ACKNOWLEDGEMENTS

I wish to express my sincere thanks to my supervisor, Dr. A.H. Jones for his friendship, supervision and encouragement during the course of the research and for his careful scrutiny of the final draft of the thesis.

I also wish to express special thanks to my parents for their unfaltering support and for the many years of education they have given me.

Abstract

In the real world there are three types of multivariable control systems. The first one is when the number of inputs is equal to the number of the outputs, this type of multivariable control system is defined as a squared multivariable control system and the main type of controller designed is a decoupling controller which minimizes interactions and gives good set-point tracking. The second type of multivariable control system is where the number of inputs is greater than the number of the outputs, for this type of system the main controller designed is a fail-safe controller. This controller remains stable if a sub-set of actuator fail. The third type of multivariable control system is the number of outputs is greater than the number of inputs, for this type of system the main controller designed is an override control system. This controller only controls a sub-set of outputs based on a lowest wins control strategy. All the three types of multivariable control systems are included in this thesis.

In this thesis the design of multivariable decoupling control, multivariable fail-safe control and multivariable override control as considered. The invention of evolutionary computing techniques has changed the design philosophy for control system design. Rather than using conventional techniques such as Nyquest plots or root-loci control systems can be designed using evolutionally algorithm. Such algorithms evolve solutions using cost functions and optimization.

There are a variety of system performance indicators such as integral squared error operator has been used as cost functions to design controllers using such algorithms.

The design of both fail-safe and override multivariable controllers is a difficult problem and there are very few analytical design methods for such controllers. Therefore, the main objective of this thesis is to use the genetic algorithms to involve both fail-safe and override controller multivariable controllers, such that they perform well in the time-domain.

Contents

ACKNOWLEDGEMENTS 1
Abstract 2
LIST OF FIGURES 6
LIST OF TABLE 13
INTRODUCTION 14
1.1 General Review of multivariable control system design 14
1.2 Non-square multivariable control system 15
1.2.1 Fail-safe control system design 17
1.2.1.1 Actuator failure in multivariable control system 17
1.2.1.2 Override control system design 18
1.3 Optimisation of Multivariable control system 20
1.3.1 Single objective genetic algorithm design for multivariable control system 23
1.3.2 Multi-objective genetic algorithm design for multivariable control system 25
1.4 Aim and objective 27
1.5 Outline of the thesis 28
1.5.1 Chapter one: introduction 28
1.5.2 Chapter two: square multivariable control system design 28
1.5.3 Chapter three: fail-safe control system design 28
1.5.4 Chapter four: override control system design 28
1.5.5 Chapter five: conclusion 28
GENETIC DESIGN OF SQUARE MULTIVARIABLE CONTROL SYSTEMS 29
2.1 Introduction 29
2.2 Genetic algorithm 38
2.2.1 Introduction 38
2.2.2 Multi-Cost function Genetic Algorithm (Multi-objective Genetic Algorithm) 45
2.2.3 Multivariable control system design methods 49
2.3 Single objective genetic algorithm multivariable control system design 57
2.3.1 The asymptotically stable plant α =2 (Non-minimum phase) 57
2.3.1.1 Single cost function genetic algorithm top design the diagonal tuning matrix 59
2.3.1.2 Single cost function GA design the whole controller parameters 62
2.3.1.3 Single cost function GA design with weight factor 65
2.3.1.4 Single cost function GA design with weight factor less than one 65
2.3.1.5 Single cost function GA design with weight factor greater than one 68
2.3.1.6 Single cost function GA design with weight factor under constraint 70
2.3.1.7 Single cost function GA design with weight factor less than one under constraint 71
2.3.1.8 Single cost function GA design with weight factor equal to one under constraint 73
2.3.1.9 Single cost function GA design with weight factor greater than one under constraint 75
2.3.2 The asymptotically stable plant when α =3 (functionally uncontrollable) 80
2.3.3 The asymptotically stable plant when α =4 (minimum phase) 85

2.3.3.1 Single cost function genetic algorithm design the tuning diagonal matrix 86
2.3.3.2 Single cost function GA design of all parameters of the controller.....90
2.4 Multi objective GA 93
2.4.1.1 Multi-objective genetic algorithm design for the asymptotically stable plantα=2 (Non-minimum phase).....95
2.4.1.2 Multi-objective genetic algorithm design for the asymptotically stable plant whenα=4 (minimum phase) 101
2.5 Conclusion 107
GENETIC DESIGN OF MULTIVARIABLE FAIL SAFE CONTROL SYSTEM .109
3.1 Introduction.....109
3.1.1 Analysis.....111
3.1.1.1 The pseudo inverse design method:.....111
3.1.1.2 The Genetic Algorithm design method.....112
3.2 Genetic Algorithm design of single actuator failure fail-safe control system design 115
3.2.1 The Pseudo Inverse design method with Global Optimisation:.....117
3.2.2 The Genetic algorithm design method with single cost function full parameters Optimisation: 120
3.2.3 The genetic algorithm design method with Worst Case Failure Optimisation:.....123
3.3 Pareto front design of single actuator failure fail-safe control system design 127
3.3.1 The minimum of "ISE for non-actuator failure" channel: 128
3.3.2 The minimum of "ISE for actuator 2 failure" channel: 129
3.3.3 The minimum of "sum of all ISE" channel "ISE for actuator 1 failure" channel and "square root of sum of all squared ISE" channel:.....130
3.4 Genetic algorithm design of single actuator and multiple actuator failure fail-safe control system 132
3.4.1 The Pseudo inverse design method with full parameters Optimisation: 135
3.4.2 The full parameter Genetic algorithm design method with constraint: 141
3.4.3 The full parameter Genetic algorithm design method with constraint and Worst Case Failure Optimisation:.....145
3.5 Pareto front design of single and multiple actuator failure fail-safe control systems.....150
3.6 Conclusion 161
GENETIC DESIGN OF MULTIVARIABLE OVERRIDE CONTROL SYSTEM.163
4.1 Introduction.....163
4.1.1 Absolute input and incremental input 165
4.1.2 Bumpless transfer: 165
4.1.3 Genetic algorithm for override control system design.....166
4.1.4 Limit cycle in override control system 168
4.2 Genetic design of single input multi output override control systems..169
4.2.1 Limit Cycle override control system.....173
4.3 Multi-objective Genetic Algorithm search for the limit cycle boundary for single input multi outputs override control systems.....174
4.4 Multi-objective Genetic Algorithm design of single input multi outputs override control systems 179

4.5 Multi inputs multi outputs override control system 181
4.5.1 No limit cycle override control: 182
4.5.2 Limit cycle override control: 186
4.6 Genetic design of multi inputs multi outputs override control system . 187
4.7 Multi-objective Genetic algorithm search for the limit cycle boundary for multi inputs multi outputs override control systems 193
4.8 Multi-objective Genetic algorithm design of multi input multi output override control systems 197
4.9 Conclusion 209
CONCLUSION 211
5.1 Overview 211
5.2 Square multivariable control system 212
5.3 Fail-safe multivariable control system 214
5.4 Override multivariable control system 216
FURTHER WORK 220
REFERENCE 221

LIST OF FIGURES

Figure 1.1: Control architecture for six rotor helicopter system……..….15
Figure 1.2: Aircraft engine's automatic control system……...16
Figure 2.1: Closed loop step response resulted from applying reposed PID parameters……35
Figure 2.2: The basic cycle of genetic algorithm…….…39
Figure 2.3: Fitness calculated by the 1/cost technique against cost……...……..41
Figure 2.4: Fitness calculated by the Max-cost technique against cost……...….42
Figure 2.5: Cost against the number of generation……..43
Figure 2.6: Flow chart of Pareto front method…….……46
Figure 2.7: The concept of Pareto dominance…….……47
Figure 2.8: The concept of Pareto dominance…….……48
Figure 2.9: The decode and scale between binary number and real number……...52
Figure 2.10: The decode and scale between binary number and real number…….53
Figure 2.11: The ISE against the number of generation with different parameter range……55
Figure 2.12: Square multivariable control system transfer function……58
Figure 2.13: The output 1 of GA design π_1, π_2, ε_1 and ε_2 with the set-point changing and interaction……..60
Figure 2.14: The output 2 of GA design π_1, π_2, ε_1 and ε_2 with the set-point changing and interaction……..60
Figure 2.15: The output 1 of GA design k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3} and k_{i4} with the set-point changing and interaction……...62
Figure 2.16: The output 2 of GA design k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3} and k_{i4} with the set-point changing and interaction……...62
Figure 2.17: The output 1 of GA design k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3} and k_{i4} with the set-point changing and interaction……...65

Figure 2.18: The output 2 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction……...65

Figure 2.19: The output 1 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction……...67

Figure 2.20: The output 2 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction…….……..67

Figure 2.21: The output 1 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction…….…..70

Figure 2.22: The output 2 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction……..…….70

Figure 2.23: The output 1 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction……..……72

Figure 2.24: The output 2 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction……...72

Figure 2.25: The output 1 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction……...……73

Figure 2.26: The output 2 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction……..….73

Figure 2.27: the output 1 and 2 of GA design whole controller……..75

Figure 2.28: the output 1 and 2 of GA design whole controller with weight factor=10 and constraint on interaction……..…….…….75

Figure 2.29: the output 1 and 2 of GA design whole controller with weight factor=1 and constraint on interaction……...…….75

Figure 2.30: the output 1 and 2 of GA design whole controller with weight factor=0.1 and constraint on interaction……..……..75

Figure 2.31: the output 1 and 2 of GA design whole controller with weight factor=10……..……75

Figure 2.32: the output 1 and 2 of GA design whole controller with weight factor=0.1……...…….……75

Figure 2.33: Square multivariable control system transfer function……78

Figure 2.34: The output 1 of GA design with the set-point changing……..80
Figure 2.35: The output 2 of GA design with the set-point changing……..……80
Figure 2.36: The output 1 of GA design with the set-point changing……..80
Figure 2.37: The output 2 of GA design with the set-point changing……..80
Figure 2.38: Square multivariable control system transfer function……..……..82
Figure 2.39: the output 1 of GA design the diagonal matrix with the set-point changing……..….85
Figure 2.40: the output 2 of GA design the diagonal matrix with the set-point changing……...85
Figure 2.41: the output 1 of GA design the whole controller with the set-point changing……...87
Figure 2.42: the output 2 of GA design the whole controller with the set-point changing…….……..…87
Figure 2.43: the concept of Pareto dominance……..…….……89
Figure 2.44: Square multivariable control system transfer function…….…..90
Figure 2.45: the Pareto front plot for example in chapter 2.4……..92
Figure 2.46: Square multivariable control system transfer function……....93
Figure 2.47: the Pareto front plot for example in chapter 2.3……..94
Figure 3.1: Figure 3.1: Control architecture…….……97
Figure 3.2: Transfer function……102
Figure 3.3: The output with non-failure…….……105
Figure 3.4: The output with actuator 1 failure……...……...……105
Figure 3.5: The output with actuator 2 failure……..……...…..105
Figure 3.6: The output with non-failure…….……107
Figure 3.7: The output with actuator 1 failure……..……...…..107
Figure 3.8: The output with actuator 2 failure……...107
Figure 3.9: The output with non-failure……...……..109
Figure 3.10: The output with actuator 1 failure…….……109
Figure 3.11: The output with actuator 2 failure……..……..…109
Figure 3.12: The comparison between normal inverse design and single objective GA design…….……110
Figure 3.13: The comparison between normal inverse design and single objective GA design…….……110

Figure 3.14: The comparison between normal inverse design and single objective GA design……110
Figure 3.15: The comparison between normal inverse design and single objective GA design……110
Figure 3.16: The output with non-failure……114
Figure 3.17: The output with actuator 1 failure……114
Figure 3.18: The output with actuator 2 failure……114
Figure 3.19: The output with non-failure……115
Figure 3.20: The output with actuator 1 failure……115
Figure 3.21: The output with actuator 2 failure……115
Figure 3.22: The output with non-failure……116
Figure 3.23: The output with actuator 1 failure……116
Figure 3.24: The output with actuator 2 failure……116
Figure 3.25: Transfer function……119
Figure 3.26: The output with non-failure……122
Figure 3.27: The output with actuator 1 failure……122
Figure 3.28: The output with actuator 2 failure……122
Figure 3.29: The output with actuator 3 failure……122
Figure 3.30: The output with actuator 4 failure……122
Figure 3.31: The output with actuator 1 and 2 failures……122
Figure 3.32: The output with actuator 1 and 3 failures……122
Figure 3.33: The output with actuator 1 and 4 failures……122
Figure 3.34: The output with actuator 2 and 3 failures……122
Figure 3.35: The output with actuator 2 and 4 failures……122
Figure 3.36: The output with actuator 3 and 4 failures……122
Figure 3.37: The output with non-failure……126
Figure 3.38: The output with actuator 1 failure……126
Figure 3.39: The output with actuator 2 failure……126
Figure 3.40: The output with actuator 3 failure……126
Figure 3.41: The output with actuator 4 failure……126
Figure 3.42: The output with actuator 1 and 2 failures……126
Figure 3.43: The output with actuator 1 and 3 failures……126
Figure 3.44: The output with actuator 1 and 4 failures……126

Figure 3.45: The output with actuator 2 and 3 failures…….126
Figure 3.46: The output with actuator 2 and 4 failures…….126
Figure 3.47: The output with actuator 3 and 4 failures…….126
Figure 3.48: The output with non-failure…….130
Figure 3.49: The output with actuator 1 failure…….130
Figure 3.50: The output with actuator 2 failure…….130
Figure 3.51: The output with actuator 3 failure…….130
Figure 3.52: The output with actuator 4 failure…….130
Figure 3.53: The output with actuator 1 and 2 failures…….130
Figure 3.54: The output with actuator 1 and 3 failures…….130
Figure 3.55: The output with actuator 1 and 4 failures……..130
Figure 3.56: The output with actuator 2 and 3 failures……..130
Figure 3.57: The output with actuator 2 and 4 failures…….130
Figure 3.58: The output with actuator 3 and 4 failures……..130
Figure 3.59: The ISE comparison between Pseudo inverse design and GA design...132
Figure 3.60: The output with non-failure……..…….138
Figure 3.61: The output with actuator 1 failure…….138
Figure 3.62: The output with actuator 2 failure…….138
Figure 3.63: The output with actuator 3 failure…….138
Figure 3.64: The output with actuator 4 failure…….138
Figure 3.65: The output with actuator 1 and 2 failures……..138
Figure 3.66: The output with actuator 1 and 3 failures……..138
Figure 3.67: The output with actuator 1 and 4 failures……..138
Figure 3.68: The output with actuator 2 and 3 failures……..138
Figure 3.69: The output with actuator 2 and 4 failures……..138
Figure 3.70: The output with actuator 3 and 4 failures……..138
Figure 3.71: The output with non-failure……...140
Figure 3.72: The output with actuator 1 failure…….140
Figure 3.73: The output with actuator 2 failure…….140
Figure 3.74: The output with actuator 3 failure…….140
Figure 3.75: The output with actuator 4 failure…….140
Figure 3.76: The output with actuator 1 and 2 failures……..140
Figure 3.77: The output with actuator 1 and 3 failures……..140

Figure 3.78: The output with actuator 1 and 4 failures……..140
Figure 3.79: The output with actuator 2 and 3 failures……..140
Figure 3.80: The output with actuator 2 and 4 failures…….…….140
Figure 3.81: The output with actuator 3 and 4 failures…….…….140
Figure 3.82: The output with non-failure…….……..142
Figure 3.83: The output with actuator 1 failure…….……142
Figure 3.84: The output with actuator 2 failure…….142
Figure 3.85: The output with actuator 3 failure…….142
Figure 3.86: The output with actuator 4 failure…….142
Figure 3.87: The output with actuator 1 and 2 failures…….……142
Figure 3.88: The output with actuator 1 and 3 failures. ……....142
Figure 3.89: The output with actuator 1 and 4 failures……142
Figure 3.90: The output with actuator 2 and 3 failures……..142
Figure 3.91: The output with actuator 2 and 4 failures……..142
Figure 3.92: The output with actuator 3 and 4 failures……..142
Figure 4.1: Aircraft engine's automatic control system…….…….. ….145
Figure 4.2: Override control……...145
Figure 4.3: Foss's override control system……148
Figure 4.4: Foss's override control system without and with limit cycle when set point changing……149
Figure 4.5: On input two outputs override control system with negative sign in steady-state transfer function matrix ……150
Figure 4.6: system output of the one input two outputs override control system with negative sign in steady-state transfer function matrix……...134
Figure 4.7: System output of the one input two outputs override control system with negative sign in steady-state transfer function matrix……...134
Figure 4.8: One input two outputs override control system……..134
Figure 4.9: Set point for no limit cycle Pareto front plot……...134
Figure 4.10: System output of the one input two outputs override control system with negative sign in steady state transfer function matrix…….. ….134
Figure 4.11: System output of the one input two output override control system with negative sign in steady-state transfer function matrix…….……. ….134
Figure 4.12: One input two outputs override control system…….134

Figure 4.13: Pareto front plot for trade-off performance……...134
Figure 4.14: Multi-inputs multi-outputs override control system block diagram…..134
Figure 4.15: Multi-inputs multi-outputs override control system for output 1……..134
Figure 4.16: Multi-inputs multi-outputs override control system for output 2……..134
Figure 4.17: Multi-inputs multi-outputs override control system for output 3……..134
Figure 4.18: Multi-inputs multi-outputs override control system……..134
Figure 4.19: Multi-inputs multi-outputs override control system for output 1……..134
Figure 4.20: Multi-inputs multi-outputs override control system for output 2……..134
Figure 4.21: Multi-inputs multi-outputs override control system for output 3……..134
Figure 4.22: Multi-inputs multi-outputs override control system block diagram…..134
Figure 4.23: Output 1 for Multi-inputs multi-outputs override control system …....134
Figure 4.24: Output 2 for Multi-inputs multi-outputs override control system …....134
Figure 4.25: Output 3 for Multi-inputs multi-outputs override control system …....134
Figure 4.26: Multi-inputs multi-outputs override control system block diagram…..134
Figure 4.27: Multi-inputs multi-outputs override control system……..134
Figure 4.28: Multi-inputs multi-outputs override control system block diagram…..134
Figure 4.29: Multi-inputs multi-outputs override control system……..134
Figure 4.30: The output 1 for multi-inputs multi-outputs override control system...134

LIST OF TABLE

Table 2.1: The ISE comparison and overshoot……..134
Table 2.2: The ISE comparison and overshoot……..134
Table 3.1: The comparison between normal inverse design and single objective GA design…….134
Table 3.2: The Pareto front plot of signal actuator failure…….134
Table 3.3: The ISE for MIMO fail-safe control system under all situations……..134
Table 3.4: The ISE for MIMO fail-safe control system under all situations compared between Pseudo inverse design and GA design global optimization…….134
Table 3.5: The ISE for MIMO fail-safe control system under all situations compared between Pseudo inverse design and GA design global optimization and GA design worst case failure optimization……..134
Table 3.6: The Pareto front solutions for MIMO fail safe control system…….134
Table 4.1: The Pareto front solutions of three ISE…….134

Chapter One

INTRODUCTION

1.1 General Review of multivariable control system design

The design of multivariable controller has been the subject of much research effort since it was introduced late 1960's. The majority of design techniques involve the design of square systems where the number of inputs equal to the number of outputs. In practice there are many situations where the multivariable control systems are not square. Such control systems have had less research effort applied to them, but are still important research topics.

