recent angle measurements from the vehicle to the wire. The path planner then iteratively calls the estimation filter and the optimal control solver to update the target estimate and the optimal path, handling projection for initial conditions and expected covariance, path blending, and data recording.

The optimal trajectory provided by the solver is spliced into the complete control history "shell," which also includes segments for the takeoff sequence, the flare mode, and a backup landing mode in case the wire is not engaged with the current plan and a further plan is never provided. Scaling of the path (and of the measurements) is provided by subroutines so that the vehicle may operate in both the ANT Center and the larger μAVIARI using the same planning algorithm.

Communication between the path planner and the dealer function is accomplished by way of a TCP connection that relies on a TCP/UDP/IP toolbox created by Peter Rydesater and can be downloaded from the MATLAB® Central File Exchange at: http://www.mathworks.com/matlabcenterl/fileexchange/345. The toolbox is built with .mex files, and a .dll file must first be created with any C compiler. The communication tools provide the ability to pass the data using the "blocking" techniques discussed in Section 7.1.3.1, allowing variable calculation timing. Breaking the message into TCP packets and reshaping them in the dealer function is performed by a subroutine written by Mr. Mark Smearcheck.

Main Path Planner Loop Code.

```
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Main Quadrotor Shell                                                  %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%This is the main program for flying the quadrotor, and it is the main loop
%for timing and communication between the Vicon/Groundstation, the optimal
%path planner, and the UKF. It can also run in simulation mode without
%input from the Vicon and the vehicle.  Running the program will walk a
%user through the pre-flight initialization and connection process.
%
%Written by LtCol Steven Ross, AFIT, 2011.


clear all;
home
global WHERE_TO_RUN WHAT_TO_RUN FASTMODE CONST
figure_cascade = 1;    %Compare multiple figure sets

%% %%%%%%%%%%%%%%%%%%%%
%% Flight Mode options %%
%% %%%%%%%%%%%%%%%%%%%%
sim        = 0;  %1=simulation mode, 0=flight mode
display_on = 0;  %1=display on in real-time (disrupts timing)
```

```
ANT        = 0;  %1=scale (both directions) 1=Fly in ANT center, 0=AFRL


%% %%%%%%%%%%%%%%%%%
%% Configure Solver %%
%% %%%%%%%%%%%%%%%%%
limits.nodes = 30;  %Global only, not used for hp local w/ GPOPS 3.3
WHERE_TO_RUN =  1;  %Location of GPOPS: 1=C drive, 2=I drive, 3=L drive
WHAT_TO_RUN  =  3;  %1=GPOPS 2.4   2=GPOPS 3.2   3=GPOPS 3.3
FASTMODE     =  1;  %1=Modified GPOPS,  else=normal GPOPS

% Subroutine to set path and initialize solver
configure_path_for_gpops(WHERE_TO_RUN, WHAT_TO_RUN)
current_solution = [];  %init


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Setup Options to define flight shell         %%
%% Coordinates are: (vicon)[x,y,z(m),psi(rad)]  %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
shell.dt_path_planner = 0.2;   %s  Time per line in shell
shell.flight_time     = 180;   %s  Makes 901 lines to dealer

% Start a/c around [-8.84 0 0  0] (8.84m is 29 ft, position not critical)
shell.flight_time_to_1m               =  5;  %s
shell.hold_time_at_1m                 =  5;  %s
shell.start_run_point                  = [-8 0 3 0];  %m
shell.flight_time_to_start_run_point = 10;  %s
shell.hold_time_at_start_run_point   = 10;  %s

%post run (abort plan if no hook engagement):
shell.time_to_descend_to_1m          = 10;  %s
shell.hold_time_before_land_at_1m    =  5;  %s
shell.flight_time_1m_to_land         = 10;  %s
shell.time_to_hold_land_position     =  5;  %s
%Total slots-landing mode slots = the time slot to transition to land mode
shell.landing_slot=(shell.flight_time - shell.time_to_descend_to_1m ...
    -shell.hold_time_before_land_at_1m - shell.flight_time_1m_to_land ...
    -shell.time_to_hold_land_position)/shell.dt_path_planner;
last_update_line = shell.landing_slot;  %init.  Changes during land mode

%hook engagement plan:
shell.hold_at_approach_point = 5;  %seconds to settle/correct height
shell.perch_speed            = 1/12;  %m/s
shell.correction_time  = 5;  %blend in vertical tgt est. changes over 5 sec
shell.stop_update_time = 5;  %s--freeze the tgt updates


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Setup first trajectory planner run  %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dt_gpops = 0.9; %Est. inclusive loop time--planner calc, delay, filter...
P=[80 0; 0 80]; %m^2 CURRENT uncertainty in target pos (not IC), Cartesian
IC.P0 = P; %Uncertainty expected at next initial planning point (Cartesian)

CONST.Pxx_f = 0.02; %Final Covariance Element to be met
CONST.Pzz_f = 0.02; %Final Covariance Element to be met

IC.t0=shell.flight_time_to_1m+shell.hold_time_at_1m+ ...
   shell.flight_time_to_start_run_point+shell.hold_time_at_start_run_point;
IC.x0 = shell.start_run_point(1); %m vicon frame, next planning point
IC.y0 = shell.start_run_point(2); %m vicon frame, next planning point
IC.z0 = shell.start_run_point(3); %m vicon frame, next planning point
IC.psi0  = 0;
IC.xdot0 = 0;         %m/s Exp obs horiz velocity at next planning point
IC.zdot0 = 0;         %m/s Exp obs vert  velocity at next planning point
```

```
CONST.x_hat_tgt = 15; %m   Inertial Coords, target initial guess
CONST.z_hat_tgt =  5; %m   Inertial Coords, target initial guess


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Define physical limitations--Room, Quadrotor, Bearing Measurement Sensor
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
limits.perch_offset_x =    0;    %m Desired offset from wire at perch point
limits.perch_offset_z = -0.4;    %m
limits.x_approach_offset =-2;    %m Desired offset from wire at app. point
limits.z_approach_offset = -0.4; %m
limits.slush    =   0;       %miss distance at app point (smoother, slower)
limits.min_alt  = 0.8;       %Hard Deck for "run" portion, vicon z (m)
limits.max_alt  = 5.5;       %Flight ceiling, vicon z (m)
limits.min_x    =  -9;       %Allowable envelope vicon x (m)
limits.max_x    =   9;       %(cameras are intermittent near the wall)
limits.beta_min = -(30)*pi/180;  %camera lower limit (rad)
limits.beta_max =  (40)*pi/180;  %camera upper limit (rad)
limits.xdot_max = 0.5;       %m/s   Horizontal velocity limit
limits.zdot_max = 0.5;       %m/s   Vertical velocity limit
limits.xddot_max= 0.5;       %m/s^2 Horizontal acceleration limit
limits.zddot_max= 0.5;       %m/s^2 Vertical acceleration limit

% Truth data for plots, measurement generation
CONST.x_tgt   = 8.555;  %m
CONST.z_tgt   = 4.02;   %m;
CONST.dt_meas = 0.33;   %sec. Expected time step of available measurements
CONST.R       = .005;   %rad^2  (std dev is a little over 4 deg)
noise=sqrt(CONST.R)*randn(shell.flight_time/CONST.dt_meas,1); %meas noise
total_measurements = 0; % init


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Setup Unscented Kalman Filter  %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
CONST.NUKFSTATES  = 2;
alpha             = 1e-3; % Sigma point spread
beta              = 2;    % Prior knowledge parameter (2 opt. for Gaussian)
kappa             = 0;    % Secondary scaling parameter
lambda            = alpha^2*(CONST.NUKFSTATES + kappa) - CONST.NUKFSTATES;
CONST.scale_param = CONST.NUKFSTATES + lambda;
CONST.W0m         = lambda/CONST.scale_param;
CONST.W0c         = CONST.W0m + (1 - alpha^2 + beta);
CONST.Wukf        = 1/2/CONST.scale_param;


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Setup Path Blending of tail for new target estimates  %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
blend_time = 5; %seconds for blending at the end of the path
blend_t    =(0:shell.dt_path_planner:blend_time)'; %time vec. for cos wave
blend_wave = .5-.5*cos(pi/blend_time*blend_t);     %gives 0 to 1 cos vector


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Setup End of Path--from approach point to perch point  %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
avg_speed_land_mode = 2;  %in/s
approach_time=(limits.perch_offset_x-limits.x_approach_offset)/ ...
    convlength(avg_speed_land_mode,'in','m');
app_lines=(0:1:ceil(approach_time / shell.dt_path_planner))';
app_wave1to0=.5+.5*cos(app_lines/app_lines(end)*pi);  % 1 to 0 cos vector
dx_from_perch=(.5+limits.perch_offset_x-limits.x_approach_offset) ...
    *app_wave1to0; %correction splice