The multivariable control system design was first investigated by Rosenbrock H.H. in 1969 (Rosenbrock, 1969). He developed a method called the "inverse Nyquist array" and used an inverse Nyquist array to design multivariable controller. The inverse Nyquist array method involved an inverse matrix which is put in series with the system's transfer function matrix with the controller matrix. It is easier to invert square matrices, so the design method carried out by Rosebrock was for square system. Moreover, the resulting controller was complicated and difficult to tune. The controller is made of three matrixes; the first matrix is a permutation matrix, the second matrix has determinant equal to one, and the third matrix is a diagonal tuning matrix. In 1973, David Q. Mayne introduced a computer-aided procedure using Nyquist diagrams and root-loci (Mayne, 1973). This method uses computer process to choose the controller parameters and both the Nyquist diagram and root-loci to check the system stability. This method designs the controller with computer process and speeds up the transient response. However, this method requires the multivariable control system to be divided into several individual single loop control system, and then designing each single loop control system, finally, the combined single loop controllers are collected together to make the multivariable controller. This method was not straight forward for designing multivariable control systems. At the same

time Macfarlane A.G. and Belletrutti J.J. introduced the characteristic locus design method (Macfarlane and Belletrutti, 1973). This method combines the Bode-Nyquist method with the state-space method to guarantee the stability. However, this method cannot guarantee to provide a high system performance of the controller. In 1981 Zames G. introduced the H infinity method (Zames, 1981). The H infinity method is used to design the controller for a square system. The benefit of this method is that the controller is designed with stability and a good performance of the system guaranteed. However this method requires a large amount of math calculations and the final controller is very complicated.

1.2 Non-square multivariable control system

In the real world most systems are multivariable control systems, for example the number of inputs (m) and the number of outputs (l) are both greater than one. There are two types of multivariable control systems: the first type is the number of inputs (m) is equal to the number of outputs (l), this is known as square multivariable control system. The second type is the number of inputs (m) is not equal to the number of outputs (l), which is known as non-square multivariable control system. The design method for non-square system has been extended from square system design method. Latawiec K.J. Banka S. and Tokarzewski J. have extended the square multivariable LTI discrete-time system design method into Non-square LTI (linear time-invariant) discrete-time system design method (Latawiec K.J. Banka S. and Tokarzewski J., 2000). Latawiec K.J. and Hunek W.P. improved the non-square LTI discrete-time system design method into non-square continuous-time system design method.(Latawiec and Hunek, 2002) Sarma K.L.N. and Chidambaram M. have extended the two simple design methods which is called Davison's method (Davison, 1976) and Tantto and Lieslehto method (Tanttu and Lieslehto, 1991) for designing centralized controller from square systems to non-square systems with right half-plane zeros (Sarma and Chidambaram, 2005). They applied those two methods into two examples, and compared each other system performance and settling time. Davison's method gives better ISE performance and less settling time compared with Tanttu and Lieslehto method.

Additionally there are two types of non-square multivariable control systems: one is where the system has more number of inputs than number of outputs (m>l) and the other is where the system has less number of inputs than the number of outputs (m<l).In the situation of m>l, there are more inputs than the outputs in the system, which means the number of actuator in the system is more than the number of output, so the fail-safe method can be used to design this kind of systems.

The aircraft pitch roll control is the one of the classic example for fail safe control system design. The aircraft pitch roll control uses the elevators, inboard ailerons, outboard ailerons and canard to control the pitch and roll angle. The control system is used to control each elevators, inboard ailerons, outboard ailerons and canard angle to control the aircraft pitch and roll angle. If one of the elevators, ailerons or canard fails the results could cause the system to become unstable. Normally, the control structure of the aircraft pitch roll can be described like figure 1.1 (Bosworth, 2012):

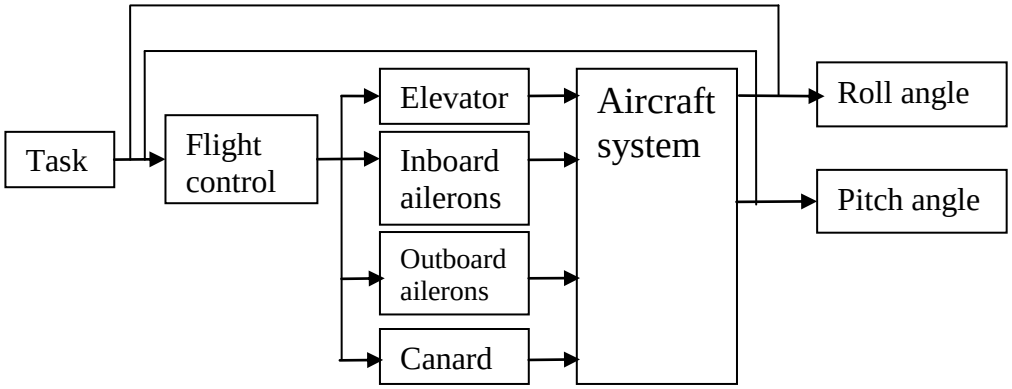

Figure 1.1: Control architecture

In this architecture, the system has four inputs: the angle of elevator, inboard ailerons, outboard ailerons and canard, and two system outputs: roll angle and pitch angle. Because there are two more actuators than the system outputs, then the system could should be able to cope with two actuator failures. Therefore, when looking into the design of aircraft roll and pitch angle control system the fail safe control theory becomes one of the options to consider.

In this fail-safe multivariable control system the system will remain stable even if some actuators fail.

However, in the case of the number of inputs is less than the number of outputs (m<l), the control system can only take control over *m* control loops and this become a lowest-wins or overrides control systems. One of the classic example of an override control system is jet engine control system. This kind of system can be shown like figure 4.1:

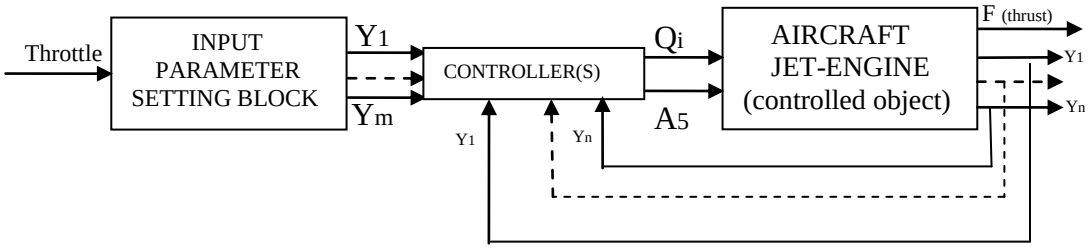

Figure 1.2: Aircraft engine's automatic control system

Mostly, aircraft engines are using fuel flow rate and inlet guide vanes to control engine's spool speed and the engine's burned gas temperature and the total thrust (Tudosie, 2011). Only two of three variables can be controlled at any time. Surprisingly, if engine speeds and temperature become too high, the engine thrust is not controlled as the control system switches to control the two variables which are too high.

1.2.1 Fail-safe control system design

1.2.1.1 Actuator failure in multivariable control system

Safety and reliability are the one of the key tasks in design multivariable controllers or complex industrial plants. Indeed, it is one of the most significant aspects of the design specification. The safety and reliability are very important in all design considerations and even relate to the cost. It may result in many financial fines if plant failures occur. This area forms a very important area for research in multivariable control system design. The fail-safe logical systems have been developed by Mine H. and Koga Y. to ensure that the system outputs remain stable even in failure situations.

A fault-tolerant controller design has been provided in the methodology for single actuator failure in multivariable control system. (Fripp, 1988) Fripp used the Pseudo Inverse method to design an controller for multivariable control system in a single actuator failure situation. The method guarantees the system's stability for a single actuator failure. Furthermore, this methodology cannot be used to design a fail-safe controller for multi actuator failure in multivariable control systems. R.N. Fripp's method can provide a stable system in both non-actuator failure and single actuator failure situations, but the failure does impact the performance of the system. Robert R.J., Medanic J.V. and Perkins W.R. has developed centralized and decentralized control design methodology to provide guaranteed stability and H_∞ performance in both non-actuator failure and actuator failure situations (Robert et al, 1992). However, their guaranteed stability and H_∞ performance only work for predesigned actuator failure. Zhao, Q. and Jiang, J. has developed a robust method to provide guaranteed system stability and acceptable performance in both non-actuator failure and actuator failure (Zhao and Jiang, 1998). Yang, G.H., Wang, J.L., Soh, Y.C. and Liao, F. has developed a reliable control design method to design an unchangeable controller to provide guaranteed stability and H_∞ performance (Yang et al, 2001 and Yang et al, 2002).

All the passive design methods can provide guaranteed stability and acceptable performance, but not optimal performance. One technique for design such controllers is the use of genetic algorithm (Porter and Jones, 1992). Genetic Algorithm could be used to design the controller to provide guaranteed stability and optimal performance for both non-actuator failure and actuator failure situations.

1.2.1.2 Override control system design

The override control method is used to deal with control system with less inputs than the outputs. Each output should maintain a designed set-point range. In this type of system only one of these outputs can be controlled by one input. Glattfelder, Schaufelberger and Fassler introduced override control for the first time in 1983

(Glattfelder et al, 1983). They investigated the stability of the override control. The controlled loop may be switched during the system when it is running if another output variable starts to go up and goes above its maximum limit. In an override control system the switching action uses rules to prioritize the lowest errors (Alejandro and Joseph, 1993). Because the error in each loop is calculated by the system output minus the set point, and the set point is the limit which the system output should not cross. Therefore, the lower value of error means the output is closer to the limit set or above the limit. Therefore, if the lowest error wins this means the variable most above it limit loop is always under control. In 1988, Glattfelder and Schaufelbereger extended their override control stability method into disctete-time single loop override control (Glattfelder and Schaufelbereger, 1988). Recently applications of override control have been developed into many different control systems. In 2007, Chen X.S., Zhai J.Y., Li Q. and Fei S.M. have combined override and model predictive control together to design the grinding control strategy (Chen et al, 2007). They just add override control into the system to avoid mill overloading and to optimize the fresh ore feed rate. However, they do not improve the override control, just use it as Glattfelder and Schaufelbereger's version. In 2010, Tran T. has added overriding control and manifest variables together into closed-loop system, to ensure the system is stabile with minimum knowledge of the system model (Tran, 2010). Again, he just used Glattfelder and Schaufelbereger's override control method.

One of the classic example of an override control system is the jet engine control system (Tudosie, 2011). Mostly, aircraft engine is using fuel flow rate and inlet guide vanes to control engine's spool speed, the engine's burned gas temperature and the total thrust. Because the jet engine control system is a closed loop system, the feedback signals become the controller's inputs; such as the engine's spool speed and engine's burned gas temperature. However, the number of the outputs (such as spool speed, burned gas temperature, the total thrust and the number of the outputs is three) is greater than the number of the input (such as fuel flow rate and exit area, and the number of inputs is two). Therefore the controller can only control two outputs at one time and the two outputs which are above their limit needs to be controlled. During this situation the controller can switch between the control loops to prevent variables in the engine exceeding safety limits and this is why override control is important.

1.3 Optimisation of Multivariable control system

In 1971 Michael Athans introduced Linear-Quadratic-Gaussian (LQG) control (Athans, 1971). LQG is one of the most fundamental optimal control theories. It is a combination of Kalman filter and linear-quadratic-regulator (LQR). The Kalman filter and LQR are designed and computed independently. LQG is going to minimize the quadratic cost function, which is related by the state variables and the system input, and two more matrixes should be chosen by the designer, the two matrixes are related to the system's performance and stability. The first disadvantage of the LQG method's is the tuning of the trade-off matrixes is very difficult, the second disadvantage is the quadratic cost function calculation is very complicated as the equation is a large matrix equation and it requires a significant amount of calculation to compute. In 1986, D.S. Bernstein, L.D. Davis and D.C. Hyland have improved the quadratic cost function for reduced-order modelling, estimation and control in the discrete-time case (Bernstein et al, 1986). However, they still need Lyapunove method for further stability checking.

In 1975 D.W.Clarke and P.J.Gawthrop introduced a self-tuning controller into single input and single output system (Clarke and Gawthrop, 1975), they used the recursive least-squares algorithm in square-root form to identify the systems parameter and then used the pole and zero method based on the identified system parameters to design the controller parameter. It 1979 Ulf Borisson introduced the multivariable minimum variance self-tuning controller (Borisson, 1979). He improved the self-tuning method into multivariable system. He used self-tuning control theory to design a controller to control unknown parameter linear multivariable system. The method also used a recursive least squares estimator to identify the system's parameter and design the controller.

In 1981 Zames G. introduced the H infinite method (Zames, 1981). The H inifinite method is used to design the controller for a squared system. The benefit of this

method is the controller is it will provide the stability of the system and a good performance of the system however this method requires a lot of math calculation.

The concept of using genetic algorithm to solve optimisation problems was introduced by Holland J.H. in 1975. This technique can be applied to the issue of control system design and control system tuning. Moreover, genetic algorithm have not been used to design multivariable control systems for non-square plants. Therefore, in this thesis, the genetic algorithm has been adopted for the design of such controllers.

To solve optimization problems the GA requires an objective function (also call cost function) and the GA will evaluate the objective function for given input parameters (Neeraj and Kumar, 2014). Because in this thesis, the genetic algorithm is applied for multivariable control system, there are two types of system output performances: set point tracking and interaction. So the objective function for the genetic algorithm used in this thesis is made up of the system output to set point changes plus the other output interactions. However, some of the system outputs required a constraint, for example, if the overshoot is too big when the optimized system is obtained, or if the interaction is too big, then the overshoot constraint should be used (Gilbert and Tan, 1991). The genetic algorithm can accommodated constraint easily. Because the number of cost functions in optimisation does not always equal to one (Ishibuchi et al, 2006). There are two main type of GA: the first type is where there is single cost function to be optimised, this type of GA is called single objective GA, the second type is where there is more than one cost function to be optimal at same time, this type of GA is called multi-objective GA. Both single cost function genetic algorithm and multi-objective genetic algorithm will be considered in this thesis.

The main purpose of decoupling multivariable control system design is finding the best transient response of the system with minimum interaction. In such control system design problems finding the controller which gives the best performance is called system optimisation. Therefore, the goal of optimisation is to design a multivariable controller which can provide a fast transient response with small interaction in all channels (Coit et al, 2004). Because the system error could be

positive or negative, and if the integral the system's errors are used, the integral of error can be positive or negative, and adding all the integral of errors together, the positive error could cancel with the negative errors. Therefore, normally the error will be squared. There are three main cost function could to define a system's performance, these are: Integral Squared Error (ISE) (Mukherjee, and Mishra, 1987)

The Integral Squared Error (ISE) is calculated by

- $ISE = \int e^2 dt$

Where e is the system output error.

Integral Absolute Error (IAE) (Graham and Lathrop, 1953) And Integral Absolute Error (IAE) is calculated by

- $IAE = \int |e| dt$

Where e is the system output error.

Integral Time-weighted Absolute Error (ITAE) (Graham and Lathrop, 1953). And the Integral Time-weighted Absoluted Error (ITAE) is calculated by

- $ITAE = \int t|e| dt$

Where e is the system output error.

All the three cost functions use a system's output performance that can includes a set point change or disturbance rejection.

If the cost function use Integral Square Error (ISE), the ISE will focus on the larger errors rather than smaller errors. The square of a large error will be much bigger, and if GA is going to minimise the ISE, and the ISE will tend to eliminate the large error quickly, but the ISE can leave small errors at the end of the transient response. Normally, at the end of running a genetic tuning algorithm which uses ISE as its cost function, the system output will exhibit a fast response, but may have low amplitude oscillation (Tavakoli and Tavakoli, 2003).

If the cost function use Integral Absolute Error (IAE), the IAE does not add any weight to any of the errors. If the GA is going to minimise IAE, at the end of running the IAE, the system output will be slower than when using the ISE, but usually the IAE result will have a quicker settling time as compared to ISE (Boz and Sari, 2009).

If the cost function use Integral Time-weighted Absoluted Error (ITAE). The ITAE will weigh the errors more at the end of the transient response. The advantage of ITAE is the system output will settle down much quicker than ISE and IAE methods. However, the disadvantage of ITAE is that the initial system response may be slower. This is because the cost function is making sure the errors at the end of the transient response will be small (Maiti et al, 2008).

In this thesis the controllers used are Proportional and Integral controllers. The integral term ensure zero steady state error. Therefore, the ITAE and IAE algorithms are not an effective. Moreover, the ISE cost function penalises the large errors because ISE algorithm involves error squared. So the initial system response should be improved. Therefore, in this thesis the cost function chosen is ISE.

1.3.1 Single objective genetic algorithm design for multivariable control system

The single objective genetic algorithm is the standard genetic algorithm and it is widely used for solving many problems. The Genetic Algorithms (GA) uses the evolution concepts such as selection, crossover and mutation to generate new solutions to optimize and to search for solutions. The genetic algorithm was first introduced by Holland J.H. in 1975 (Holland, 1975). This is the standard genetic algorithm which includes selection process which is based on the fitness of each individual; the fitness is calculated using a cost function. This also includes the crossover and mutation to generate the next generation of the population. In the same year, Holland J.H. improved his standard genetic algorithm into a steady-state genetic algorithm (Holland, 1975). This algorithm is not like the standard version, the standard genetic algorithm will generate a new population from the previous population, but the steady-state genetic algorithm maintains the population and

updates the individuals. In this technique the diversity of the population is poor. In 1989, Goldberg D.E. introduced messy genetic algorithm (Goldberg, 1989). Compared with the standard genetic algorithm, the messy genetic algorithm has three differences. The first one is messy coding: the length of an individual is not fixed; and second one is the messy operator: the crossover is no longer exists and instead a splice and cut method is used with this method, individuals will cut at any position and the second part of each individual will switch; and the third difference is tournament selection. The tournament selection randomly chooses several individuals from the population, and they are comparing each other, and the best fitness individual wins. To compare this with standard selection, the tournament selection selects the best individual from fixed size random individuals rather than all of the population, so the tournament selection is much quicker. However, the tournament selection has a chance to miss the global best individual. Furthermore, this algorithm is very complicated. In 1995, Rowe J. and East I. introduced the direct replacement genetic algorithm (Rowe and East, 1995). This algorithm is very similar to the steady-state genetic algorithm but this algorithm does not have mutation. Therefore, this algorithm is quicker than the steady-state genetic algorithm. After 1990 many researchers combined the genetic algorithm with other optimization methods, this kind of combination genetic algorithm is called Hybrid Genetic algorithm. Like Weare R., Burke E. and Elliman D., they combine different crossover operator to make the better children (Weare et al, 1995). Like Wan W. and Birch B. has combined the genetic algorithm with a new local search procedure (Wan and Birch, 2013). They used the new local search procedure to generate new children, the main idea is that the new child is only generated by the best parent and it will be kept if and only if it is the best in the children population and also is the best in the parent population. This procedure will slow down the genetic algorithm but it has a better chance to find the best globally individual.

1.3.2 Multi-objective genetic algorithm design for multivariable control system

Many real world problems involve the optimisation of more than one variable, and requires a group of variables to be simultaneously optimised. The single cost function genetic algorithm can deal with this situation by adding a weighting factor to each individual objective. However, it is difficult to choose the weighting factor. Because of this the multi-objective genetic algorithm has been developed (Ishibuchi et al, 2006). Moreover, many real-world problems have no single optimal solution, but have a set of optimized solutions; these solutions are optimal in the wider sense. No single solution is better than the others when all objectives are considered. This type of solutions is called a non-dominated Pareto-optimal solutions (Weile et al, 1996).

During 1993 to 1995, Fonseca and Fleming's (Fonseca and Fleming, 1993) extended the single objective genetic algorithm to multi-objective genetic algorithm. There is no cost function in multi-objective genetic algorithm, and instead the ranking is done by the domination method. All the best non-dominated solutions are Pareto-optimal solutions and by including a fitness sharing method to maintain the diversity of the solution a set of non-dominated optimal solutions can be obtained. Srinivas and Deb's (Srinivas and Deb, 1995) also improved the single objective evolution algorithm to non-dominated Sorting Genetic Algorithm (NSGA). This algorithm needs to find member of the population which dominate other solutions and has to search the population again and again to find out the rank of each individual therefore this algorithm is very slow. Horn, Nafploitis and Goldberg (Horn et al, 1994) has improved the single objective genetic algorithm to Niched Pareto Genetic Algorithm (NPGA). This algorithm uses tournament selection based on Pareto dominance. This algorithm is fast but it has the chance to lose the best solutions. Because tournament selection is not going to select the best individual from the total population, but it selects the best individual from a fixed size individual, and those individuals are randomly chosen from total population. Therefore, there is a chance that the local best individual is not going to be selected, and then the best is missed.

Zitzler and Thiele improved the multi-objective evolution algorithm by using the Strength-Pareto Evolution Algorithm (SPEA) (Zitzler and Thiele, 1998). In the SPEA, the an extra population is added, and this extra population keeps the non-dominated solutions, and combines this extra population with the current generation population. However, the size of the extra population can easily grow too large and this could slow down the search procedure. Knowles and Corns improved the multi-objective evolution algorithm by using Pareto-Archived Evolution Strategy (PAES) (Knowles and Corne, 1999). In the PAES, the single-parent and single-child algorithm is used. This compares the child with the parent, if the child is dominating the parent, the child is the parent for the next generation; if the child is not dominating the parent, this child will be killed and a new child will be found. But this algorithm needs an extra algorithm to maintain diversity of population. An interesting multi-objective genetic algorithm has been developed by Deb K., Pratap A., Agarwal S. and Meyarian T. in 2002 called the Non-dominated Sorting Genetic Algorithm II (Deb et al, 2002). This algorithm is an improved version of the NSGA. This algorithm not only finds out if the individual is dominated by other individuals but also finds out how many other individuals are dominated by it. The greater the number of individuals dominated the higher fitness the individual gets. This algorithm used crowding distance method to maintain the diversity of the population. Crowding distance methods finds out the distance of the two closest solutions, a larger crowding distance is better. In 2011 Yan T., Guo G. and Wu L. improved multi-objective genetic algorithm by using granular ranking and distant reproduction (Yan et al, 2011). The granular ranking is dominated, if the individual is not dominated by others; the rank which is highest belongs to the Pareto front solution. The distance reproduction is calculating the distance between the solutions; if the distance is small it will not be chosen to generate the child. This maintains the diversity of the population.

1.4 Aim and objective

The aim of this thesis is using single cost function genetic algorithm and multi-objective genetic algorithm to optimise the design of square and non-square multivariable decoupling control systems. The performance of the multivariable system is measured by the set point tracking performance plus the interaction performance defined as an ISE cost function. For square multivariable control systems design, single cost function genetic algorithm and multi-objective genetic algorithms using the Pareto front method are used to design controllers which provide optimal performances under different situations.

There are two type of non-square multivariable control system design method: fail safe method and override method. The fail safe method is when for the number of inputs is greater than the number of outputs. There is no formal design method for multiple actuator failures. The single cost function genetic algorithm is used to design both multivariable fail safe system under single actuator failure and multiple actuator failures. The genetic algorithm can optimise system performance under all non-failure and failure situations. Moreover, multi-objective genetic algorithm using the Pareto front method can design a family of solution which provide optimise solution under different situations.

The override method is where the number of outputs is greater than the number of inputs. There are not many design methods and there is no general method for addressing the design. However, genetic algorithm could design the controller for this kind of system. Moreover, if the controller exists then the genetic algorithm will design a good controller that makes the system have good transient response.

1.5 Outline of the thesis

1.5.1 Chapter one: introduction

This introduction will introduce the two types of multivariable control systems: square multivariable control system and non-square multivariable control system. It will also review the methods for the design of each type of multivariable control system.

1.5.2 Chapter two: square multivariable control system design

In this chapter the design of square multivariable control systems by using single objective genetic algorithm and multi-objective genetic algorithm will be used to design multivariable control systems.

1.5.3 Chapter three: fail-safe control system design

In this chapter the design of single failure and multi failure fail-safe control system will be used to design multivariable control systems which can deal with single and multiple actuator failures.

1.5.4 Chapter four: override control system design

In this chapter the single input multi outputs override control system and multi inputs multi outputs override control systems will be addressed. The issue of limit cycling in override control system is reviewed. Single objective and multi-objective genetic algorithm design for both override control system are considered. Finally, the multi-objective genetic algorithm is shown to be able to determine the range of set points over with a designed controller will either limit cycle or nor limit cycle.

1.5.5 Chapter five: conclusion

In this chapter all the results of design will be discussed and single objective and multi-objective genetic algorithm methods are discussed.

Chapter Two

GENETIC DESIGN OF SQUARE MULTIVARIABLE CONTROL SYSTEMS

2.1 Introduction

A square multivariable control system is when the number of the input is equal to the number of the output and both the centralized and decentralized control systems are square multivariable control system. The centralized control system uses all process input and output measurements to simultaneously determine all the manipulated variables. In such controller's decoupling is the main design issue. This involves good set-point tracking characteristic and corresponding low levels of interaction on other channels. The decentralized control system uses one process input and one output measurement to separately determine each manipulated variable, and can give rise to significantly more interaction.

The multivariable control system design was first investigated by Rosenbrock H.H. in 1969 (Rosenbrock, 1969). He used a method called "inverse Nyquist array" and drew an inverse Nyquist array corresponding to design the controller. The inverse Nyquist array method used an inverse matrix which is the system's transfer function matrix with the controller matrix, and only squared matrix can be inverted in this way, so the system has to be square. Moreover, the analysis for the controller in this design method is carried out in the frequency domain. The controller in this design is made up to three matrixes; the first matrix is a permutation matrix, the second matrix has determinant equal to one, and the third matrix should be diagonal. In 1973, David Q. Mayne introduced a computer-aided procedure using Nyquist diagrams and root-loci (Mayne, 1973). This method uses an algorithm to design the controller parameters and both the Nyquist diagram and root-loci to check the system stability. However, this

method requires the multivariable control system to be divided into several individual single loop control systems, and then designing the single loop control system, then combine those single loop controllers together to make the multivariable control system. This method is not a straight forward method and it is difficult to do. In the same year Macfarlane A.G. and Belletrutti J.J. introduced the characteristic locus design method (Macfarlane and Belletrutti, 1973). This method combines the Bode-Nyquist method with the state-space method to guarantee stability. However, this method cannot guarantee to provide a high system performance. In 1981 Zames G. introduced the H_∞ method (Zames, 1981). The H_∞ method is used to design the controller for a squared system. The benefit of this method is that the controller is designed with stability and a good performance of the system. However this method requires a large amount of computation and the final controller has a very complex structure which makes implementation difficult.

In order to control the square multivariable control system, the Proportional, integral and derivative (PID) method is widely used (Singh and Mitra, 2014). The reason PID methods are widely used is that PID controllers have a simple structure, have good robustness as well as good performance. When a PID controller is used in a feedback control loop, it calculates an error value as the difference between a measured output and a designed set point (Rusnak, 2000). The PID controller attempts to minimize the error by changing the system input. The main design problem on the PID controller is the design of the Proportional, integral and derivative parameters of PID controller. The proportional term produces an output value that is proportional to the current error value, the integral term is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously, the derivative term is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain (Ying, 2011). However, the derivative term is not commonly used in practice because it can amplify noise in the control system and cause the control system to become erratic. Because of the poor noise performance of derivative control, the PI controller is considered in this thesis.

There are lots of developments of the PID method in single-input and single-output system which have been produced. In 1986, Rivera, D.E., S. Skogestad and M. Morari

introduced an internal model control based on PID controller design using a first order process model (Rivera et al, 1986). Their development was to improve the performance of the single input single output (SISO) system with time delay. In 1988, Chein, I.L. extended their work into a second order process model (Chein, 1988). After that, Wang, Q.G., C.C. Hang and X.P. Yang used the frequency response approach with least squares algorithm to develop a PID controller, this method can make the controller high-order to achieve high performance. Katebi, M.R. and M.H. Moradi introduced the predictive PID controller for SISO and first or second order systems however the performance is not improved much (Katebi, and Moradi, 2001). One year later M.H. Moradi., M.R. Katebi, and M.A. Johnson extended the SISO predictive PID controller into MIMO systems by using a polynomial form (Moradi et al, 2002). Tan, K.K., S.N. Huang, and T.H. Lee presented a PID control design based on the generalized predictive approach for a second order system with time delay (Tan et al, 2000). However, their method only dealt with SISO system. Later on Qamar Saeed, VAli Uddin and Reza Katebi developed a multi inputs and multi outputs predictive PID controller using the same approach (Saeed et al, 2010).

One of the most effect design technique for multivariable control system was proposed by B. Porter and A. Bradshaw at 1979 (Porter and Bradshaw, 1979) for stable or unstable plants. One of the main task is designing an effective multivariable control system depends on where the system transmission zeros are. The method to determine the multivariable transmission zeros was introduced by B. Porter and J.J. D'Azzo (Porter and D'Azzo, 1977) and indicated if the system is minimum phase system or non-minimum phase system. If the system is minimum phase system that the control system can be generally designed to be a good performance. However, if the system is non-minimum phase system then those control system may exhibit poor closed-loop performance. F.A. Himmelstoss, J.W. Kolar and F.C. Zach shows the stabilization of the non-minimum phase system is considerably more difficult compared with minimum phase system (Himmelstoss et al, 1991).

An example of a multivariable system which have both minimum phase and non-minimum phase was introduced by B. Porter and A.H. Jones (Porter and Jones, 1984). In this system, the zero's location is depend on the value of α .The

asymptotically stable plant is two input and two output system, and the transfer function is:

$$G(s) = \begin{bmatrix} \frac{1}{s+1} & \frac{\alpha}{s+3} \\ \frac{1}{s+1} & \frac{1}{s+1} \end{bmatrix} \quad \text{.....(2.1)}$$

- When $\alpha = 2$, the square multivariable system is non-minimum phase system, because the poles and zeros of the system are in the right hand place;
- when $\alpha = 3$, the square multivariable system is functionally uncontrollable, because when $\alpha = 3$, the zero is at the origin;
- when $\alpha = 4$, the square multivariable system is minimum phase system, because the zeros is in the left hand place.

E. J. Davison also introduced multivariable control system design method (Davison, 1976). He introduced a method which deals with problem of finding a controller for an unknown system, and the system should exhibit asymptotic tracking independent of input disturbances and parameter variations in the system. J. Pentinnen and H.M. Koivo also introduced a method which determining a multivariable robust PI controller for an unknown linear multivariable stable system (Pentinnen and koivo, 1980). They design the Proportional controller to use the interaction of the system, these interactions are detected by the observing the output of the system to step inputs. B. Porter and A.H. Jones introduced a method to design PID controller for square multivariable control system (Porter and Jones, 1986). They improved Davison's method for multivariable control system design. This method uses the steady state equation matrix to design the integral controller parameters and used the decoupling matrix (Parzen, 1997) to design the proportional and derivative controller parameters. The design techniques involve manual tuning diagonal controller matrix such that the control system archival stable performance with good tracking and low interaction effect. However, this method only works for minimum phase system.

Furthermore, there is no suitable controller tuning method for multivariable control systems. Moreover, most of multivariable control systems cannot be simplified into single-input and single-output system. Moreover, the number of controller parameters

in multivariable control system is much more than the number of controller parameters in single-input and single-output system. Therefore, automatically tuning of the PID controller parameters for multivariable control systems has not been achieved with using theoretical methods.

Genetic algorithm can be used to tune multivariable control system. There are two main approaches. One is to tune the decoupling controllers such as the one proposed by Porter and Jones (Porter and Jones, 1992). In such cases there are a set of proportional, integral and derivative gains to select. The other option is to allow the genetic algorithm to choose all of the parameters of the multivariable controller. In such case there are more parameters but this may result in better performance. In order to facilitate a comparison of the genetic design of multivariable control system, two controllers are considered. The first is a structured decoupling controllers as proposed by Porter and Jones which as a set of diagonal tuning parameters for the genetic algorithm to optimal. The second is a fully parameters controller, where the genetic algorithm has to optimal all of these controller parameters.