```

```
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Get User inputs--check IP/subnet settings, get initial start point    %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if sim==1  %for the sim, just pick a hardcoded spot
    current_pos=get_current_position_sim(0,[]);
    if ANT==1
        current_pos(2)=0;
    end
    sprintf('%s','NOT ONLINE--SIMULATING CURRENT POS AND MEASUREMENT DATA')
else
    confirm1='n';
    while confirm1~='y'
        disp('--')
        disp('--')
        disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')
        disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')
        disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')
        disp('--')
        disp('--')
        disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')
        disp('%%System initialized in real-world flight mode.%%')
        disp('%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%')

        disp('--')
        disp('--')
        if ANT==1
            disp('System is in ANT Center mode--flight arena scaled')
            disp('Acknowledge with any key')
        else
            disp('System is in AFRL mode--no scaling')
            disp('Acknowledge with any key')
        end
        pause

        disp('Apply Network Settings:')
        disp('IP: 192.168.10.91')
        disp('Subnet mask: 255.255.255.0')
        confirm1=input('Update settings.  When correct, enter ''y'':','s');
    end
        confirm2='n';
    while confirm2~='y'
        disp('--')
        disp('--')
        disp('Confirm GS/UAV/vicon on, check frame rates, reasonable data')
        disp('Input current (start flight) position, Vicon Frame--')
        initial_x       = input('x(m)=');
        initial_y       = input('y(m)=');
        initial_z       = input('z(m)=');
        initial_psi_deg = input('psi(deg)=');
        disp('Confirm Initial Position (case sensitive)')
        initial_x
        initial_y
        initial_z
        initial_psi_deg
        confirm2=input('If correct, enter ''y'', else enter ''n'':','s');
    end
    %set initial position [line_num x y z psi(rad)]
    current_pos=[0 initial_x initial_y  initial_z initial_psi_deg*pi/180];
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Create first path shell for dealer  %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if ANT==1
    current_pos=scale_pos_from_ANT(current_pos);
end
```

```
%init:
data_for_dealer_all=zeros(shell.flight_time/shell.dt_path_planner+1,4,200);
path = build_front_shell_for_dealer(shell, current_pos);       %build path

%% Fill in a sample set of run data
Start_time = cputime;
current_solution = trajectory_planner(IC,limits,current_solution);

GPOPS_init_time = cputime-Start_time  %Display initial epoch planning time
current_solution.run_time=GPOPS_init_time;
path = build_data_for_dealer(path, current_solution, shell);

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Set up file saving for post processing  %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
data_for_dealer_all(:,:,1) = path;                        %save each path
gpops                        = cell(200,1);              %save GPOPS solutions
meas_save=cell(ceil(shell.flight_time/CONST.dt_meas),1); %save measurements
loop_time_save       = zeros(300,1);        %init
actual_position      = zeros(300,5);        %init
actual_position(1,:) = current_pos;         %init
current_solution.IC  = IC;
x_est                = zeros(300,1);        %init
z_est                = zeros(300,1);        %init
x_est(1)             = CONST.x_hat_tgt;
z_est(1)             = CONST.z_hat_tgt;
gpops{1}              = current_solution;
P_save               = cell(300,1);         %init
break_loop           = 0;


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Plot initial path for visual error checking prior to takeoff %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
zero_to_2pi = (0:2:360) * (pi/180); %rad (to display covariance circle)
%2xN, pairs of points on unit circle:
points_on_unit_circle=[cos(zero_to_2pi); sin(zero_to_2pi)];
old=plot_init_96 ...     %initialize main plot
    (current_solution, IC,limits,current_pos,points_on_unit_circle);
if ANT==1
    scaled_path=scale_path_to_ANT(path);
    plot_dealer_path(scaled_path,shell,0,figure_cascade)
else
    plot_dealer_path(path,shell,0,figure_cascade)
end
disp('--')
disp('--')
disp('System Initialization Complete')
disp('Confirm valid initial path, press any key')
pause


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Connect to dealer function, hold until dealer accepts initial path   %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if sim~=1

    % IP Settings
    PORT_CMDLST = 49993;
    PORT_TRUTH  = 49992;

    % Info for cutting path packets
    NUM_COMMANDS_PER_LIST   = 901;
    NUM_COMMANDS_PER_PACKET =  19;
    NUM_VALUES_PER_COMMAND  =   4;

    % Create a TCP Server to send command lists
    sockconCMDLST = pnet('tcpsocket',PORT_CMDLST);
    if sockconCMDLST == -1
        error('Specified TCP port unavailable for command lists');
    end
    disp('Command List Server created');

    % Create a TCP Server to poll for truth and truth history
    sockconTRUTH = pnet('tcpsocket',PORT_TRUTH);
    if sockconTRUTH == -1
        error('Specified TCP port unavailable for truth messages');
    end
    disp('Truth Server created');

    disp('Waiting for connections...Start the Dealer Now')

    % Blocks indefinitely until client connects for command lists
    conCMDLST = pnet(sockconCMDLST, 'tcplisten');
    disp('Command List Connection accepted');

    % Blocks indefinitely until client connects for truth and truth history
    conTRUTH = pnet(sockconTRUTH, 'tcplisten');
    disp('Truth Connection accepted');

    disp('Connection to Dealer valid.  Press "Connect" in ground station')

    % Wait for the ok before sending command lists
    okCMDLST = pnet(conCMDLST, 'read', 1, 'swap');
    if okCMDLST(1) ~= '1'
        disp('Error receiving ok to start sending command lists ');
    end

    % Send path to dealer
    if ANT==1
        scaled_path=scale_path_to_ANT(path);
        SendPathList(conCMDLST,scaled_path);
    else
        SendPathList(conCMDLST,path);
    end
end

%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Pre-run loop: takeoff to start_run point %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
dist_from_start_run_point=10;            %init (big)
dist_from_approach_point =10;            %init (big)
disp('Waiting to start Pre-loop--')
disp('Turn on Data Recorder, check valid IMU output, hit any key')
pause
disp('Start Ground station first, then Matlab:')
disp('                    1) IP Flight')
disp('                    2) Motors on')
disp('                    3) Start IP Command RX')
disp('                    4) Start MATLAB pre-loop (hit any key)')
pause
sprintf('Autopilot engaged, preloop--climbing to start run position')

if sim==1      %For simulation, just enter the preloop for a few seconds
    run_preloop_time    = 3;  %seconds
    time_start_preloop = cputime;
    time_integer       = floor(run_preloop_time);
else
    time_integer = floor(IC.t0);
end

%Run preloop until the start run time
```

```
while current_pos(1)*shell.dt_path_planner <= IC.t0-1

    if sim==1   %for simulation, just exercises the preloop
     preloop_time_remaining=run_preloop_time-(cputime-time_start_preloop);
      if preloop_time_remaining <= time_integer
        sprintf('SIM MODE.  Time remaining in pre-loop=%1.0f',time_integer)
        time_integer=time_integer-1;
      end
      if preloop_time_remaining <=0
        %set sim to 1 second prior to run, hovering at start run point
        current_pos=[ceil((IC.t0)/shell.dt_path_planner) ...
            IC.x0 IC.y0 IC.z0 IC.psi0];
      end
    else     %get current position from Dealer
        current_pos = get_current_position(conTRUTH);

        % AFRL vs ANT lab sign swap--temp fix  (number 1/3)
        current_pos(4)=-current_pos(4);

        if ANT==1    %scale if in ANT center
            current_pos=scale_pos_from_ANT(current_pos);
        end

        %display time remaining every integer
        preloop_time_remaining=floor(IC.t0-current_pos(1) ...
            *shell.dt_path_planner);
        if preloop_time_remaining<time_integer
            time_integer=preloop_time_remaining;
            sprintf('FLY MODE.  Time to start run: %1.0f',time_integer)
        end
    end
end


%% %%%%%%%%%%%%%%%%%%%%%
%% Init Main Run loop  %%
%% %%%%%%%%%%%%%%%%%%%%%
loop_ctr=2; %counter for knowing which slice to put data_for_dealer into
sprintf('Begin Main Run Loop')
slot_last_meas=ceil((IC.t0-CONST.dt_meas)/shell.dt_path_planner);
meas = [];   %no initial measurements
approach_point=[CONST.x_hat_tgt+limits.x_approach_offset ...
    CONST.z_hat_tgt+limits.z_approach_offset];
start_loop_time=cputime;

if sim==1    %sim mode: generate a fake list of time slots for measurements
    line_meas_sim= ...
        ceil(152:CONST.dt_meas/shell.dt_path_planner:shell.landing_slot);
    sim_clock     = cputime;
    display_fudge = 0;  %compensation factors (sim only) for graphics time
    display_time  = 0;
end


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Main Run Loop. Initiated from hover at start run point %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
while dist_from_approach_point>1 || P(1,1)>CONST.Pxx_f ||P(2,2)>CONST.Pzz_f

    % Splice GPOPS into shell, recalc the land mode to match end point
    [path_to_old_tgt approach_slot] ...
        =build_data_for_dealer(path, current_solution, shell);

    % Blend tail for changes in target estimate
    path=blend_path_to_updated_tgt(path_to_old_tgt, current_pos, shell, ...
        blend_time, blend_wave,approach_slot, approach_point);
```

```matlab

    % Save for post processing
    data_for_dealer_all(:,:,loop_ctr)=path;

    % Send path to Dealer
    if sim~=1
        if ANT==1
            scaled_path=scale_path_to_ANT(path);
            SendPathList(conCMDLST,scaled_path);
        else
            SendPathList(conCMDLST,path);
        end
    end

    %% Project down path, calc the conditions and expected cov at
    %% next GPOPS update time, bias for being off of commanded position
    pos_error=path(current_pos(1),:) - current_pos(2:5);
    IC=Calc_next_IC(current_pos, pos_error, IC,limits,current_solution, ...
        P, shell.dt_path_planner,dt_gpops, slot_last_meas);

    %% Plan next optimal path
    Start_time                = cputime;
    current_solution ...
        =trajectory_planner(IC,limits,current_solution);
    current_solution.run_time = cputime-Start_time;
    loop_time_save(loop_ctr)  = cputime-start_loop_time;
    start_loop_time           = cputime;

    %% Get position, time
    if sim==1
        display_fudge=display_fudge+display_time; %sim: add graphics time
        %obtain position from ground station [line_num x y z psi]:
        current_pos=get_current_position_sim(floor((29+cputime ...
            -sim_clock-display_fudge)/shell.dt_path_planner),path);
    else
        %obtain position from ground station [line_num x y z psi]
        current_pos=get_current_position(conTRUTH);

        % AFRL vs ANT lab sign swap--temp fix (number 2/3)
        current_pos(4)=-current_pos(4);

        % Scale back up if in ANT Center
        if ANT==1
            current_pos=scale_pos_from_ANT(current_pos);
        end
    end


    % Run the display, if on
    if display_on==1
       start_display_time=cputime;
       old=plot_single_display96(current_solution, IC,limits,old, ...
           current_pos,path,meas,points_on_unit_circle);
       display_time=cputime-start_display_time; %time spent making display
    end

    %% Record Data for post processing
    actual_position(loop_ctr,:) = current_pos;
    current_solution.IC          = IC;
    gpops{loop_ctr}              = current_solution;