The first type of controller is:

$$u = B^* \times \Delta \times e(t) + \left(G(0)\right)^{-1} \times \varepsilon \times \int_0^t e(t)dt$$ (Porter and Jones, 1986) ……...(2.2)

where

$$B^* = CA^{-1}\left(e^{At} - I_n\right)B$$ ……..……..…….(2.3)

and

$$G(0) = -CA^{-1}B$$ ……..……(2.4)

and

A, B and C are steady space equation matrix, t is the sampling time, I_n is the identity matrix with n by n size;

Where

$$\Delta = \begin{bmatrix} \Delta_1 & 0 & \cdots & 0 & 0 \\ 0 & \Delta_2 & 0 & \vdots & 0 \\ \vdots & 0 & \ddots & 0 & \vdots \\ 0 & \vdots & 0 & \Delta_{m-1} & 0 \\ 0 & 0 & \vdots & 0 & \Delta_m \end{bmatrix} \quad \text{.....(2.5)}$$

And

$$\varepsilon = \begin{bmatrix} \varepsilon_1 & 0 & \cdots & 0 & 0 \\ 0 & \varepsilon_2 & 0 & \vdots & 0 \\ \vdots & 0 & \ddots & 0 & \vdots \\ 0 & \vdots & 0 & \varepsilon_{m-1} & 0 \\ 0 & 0 & \cdots & 0 & \varepsilon_m \end{bmatrix} \quad \text{.....(2.6)}$$

are diagonal matrix. Therefore, in square multivariable control system, all the Δ and ε to be tuned, and the total number of parameters need to be tuning are Δ_i and ε_i (i=1,2,3 … m).

The second type of controller is where all the parameters of the control matrix kp and ki can be searched for. In this system the controller equations for the multivariable Proportional-Integral controller is given by:

$$u = k_p \times e(t) + k_i \times \int_0^t e(t)dt \quad \text{.....(2.7)}$$

Where u is the system input, k_p is the proportional gain, k_i is the integral gain, and $e(t)$ is the error of the system output and set point. If this is a two input two output system,

$$k_p = \begin{bmatrix} k_{p11} & \cdots & k_{p1m} \\ \vdots & \ddots & \vdots \\ k_{pm1} & \cdots & k_{pmm} \end{bmatrix} \quad \text{.....(2.8)}$$

and

$$k_i = \begin{bmatrix} k_{i11} & \cdots & k_{i1m} \\ \vdots & \ddots & \vdots \\ k_{im1} & \cdots & k_{imm} \end{bmatrix} \quad \text{.....(2.9)}$$

and in square multivariable control systems, both controller parameters are $m \times m$ matrices. Therefore, the total number of parameters need to be tuned are k_{pij} and k_{iij} (i=1,2,3…m and j=1,2,3,…m).

The GA is going to find the optimal solution using a cost function. Therefore, the cost function is very important to GA. The cost function selection is also a measure of the controlled system's performance. These measures are used to compare the system's performance between different control situation or different controller parameters. There are three main cost function of a controlled system's performance, these are: Integral Squared Error (ISE)(Mukherjee and Mishra, 1987),

The Integral Squared Error (ISE) is calculated by

- $ISE = \int e^2 dt$ ……..(2.10)

Where e is the system output error.

Integral Absolute Error (IAE) (Graham and Lathrop, 1953) And Integral Absolute Error (IAE) is calculated by

- $IAE = \int |e| dt$ ……..(2.11)

Where e is the system output error.

And Integral Time-weighted Absolute Error (ITAE) (Graham and Lathrop, 1953). And the Integral Time-weighted Absolute Error (ITAE) is calculated by

- $ITAE = \int t|e|dt$ ……..(2.12)

Where e is the system output error.

All the three cost functions use the system's output simulated performance and can includes a set point change and may include a rejection to a disturbance. The system is tuned under a fixed situation, which will involve a set point change. The running time of any simulation should be long enough for the system responses to settle down.

If the cost function used is Integral Square Error (ISE), the ISE will focus on the larger errors rather than smaller errors. The square of a large error will be much bigger, and if GA is going to minimise ISE, and the ISE will tend to eliminate the

large error quickly. Normally, at the end of running a genetic tuning algorithm which uses ISE as its cost function, the system output will exhibit the fastest response which possible under this type of control (Tavakoli and Tavakoli, 2003).

If the cost function used is Integral Absolute Error (IAE), the IAE does not add any weight to any of the errors. If the GA is going to minimise IAE, at the end of the simulation, the system output will be slower than the ISE result, but usually the IAE result will have quicker settling time as compared to ISE (Boz and Sari, 2009).

If the cost function used is Integral Time-weighted Absoluted Error (ITAE), The ITAE will weight the errors more at the end of the system's output. The advantage of ITAE is the system output will settle down much quicker than ISE and IAE methods. But the disadvantage of ITAE is the slow system response (Maiti et al, 2008).

The single input and single output system has open-loop transfer function:

$$G(s) = \frac{0.4e^{-1.8s}}{0.9s + 1} \quad \text{.....(2.13)}$$

is chosen as an example that shows the different closed loop step responses results for a proportional-integral controller which have been genetically tuned for optimal performance. In this case ISE, IAE and ITAE where each used as the cost functions. Then the closed loop step response results is shown below (Tavakoli and Tavakoli, 2003):

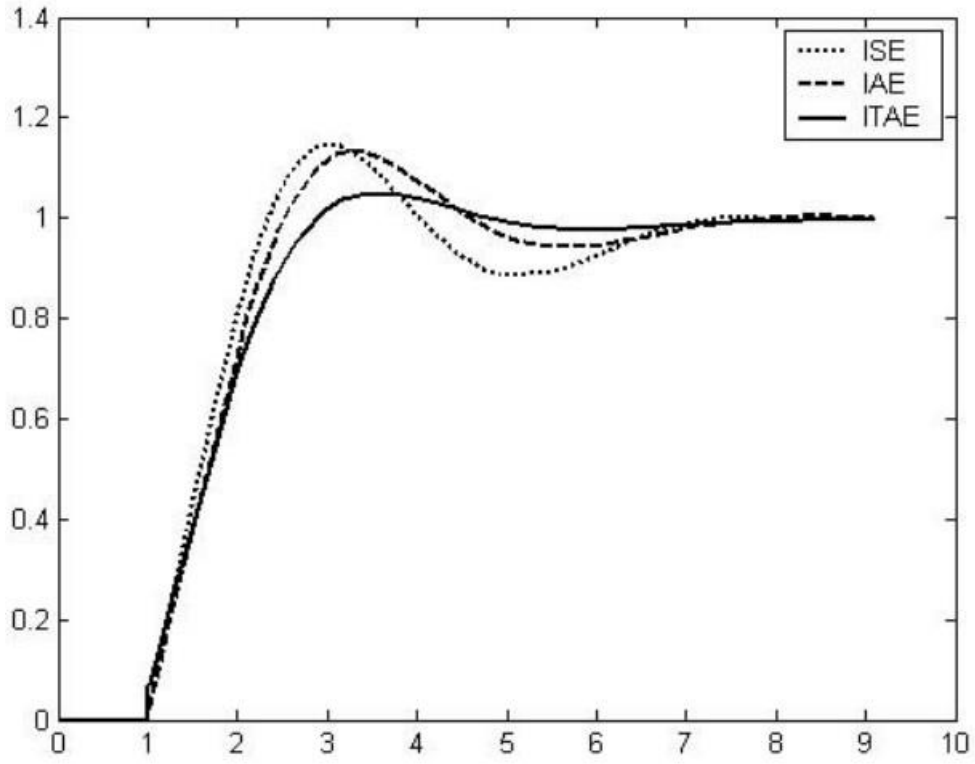

Figure 2.1: Closed loop step response resulted from applying proposed PID parameters

As figure 2.1 shows, the closed loop step response result optimized by ISE has the quickest system response, and the settling time of ISE, IAE and ITAE are similar. Because the GA could use any cost function, ISE, IAE or ITAE could be used as cost functions in GA to optimise the system's performance. In this work the ISE is used because it gives the fastest response.

The total ISE is equal to the sum of ISE for each output, and each output ISE is equal to the ISE calculated by the set point tracking plus the ISE calculated by the interactions in the other channels due to the set-point change. For example, there is a three inputs and three output system:

$$G(s) = \begin{bmatrix} g_{11}(s) & g_{12}(s) & g_{13}(s) \\ g_{21}(s) & g_{22}(s) & g_{23}(s) \\ g_{31}(s) & g_{32}(s) & g_{33}(s) \end{bmatrix} \quad \text{.....(2.14)}$$

there are set point tracking on $g_{11}(s), g_{22}(s)$ and $g_{33}(s)$, all others are interactions.

Then ISE for each individual outputs will be like below:

$$\begin{bmatrix} \mathrm{ISE}_{11} & ISE_{12} & ISE_{13} \\ ISE_{21} & ISE_{22} & ISE_{23} \\ ISE_{31} & ISE_{32} & ISE_{33} \end{bmatrix}$$

Where ISE_{11}, ISE_{22} and ISE_{33} are the ISE caused by the set point change, ISE_{21} and ISE_{31} are the interaction ISE caused by the set point 1 change. ISE_{12} and ISE_{32} are the interaction ISE caused by the set point 2 change. ISE_{13} and ISE_{23} are the interaction ISE caused by the set point 3 change.

The total ISE for the three outputs cause by the set point change in output 1:
$ISE_1=ISE_{11} + ISE_{21} +ISE_{31}$
The total ISE for the three outputs cause by the set point change in output 2:
$ISE_2=ISE_{12} + ISE_{22} +ISE_{32}$
The total ISE for the three outputs cause by the set point change in output 3:
$ISE_3=ISE_{13} + ISE_{23} +ISE_{33}$

Therefore, for the single cost function GA, the cost function ISE (Integral Square of Error) is calculated by:

$$e = \int (e)^2 dt = (\varpi_1 \times ISE_1 + \omega_2 \times ISE_2 + \varpi_3 \times ISE_3)$$

where ϖ_i is the weight factor of each ISE.

2.2 Genetic algorithm

2.2.1 Introduction

Evolutionary Algorithms (EAs) are a rapidly growing area of artificial intelligence. In 1973 Rechenberg I. introduced evolutionary computing in his work "Evolution strategies". The Genetic Algorithm (GA) developed in 1975 (Holland, 1975). Genetic algorithm is the one of the best known evolutionary computation methods which is able to deal with a wide range of difficult optimisation engineering problems.
The Genetic algorithm starts with an initialled population of solutions and improves the population towards the optimum solution (Back, 1996). This process is improved by an evaluation procedure (fitness function) that finds out the fitness of each member

of the population. As an optimisation tool (Bach and Schwefel, 1993) GA is a random search algorithm that uses random choice as a tool and coding of the parameter space to improve the population. A GA is different from traditional optimisation and searches procedures in the following respects:

- GA works with a coding of the parameter set, not the parameters themselves.
- GA searches from a population of points, not a single point
- GA uses objective function information, not derivative or other auxiliary knowledge.
- GA uses probabilistic transition, not deterministic rules.

The advantage of GA is it tracks the solution in a search space with more individuals so they are less likely to get stuck in a local minimum like some other methods (Back and Schwefel, 1991). The genetic algorithm is good, flexible and easy to implement, allowing a variety of problems to be formulated and solved without much change or improvement to the code. The disadvantage of GA is the computational time. It can be slower than some other methods, but with modern computers it is not as much of a problem.

The basic idea behind GA is simply to do what nature does. The population improves over time through competition (keep the fittest) and controlled variation (crossover & mutation). In this way, the best elements of the current population has a higher chance to be used in the next population (child population) and whether the element is good or bed depends on the fitness, a stronger fitness is better. For example, the parent elements are selected according to their fitness; the more suitable they are the more chances they have to reproduce. Since the child population is always produced by the best elements (the highest fitness) of the parent population, the average fitness of the population will improve as the generation grows and the overall fitness of the new generation will always be better than the old one (Clark et al, 1987). The first population can be initialised randomly from the search space. The cycle of evolution is repeated until the optimal result has reached.

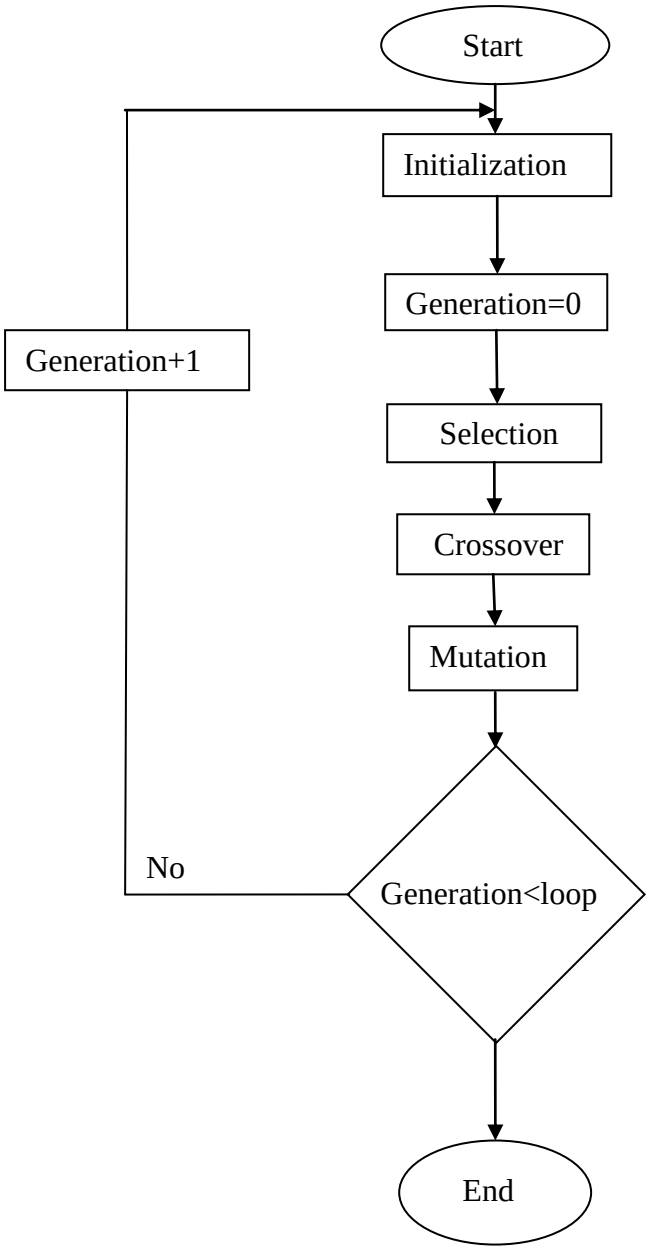

Figure 2.2: The basic cycle of genetic algorithm

In general, the outline of the basic genetic algorithm (2.2) is as follows:

[Start] Generate random population of n chromosome x in the population

[Fitness] Evaluate the fitness f(x) of each chromosome x in the population.

[New population] Create a new population by repeating following steps:

[Selection] Select cow parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected)

[Cross over] With a crossover probability cross over the parents to form a new offspring (children).

[Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).

[Accepting] Place new offspring in a new population.

[Replace] Use new generated population for a further run of algorithm.

[Test] If the end condition is satisfied, stop and return the best solution.

[Loop] Go to step 2.

The GA has two main types: the first is the single cost function GA, the second is multi cost function GA(multi objective GA). The main goal of single cost function GA is to find the best solution. The best solution could be the minimum or the maximum value of a single cost function (Bandyopadhyay and Saha, 2013). The single cost function could be single objective or multi objectives, in multi objectives situation, each objective requires a weighting factor to be added into each objective, and the final cost is the sum of weighted costs. This technique will convert the multi cost function into a single cost function (Ishibuchi et al, 2006). The minimization or maximization of the objective is dependent on each problem. If the problem is a maximization optimization problem, the genetic algorithm just selects the maximum fitness values of each individual; if the problem is a minimization problem, the genetic algorithm need an extra technique to convert the minimization problem into the maximization problems. There are two types of ways to convert minimization problems into maximization problems: the first one is let the

- $Fitness = \frac{1}{Cost}$ .....(2.15)

And the second way is

- $Fitness = Max - Cost$ .....(2.16)

Where Max is chosen by the designer (Rani and Kumar, 2012). For the first case, the minimization problem involve converting the problem into an maximization problem. However, if the value of Cost is becomes small, the value of Fitness can become to big. Furthermore, if the differences between two Costs are small, then the difference between Fitness will even be smaller, this will make the Genetic Algorithm program less able to differentiate between good and very good solutions. The second technique is to calculate the Fitness by choosing a positive constant (Max) and subtractly the cost from the GA cost, If the value of “Cost” is small, the value of “Fitness” is going

to big. However, the value of Max should choose carefully if the value of Max is chosen too big, the difference between Fitness will be large, then the Genetic algorithm program again cannot differentiate between two costs; and if the value of Max is chosen too small, the Genetic Algorithm program will not work as (Max – Cost) become negative. Moreover, the genetic algorithm is going to optimise the system's performance, in control system design the cost function used is ISE, and the ISE need to be minimised to optimise the system's performance. In addition, the Fitness calculated by the first technique is non-linear, and the Fitness calculated by the second technique is linear if the value of Max is chosen carefully. For example, if the genetic algorithm has improved the ISE from 20 to 5, and the value of Max is chosen as 21. So plot out the Fitness calculated by the two techniques are show below:

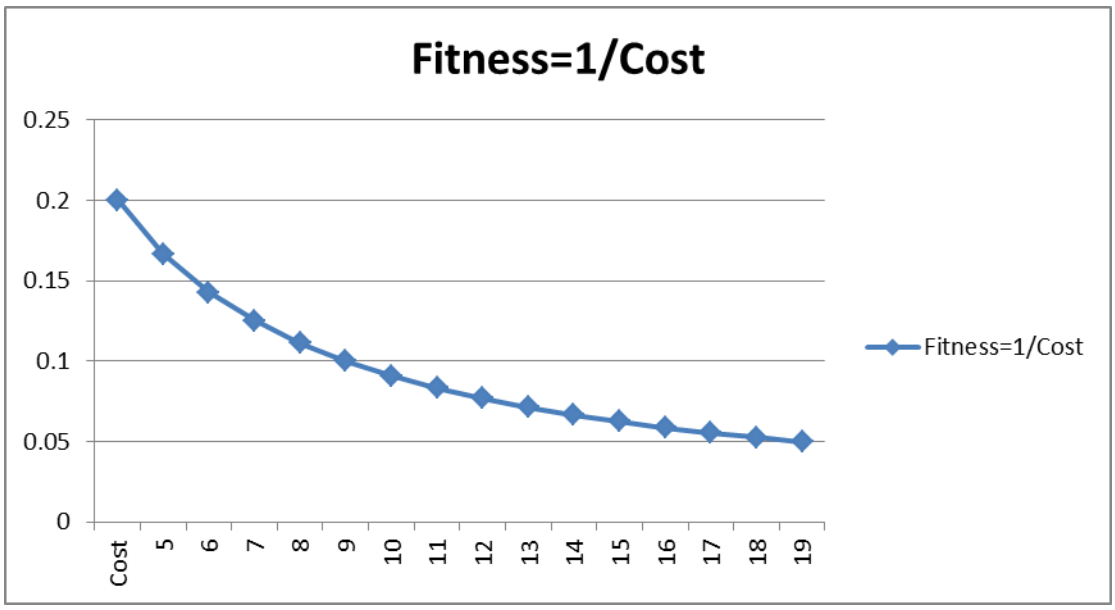

Figure 2.3: Fitness calculated by the first technique against cost

As figure 2.3 shows, the Fitness is calculated by the reciprocal of cost technique, with this technique, the Fitness increase is going to become bigger as the Cost reduces. This means the smaller cost get a much better chance to be select by the GA.

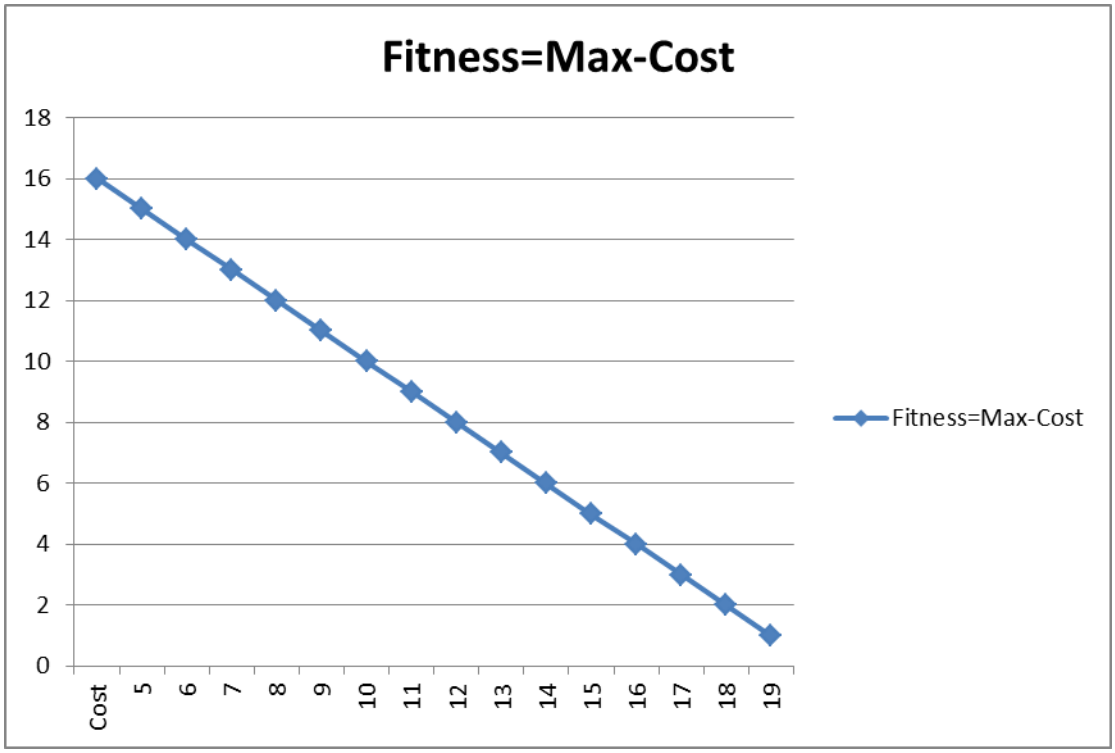

Figure 2.4: Fitness calculated by the second technique against cost

As figure 2.4 shows, the Fitness is calculated by the max minus cost technique, as it shows, the fitness is linear increase as cost improvement. However, this technique require the designer to choose the value of Max carefully, if the value of Max is chosen too big, the difference between Fitness will be large, then the Genetic algorithm program again cannot differentiate between two costs; and if the value of Max is chosen too small, the Genetic Algorithm program will not work as (Max – Cost) becomes negative. To compare these two techniques, the advantage of the first technique is the calculation is robust, because the Fitness is all ways positive. The advantage of the second technique is that if the value of Max is chosen carefully, the Fitness calculated by this technique will be better than the Fitness calculated by the first technique, because this technique will directly kill those individuals that do not have such good cost, as the (Max – cost) becomes negative. The disadvantage of the first technique is the difference between two costs is non-linear, if the difference between two costs is very small and the cost is large, then GA under this technique will find it is very hard to differentiate which is better; the disadvantage of the second technique is the value of Max is very hard to choose at the first time the algorithm is ran, if it is too big, then the GA is very hard to differential two costs, if it is too small,

then the GA is not going to be able to initiate a population, so the value of Max may have to be found by running the GA a number of times to find out typical values of cost, then the value of Max is chosen a little bigger than the value of cost. For example, consider the system which has open loop transfer function:

$$G(s) = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} = \begin{bmatrix} \frac{1}{s+1} & \frac{2}{s+3} \\ \frac{1}{s+1} & \frac{1}{s+1} \end{bmatrix} \quad \text{.....(2.17)}$$

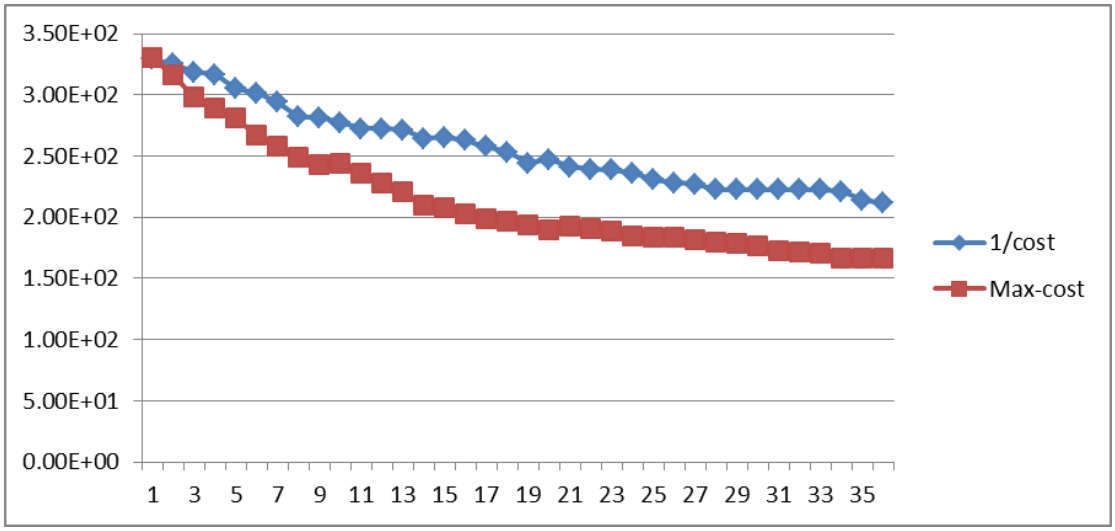

Figure 2.5: Fitness against the number of generation

As the figure 2.5 shows above, the fitness under two cost techniques shows the (max – cost) technique coverage quicker than the reciprocal of cost technique. The genetic algorithm will find the optimal solution by using both cost function if the running time is infinity. However, the convergence of max minus cost better, so in this thesis the max-cost technique is used.

In a genetic algorithm, the parameter range is very important, because it is related to how the GA finds out the optimal solutions. Moreover, the parameter range normally is not easy to choose. Therefore, the concept of parameter range movement has been included into the GA program. The parameter range movement method is when the best individual becomes close to the upper or the lower range limit, and then the whole range moves up or down. Normally the "close to the upper or the lower range limit" means 10% of the total range size, and how much percentage is chosen by the

designer. Normally if the range needs to be moved, the range will move up or down by 20% of the total range, the exact percentage to move is chosen by the designer. After the range movement, the GA needs to re-decode and re-scale the parameters again with the new range, to make sure the GA binary number and the real parameter number are not changed. Without adopting this technique, the parameter range is not granteed to coverage to the optimal solution.

2.2.2 Multi-Cost function Genetic Algorithm (Multi-objective Genetic Algorithm)

In many control problems, there is more than one objective that needs to be optimised, the technique used to optimization these kind of problems is called Multi-objective optimization (Osyczka, 1985). The Multi-objective optimization problem is going to optimizes all the objective functions and normally those objective functions are conflicting or against each other. Such that improving one objective and reduces other objective. To optimize all objective function means to find out such a solution which will make one individual objective function optimal and have the other cost functions as optimal as possible. In multi-objective optimization problems, there is no single optimal solution, indeed there are family of optimization solutions. Because in multi-objective problems, the optimal solutions are the a set of compromise (or trade-offs) solutions (Edgeworth, 1881). Vilfredo Pareto used this idea to introduce the Edgeworth-Pareto optimum method or Pareto optimal (Pareto, 1896). In this method, a family of solutions are found which would improve one objective and at the same time does not make the other objectives worse. The set of Pareto optimal solution are called non-dominated solution, because those solutions are not dominated by other solutions. The plot of the non-dominated solution is called the Pareto front. In 1989, the Pareto-based technique called Vector Evaluated Genetic Algorithm (VEGA) suggested by Goldberg and Schaffer (Goldberg, 1989). This VEGA uses non-dominated ranking and selection to find out the Pareto front. They used the technique called “Fitness Sharing” to maintain the diversity of the population (Goldberg and Richardson, 1987). The fitness sharing technique will reduce the individual’s fitness if two individuals are similar. The Genetic algorithm based on the Pareto-front method is easy to implement, and does not involve choosing any

weighting factors which can be very difficult to choose in single cost function Genetic Algorithm. Moreover, the weight factor is even more difficult to choose if the objectives are in different variables and the multi-objective genetic algorithm avoids this problem. Therefore, the multi cost function GA has been used in this thesis. There is no straight minimum or maximum cost function in multi-objective genetic algorithm, but instead the domination method is used. All the best non-dominated solutions are Pareto-optimal solutions and include the fitness sharing method to maintain the diversity of the solution. Srinivas and Deb's (Srinivas and Deb, 1995) has improved the single objective evolution algorithm to Nondominated Sorting Genetic Algorithm (NSGA). This algorithm needs to dominate the population again and again to find out the rank of each individual therefore this algorithm is very slow. Horn, Nafploitis and Goldberg (Horn et al, 1994) has improved the single objective genetic algorithm to Niched Pareto Genetic Algorithm (NPGA). This algorithm uses tournament selection based on Pareto dominance. This algorithm is fast but it has the chance to lose the local best. Because tournament selection is not going to select the best individual from the total population, but it selects the best individual from a fixed size individual, and those individuals are randomly chosen from total population. Therefore, there is a chance that the local best is missed. In 1995, Chipperfield A. and Fleming P. introduced the multi-objective genetic algorithm (MOGAs) into the design of a squared multivariable control system for a gas turbine engine (Chipperfield and Fleming, 1995). This multi-objective genetic algorithm method is evolving a family of Pareto-front solutions, these solutions allow the designer to examine the trade-off the individual objectives.

For multi-objective GA, the Pareto optimal method is used. And the chart of this method is shown below:

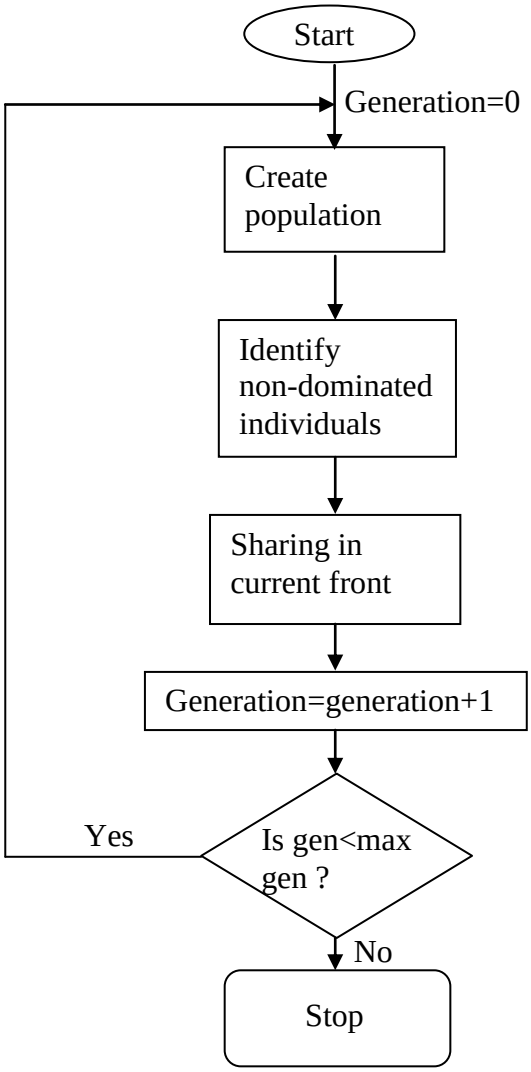

Figure 2.6: Flow chart of pareto front method

In the Pareto front method, the rank of non-dominated is equal to the number of the result better than it. Which means the rank of non-dominated will be plus one if there is one other result which dominates it (Zitzler and Thiele, 1998). The rest of the procedure is similar to the single objective GA. Compared with the single objective GA, the multi objective GA optimises all the multi objective functions simultaneously. Therefore, there is no single best solution to be found, but a family of Pareto solutions exist. An individual solution belongs to the family of Pareto solutions as there is no other solution that can improve at least one of the objectives and improve another objective simultaneously (Horn et al, 1994).

In addition, as different optimisation problem requires the minimization or maximization, the Pareto front solution can be optimisation by minimization or maximization. If the problem involves minimization, the Pareto front has the form show below:

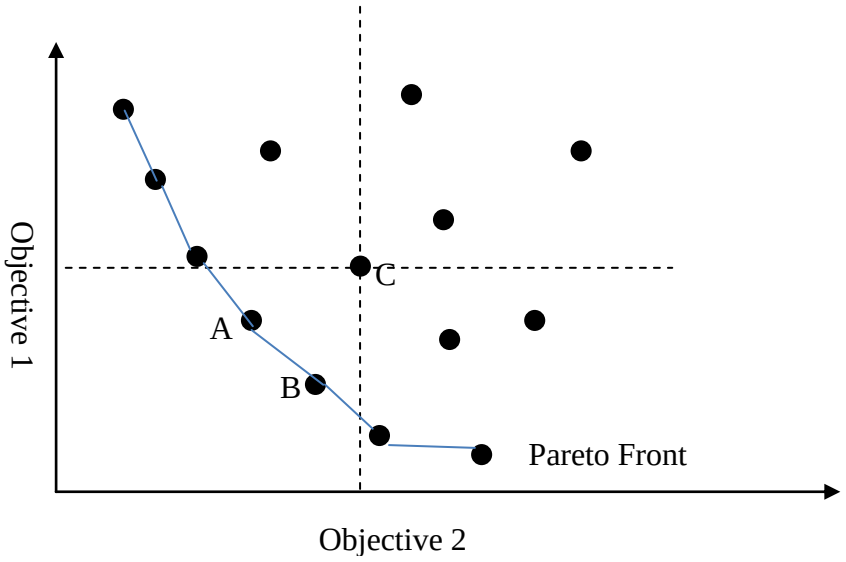

Figure 2.7: the concept of Pareto dominance.