    x_est(loop_ctr,1)           = CONST.x_hat_tgt;
    z_est(loop_ctr,1)           = CONST.z_hat_tgt;


    %% Get measurement locations from Ground station
    if sim==1
```

```
     %12x5 [line_num x y z psi] (last 12 meas locations, not in order)
     meas_locations=get_meas_locations_sim(path,current_pos,line_meas_sim);
    else
      meas_locations=get_meas_locations(conTRUTH);

      if ANT==1
       %scale up the meas locations
       meas_locations=scale_pos_from_ANT(meas_locations);
      end
    end

    %% Sort measurements that have not been incorporated (may be empty)
    [meas slot_last_meas] = get_beta (meas_locations, slot_last_meas);

    %% If there are new measurements, generate new target estimate and cov
    %% matrix with Unscented Kalman Filter
    new_meas=length(meas(:,1));

    if new_meas > 0
        %add measurement noise
        meas(:,3)= meas(:,3) ...
            + noise (total_measurements+1:total_measurements + new_meas);
        total_measurements=total_measurements + new_meas;
        for i=1:new_meas %UKF
            [Xrel P]=UKF_Cartesian(meas(i,:),P);
            CONST.x_hat_tgt=Xrel(1)+meas(i,1);   %update tgt estimate
            CONST.z_hat_tgt=Xrel(2)+meas(i,2);
        end
    end

    %% Save for post processing
    meas_save{loop_ctr}=meas;

    %% Update approach point with new tgt estimate
    approach_point=[CONST.x_hat_tgt+limits.x_approach_offset ...
        CONST.z_hat_tgt+limits.z_approach_offset];
    dist_from_approach_point = norm(approach_point - current_pos([2 4]));

    loop_ctr=loop_ctr+1;
    if display_on==1
sprintf('gpops: %g sec, loop: %g sec, dx= %g,  dz= %g', current_solution.run_time,loop_time_save(loop_ctr), ...
            CONST.x_tgt-CONST.x_hat_tgt,CONST.z_tgt-CONST.z_hat_tgt)
    end

end   % end main loop


sprintf('Exiting Main Loop--Required Position & Covariance Achieved')
Tgt_est_error_at_approach_point_inches=convlength([CONST.x_tgt...
    -CONST.x_hat_tgt CONST.z_tgt-CONST.z_hat_tgt],'m','in')
%update display when achieving approach position
if sim==1 && display_on==1
    old=plot_single_display96(current_solution, IC,limits, ...
        old,current_pos,path,meas,points_on_unit_circle);
    titleA=sprintf('Approach Parameters Achieved--True Tgt Err: x=%1.3g(in) z=%1.3g(in)', ...
        Tgt_est_error_at_approach_point_inches(1), Tgt_est_error_at_approach_point_inches(2));
    title(titleA,'fontsize',14)
end


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Initialize Landing Mode, splice approach into shell  %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
row2=approach_slot;

% X perching profile
```

```
x_remaining=limits.perch_offset_x-path(approach_slot,1);
%use the wave to smooth in, starting backwards from tgt and using only the
%distance that you have left
path_x=CONST.x_hat_tgt+limits.perch_offset_x-dx_from_perch...
    (dx_from_perch<x_remaining);
row3=row2+length(path_x);   %find slot to splice into
path(row2+1:row3,1)=path_x; %splice in x approach profile
path(row3+1:end,1)=path(row3,1); %hold final x for the rest of the flight

% Z perching profile
z_perch=CONST.z_hat_tgt+limits.perch_offset_z;
%hold and additional 2 seconds past when x finishes
row3_plus2sec=row3+ceil(2/shell.dt_path_planner);
path(row2+1:row3_plus2sec,3)=z_perch;
dist_per_line_1in_per_sec=convlength(1,'in','m')*shell.dt_path_planner;
% Drop 5 inches (.127m) at 1 in per sec, then 1m at 2in per sec
append3_z=[z_perch:-dist_per_line_1in_per_sec:z_perch-.127, ...
    z_perch-.127:-2*dist_per_line_1in_per_sec:z_perch-1]';
row_complete=row3_plus2sec+length(append3_z);
path(row3_plus2sec+1:row_complete,3)=append3_z;

if row_complete<shell.landing_slot
    % If no wire engagement, hold the last z position until landing time
    path(row_complete+1:shell.landing_slot,3)=path(row_complete,3);
end

% Recalc z from end of run point to 1 m hover (the rest doesn't change)
row_1m=shell.landing_slot+shell.time_to_descend_to_1m ...
    /shell.dt_path_planner;
path(shell.landing_slot+1:row_1m,3) ...
    =linspace(path(shell.landing_slot,3),1,row_1m-shell.landing_slot);

last_z_update=1;  %init, just for recording the last updated value


%% Send path
if sim~=1
    if ANT==1
        scaled_path=scale_path_to_ANT(path);
        SendPathList(conCMDLST,scaled_path);
    else
        SendPathList(conCMDLST,path);
    end
end

%% Save for post process
data_for_dealer_all(:,:,loop_ctr) = path;           %save path
actual_position(loop_ctr,:)       = current_pos;    %save actual position
P_save{loop_ctr}                  = P;              %save est covariance
loop_time_save(loop_ctr) = cputime-start_loop_time; %save loop process time
start_loop_time                   = cputime;        %restart loop time
x_est(loop_ctr,1)                 = CONST.x_hat_tgt;%save current tgt est
z_est(loop_ctr,1)                 = CONST.z_hat_tgt;%save current tgt est
perch_loop_ctr                    = loop_ctr;%identify when main loop ended
loop_ctr                          = loop_ctr+1;     %increment loop ctr


%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Perch loop:  Check position, get measurments (only count those in FOV %%
%% limits).  Update tgt if valid measurements. Correct path if valid     %%
%% tgt update.  Keep loop going until 5 seconds after the system should  %%
%% have perched.                                                         %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

sprintf('Entered Perching Mode Loop')
while current_pos(1) < row_complete
```

```

%% Get current position and measurement locations
if sim==1   %simulate position from ground station [line_num x y z psi]
 current_pos=get_current_position_sim(floor((29+cputime ...
       -sim_clock-display_fudge)/shell.dt_path_planner),path);
   %12x5 [line_num x y z psi] (last 12 meas locations, not in order)
 meas_locations=get_meas_locations_sim(path,current_pos,line_meas_sim);
else        %obtain position from ground station [line_num x y z psi]
    current_pos=get_current_position(conTRUTH);
    meas_locations=get_meas_locations(conTRUTH);

    % AFRL vs ANT lab sign swap--temp fix  (number 3/3)
    current_pos(4)=-current_pos(4);

    if ANT==1
        %scale up the meas locations
        meas_locations=scale_pos_from_ANT(meas_locations);
        %scale up current position
        current_pos=scale_pos_from_ANT(current_pos);
    end
end

%% Check for valid measurments (discard those outside of true FOV)
meas_locations=discard_meas_outside_FOV ...
    (meas_locations,limits.beta_min,limits.beta_max);

%% Sort measurements that have not been incorporated (may be empty)
[meas slot_last_meas] = get_beta (meas_locations, slot_last_meas);

%% If there are new measurements, update target estimate and cov
%% matrix with Unscented Kalman Filter, then update the path
if ~isempty(meas)
    new_meas=length(meas(:,1));
    meas(:,3) = meas(:,3)  ...
        + noise (total_measurements+1:total_measurements + new_meas);
    total_measurements=total_measurements + new_meas;
    for i=1:new_meas %UKF
        [Xrel P]=UKF_Cartesian(meas(i,:),P);
        CONST.x_hat_tgt=Xrel(1)+meas(i,1);   %update tgt estimate
        CONST.z_hat_tgt=Xrel(2)+meas(i,2);
    end

    %%  Update the path
    %% (if past hold at approach point and more than 4 in from perch)

    %once past the hold, resume updating path
    if current_pos(1) > row2 && current_pos(1)<last_update_line

        %Update x:
        x_remaining=CONST.x_hat_tgt+limits.perch_offset_x-path ...
            (current_pos(1),1); %dist in x still to go
        %(don't go from actual position, else you'll correct errors
        %for the integrator and never allow it to zero out).
        path_x=CONST.x_hat_tgt+limits.perch_offset_x-dx_from_perch ...
            (dx_from_perch<x_remaining);

        %append to path
        rowPerch=current_pos(1)+length(path_x);
        path(current_pos(1)+1:rowPerch,1)=path_x;

        %hold that x for the rest of the flight
        path(rowPerch+1:end,1)=path(rowPerch,1);

        %Update z (move at fixed velocity to correct error)
        z_perch=CONST.z_hat_tgt+limits.perch_offset_z;
        if path(current_pos(1),3) <= z_perch %if cmd is low, move up
```

```
                append4_z=[path(current_pos(1),3) ...
                +dist_per_line_1in_per_sec:dist_per_line_1in_per_sec ...
                :z_perch, z_perch]';
            else %if current cmd is high, move down
                append4_z=[path(current_pos(1),3) ...
                    -dist_per_line_1in_per_sec ...
                    :-dist_per_line_1in_per_sec:z_perch, z_perch]';
            end

            %stop sending new paths at 4 in (should be out of FOV anyway)
            if current_pos(1)+append4_z > last_update_line  %freeze path
                append4_z=append4_z(1:last_update_line-current_pos(1));
            end

            row4z=current_pos(1)+length(append4_z);
            path(current_pos(1)+1:row4z,3)=append4_z;

            %should have frozen at 4in, so the if is redundant--hold until
            %2 sec after x reaches perch
            if row4z<rowPerch
                rows2sec=ceil(2/shell.dt_path_planner);
                path(row4z+1:rowPerch+rows2sec,3)=path(row4z,3);
            end

            % Drop 5 inches (.127m) at 1 in per sec, then 1m at 2in per sec
            append5_z=[path(row4z,3):-dist_per_line_1in_per_sec ...
                :path(row4z,3)-.127,  path(row4z,3)-.127:-2 ...
                *dist_per_line_1in_per_sec:path(row4z,3)-1]';
            row_complete=rowPerch+rows2sec+length(append5_z);
            path(rowPerch+rows2sec+1:row_complete,3)=append5_z;

            if row_complete < shell.landing_slot
                % Hold the last z position until landing mode
                path(row_complete+1:shell.landing_slot,3) ...
                    =path(row_complete,3);
            end

            % Recalc z from end of run to 1m hover (rest doesn't change)
            path(shell.landing_slot+1:row_1m,3)=linspace ...
                (path(shell.landing_slot,3),1,row_1m-shell.landing_slot);

            %make last update line happen 4 in from the perch
            x_perch_minus_4in = CONST.x_hat_tgt+limits.perch_offset_x-.1;
            slots_past4in     = find(path(:,1)>x_perch_minus_4in);
            last_update_line  = slots_past4in(1);



            % Send path
            if sim~=1
                if ANT==1
                    scaled_path=scale_path_to_ANT(path);
                    SendPathList(conCMDLST,scaled_path);
                else
                    SendPathList(conCMDLST,path);
                end
            end
        end

    end

    %% Save for post process
    data_for_dealer_all(:,:,loop_ctr) = path;
    actual_position(loop_ctr,:)       = current_pos;
    loop_time_save(loop_ctr)          = cputime-start_loop_time;
    start_loop_time                   = cputime;
    x_est(loop_ctr,1)                  = CONST.x_hat_tgt;
    z_est(loop_ctr,1)                  = CONST.z_hat_tgt;
    meas_save{loop_ctr}                = meas;
    P_save{loop_ctr}                   = P;

    loop_ctr                           = loop_ctr+1;

end  %end perch mode loop

x_hat_at_freeze = CONST.x_hat_tgt;
z_hat_at_freeze = CONST.z_hat_tgt;

sprintf('No more path updates being sent')

Final_Tgt_est_error_inches=convlength ...
    ([CONST.x_tgt-CONST.x_hat_tgt CONST.z_tgt-CONST.z_hat_tgt],'m','in')

save last_run   %save workspace

%% %%%%%%%%%%%%%%%%%%%%%%%%%
%% CLOSE THE CONNECTIONS  %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%
if sim~=1
    RequestConnectionClose(conTRUTH);

    % Close the connection and socket
    pnet(conCMDLST, 'close');
    pnet(sockconCMDLST, 'close');
    disp('TCP/IP Command Connection closed');

    % Close the connection and socket
    pnet(conTRUTH, 'close');
    pnet(sockconTRUTH, 'close');
    disp('TCP/IP TRUTH Connection closed');
end
```