As figure 2.7 shows, Point C is dominated by points A and B. Point A and B are better than point C both for objective 1 and objective 2. Point A does not dominate point B and point B does not dominate point A, because point A is the best point with respect to the objective 2 compare with point B and point B is the best one with respect to objective 1 compare with point A. In fact, points on the full line are not dominating each other. Hence all the points are located on the full line are non-dominated and possible optimal solutions, and they belong to the non-dominated Pareto solution (Kalyanmoy, 2002).

If the Pareto front solution involves maximisation, the Pareto front has the form show below:

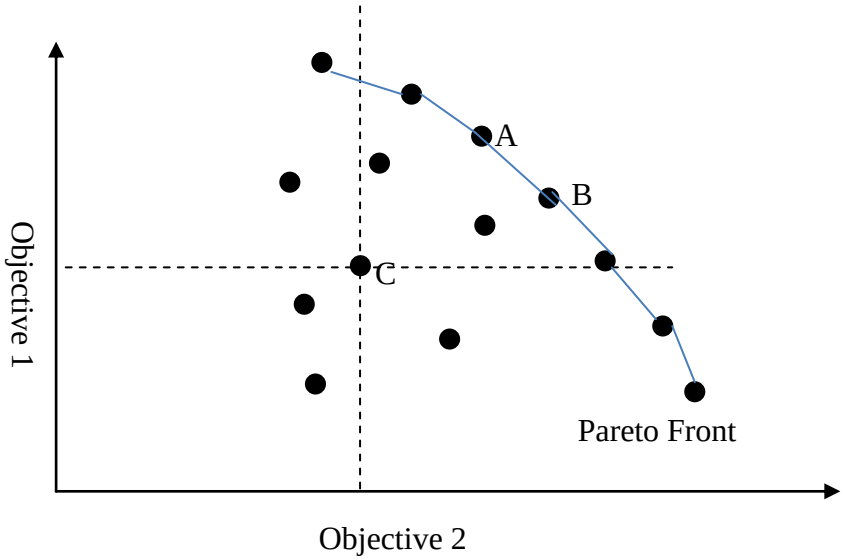

Figure 2.8: the concept of Pareto dominance.

As figure 2.8 shows, Point C is dominated by points A and B. Point A and B are better than point C both for objective 1 and objective 2. Point A does not dominate point B and point B does not dominate point A, because point A is the best point with respect to the objective 2 compare with point B and point B is the best one with respect to objective 1 compared with point A. In fact, points on the full line are not dominating each other. Hence all the points are located on the full line are non-dominated and efficient solutions, and they belong to Pareto solution (Kalyanmoy, 2002).

2.2.3 Multivariable control system design methods

2.2.3.1 Introduction

The design of multivariable control systems involves the selection of controller matrix parameters and optimizing the controller parameters against a cost function. The cost function could include set-point tracking and interaction and could involve a number of channels each with different variables. The optimisation could also be done using either a single cost function or a multi-objective cost functions.

In order to design digital set-point tracking PI controllers, it requires mathematical models of linear multivariable plants. Both state space and transfer function matrix forms are able to model the linear multivariable systems. The Proportional and Integral controller matrices embodied in the tuneable digital PI controllers introduced by B. Porter and A.H. Jones (Porter and Jones, 1984) is considered firstly. They are similar to those introduced by Davison, Pentinnen and Koivo. The controller uses the structure of the multivariable system to select both the proportional and integral controller structures, and then requires the tuning of *m* controller gains associated with the proportional controller and *m* controller gain associated with the integral controller.

2.2.3.2 Analysis

Consider the state space equation of the open-loop system is:

$$\dot{X} = A \times X + B \times u \quad \text{.....(2.18)}$$

and

$$y = C \times X \quad \text{.....(2.19)}$$

Where $X \in R^n$ is the state vector and $u \in R^m$ is the input vector, $y \in R^m$ is the output vector. The plant transfer function matrix is:

$$G(s) = C(sI_n - A)^{-1}B \quad \text{.....(2.20)}$$

And

$$G(0) = -CA^{-1}B \in R^{m \times m} \quad \text{.....(2.21)}$$

And

$$H(t) = CA^{-1}(e^{At} - I_n)B \in R^{m \times m} \text{ (Porter and Jones, 1984) .....(2.22)}$$

In order to design digital error-actuated set-point tracking PID controller for the open loop steady space equation system, the discrete time system set time $T_T = \{0, T, 2T, \ldots\}$. Then the state and output equation become:

$$X_{k+1} = \phi X_k + \varphi u_k \quad \text{.....(2.23)}$$

And

$$y_k = \Gamma x_k \quad \text{.....(2.24)}$$

Where

$$x_k = x(kT) \in R^n$$

$$u_k = u(kT) \in R^m$$

$$y_k = y(kT) \in R^m$$

$$\phi = \exp(AT)$$

$$\varphi = \int_0^T \exp(AT)Bdt$$

$$\Gamma = C$$

And the T is the sampling period. (Porter, 1982)

The controller equation is:

$$u_k = TK_p e_k + TK_i z_k + K_d(e_k - e_{k-1}) \quad \text{.....(2.25)}$$

Therefore, the closed loop steady space equation becomes:

$$\begin{bmatrix} x_{k+1} \\ z_{k+1} \\ f_{k+1} \end{bmatrix} = \begin{bmatrix} \phi - T\varphi K_p - \varphi K_d \Gamma & T\varphi K_i & -\varphi K_d \\ -T\Gamma & I_m & 0 \\ -\Gamma & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ z_k \\ f_k \end{bmatrix} + \begin{bmatrix} T\varphi K_p + \varphi K_d \\ TI_m \\ I_m \end{bmatrix} v \quad \text{.....(2.26)}$$

And

$$y_k = \begin{bmatrix} \Gamma & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ z_k \\ f_k \end{bmatrix} \quad \text{.....(2.27)}$$

where $e_k = v - y_k \in R^m$ is the error vector, $f_k = e_{k-1} \in R^m$ is the stored error vector, $v \in R^m$ is the set-point input vector, the digital integral of error vector

$$z_k = z_{k-1} + Te_{k-1} \in R^m \quad \text{.....(2.28)}$$

And the controller parameter

$$K_p = H^{-1}(T)\pi \in R^{m \times m} \quad \text{.....(2.29)}$$

$$K_i = G^{-1}(0)\varepsilon \in R^{m \times m} \quad \text{.....(2.30)}$$

$$K_d = H^{-1}(T)\Delta \in R^{m \times m} \quad \text{.....(2.31)}$$

where

$$\pi = diag\{\pi_1, \pi_2, \ldots, \pi_m\} = \begin{bmatrix} \pi_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \pi_m \end{bmatrix} \quad \text{.....(2.32)}$$

$$\varepsilon = diag\{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m\} = \begin{bmatrix} \varepsilon_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \varepsilon_m \end{bmatrix} \quad \text{.....(2.33)}$$

$$\Delta = diag\{\Delta_1, \Delta_2, \ldots, \Delta_m\} = \begin{bmatrix} \Delta_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \Delta_m \end{bmatrix} \quad \text{.....(2.34)}$$

are positive diagonal tuning matrices which chosen by the designer to achieve satisfactory closed-loop control system performance. Moreover, the three positive diagonal tuning matrices can be designed by the Genetic algorithm, and $\pi_1, \pi_2, \ldots, \pi_m$, $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m$ and $\Delta_1, \Delta_2, \ldots, \Delta_m$ are the parameters designed by Genetic algorithm. However, in this thesis, only PI controllers are considered, so only $\pi_1, \pi_2, \ldots, \pi_m$ and $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m$ need to be designed, and the total number of parameters is 2m.

Because Genetic algorithm can design any number of parameters, the Genetic algorithm can design the whole controller matrix which is

$$u_k = K_p e_k + K_i z_k \quad \text{.....(2.35)}$$

Where

$$z_k = z_{k-1} + Te_{k-1} \in R^m \quad \text{.....(2.36)}$$

$$K_p = \begin{bmatrix} k_{p11} & \ldots & k_{p1n} \\ \vdots & \ddots & \vdots \\ k_{pm1} & \cdots & k_{pmn} \end{bmatrix} \quad \text{.....(2.37)}$$

and

$$K_i = \begin{bmatrix} k_{i11} & \cdots & k_{i1n} \\ \vdots & \ddots & \vdots \\ k_{im1} & \cdots & k_{imn} \end{bmatrix} \quad \text{.....(2.38)}$$

where $k_{p11} \cdots k_{pmn}$ and $k_{i11} \cdots k_{imn}$ are the parameters for the proportional controller and the integral controller, in this case there are m^2 parameters for the proportional

controllers and m^2 parameters for the integral controllers need to be chosen. To compare with the previous method the total number of parameters is $2m$, the GA will have to search for a larger number of parameters, so this method will take longer time to coverage but should give better results.

Because binary numbers are used in genetic algorithm not real number, but the parameter ranges are chosen using real numbers. Therefore, the genetic algorithm needs to decode and scale the real number parameter range into binary number parameter range. Therefore, if the parameter range is too big, and the number of binary bit is not long enough, the accuracy of the decoded and scaled binary number will be poor. So the parameter range should be chosen very carefully by the designer. There are two examples to show the parameter accuracy after decode and scale in two ranges.

For example, in GA, each parameter could be coded by a 10 bits binary number, the parameter range is from 0 to 1024. So the decode and scale the binary number will be:

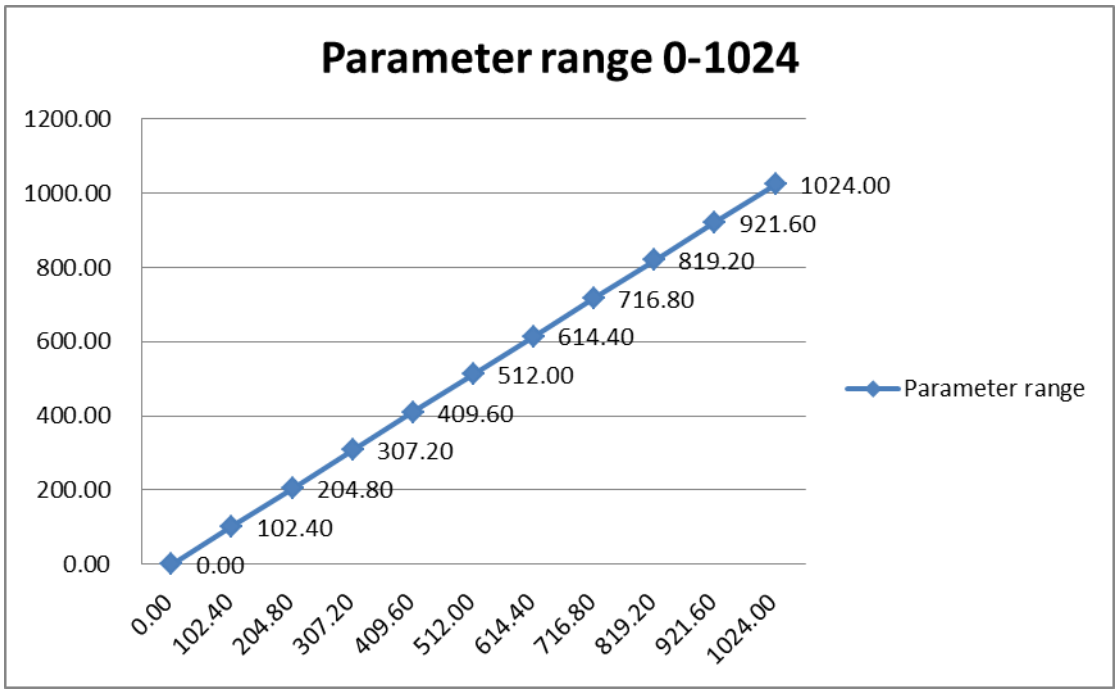

Figure 2.9: The decode and scale between binary number and real number

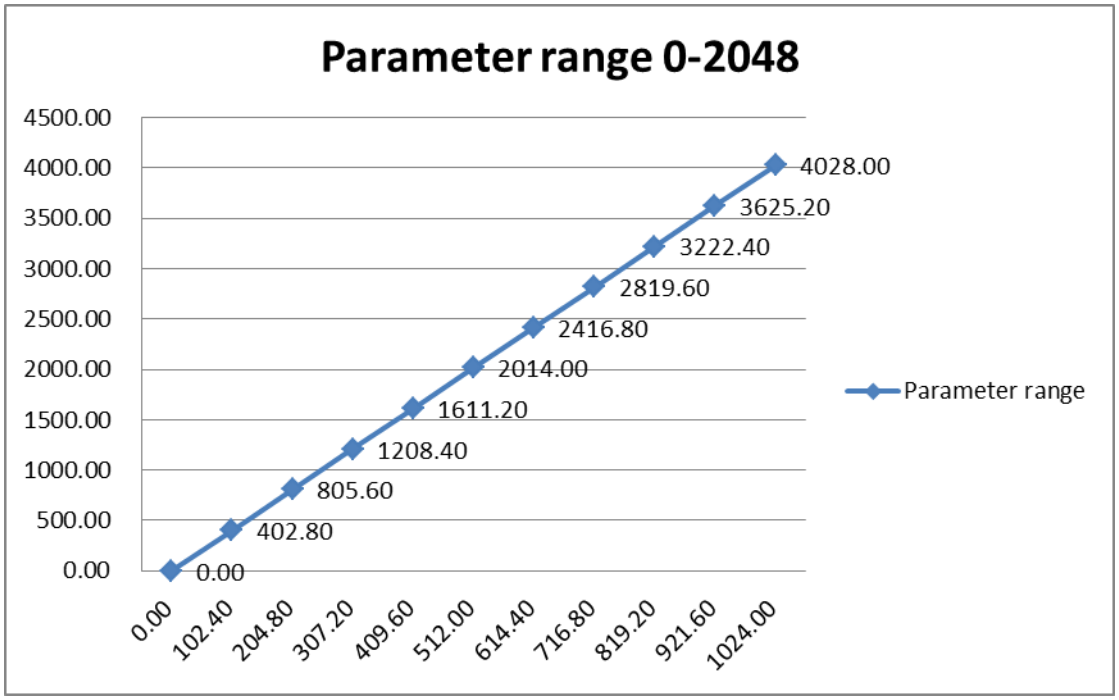

Figure 2.10: The decode and scale between binary number and real number

As figure 2.9 and 2.10 shows above, if GA code parameter in 10 bits binary number, that means there are 1024 points in the parameter range. If the parameter range is from 0 to 1024, binary number can scale the parameter range in 1024 point, that is 1,2,3…1024 integer numbers. The difference between each number is one, and all the number between are missing. If the parameter range is from 0 to 2048, binary number can scale the parameter range in 1024 point, that is 2,4,6…2048 even integer numbers. The difference between each number is two, and all the number between are missing. Therefore, if the parameter range is increased and the number of binary number bit are not changed; the parameter will lose accuracy. Therefore, the range of parameter should be narrow enough and the true optimal solution should lie inside the range.

To demonstrate the effect of parameter range select, the following minimum phase system is chosen as an example for the parameter range variation:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{s+1} & \frac{1}{s+3} \\ \frac{1}{s+1} & \frac{1}{s+1} \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \text{.....(2.39)}$$

Where y_1 and y_2 are the system output; u_1 and u_2 are the system controller input. This system have PI controller:

$$u_k = K_p e_k + K_i z_k \quad \text{.....(2.40)}$$

Where

$$z_k = z_{k-1} + Te_{k-1} \in R^m \quad \text{.....(2.41)}$$

The steady space equation is:

$$\dot{X} = A \times X + B \times u$$

$$\dot{X} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \times X + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \times u \quad \text{.....(2.42)}$$

and

$$y = C \times X = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \times X \quad \text{.....(2.43)}$$

Because this is minimum phase system, then the PI controller can be calculated by:

$$K_p = H^{-1}(T)\Delta = (C \times B)^{-1} \times \pi = \left[\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \right]^{-1} \times \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix}$$

$$= \begin{bmatrix} 0.25 & 0.25 \\ 0.25 & 0.25 \end{bmatrix} \times \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix}$$

$$K_i = G^{-1}(0)\varepsilon = \left[-CA^{-1}B \right]^{-1} \times \varepsilon$$

$$= \left[-\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}^{-1} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \right]^{-1} \times \begin{bmatrix} \varepsilon_1 & 0 \\ 0 & \varepsilon_2 \end{bmatrix}$$

$$= \begin{bmatrix} 1.4999 & -0.5 \\ -1.4999 & 1.5 \end{bmatrix} \times \begin{bmatrix} \varepsilon_1 & 0 \\ 0 & \varepsilon_2 \end{bmatrix}$$

To compare the GA convergence speed for different parameter ranges, the genetic algorithm was used to tune in the same situation but with different parameter ranges. In this example, GA is choosing Δ_1 Δ_2 ε_1 and ε_2, and each parameters range is zero to

one and the GA is run for 5000 generations. Then the parameter range was change to zero to 10, and then the GA was re-run for same number of generations. Figure 2.11 shows the ISE against the number of generation with different parameter range.

The graph below is the plot:

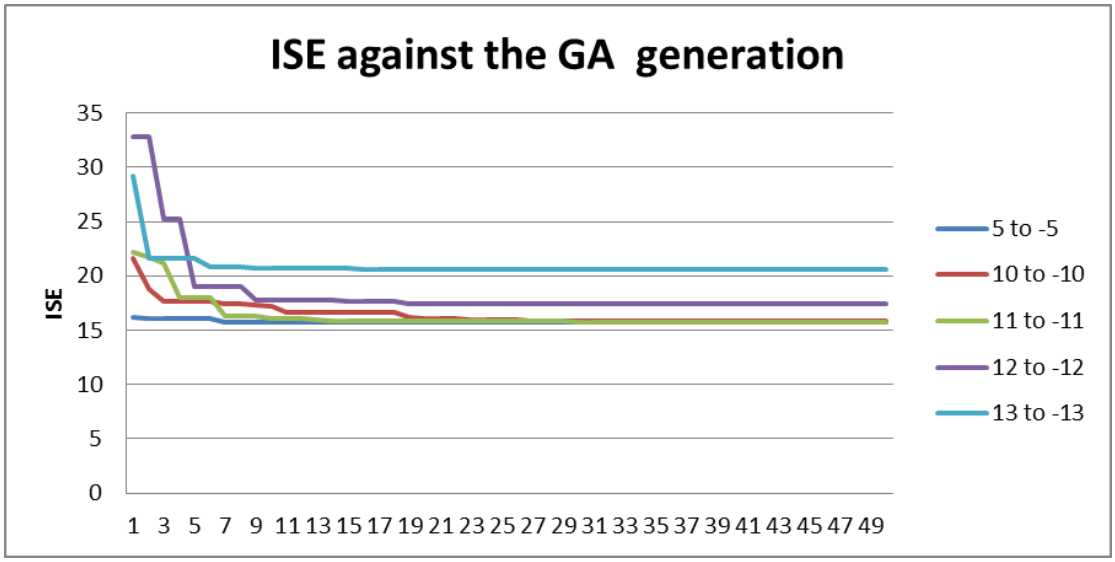

Figure 2.11: the ISE against the number of generation with different parameter range

As above the result shows, if the parameter range is too wide, genetic algorithm program will take a longer time to converge. Therefore, the right range of parameter is very important to speed up the convergence of the genetic algorithm.

When designing multivariable controllers not only the range of parameters is important, but the sign of each of the parameters has to be chosen as well. This is because in a multivariable system the sign of the controller parameters is not easily evident. When design decoupling controllers as proposed by Porter and Jones, the parameter sign can be chosen by the sign of H^{-1} and $G^{-1}(0)$. Moreover, because the diagonal tuning matrices π, ε and Δ are positive, so the sign of them are easy to choose. The parameter range size can be chosen by plus and minus 10% to 20% of H^{-1} and $G^{-1}(0)$ multiply by the diagonal tuning matrices π, ε and Δ. If this is done then the range of parameter should be well suited for running the GA program.

2.3 Single objective genetic algorithm multivariable control system design

In this chapter, an asymptotically stable plant is used as example, and $\alpha = 2,3,4$. This is interesting because the system's zero location is depend on the value of α.The asymptotically stable plant is two input and two output system, and the transfer function is:

$$G(s) = \begin{bmatrix} \frac{1}{s+1} & \frac{\alpha}{s+3} \\ \frac{1}{s+1} & \frac{1}{s+1} \end{bmatrix} \quad \text{....(2.44)}$$

- When $\alpha = 2$, the square multivariable system is non-minimum phase system, because the poles and zeros of the system are in the right hand place;
- when $\alpha = 3$, the square multivariable system is functionally uncontrollable, because when $\alpha = 3$, the zero is at the origin;
- when $\alpha = 4$, the square multivariable system is minimum phase system, because the zeros is in the left hand place.

2.3.1 The asymptotically stable plant α=2 (Non-minimum phase)

The asymptotically stable plant when $\alpha = 2$ is chosen, and the two inputs and two outputs closed loop control system has open loop transfer function

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} g_{11}(s) & g_{12}(s) \\ g_{21}(s) & g_{22}(s) \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{s+1} & \frac{2}{s+3} \\ \frac{1}{s+1} & \frac{1}{s+1} \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \text{.....(2.45)}$$

Therefore, the state space equation is given by:

$$\dot{X} = A \times X + B \times u$$

$$\dot{X} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \times X + \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \times u \quad \text{.....(2.46)}$$

and

$$y = C \times X = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \times X \quad \text{.....(2.47)}$$

And the system input $u = k_p \times e(t) + k_i \times \int_0^t e(t)dt$ .....(2.48)

Where

$e = v - y$ .....(2.49)

v is the set point and y is the system output

The controller sign is given by

$K_p = inv(CB) \times \Delta$ .....(2.50)

and

$K_i = (G(0))^{-1} \times \varepsilon$ .....(2.51)

Where

$$\Delta = \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} \quad \text{.....(2.52)}$$

And

$$\varepsilon = \begin{bmatrix} \varepsilon_1 & 0 \\ 0 & \varepsilon_2 \end{bmatrix} \quad \text{.....(2.53)}$$

And the Simulink block diagram is shown in Figure 2.12:

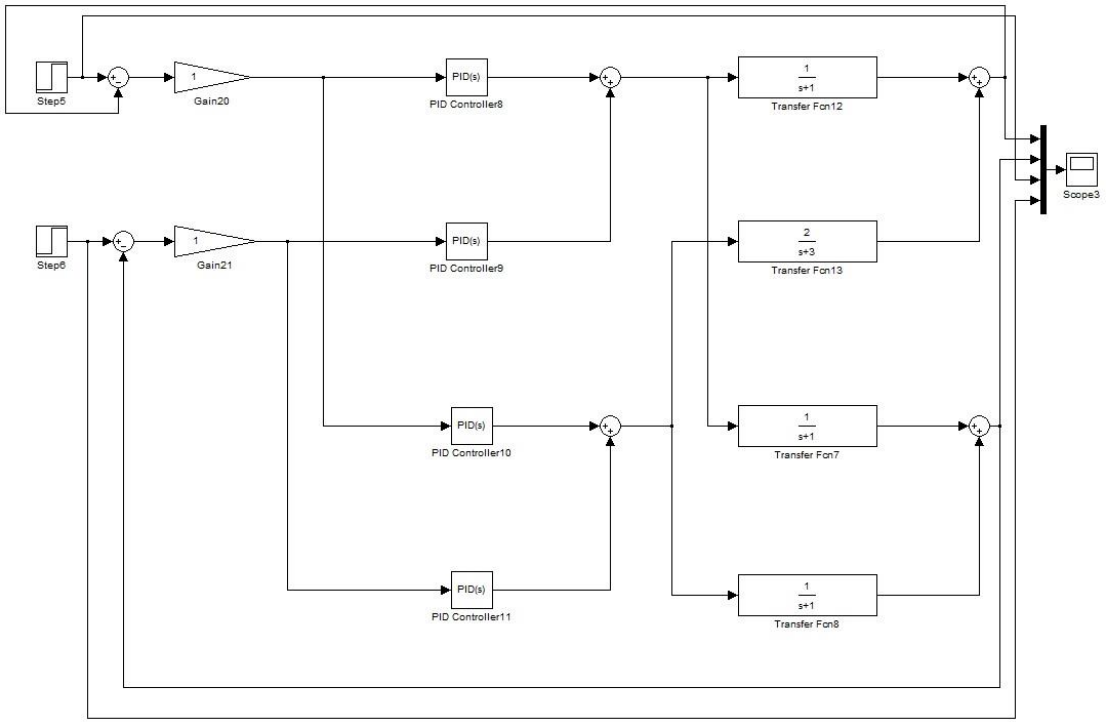

Figure 2.12: Square multivariable control system transfer function

2.3.1.1 Single cost function genetic algorithm top design the diagonal tuning matrix

Because, the asymptotically stable plant when $\alpha = 2$ is non-minimum phase system, the initial decoupling approach of Porter and Jones does not work. For this reason the approach of Davidson will be adopted where both proportional and integral controller structure are based on $(G(0))^{-1}$. Therefore, $K_p = (G(0))^{-1}$ instead it. Therefore, the Kp is calculated by:

$$K_p = (G(0))^{-1} \times \Delta =$$

$$= \left(-\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}^{-1} \times \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \right)^{-1} \times \Delta \quad \text{.....(2.54)}$$

$$= \begin{bmatrix} 3 & -2 \\ -3 & 3 \end{bmatrix} \times \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix}$$

And

$$K_i = (G(0))^{-1} \times \varepsilon =$$

$$= \left(- \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}^{-1} \times \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \right)^{-1} \times \varepsilon \quad \text{.....(2.55)}$$

$$= \begin{bmatrix} 3 & -2 \\ -3 & 3 \end{bmatrix} \times \begin{bmatrix} \varepsilon_1 & 0 \\ 0 & \varepsilon_2 \end{bmatrix}$$

In this controller the parameter of the proportional and integral controller structures are fixed and there are four tuning parameters π_1, π_2, $\varepsilon_1 and \varepsilon_2$ to be chosen by GA. All the four parameters ranges are:

$$\Delta_1 = [0 \quad 1]$$

$$\Delta_2 = [0 \quad 1]$$

$$\varepsilon_1 = [0 \quad 1]$$

$$\varepsilon_2 = [0 \quad 1]$$

When the single cost function GA has applied, the Max is chosen as 100, the parameter range movement code is used, and the cost function minimizes the set point tracking property plus the interaction ISE on the other loop for both channels. The genetic algorithm was run for 5000 generations with a population size of 100, the following values were then obtained from the genetic algorithm:

The four parameters are:

$$\Delta = \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} = \begin{bmatrix} 0.5914 & 0 \\ 0 & 0.5758 \end{bmatrix} \text{.....(2.56)}$$

And

$$\varepsilon = \begin{bmatrix} \varepsilon_1 & 0 \\ 0 & \varepsilon_2 \end{bmatrix} = \begin{bmatrix} 0.3607 & 0 \\ 0 & 0.5992 \end{bmatrix} \text{.....(2.57)}$$

Therefore, the total PI controllers are:

$$K_p = \begin{bmatrix} 3 & -2 \\ -3 & 3 \end{bmatrix} \times \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} = \begin{bmatrix} 3 & -2 \\ -3 & 3 \end{bmatrix} \times \begin{bmatrix} 0.5914 & 0 \\ 0 & 0.5758 \end{bmatrix} = \begin{bmatrix} 1.7742 & -1.1515 \\ -1.7742 & 1.7273 \end{bmatrix} \quad \text{.....(2.58)}$$

$$K_i = \begin{bmatrix} 3 & -2 \\ -3 & 3 \end{bmatrix} \times \begin{bmatrix} \varepsilon_1 & 0 \\ 0 & \varepsilon_2 \end{bmatrix} = \begin{bmatrix} 3 & -2 \\ -3 & 3 \end{bmatrix} \times \begin{bmatrix} 0.3607 & 0 \\ 0 & 0.5992 \end{bmatrix} = \begin{bmatrix} 1.0821 & -1.1984 \\ -1.0821 & 1.7977 \end{bmatrix} \quad \text{.....(2.59)}$$

The system performance is calculated using an ISE, and each ISE_{ij} where i is the number of the output and j is the number of the input.

And the output of the system is shown below:

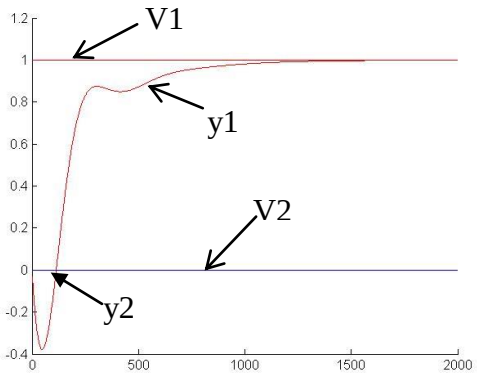

Figure 2.13: the output 1 of GA design $\Delta_1, \Delta_2, \varepsilon_1 and \varepsilon_2$ with the set-point changing and interaction

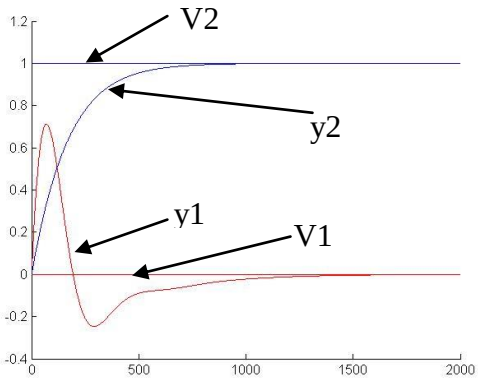

Figure 2.14: the output 2 of GA design $\Delta_1, \Delta_2, \varepsilon_1 and \varepsilon_2$ with the set-point changing and interaction

Where V1 is the set point 1; V2 is the set point 2; y1 is the output 1, y2 is the output 2.

As the above graph shows, the ISE (Integral Square of Error) is considered for each output and it is calculated by:

$$ISE_1 = \varpi_1 \times ISE_{11} + \varpi_2 \times ISE_{21}$$

And

$$ISE_2 = \varpi_3 \times ISE_{12} + \varpi_4 \times ISE_{22}$$

In this design, the weight factor ϖ_i are all equal to one. ISE_{11} is the ISE of output 1 for a change in set-point 1. ISE_{21} is the interaction ISE of output 2 due to change in

set-point 1. ISE_{22} is the ISE of output 2 for a change in set-point 2, and ISE_{12} is the interaction ISE of output 1 due to change in set-point 2. Therefore, the ISE for single GA design controller are:

The $ISE_1 = 227.533773421518$

The $ISE_2 = 142.977733012244$

The total cost is equal to

$$ISE_{total} = ISE_1 + ISE_2 = 227.533773421518 + 142.977733012244$$
$$= 370.5115064337620$$

2.3.1.2 Single cost function GA design the whole controller parameters

It is possible or the GA to design the whole controller, which is given by:

$$K_p = \begin{bmatrix} k_{p1} & k_{p2} \\ k_{p3} & k_{p4} \end{bmatrix} \quad \text{.....(2.60)}$$

And

$$K_i = \begin{bmatrix} k_{i1} & k_{i2} \\ k_{i3} & k_{i4} \end{bmatrix} \quad \text{.....(2.61)}$$

Therefore, $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} are the parameters need to be design by GA.

When the single cost function GA has applied, the Max is chosen as 300, the parameter range movement code is used, and the cost function minimizes the set point tracking property plus the interaction ISE on the other loop for both channels. The genetic algorithm was run for 5000 generations with a population size of 100, the parameter range movement code has also been included, and the cost function minimizes the sum of ISE for the two outputs. The range of each parameters are:

$k_{p1} = [2.01 \quad 2.41]$

$k_{p2} = [-0.81 \quad -0.41]$

$k_{p3} = [-1.38 \quad -0.98]$

$k_{p4} = [1.56 \quad 1.96]$

$$k_{i1} = \begin{bmatrix} 1.74 & 2.14 \end{bmatrix}$$

$$k_{i2} = \begin{bmatrix} -0.24 & 0.16 \end{bmatrix}$$

$$k_{i3} = \begin{bmatrix} -0.99 & -0.59 \end{bmatrix}$$

And

$$k_{i4} = \begin{bmatrix} 1.44 & 1.84 \end{bmatrix}$$

When the single cost function GA has applied, the following values were then obtained from the genetic algorithm:

$$K_p = \begin{bmatrix} 2.3007 & -0.6064 \\ -1.0720 & 1.8792 \end{bmatrix} \quad \text{.....(2.62)}$$

$$K_i = \begin{bmatrix} 1.9427 & 0.0483 \\ -0.8118 & 1.6490 \end{bmatrix} \quad \text{.....(2.63)}$$

And the output of the system is shown below:

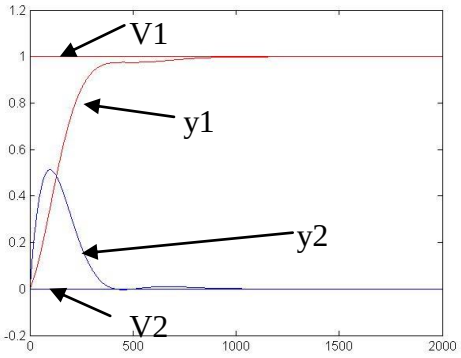

Figure 2.15: the output 1 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction

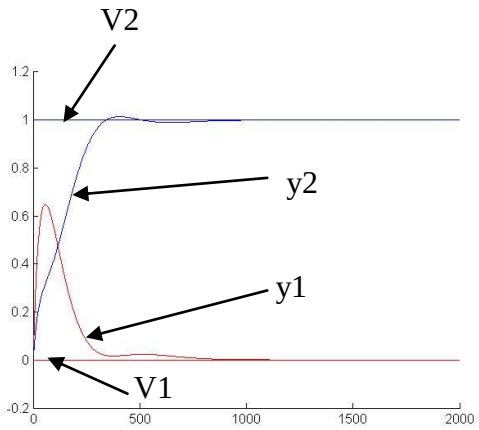

Figure 2.16: the output 2 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction

Where V1 is the set point 1; V2 is the set point 2; y1 is the output 1, y2 is the output 2.