B.2 Trajectory Planner *GPOPS* Interface

The optimal solver calling function is included to provide an example of a *GPOPS* interface that is set up to run recursively, for real-time control applications. It provides an example of how to trim and bootstrap a previous guess, and it highlights the different inputs required for use with *GPOPS* 2.4, 3.2, and 3.3. In particular, the order and size of the outputs change between different versions of the software, but this is not addressed in any of the current documentation. This can cause significant errors, particularly with analytic derivatives in relation to the cost, DAE, and event functions. These are provided with correct output examples for all cases.

Trajectory Planner GPOPS Interface Code.

```
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Trajectory planner                                                 %%
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This mfile sets up the optimal control problem for the GPOPS software
% Written By:  LtCol Steven Ross, AFIT/ENY 2010.
%
% Inputs are initial conditions (the projected position when solution is
% expected to become available), limits (room boundaries, etc.), the last
% solution (the "already flown" portion is cut off and the remainder is
% used as the guess), output is output.solution from GPOPS. If the solution
% does not converge, the old solution is sent back out (so the next next
% round will not use the new (bad) solution for the guess), and an error
% message is recorded and displayed. x_hat_tgt, z_hat_tgt is the current
% target location estimate

function current_solution = trajectory_planner (IC, limits, Last_Solution)

global CONST FASTMODE WHAT_TO_RUN

%% Setup, Define Final approach point
% check limits (if a really bad estimate has put it out of ceiling/floor
% limits, put it on the limit)
xf=CONST.x_hat_tgt+limits.x_approach_offset;
zf=min(max(CONST.z_hat_tgt+limits.z_approach_offset,limits.min_alt), ...
    limits.max_alt);

FIM0=pinv(IC.P0);          %Initial Fisher Information Matrix
straight_time=(xf-IC.x0)/limits.xdot_max;  %min possible time

%% Bounds on initial and terminal values of time
limits.time.min = [IC.t0 IC.t0+straight_time];           %[t0 min  tf min]
limits.time.max = [IC.t0 IC.t0+max(10,3*straight_time)]; %[t0 max  tf max]
% use suitable buffer of time, NLT 10 seconds if close to target

%% State Bounds
%x  (Using "wall" at approach point):
limits.state.min(1,:) = [IC.x0   IC.x0-2    xf-limits.slush];
limits.state.max(1,:) = [IC.x0      xf             xf       ];

% z
limits.state.min(2,:) ...textcolorcomment
    =[IC.z0 limits.min_alt  max(limits.min_alt,zf-limits.slush)];
limits.state.max(2,:) ...
    =[IC.z0 limits.max_alt  min(limits.max_alt,zf+limits.slush)];

% x_dot
limits.state.min(3,:) = [IC.xdot0 -limits.xdot_max 0];
limits.state.max(3,:) = [IC.xdot0  limits.xdot_max 0];

% z_dot
limits.state.min(4,:) = [IC.zdot0  -limits.zdot_max 0];
limits.state.max(4,:) = [IC.zdot0   limits.zdot_max 0];

%zeta1
limits.state.min(5,:) = [FIM0(1,1)   -100     0];
limits.state.max(5,:) = [FIM0(1,1)  10000 10000];

%zeta2
limits.state.min(6,:) = [FIM0(2,2)   -100      0];
limits.state.max(6,:) = [FIM0(2,2)  10000  10000];

```

```
%zeta3
limits.state.min(7,:) = [FIM0(1,2)  -10000  -10000];
limits.state.max(7,:) = [FIM0(1,2)   10000   10000];

%% Control Bounds
limits.control.min    = [-limits.xddot_max; -limits.zddot_max];
limits.control.max    = [ limits.xddot_max;  limits.zddot_max];

%Bounds on an unknown static parameter
limits.parameter.min  = [];
limits.parameter.max  = [];

%% Path Limits (maintain FOV)
limits.path.min       = limits.beta_min;
limits.path.max       = limits.beta_max;

%% Event Constraints (any positive num indicates final covariance is met)
limits.event.min      = [  0;    0];
limits.event.max      = [1e6;  1e6];



%% Initial Guess==>bootstrap if Last_Solution is provided
test = isfield(Last_Solution,'time');
if test   %see if the Last_Solution exists (won't if deleted, or 1st run)
    %crop out any parts of the guess that will have been flown past
    index=find(Last_Solution.time>IC.t0);  %get index of future slots
    if ~isempty(index) %if there are future points, use as guess
        guess.time         = [IC.t0; Last_Solution.time(index)];
        guess.state        = [IC.x0 IC.z0 IC.xdot0 IC.zdot0 FIM0(1,1) ...
            FIM0(2,2) FIM0(1,2); Last_Solution.state(index,:)];
        guess.control      = [0 0; Last_Solution.control(index,:)];
        guess.parameter    = [];
    else %May not be future points (i.e. end of path, final cov not met)
        guess.time         = [IC.t0;     IC.t0+1+straight_time];
        guess.state(:,1)   = [IC.x0;     xf];
        guess.state(:,2)   = [IC.z0;     zf];
        guess.state(:,3)   = [IC.xdot0;  0];
        guess.state(:,4)   = [IC.zdot0;  0];
        guess.state(:,5)   = [FIM0(1,1); 200];
        guess.state(:,6)   = [FIM0(2,2); 200];
        guess.state(:,7)   = [FIM0(1,2); 200];
        guess.control(:,1) = [0;         0];
        guess.control(:,2) = [0;         0];
        guess.parameter    = [            ];
    end
else
    guess.time             = [IC.t0;     IC.t0+straight_time];
    guess.state(:,1)       = [IC.x0;     xf];
    guess.state(:,2)       = [IC.z0;     zf];
    guess.state(:,3)       = [IC.xdot0;  0];
    guess.state(:,4)       = [IC.zdot0;  0];
    guess.state(:,5)       = [FIM0(1,1); 200];
    guess.state(:,6)       = [FIM0(2,2); 200];
    guess.state(:,7)       = [FIM0(1,2); 200];
    guess.control(:,1)     = [limits.xdot_max; -limits.xdot_max];
    guess.control(:,2)     = [0;         0];
    guess.parameter        = [            ];
end

% Setup part of the problem
setup.name              = mfilename;
setup.funcs.cost        = 'trajectory_planner_cost';
setup.funcs.dae         = 'trajectory_planner_dae';
setup.funcs.event       = 'trajectory_planner_event';
setup.funcs.link        = '';
setup.limits            = limits;
setup.guess             = guess;
setup.linkages          = [];
setup.direction         = 'increasing'; %of independent variable
setup.autoscale         = 'on';
setup.derivatives       = 'analytic';
setup.checkDerivatives  = 0;
setup.maxIterations     = 500;

if WHAT_TO_RUN==2   %Additional Options for GPOPS 3.2
    %required inputs:
    setup.mesh.grid='hp';       %'hp' / 'global'
    setup.mesh.nodesbottom=2;   %fewest number of nodes to use
    setup.mesh.on='yes';        %('yes' / 'no')
    setup.method='radau';       %'radau','gauss','lobatto'
    setup.solver='snopt';       %ipopt not working yet
    setup.limits.intervals=3;
    setup.limits.nodesperint=5;

    %Optional inputs:
    %setup.meshdisplay='yes';
    %setup.mesh.tolerance;         OPTIONAL (Default = 1e-3)
    %setup.mesh.iteration;         OPTIONAL (Default = 20)
    %setup.mesh.guess;             OPTIONAL (Default = 'yes')
    %setup.controlinterp;          OPTIONAL (Default = 'lagrange')
    %setup.mesh.nodestop;          OPTIONAL (Default = setup.nodesbottom+5)
    %setup.mesh.splitmult;         OPTIONAL (Default = 2)
    %setup.mesh.warm='yes';%       OPTIONAL (Default = 'no')

elseif WHAT_TO_RUN==3 %Additional Options to run GPOPS 3.3
    setup.mesh.on='yes';           %('yes' / 'no')
    setup.mesh.grid='hp';          %'local','hp','global'
    setup.mesh.tolerance=1e-3;    %OPTIONAL (Default = 1e-3)
    setup.mesh.iteration=2;
    setup.mesh.guess='yes';
    setup.controlinterp='lagrange';%'lagrange','linear','cubic','spline'
    setup.mesh.nodesbottom=2;     %fewest number of segment nodes
    setup.mesh.nodestop=12;       %OPTIONAL generally should be bottom + 10
    setup.method='radau';         %'radau','gauss','lobatto'
    setup.solver='snopt';         %ipopt not working yet
    setup.limits.intervals=3;
    setup.limits.nodesperint=5;
    setup.mesh.warm='no';         %OPTIONAL (Default = 'no')
    %setup.mesh.splitmult=2;      %OPTIONAL (Default = 2)
end

%Call main function
if FASTMODE ==1                 %Use my modified GPOPS (Ross_gpops)
    setup.fastmode=1;           %don't add this if using normal GPOPS
    output = Ross_gpops(setup);
else
    output = gpops(setup);      %use standard GPOPS
end

if  output.SNOPT_info == 1
    current_solution=output.solution;
else
    sprintf('*******DID NOT CONVERGE, FORWARDING PREVIOUS SOLUTION*******')
    current_solution=Last_Solution;
end
current_solution.SNOPT_info=output.SNOPT_info;
```

Trajectory Planner Cost Function.

```
function [Mayer,Lagrange,DerivMayer,DerivLagrange] ...
    =trajectory_planner_cost(solcost)

% This function works with the optimal path solver, and provides the cost
% and all of the partial derivatives when provided with the path

tf   = solcost.terminal.time;
U    = solcost.control;
% t0 = solcost.initial.time;
% X0 = solcost.initial.state;
% Xf = solcost.terminal.state;
% t  = solcost.time;
% X  = solcost.state;
% p  = solcost.parameter;
% iphase = solcost.phase

Mayer = tf;   % min final time
w     = .1;   % Slightly weight control to avoid singular arc
Lagrange = w*(U(:,1).^2+U(:,2).^2);

% Analytic Derivatives:


if nargout == 4 % Can be used for analytic derivatives an another option
  [N , m]=size(U);
  DerivMayer=[zeros(1,15) 1];%[dphi/dX(t0) dphi/dt_0 dphi/dX(tf) dphi/dt_f]
  dL_dX=zeros(N,7);
  dL_dU=2*w*U;
  dL_dt=zeros(N,1);
  DerivLagrange=[dL_dX dL_dU dL_dt];
else
  DerivMayer=[];
  DerivLagrange=[];
end
```

Trajectory Planner Differential Algebraic Equations Function.

```
function [output1 output2 output3]=trajectory_planner_dae(soldae)

% This mfile provides the differential algebraic equations for the
% trajectory planner, and provides all of the partial derivatives when
% provided with the path.  A path constraint is added to keep the UAV
% within camera FOV limits. Outputs are different based on which version of
% GPOPS is being run, and whether or not the analytic derivatives are being
% used.
%
% Output Formatting:
%
% gpops 2.4, auto derivs:
%   output1=[xdot path];  output2=[]; output3=[];
% gpops 2.4, analytic derivs:
%   output1=[xdot path];  output2=[deriv_dae]; output3=[];
% gpops 3.~, auto derivs:
%   output1=[xdot];  output2=[path]; output3=[];
% gpops 3.~, analytic derivs:
%   output1=[xdot];  output2=[path]; output3=[deriv_dae];

```

```
global CONST WHAT_TO_RUN

X   = soldae.state;
U   = soldae.control;
% p = soldae.parameter;
% t = soldae.time;
% iphase = soldae.phase

rx        = CONST.x_hat_tgt-X(:,1);                %relative x
rz        = CONST.z_hat_tgt-X(:,2);                %relative z
xdot      = X(:,3);                                 %velocity x
zdot      = X(:,4);                                 %velocity z
xddot     = U(:,1);                                 %acceleration x
zddot     = U(:,2);                                 %acceleration z
r2        = rx.^2+rz.^2;                            %range squared
zeta1_dot = 1/CONST.dt_meas/CONST.R * (rz./r2).^2; %Deriv of FIM elements
zeta2_dot = 1/CONST.dt_meas/CONST.R * (rx./r2).^2;
zeta3_dot = 1/CONST.dt_meas/CONST.R * -(rx.*rz)./(r2.^2);

Xdot      = [xdot zdot xddot zddot zeta1_dot zeta2_dot zeta3_dot];
path      =  atan2(rz,rx);
Xdot_path = [Xdot path];

%% Calculate analytic derivatives

if (WHAT_TO_RUN==1 && nargout==2)  || (WHAT_TO_RUN==2 && nargout==3) ...
        || (WHAT_TO_RUN==3 && nargout==3) %if analytic deriv's are used

    [N n]=size(X);

    DerivDAE=zeros((n+1)*N, 10);  %init.  dimensions:  N(n+c) x (n+m+q+1)
    %(N=nodes, n=states, m=controls, q=parameters, c=paths)

    %%f1:  dX_1/dt = xdot      f is the derivatives of the states
    %df1=[df1_dx df1_dz df1_dxdot df1_dzdot df1_dzeta1 df1_dzeta2 ...
    %     df1_dzeta3 df1_du1 df1_du2 df1_dt];

    df1_dxdot      = ones(N,1);  %Calculate the non-zero partials
    DerivDAE(1:N,3)= df1_dxdot;  %Update the elements that are non-zero


    %%f2: dX_2/dt = zdot
    %df2=[df2_dx df2_dz df2_dxdot df2_dzdot df2_dzeta1 df2_dzeta2 ...
    %     df2_dzeta3 df2_du1 df2_du2 df2_dt];

    df2_dzdot           = ones(N,1); %Calculate the non-zero partials
    DerivDAE(N+1:2*N,4) = df2_dzdot; %Update the elements that are non-zero


    %%f3: dX_3/dt=xddot
    %df3=[df3_dx df3_dz df3_dxdot df3_dzdot df3_dzeta1 df3_dzeta2 ...
    %     df3_dzeta3 df3_du1 df3_du2 df3_dt];

    df3_du1               = ones(N,1); %Calculate the non-zero partials
    DerivDAE(2*N+1:3*N,8) = df3_du1; %Update the elements that are non-zero

    %%f4: dX_4/dt=zddot
    % df4 = [df4_dx df4_dz df4_dxdot df4_dzdot df4_dzeta1 df4_dzeta2 ...
    %        df4_dzeta3 df4_du1 df4_du2 df4_dt];

    df4_du2              = ones(N,1); %Calculate the non-zero partials
    DerivDAE(3*N+1:4*N,9)= df4_du2;   %update the non-zero elements


    %%f5: dX_5/dt=1/dt_meas/R * rz^2/(rx^2+rz^2)^2
    % df5 = [df5_dx df5_dz df5_dxdot df5_dzdot df5_dzeta1 df5_dzeta2 ...
    %        df5_dzeta3 df5_du1 df5_du2 df5_dt];

    %Calculate the non-zero partials
    df5_dx        = 4/CONST.dt_meas/CONST.R*rz.^2.*rx./r2.^3;
    df5_dz        = -2/CONST.dt_meas/CONST.R*rz.*(rx+rz).*(rx-rz)./r2.^3;
    DerivDAE(4*N+1:5*N,1:2)=[df5_dx df5_dz];  %update the non-zero elements

    %%f6: dX_6/dt=1/dt_meas/R * rx^2/(rx^2+rz^2)^2
    % df6=[df6_dx df6_dz df6_dxdot df6_dzdot df6_dzeta1 df6_dzeta2 ...
    %      df6_dzeta3 df6_du1 df6_du2 df6_dt];

    %Calculate the non-zero partials
    df6_dx        = 2/CONST.dt_meas/CONST.R*rx.*(rx+rz).*(rx-rz)./r2.^3;
    df6_dz        = 4/CONST.dt_meas/CONST.R*rx.^2.*rz./r2.^3;
    DerivDAE(5*N+1:6*N,1:2)=[df6_dx df6_dz];  %update the non-zero elements


    %%f7: dX_7/dt=1/dt_meas/R * -rx*rz/(rx^2+rz^2)^2
    % df7 = [df7_dx df7_dz df7_dxdot df7_dzdot df7_dzeta1 df7_dzeta2 ...
    %        df7_dzeta3 df7_du1 df7_du2 df7_dt];

    %Calculate the non-zero partials
    df7_dx        = 1/CONST.dt_meas/CONST.R*rz.*(rz.^2-3*rx.^2)./r2.^3;
    df7_dz        = 1/CONST.dt_meas/CONST.R*rx.*(rx.^2-3*rz.^2)./r2.^3;
    DerivDAE(6*N+1:7*N,1:2)=[df7_dx df7_dz];  %update the non-zero elements


    % Path Constraint C1: atan2(rz,rx)
    % dc1 = [dc1_dx dc1_dz dc1_dxdot dc1_dzdot dc1/dzeta1 dc1/dzeta2 ...
    %        dc1/dzeta3 dc1_du1 dc1_du2 dc1_dt];

    %Calculate the non-zero partials
    dc1_dx                  = rz./r2;
    dc1_dz                  =-rx./r2;
    DerivDAE(7*N+1:8*N,1:2) = [dc1_dx dc1_dz];%update the non-zero elements
end

if WHAT_TO_RUN==1 %Format for GPOPS 2.4
    output1=Xdot_path;
    output3=[];
    if nargout==2
        output2=DerivDAE;
    else
        output2=[];
    end
elseif WHAT_TO_RUN==2 || WHAT_TO_RUN==3  %Format for GPOPS 3.2 & GPOPS 3.3
    output1=Xdot;
    output2=path;
    if nargout==3
        output3=DerivDAE;
    else
        output3=[];
    end
end
```

Trajectory Planner Event Function.

```
function [events Derivevents]=trajectory_planner_event(solevents)

% This function provides the evaluation of the event constraint
% (boundary condition on a combination of states), and the analytic
% partial derivatives about the constraint.  The event constraint used is
% positive when the required final covariance in the associated axis is
% expected to be met.

global CONST WHAT_TO_RUN

Xf   = solevents.terminal.state;
% t0 = solevents.intial.time;
% X0 = solevents.intial.state;
% tf = solevents.terminal.time;
% p  = solevents.parameter;
% iphase=solevents.phase;

zeta1_tf = Xf(5);
zeta2_tf = Xf(6);
zeta3_tf = Xf(7);

den1 = (zeta1_tf*zeta2_tf-zeta3_tf^2); %Calculate a common denominator once

event1 = CONST.Pxx_f * den1 - zeta2_tf;
event2 = CONST.Pzz_f * den1 - zeta1_tf;

events = [event1;  event2];

if nargout==2   %Calculate analytic partial derivatives

    %% NOTE:  The order of the derivatives has changed. The old order for
    %% GPOPS 2.~ is Derivevents=[dE/dX(t0) dE/dt0 dE/dX(tf) dE/dtf dE/dp]
    %% and is reflected in the body below. The order is changed at the
    %% bottom for GPOPS 3.~

    Derivevents=zeros(2,16);   %init.  size= (e, 2n+2+q)

    %%E1=Pxx_f(zeta1_f*zeta2_f-zeta3_f^2)-zeta2_f
    % dE1=[dE1_dx0 dE1_dz0 dE1_dxdot0 dE1_dzdot0 dE1_dzeta1_0 ...
    %      dE1_dzeta2_0  dE1_dzeta3_0 dE1_dt0 dE1_dxf dE1_dzf ...
    %      dE1_dxdotf dE1_dzdotf dE1_dzeta1_f dE1_dzeta2_f dE1_dzeta3_f ...
    %      dE1_dtf dE1_dp];

    %Calculate non-zero partial derivatives
    dE1_dzeta1_f=CONST.Pxx_f*zeta2_tf;
    dE1_dzeta2_f=CONST.Pxx_f*zeta1_tf-1;
    dE1_dzeta3_f=-2*CONST.Pxx_f*zeta3_tf;

    %%E2=Pzz_f(zeta1_f*zeta2_f-zeta3_f^2)-zeta1_f
    % dE2=[dE2_dx0 dE2_dz0 dE2_dxdot0 dE2_dzdot0 dE2_dzeta1_0 ...
    %      dE2_dzeta2_0 dE2_dzeta3_0 dE2_dt0 dE2_dxf dE2_dzf ...
    %      dE2_dxdotf dE2_dzdotf dE2_dzeta1_f dE2_dzeta2_f ...
    %      dE2_dzeta3_f dE2_dtf dE2_dp];

    %Calculate non-zero partial derivatives
    dE2_dzeta1_f=CONST.Pzz_f*zeta2_tf-1;
    dE2_dzeta2_f=CONST.Pzz_f*zeta1_tf;
    dE2_dzeta3_f=-2*CONST.Pzz_f*zeta3_tf;

    %update non-zero elements  (note--the order of the derivatives has
    %changed between GPOPS 2.~ series and GPOPS 3.~ series).
    if WHAT_TO_RUN==1   %Format for GPOPS 2.~
        Derivevents(:,13:15)=[dE1_dzeta1_f dE1_dzeta2_f dE1_dzeta3_f;...
                              dE2_dzeta1_f dE2_dzeta2_f dE2_dzeta3_f];
    elseif WHAT_TO_RUN==2 || WHAT_TO_RUN==3 %Format for GPOPS 3.~
        Derivevents(:,10)=[dE1_dzeta1_f; dE2_dzeta1_f];
        Derivevents(:,12)=[dE1_dzeta2_f; dE2_dzeta2_f];
        Derivevents(:,14)=[dE1_dzeta3_f; dE2_dzeta3_f];
    end
else
```

```
    Derivevents=[];
end
```

Bibliography

[1] Aidala, Vincent and Sherry Hammel. “Utilization of Modified Polar Coordinates for Bearings-Only Tracking”. *IEEE Transactions on Automatic Control*, volume AC-28, 283–294. March 1983.

[2] Athans, Michael. “Editorial On the LQG Problem”. *IEEE Transactions on Automatic Control*, AC-16(6):528, 1971.

[3] Bellman, Richard. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

[4] Benson, David. *A Gauss Pseudospectral Transcription for Optimal Control.* Ph.D. Dissertation, Massachusetts Institute of Technology, 2005.

[5] Benson, David, Geoffrey T. Huntington, Tom P. Thorvaldsen, and Anil V. Rao. “Direct Trajectory Optimization and Costate Estimation via an Orthogonal Collocation Method”. *Journal of Guidance, Control, and Dynamics*, 29(6):1435–1440, 2006.

[6] Betts, J. T. and W. P. Huffman. “A Sparse Nonlinear Optimization Algorithm”. *Journal of Optimization Theory and Applications*, 82:519–541, 1994.

[7] Betts, John T. “Survey of Numerical Methods for Trajectory Optimization”. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, 1998.

[8] Betts, John T. *Practical Methods for Optimal Control Using Nonlinear Programming.* Advances in Design and Control. SIAM, Philadelphia, PA, 2001.

[9] Bishop, Adrian N. and Pubudu N. Pathirana. “Optimal Trajectory Characterization for a Pursuer Navigation Scheme”. *IEEE International Conference on Networking, Sensing and Control*, 998–1003. Sanya, China, 6-8 April 2008.

[10] Bollino, Kevin and I. Michael Ross. “A Pseudospectral Feedback Method for Real-Time Optimal Guidance of Reentry Vehicles”. *Proceedings of the 2007 American Control Conference*, 3861–3867. New York City, NY, 11-13 July 2007.

[11] Brown, Robert G. and Patrick Y. C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering.* John Wiley & Sons, New York, NY, 3rd edition, 1997.

[12] Bryson, Jr., Arthur E. “Optimal Control-1950 to 1985”. *IEEE Control Systems*, 16(3):26–32, 1996.

[13] Bryson, Jr., Arthur E. and Yu Chi Ho. *Applied Optimal Control.* Hemisphere Publishing Corp., Washington D.C., 1975.

[14] Burl, Jeffery B. *Linear Optimal Control.* Addison Wesley Longman, Inc., Menlo Park, CA, 1999.

[15] Byrd, R. H., J. Nocedal, and R. A. Waltz. "KNITRO: An Integrated Package for Nonlinear Optimization". *Large Scale Nonlinear Optimization*, 35–39, 2006.

[16] Canuto, C., M. Y. Hussaini, A. Quarteroni, and T. A. Zang. *Spectral Methods in Fluid Dynamics.* Springer-Verlag, New York, NY, 1988.

[17] Chavanne, Bettina H. "Small UAVs May Recharge on Power Lines". Military.com, 13 Dec 2007. URL `http://www.military.com/features/0,15240,158240,00.html`.

[18] Chen, Jyun-Jye F. and Bruce A. Conway. "Neighboring Optimal Feedback Control Using Collocation and Nonlinear Programming". *AIAA/AAS Astrodynamics Conference*, 512–520. Hilton Head Island, SC, 12 August 1992.

[19] Cory, Rick and Russ Tedrake. "Experiments in Fixed-Wing UAV Perching". *Proceedings of the 2008 AIAA Guidance, Navigation, and Control Conference.* Honolulu, HI, 18-21 August 2008.

[20] Cuthrell, J. E. and L. T. Biegler. "Simultaneous Optimization and Solution Methods for Batch Reactor Control Profiles". *Computers and Chemical Engineering*, 13(1/2):49–62, 1989.

[21] Darby, Christopher L., William W. Hager, and Anil V. Rao. "An hp-Adaptive Pseudospectral Method for Solving Optimal Control Problems". *Optimal Control Applications and Methods*, 32, August, 2010.

[22] Dickmanns, E. D. and H. Well. "Approximate solution of Optimal Control Problems Using Third-Order Hermite Polynomial Functions". *Proceedings of the 6th Technical Conference on Optimization Techniques.* Springer-Verlag, 1975.

[23] Dobrokhodov, Vladimir N., Isaac I. Kaminer, and Kevin D. Jones. "Vision-Based Tracking and Motion Estimation for Moving Targets Using Unmanned Air Vehicles". *Journal of Guidance, Control, and Dynamics*, 31(4):907–917, July-August, 2008.

[24] Eele, Alison and Arthur Richards. "Comparison of Branching Strategies for Path-Planning with Avoidance using Nonlinear Branch-and-Bound". *AIAA Guidance, Navigation and Control Conference and Exhibit.* Honolulu, Hawaii, 18-21 August 2008.

[25] Elnagar, Gamal, Mohammad Kazemi, and Mohsen Razzaghi. "Pseudospectral Legendre Method for Discretizing Optimal Control Problems". *IEEE Transactions on Automatic Control*, 40(10):1793–1796, 1995.

[26] Euler, L. *Institutiones Calculi Integralis*. Academy of Sciences, St. Petersburg, Russia, 1768.

[27] Fahroo, Fariba and I. Michael Ross. "Pseudospectral Methods for Infinite-Horizon Optimal Control Problems". *Journal of Guidance, Control, and Dynamics*, 31(4):927–936, 2008.

[28] Farrokh, Arsolan and Vikram Krishnamurthy. "Optimal Threshold Policies for Hard-Kill of Enemy Radars With High Speed Anti-Radiation Missiles (HARMS)". *ICASSP Proceedings of the IEEE International Conference on on Acoustics, Speech, and Signal Processing*, volume 3. IEEE, Toulouse, France, 14-19 May 2006.

[29] Fawcett, J. A. "Effect of Course Maneuver on Bearings-Only Range Estimation". *IEEE Transactions on Acoustic, Speech, and Signal Processing*, 36(8), August 1988.

[30] Feldbaum, A. A. "Dual Control Theory I-IV". *Automation and Remote Control*, 21, 22:874–880, 1033–1039, 1–12, 109–121, 1960-61.

[31] Flatley, Joseph L. "Cyber Technology's UAV Perches, Stares, Makes Us a Little Uncomfortable". Engadget.com, 17 December 2009. URL `http://www.engadget.com/2009/12/17/cyber-technologys-uav-perches-stares-makes-us-a-little-uncomf/`.

[32] Fornberg, Bengt. *A Practical Guide to Pseudospectral Methods*. Cambridge University Press, New York, NY, 1998.

[33] Frew, Eric W. *Observer Trajectory Generation for Target-Motion Estimation Using Monocular Vision*. Ph.D. Dissertation, Stanford University, August 2003.

[34] Garg, D., M. A. Patterson, C. L. Darby, F. Francolin, G. T. Huntington, W. W. Hager, and A. V. Rao. "Direct Trajectory Optimization and Costate Estimation of Finite-Horizon and Infinite-Horizon Optimal Control Problems Using a Radau Pseudospectral Method". *Computational Optimization and Applications*, 6 October 2009.

[35] Garg, Divya, Michael Patterson, Anil V. Rao, William W. Hager, David A. Benson, and Geoffrey T. Huntington. "A Unified Framework for the Numerical Solution of Optimal Control Problems Using Pseudospectral Methods". *Automatica*, 1–9, 2010.

[36] Gavish, Motti and Anthony Weiss. "Performance Analysis of Bearing-Only Target Location Algorithms". *IEEE Transactions of Aerospace and Electronic Systems*, 28(3):817–828, 1992.

[37] Geiger, Brian R. and Joseph F. Horn. "Neural Network Based Trajectory Optimization for Unmanned Aerial Vehicles". *47th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition.* American Institute of Aeronautics and Astronautics, Orlando, FL, 5-8 January 2009.

[38] Geiger, Brian R., Joseph F. Horn, Gregory L. Sinsley, James A. Ross, Lyle N. Long, and Albert F. Niessner. "Flight Testing a Real-Time Direct Collocation Path Planner". *Journal of Guidance, Control, and Dynamics*, 31(6):1575–1586, 2008.

[39] Gill, Philip E., Walter Murray, and Michael A. Saunders. "SNOPT: An SPQ Algorithm for Large-Scale Constrained Optimization". *SIAM Journal on Optimization*, 12(4):979–1006, 2002.

[40] Goldstine, H. *A History of the Calculus of Variations from the 17th Century through the 19th Century.* Springer-Verlag, 1980.

[41] Gong, Qi, Wei Kang, S. Bedrossian, Nazareth, Fariba Fahroo, Pooya Sekhavat, and Kevin Bollino. "Pseudospectral Optimal Control for Military and Industrial Applications". *Proceedings of the 47th IEEE Conference on Decision and Control*, 4128–4142. New Orleans, LA, December 12-14 2007.

[42] Goodman, G. and J. Knowles. "Joint DIRCM Program Nearing RFP Stage". *Journal of Electronic Defense*, 32(10):18–20, 2009.

[43] Gutierrez, Gustavo. Personal interview. Commander of Ohio class submarine USS Pennsylvania (SSBN-735), 5 January 2010.

[44] Hammel, S. E., P. T. Liu, E. J. Hilliard, and K. F. Gong. "Optimal Observer Motion for Localization with Bearing Measurements". *Computers and Mathematics with Applications*, 18(1-3):171–186, 1989.

[45] Hammel, Sherry and Vincent J. Aidala. "Observability Requirements for Three-Dimensional Tracking via Angle Measurements". *IEEE Transactions on Aerospace and Electronic Systems*, volume AES-21, 200–207. March 1985.

[46] Hammel, Sherry, Vincent J. Aidala, Kai F. Gong, and Allen G. Lindgren. "Recursive versus Batch Processing Algorithms for Bearings-Only Tracking". *Oceans Conference Record*, 50–61, 1983.

[47] Hammel, Sherry E. *Optimal Observer Motion for Bearings-Only Localization and Tracking.* Ph.D. Dissertation, University of Rhode Island, 1988.

[48] Hargraves, C. R. and S. W. Paris. "Direct Trajectory Optimization Using Nonlinear Programming and Collocation". *Journal of Guidance, Control, and Dynamics*, l0(4):338–342, 1987.

[49] Helferty, James P. and David R. Mudgett. "Optimal Observer Trajectories for Bearings-only Tracking by Minimizing the Trace of the Cramer-Rao Lower Bound". *Proceedings of the IEEE 32nd Conference on Decision and Control*. San Antonio, TX, December 1993.

[50] Helferty, James P., David R. Mudgett, and John E. Dzielski. "Trajectory Optimization for Minimum Range Error in Bearings-Only Source Localization". *Proceedings of the Conference on Oceans '93*, volume 2, 229–234. 12-18 Oct 1993.

[51] Ho, K. C. and Y. T. Chan. "An Unbiased Estimator for Bearings-Only Tracking and Doppler-Bearing Tracking". *IEEE Proceedings ICASSP '03*, 169–172. Hong Kong, China, 3 April 2003.

[52] Hodgson, Jeremy A. "Trajectory Optimization Using Differential Inclusion to Minimize Uncertainty in Target Location Estimation". *AIAA Guidance, Navigation, and Control Conference and Exhibit*. San Francisco, CA, 15-18 August 2005.

[53] Hull, D., J. Speyer, and D. Burris. "Linear-Quadratic Guidance Law for Dual Control of Homing Missiles". *AIAA Journal of Guidance, Control, and Dynamics*, 13(1), 1990.

[54] Humbert, Sean J., Richard M. Murray, and H. Dickinson, Michael. "Pitch-Altitude Control and Terrain Following Based on Bio-Inspired Visuomotor Convergence". *AIAA Guidance, Navigation, and Control Conference and Exhibit*. San Francisco, CA, 15-18 August 2005.

[55] Hurni, Michael A., Pooya Sekhavat, and I. Michael Ross. "Autonomous Trajectory Planning Using Real-Time Information Updates". *26th AIAA Applied Aerodynamics Conference*. Honolulu, HI, 18-21 August 2008.

[56] Jain, S. and P. Tsiotras. "Trajectory Optimization Using Multiresolution Techniques". *Journal of Guidance, Control, and Dynamics*, 31(5):1424–1436, September-October 2008.

[57] Johnson, Eric N., Anthony J. Calise, Yoko Watanabe, Jincheol Ha, and James C. Neidhoefer. "Real-Time Vision-Based Relative Aircraft Navigation". *Journal of Aerospace Computing, Information, and Communication*, 4(4):707–738, 2004.

[58] Jorris, Timothy R. *Common Aero Vehicle Autonomous Reentry Trajectory Optimization Satisfying Waypoint and No-Fly Zone Constraints*. Ph.D. Dissertation, Air Force Institute of Technology, 2007.

[59] Julier, Simon J. and Jeffrey K. Uhlmann. "Unscented Filtering and Nonlinear Estimation". *Proceedings of the IEEE*, 92(3):401–422, 2004.

[60] Kalman, Rudolph E. *Contributions to the Theory of Optimal Control.* Boletin de la Sociedad Mathematica Mexicana, 1960.

[61] Kalmár-Nagy, Tamás, Raffaello D'Andrea, and Pritam Ganguly. "Near-Optimal Dynamic Trajectory Generation and Control of an Omnidirectional Vehicle". *Robotics & Autonomous Systems*, 46(1):47–65, January 2004.

[62] Karelahti, Janne and Kai Virtanen. "Automated Generation of Realistic Near-Optimal Aircraft Trajectories". *Journal of Guidance, Control, and Dynamics*, 31(3):674, 2008.

[63] Karlsson, Rickard and Fredrik Gustafsson. "Range Estimation Using Angle-Only Target Tracking with Particle Filters". *Proceedings of the American Control Conference*, volume 5, 3743–3748. Arlington, VA, 25-27 June 2001.

[64] Kim, Jinwhan and Stephen Rock. "Stochastic Feedback Controller Design Considering the Dual Effect". *AIAA Guidance, Navigation, and Control Conference.* 2006.

[65] Kincaid, David and Ward Cheney. *Numerical Analysis: Mathematics of Scientific Computing.* American Mathematical Society, Providence, RI, third edition, 2002.

[66] Kirk, Donald E. *Optimal Control Theory, An Introduction.* Dover, Mineola, NY, 2004.

[67] Le Cadre, J. P. "Optimization of the Observer Motion for Bearings-Only Target Motion Analysis". *Proceedings of the 36th Conference on Decision and Control*, volume 4, 3126–3131. San Diego, CA, December 1997.

[68] Le Cadre, J. P. and O. Tremois. "Bearings-Only Tracking for Maneuvering Sources". *IEEE Transactions on Aerospace and Electronic Systems*, 34(1):179–193, 1998.

[69] Lerro, Don and Yaakov Bar-Shalom. "Tracking with Debiased Consistent Converted Measurements Versus EKF". *IEEE Transactions on Aerospace and Electronic Systems*, 29(3):1015–1022, 1993.

[70] Leung, Cindy, Shoudong Huang, and Gamini Dissanayake. "Trajectory Planning for Multilple Robots in Bearing-Only Target Localisation". *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2312–2317. Edmonton, AB, Canada, 2-6 August 2005.

[71] Lewis, L. R., I. Michael Ross, and Qi Gong. "Pseudospectral Computational Methods for Motion Planning and Obstacle Avoidance". *Proceedings of the 47th IEEE Conference on Decision and Control.* New Orleans, LA, 12-14 December 2007.

[72] Lindgren, Allen G. and Kai F. Gong. "Position and Velocity Estimation via Bearing Observations". *IEEE Transactions on Aerospace and Electronic Systems*, AES-14(4):564–577, 1978.

[73] Liu, P. T. "An Optimum Approach in Target Tracking with Bearing Measurements". *Journal of Optimization Theory and Applications*, 56(2):205–214, 1988.

[74] Logothetis, A., A. Isaksson, and R. J. Evans. "An Information Theoretic Approach to Observer Path Design for Bearings-Only Tracking". *Proceedings of the 36th IEEE Conference on Decision and Control*, 4:3132–3137, 1997.

[75] Luenberger, David G. *Optimization by Vector Space Methods.* John Wiley & Sons, New York, NY, 1969.

[76] MacCabe, B. J. "Accuracy and Tactical Implications of Bearings-Only Ranging Algorithms". *Operation Research*, 33(1):95–109, 1985.

[77] Maybeck, Peter S. *Stochastic Models, Estimation, and Control*, volume 2. Academic Press, New York, NY, 1982.

[78] Maybeck, Peter S. *Stochastic Models, Estimation, and Control*, volume 1. Navtech Book and Software Store, Arlington, VA, 1994.

[79] Metthies, L. and T. Kanade. "Kalman Filter-based Algorithms for Estimating Depth from Image Sequences". *International Journal of Computer Vision*, 3:209–236, 1989.

[80] Milam, Mark B., Kudah Mushambi, and Richard M. Murray. "A New Computational Approach to Real-Time Trajectory Generation for Constrained Mechanical Systems". *39th IEEE Conference on Decision and Control*, 845–851. Sydney, NSW, Australia, 12-15 December 2000.

[81] Misra, Pratap and Per Enge. *Global Positioning System: Signals, Measurements, and Performance.* Ganga-Jamuna Press, Lincoln, MA, 2001.

[82] Nardone, Steven C. and Vincent J. Aidala. "Observability Criteria for Bearings-Only Target Motion Analysis". *IEEE Transactions on Aerospace and Electronic Systems*, AES-17(2):162–166, March 1981.

[83] Nelson, Robert C. *Flight Stability and Automatic Control.* McGraw-Hill Companies, Inc., Boston, MA, 2nd edition, 1998.

[84] Oshman, Yaakov and Pavel Davidson. "Optimal Observer Trajectories for Passive Target Localization Using Bearing-Only Measurements". *AIAA Guidance, Navigation, and Control Conference.* San Diego, CA, 29-31 July 1996.

[85] Oshman, Yaakov and Pavel Davidson. “Optimization of Observer Trajectories for Bearings-Only Target Localization”. *IEEE Transactions on Aerospace and Electronic Systems*, 35(3):892–902, July 1999.

[86] Paris, S. W. “Enhanced Procedures for Direct Trajectory Optimization Using Nonlinear Programming and Implicit Integration”. *Collection of Technical Papers - AIAA/AAS Astrodynamics Specialist Conference*, volume 2. 21-24 August 2006.

[87] Passerieux, J. M. and D. Van Cappel. “Optimal Observer Maneuver for Bearings-Only Tracking”. *IEEE Transactions on Aerospace and Electronic Systems*, 34(3):777–7888, 1998.

[88] Ponda, Sameera S., Richard M. Kolacinski, and Emilio Frazzoli. “Trajectory Optimization for Target Localization Using Small Unmanned Aerial Vehicles”. *AIAA Guidance, Navigation, and Control Conference*. Chicago, IL, 10-13 August 2009.

[89] Pontryagin, L. S., V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mischenko. *The Mathematical Theory of Optimal Processes*. Interscience, New York, NY, 1962.

[90] Rao, A. V., D. A. Benson, C. L. Darby, M. A. Patterson, C. Francolin, and G. T. Huntington. “Algorithm 902: GPOPS, A MATLAB$^{®}$ Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method”. *ACM Transactions on Mathematical Software*, 37(2):1–39, April-June 2010.

[91] Rao, Anil V. “Extension of a Pseudospectral Legendre Method to Non-Sequential Multiple-Phase Optimal Control Problems”. *AIAA Guidance, Navigation, and Control Conference and Exhibit*. Austin, TX, Aug.11-14 2003.

[92] Rao, Anil V., David Benson, Christopher L. Darby, Camila Francolin, Michael Patterson, Ilyssa Sanders, and Geoffrey T. Huntington. “User’s Manual for GPOPS Version 2.3: A MATLAB$^{®}$ Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method”. August, 2009.

[93] Rao, Anil V., Sean Tang, and Wayne P. Hallman. “Numerical Optimization Study of Multiple-Pass Aeroassisted Orbital Transfer”. *Optimal Control Aplications and Methods*, 23:215–238, 2002.

[94] Reddien, G. W. “Collocation at Gauss Points as a Discretization in Optimal Control”. *SIAM Journal on Control and Optimization*, 17(2), March 1979.

[95] Ross, I. Michael. “Pseudospectral Feedback Control: Foundations, Examples and Experimental Results”. *AIAA Guidance, Navigation, and Control Conference*, volume 4. Keystone, CO, 21-24 August 2006.

[96] Ross, I. Michael and Fariba Fahroo. “A Direct Method for Solving Nonsmooth Optimal Control Problems”. *Proceedings of the 15th IFAC World Congress*, volume 15. Barcelona, Spain, 2002.

[97] Ross, James A., Brian R. Geiger, Gregory L. Sinsley, Joseph F. Horn, Lyle N. Long, and Albert F. Niessner. “Vision-Based Target Geolocation and Optimal Surveillance on an Unmanned Aerial Vehicle”. *AIAA Guidance, Navigation and Control Conference and Exhibit*. Honolulu, HI, 18-21 August 2008.

[98] Royce, Robert. Personal Interview. Powerline Urban Sentry (PLUS) Program Control, Defense Research Associates, Inc., 9 February and 13 April 2010.

[99] Rysdyk, R. “UAV Path Following for Constant Line-of-Sight”. *2nd AIAA Unmanned Unlimited Conference, Workshop, and Exhibit*. San Diego, CA, 15-18 September 2003.

[100] Sekhavat, Pooya, Andrew Fleming, and I. Michael Ross. “Time-Optimal Nonlinear Feedback Control for the NPSAT1 Spacecraft”. *Proceddings of the 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 843–850. Monterey, CA, 24-28 July 2005.

[101] Seywald, Hans, Renjith R. Kumar, and Todd A. Wetzel. “Does the Pinciple of Optimality Hold Along Collocation Solutions?” *AIAA Guidance, Navigation and Control Conference*. San Diego, CA, 29-31 July 1996.

[102] Skoglar, Per, Umut Orguner, and Fredrik Gustafsson. “On Information Measures based on Particle Mixture for Optimal Bearings-only Tracking”. *IEEE Aerospace Conference Proceedings*, 1–13, 2009.

[103] Speyer, J., D. Hull, V. Tseng, and S. Larson. “Estimation Enhancement by Trajectory Modulation for Homing Missiles”. *AIAA Journal of Guidance, Control, and Dynamics*, 7(3), 1984.

[104] Spiess, F. N. “Complete Solution of the Bearings Only Approach Problem”. *UC San Diego: Scripps Institution of Oceanography, MPL Technical Memorandum*. 15 December 1953.

[105] Spingarn, Karl. “Passive Position Location Estimation Using the Extended Kalman Filter”. *IEEE Transactions on Aerospace and Electronic Systems*, volume AES-23, 558–567. July 1987.

[106] Strizzi, Jon, I. Michael Ross, and Fariba Fahroo. “Towards Real-Time Computation of Optimal Controls for Nonlinear Systems”. *AIAA Guidance, Navigation, and Control Conference and Exhibit*. Monterey, CA, 5-8 Aug 2002.

[107] von Stryk, Oskar. “Numerical Solution of Optimal Control Problems by Direct Collocation”. *Optimal Control Theory and Numerical Methods*, 111:129–143, 1993.

[108] Taylor, James H. "The Cramér-Rao Estimation Error Lower Bound Computation for Deterministic Nonlinear Systems". *IEEE Transactions on Automatic Control*, AC-24(2):343–344, 1979.

[109] The Small Unmanned Aerial Systems News Source. "sUAS News", March 2010. URL `http://www.suasnews.com/`.

[110] Tsuchiya, Takeshi, Masahiro Miwa, and Shinji Suzuki. "Real-Time Flight Trajectory Optimization and Its Verification in Flight". *Journal of Aircraft*, 46(4):1468–1470, July-August 2009.

[111] Van Trees, H. L. *Detection, Estimation, and Modulation Theory, Part I.* Wiley, New York, 1968.

[112] Veth, Michael J. *Fusion of Imaging and Inertial Sensors for Navigation.* Ph.D. Dissertation, Air Force Institute of Technology, 2006.

[113] Watanabe, Yoko, Eric N. Johnson, and Anthony J. Calise. "Vision-Based Guidance Design from Sensor Trajectory Optimization". *AIAA Guidance, Navigation, and Control Exhibit.* Keystone, CO, 21-24 Auguest 2006.

[114] Watanabe, Yoko, Eric N. Johnson, and Anthony J. Calise. "Stochastically Optimized Monocular Vision-Based Guidance Design". *AIAA Guidance, Navigation and Control Conference and Exhibit.* Hilton Head, South Carolina, 20-23 August 2007.

[115] Watanabe, Yoko, Eric N. Johnson, and Anthony J. Calise. "Stochastic Guidance Design for UAV Vision-Based Control Applications". *AIAA Guidance, Navigation, and Control Conference and Exhibit.* Honolulu, HI, 18-21 August 2008.

[116] Wickenheiser, Adam M. and Ephrahim Garcia. "Optimization of Perching Maneuvers Through Vehicle Morphing". *Journal of Guidance, Control, and Dynamics*, 31(4):815–823, August 2008.

[117] Williams, Paul. "Three-Dimensional Aircraft Terrain-Following via Real-Time Optimal Control". *Journal of Guidance, Control, and Dynamics*, 30(4):1201–1205, 2007.

[118] Wise, Richard and Rolf T. Rysdyk. "UAV Coordination for Autonomous Target Tracking". *AIAA Guidance, Navigation and Control Conference and Exhibit.* Keystone, CO, 21-24 August 2006.

[119] Yan, Hui. "Real-time Computation of Neighboring Optimal Control Laws". *AIAA Guidance, Navigation, and Control Conference and Exhibit.* Monterey, CA, 5-8 Aug 2002.

[120] Zadeh, L. A. "The Evolution of Systems Analysis and Control: A Personal Perspective". *IEEE control systems /IEEE Control Systems Society*, 16(3):95–98, June 1996.

Index

This index is conceptual and does not designate every occurrence of a keyword

Vita

Lieutenant Colonel Steven M. Ross was raised in Sacramento, California, and was a distinguished graduate of the United States Air Force Academy's class of 1996, receiving a Bachelor of Science degree in Engineering Sciences and a minor in Japanese Language Studies. Following his commission, then Lieutenant Ross was happily married and served as a sailplane instructor pilot before moving to Sheppard AFB, Texas for the Euro-NATO Joint Jet Pilot Training Program (ENJJPT). After completing Introduction to Fighter Fundamentals (IFF) at Columbus AFB, Missouri, and a transition course at Tyndall AFB, Florida, he served as a combat F-15C pilot based at Kadena AFB, Japan.

Lieutenant Colonel Ross returned to Sheppard AFB and was recognized as the outstanding instructor for two pilot training classes before entering the AFIT/Test Pilot School joint program, in Dayton, Ohio, and Edwards AFB, California, respectively. He completed both programs as a distinguished graduate, and his thesis work in automated aerial refueling included the first ever fully autonomous close formation flight, winning the AFIT Commandant's and Dean's Awards for best thesis and the Air Force Association's national Von Karman Award for his contribution to science.

Lieutenant Colonel Ross most recently served at Eglin AFB, Florida, as a test pilot for many programs on the F-15C and F-15E before returning to Dayton for his Ph.D. He is a senior pilot with over 1900 flying hours in over 45 civilian and military aircraft types, and will return to Edwards AFB as a Test Pilot School Instructor. Without comparison, his greatest blessings in life are his wife and five wonderful children who fill his days with joy and laughter.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)* 16–09–2011	**2. REPORT TYPE** Doctoral Dissertation	**3. DATES COVERED** *(From — To)* Sep 2008 — Sep 2011

4. TITLE AND SUBTITLE Stochastic Real-Time Optimal Control: A Pseudospectral Approach for Bearing-Only Trajectory Optimization	**5a. CONTRACT NUMBER** **5b. GRANT NUMBER** **5c. PROGRAM ELEMENT NUMBER**
6. AUTHOR(S) Ross, Steven M., LtCol, USAF	**5d. PROJECT NUMBER** 11Y270 **5e. TASK NUMBER** **5f. WORK UNIT NUMBER**
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/DS/ENY/11-24
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory, Air Vehicles Directorate Attn: Gregory H. Parker 24B 2145 Fifth Street WPAFB, OH 45433 (937)255-7550 Gregory.Parker@wpafb.af.mil	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RB **11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

12. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States

14. ABSTRACT

A method is presented to couple and solve the optimal control and the optimal estimation problems simultaneously, allowing systems with bearing-only sensors to maneuver to obtain observability for relative navigation without unnecessarily detracting from a primary mission. A fundamentally new approach to trajectory optimization and the dual control problem is developed, constraining polynomial approximations of the Fisher Information Matrix to provide an information gradient and allow prescription of the level of future estimation certainty required for mission accomplishment. Disturbances, modeling deficiencies, and corrupted measurements are addressed with recursive updating of the target estimate with an Unscented Kalman Filter and the optimal path with Radau pseudospectral collocation methods and sequential quadratic programming. The basic real-time optimal control (RTOC) structure is investigated, specifically addressing limitations of current techniques in this area that lose error integration. The resulting guidance method can be applied to any bearing-only system, such as submarines using passive sonar, anti-radiation missiles, or small UAVs seeking to land on power lines for energy harvesting. Methods and tools required for implementation are developed, including variable calculation timing and tip-tail blending for potential discontinuities. Validation is accomplished with simulation and flight test, autonomously landing a quadrotor helicopter on a wire.

15. SUBJECT TERMS

RTOC, Real-Time Optimal Control, Bearing-only, Trajectory Optimization, UAV, SUAS, Power Line, Pseudospectral, Quadrotor, Energy Harvesting

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Richard G. Cobb (ENY)
U	U	U	UU	247	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255-3636, x4559; richard.cobb@afit.edu