As the figure 2.15 and 2.16 shows, the ISE (Integral Square of Error) is considered for each output is calculated by:

$$ISE_1 = \varpi_1 \times ISE_{11} + \varpi_2 \times ISE_{21}$$

And

$$ISE_2 = \varpi_3 \times ISE_{12} + \varpi_4 \times ISE_{22}$$

In this design, the weight factor ϖ_i are all equal to one. ISE_{11} is the ISE of output 1 for a change in set-point 1. ISE_{21} is the interaction ISE of output 2 due to change in set-point 1. ISE_{22} is the ISE of output 2 for a change in set-point 2, and ISE_{12} is the interaction ISE of output 1 due to change in set-point 2. Therefore, the ISE for single GA design controller are

The $ISE_1 = 134.919606128765$

The $IEE_2 = 119.496152596949$

The total cost is equal to

$$ISE_{total} = ISE_1 + ISE_2 = 134.919606128765 + 119.496152596949$$
$$= 254.4157587257140$$

A comparison of the performance of the designed tuning approach or full parameter tuning approach is show in table 2.1:

	ISE1 (Combine ISE for change in set-point 1)	ISE2 (Combine ISE for change in set-point 2)	ISE_1+ISE_2
Genetic algorithm design for tuning the diagonal matrix	227.5338	142.9777	370.51151
Genetic algorithm design for tuning the full controller	134.9196	119.4962	254.41576

Table 2.1: the ISE comparison between genetic algorithm diagonal matrix design and full controller design

As the table 2.1 shows, the ISE_1, ISE_2 and total ISE from the full controller design are better than the diagonal matrix design. This means that when the single cost function genetic algorithm is used to design the full controller it is obtains a better result than when the single cost function genetic algorithm is used to design the diagonal tuning matrix.

However, as figure above shows, the ISE of the system output is small, but the interaction is very large and it may be too big, so it will have to be reduced by re-designing the controller. There are two ways to reduced it. The first is adding a

weighting factor on the interaction. The second technique is where the cost is increased if the interaction is greater than the designed amount and the "designed amount" is choosing by the designer.

2.3.1.3 Single cost function GA design with weight factor

Because the multivariable controller parameter design can have lead to large there may be a requirement of reduce the amount of interaction. There are two ways to reduce the interaction: the first one is adding a weighting factor on the interaction ISE; the second is using a constraint on the interaction. For the weighting factor on interaction method, the weighting factor is chosen by the designer. Because this design method uses weight factor to reduce the interaction, the GA will pay more attention on interaction. The weight factor is choosing by the designer and there are three situations: the first one is the weight factor is less than one, in this case, the GA will pay more attention on the set point tracking output because of reducing the weight of interaction ISE; the second one is the weight factor is equal to one, this result will be the same as the pervious full controller parameter design; the third one is the weight factor is greater than one, in this case, the GA will pay more attention on the interaction because of increase the weight of interaction ISE.

2.3.1.4 Single cost function GA design with weight factor less than one

In this design situation, the weighting factor added for the interaction will be less than one. So the two weight factor will be chosen as 0.1 for example, two cost function become:

$$ISE_1 = ISE_{11} + 0.1 \times ISE_{21} \quad \text{.....(2.64)}$$

And

$$ISE_2 = 0.1 \times ISE_{12} + ISE_{22} \quad \text{.....(2.65)}$$

In this design, ISE_{11} is the ISE of output 1 for a change in set-point 1. ISE_{21} is the interaction ISE of output 2 due to change in set-point 1. ISE_{22} is the ISE of output 2

for a change in set-point 2, and ISE_{12} is the interaction ISE of output 1 due to change in set-point 2.

Then the single cost function GA applied again, and the GA is design the full controller parameters which are k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3} and k_{i4}. The Max is chosen as 300, the parameter range movement code is used, and the cost function minimizes the set point tracking property plus the interaction ISE on the other loop for both channels. The genetic algorithm was run for 5000 generations with a population size of 100. The range of each parameters are:

$$k_{p1} = \begin{bmatrix} 3.42 & 3.82 \end{bmatrix}$$

$$k_{p2} = \begin{bmatrix} 0.33 & 0.73 \end{bmatrix}$$

$$k_{p3} = \begin{bmatrix} -1.09 & -0.69 \end{bmatrix}$$

$$k_{p4} = \begin{bmatrix} 2.52 & 2.92 \end{bmatrix}$$

$$k_{i1} = \begin{bmatrix} 2.70 & 3.10 \end{bmatrix}$$

$$k_{i2} = \begin{bmatrix} 0.51 & 0.91 \end{bmatrix}$$

$$k_{i3} = \begin{bmatrix} 0.13 & 0.53 \end{bmatrix}$$

And

$$k_{i4} = \begin{bmatrix} 2.41 & 2.81 \end{bmatrix}$$

When the single cost function GA has applied, the following values were then obtained from the genetic algorithm:

$$K_p = \begin{bmatrix} 3.7264 & 0.6511 \\ -0.8793 & 2.8344 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 2.9690 & 0.8281 \\ 0.4293 & 2.6717 \end{bmatrix}$$

And the output of the system is shown below:

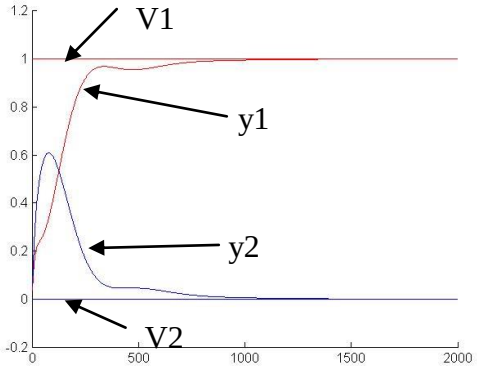

Figure 2.17: the output 1 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction

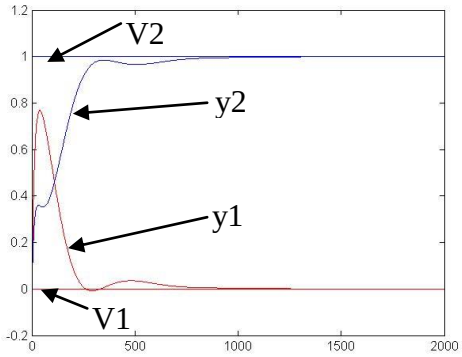

Figure 2.18: the output 2 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction

Where V1 is the set point 1; V2 is the set point 2; y1 is the output 1, y2 is the output 2.

As the above graph shows, the ISE (Integral Square of Error) is considered for each output is calculated by:

$$ISE_1 = ISE_{11} + ISE_{21}$$

And

$$ISE_2 = ISE_{12} + ISE_{22}$$

In this design, ISE_{11} is the ISE of output 1 for a change in set-point 1. ISE_{21} is the interaction ISE of output 2 due to change in set-point 1. ISE_{22} is the ISE of output 2 for a change in set-point 2, and ISE_{12} is the interaction ISE of output 1 due to change in set-point 2. Therefore, the ISE for single GA design controller are

The $ISE_1 = 125.270722921406$

The $ISE_2 = 114.773915309684$

The total cost is equal to

$$ISE_{total} = ISE_1 + ISE_2 = 125.270722921406 + 114.773915309684$$
$$= 240.0446382310900$$

2.3.1.5 Single cost function GA design with weight factor greater than one

In this design situation, the weight factor added for the interaction will be greater than one. So the two weight factor will be chosen as 10 for example, two cost function become

$$ISE_1 = ISE_{11} + 10 \times ISE_{21} \quad \text{.....(2.66)}$$

And

$$ISE_2 = 10 \times ISE_{12} + ISE_{22} \quad \text{.....(2.67)}$$

In this design, ISE_{11} is the ISE of output 1 for a change in set-point 1. ISE_{21} is the interaction ISE of output 2 due to change in set-point 1. ISE_{22} is the ISE of output 2 for a change in set-point 2, and ISE_{12} is the interaction ISE of output 1 due to change in set-point 2.

Then the single cost function GA applied again, and the GA is design the full controller parameters which are $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4}. The Max is chosen as 300, the parameter range movement code is used, and the cost function minimizes the set point tracking property plus the interaction ISE on the other loop for both channels. The genetic algorithm was run for 5000 generations with a population size of 100. The range of each parameters are:

$$k_{p1} = [1.62 \quad 2.02]$$

$$k_{p2} = [-1.41 \quad -1.01]$$

$$k_{p3} = [-1.87 \quad -1.47]$$

$$k_{p4} = [0.90 \quad 1.30]$$

$$k_{i1} = [1.32 \quad 1.72]$$

$$k_{i2} = [-0.75 \quad -0.35]$$

$$k_{i3} = [-1.79 \quad -1.39]$$

and

$$k_{i4} = [0.85 \quad 1.25]$$

When the single cost function GA has applied, the following values were then obtained from the genetic algorithm:

$$K_p = \begin{bmatrix} 1.7208 & -1.1061 \\ -1.6092 & 0.9993 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 1.5401 & -0.5515 \\ -1.5173 & 1.0491 \end{bmatrix}$$

And the output of the system is shown below:

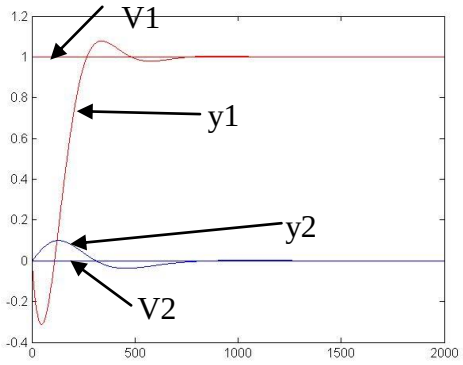

Figure 2.19: the output 1 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction

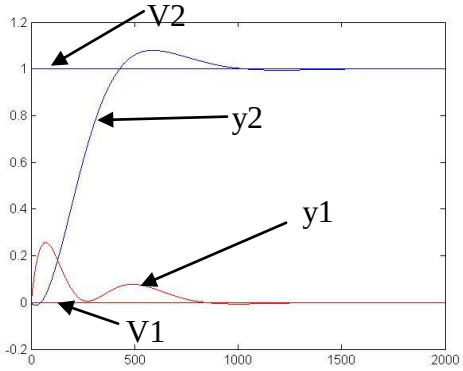

Figure 2.20: the output 2 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction

Where V1 is the set point 1; V2 is the set point 2; y1 is the output 1, y2 is the output 2.

As the above graph shows, the ISE (Integral Square of Error) is considered for each output is calculated by:

$$ISE_1 = ISE_{11} + ISE_{21}$$

And

$$ISE_2 = ISE_{12} + ISE_{22}$$

In this design, ISE_{11} is the ISE of output 1 for a change in set-point 1. ISE_{21} is the interaction ISE of output 2 due to change in set-point 1. ISE_{22} is the ISE of output 2 for a change in set-point 2, and ISE_{12} is the interaction ISE of output 1 due to change in set-point 2. Therefore, the ISE for single GA design controller are

The $ISE_1 = 205.620932183817$

The $ISE_2 = 183.074238140837$

The total cost is equal to

$ISE_{total} = ISE_1 + ISE_2 = 205.620932183817 + 183.074238140837$
$= 388.6951703246540$

As the weight factor results shows, the weight factor could affect the GA design. If the weight factor is less than one, the set point tracking will be faster but the interaction will also increase. If the weight factor is greater than one, the interaction decrease but the set point tracking performance is reduced. Therefore, there is an extra method could added into design that is constraint.

2.3.1.6 Single cost function GA design with weight factor under constraint

Because this design method uses a weight factor and constraint together, only the controller with interaction under a certain value (this is chosen by the designer) will be selected by the GA. When the weight factor is less than one, the GA program will pay more attention on the set point tracking, this will make the set point tracking has best performance and large interactions. If the weight factor is equal to one, the GA program will pay equal attention on the set point tracking and interaction, this will make the set point tracking have less performance than the weight factor less than one, but the interaction will be less than the weight factor less than one case. If the weight factor is greater than one, the GA program will pay more attention on the interaction, this will make the interaction smaller but the set point tracking performance will not be as good.

2.3.1.7 Single cost function GA design with weight factor less than one under constraint

In this design situation, the weight factor added in interaction will be less than one. So the two weight factor will be chosen as 0.1 for example, two cost function become:

$$ISE_1 = ISE_{11} + 0.1 \times ISE_{21} \quad \text{.....(2.68)}$$

And

$$ISE_2 = 0.1 \times ISE_{12} + ISE_{22} \quad \text{.....(2.69)}$$

In this design, ISE_{11} is the ISE of output 1 for a change in set-point 1. ISE_{21} is the interaction ISE of output 2 due to change in set-point 1. ISE_{22} is the ISE of output 2 for a change in set-point 2, and ISE_{12} is the interaction ISE of output 1 due to change in set-point 2.

The constraint is choosing as the interaction overshot will less than 20% as example, this means the interaction overshoot will be less than 20% of the set-point change causing that interaction. This "20%" is choosing by the designer, in this example, if the overshot in interaction is bigger than 20%, then the fitness of this individual is setting to zero. Then the single cost function GA applied again, and the GA is design the full controller parameters which are k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3} and k_{i4}. The Max is chosen as 300, the parameter range movement code is used, and the cost function minimizes the set point tracking property plus the interaction ISE on the other loop for both channels. The genetic algorithm was run for 5000 generations with a population size of 100. The range of each parameters are:

$$k_{p1} = [1.52 \quad 1.92]$$

$$k_{p2} = [-1.31 \quad -0.91]$$

$$k_{p3} = [-1.75 \quad -1.35]$$

$$k_{p4} = [0.68 \quad 1.08]$$

$$k_{i1} = [1.34 \quad 1.74]$$

$$k_{i2} = [-0.69 \quad -0.29]$$

$$k_{i3} = [-1.72 \quad -1.32]$$

and

$$k_{i4} = \begin{bmatrix} 0.85 & 1.25 \end{bmatrix}$$

When the single cost function GA has applied, the following values were then obtained from the genetic algorithm:

$$K_p = \begin{bmatrix} 1.7151 & -1.0586 \\ -1.4681 & 0.8295 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 1.5598 & -0.4972 \\ -1.5421 & 1.1197 \end{bmatrix}$$

And the output of the system is shown below:

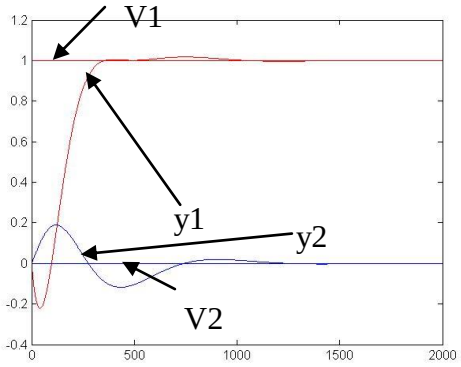

Figure 2.21: the output 1 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction

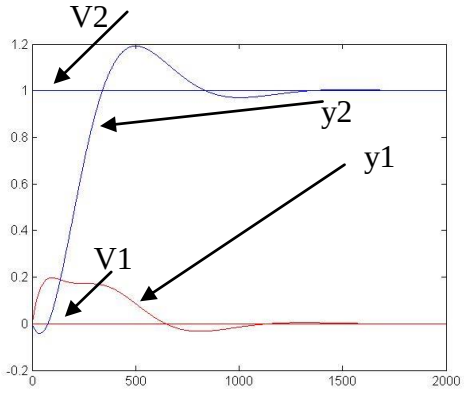

Figure 2.22: the output 2 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction

Where V1 is the set point 1; V2 is the set point 2; y1 is the output 1, y2 is the output 2.

As the above graph shows, the ISE (Integral Square of Error) is considered for each output is calculated by:

$$ISE_1 = ISE_{11} + ISE_{21}$$

And

$$ISE_2 = ISE_{12} + ISE_{22}$$

In this design, ISE_{11} is the ISE of output 1 for a change in set-point 1. ISE_{21} is the interaction ISE of output 2 due to change in set-point 1. ISE_{22} is the ISE of output 2

for a change in set-point 2, and ISE_{12} is the interaction ISE of output 1 due to change in set-point 2. Therefore, the ISE for single GA design controller are

The $ISE_1 = 187.338809225897$

The $ISE_2 = 195.004732363057$

The total cost is equal to

$ISE_{total} = ISE_1 + ISE_2 = 187.338809225897 + 195.004732363057$
$= 382.3435415889540$

2.3.1.8 Single cost function GA design with weight factor equal to one under constraint

In this design situation, the weight factor added for the interaction will be equal to one. So the two cost function become:

$ISE_1 = ISE_{11} + 1 \times ISE_{21}$ .....(2.70)

And

$ISE_2 = 1 \times ISE_{12} + ISE_{22}$ .....(2.71)

In this design, ISE_{11} is the ISE of output 1 for a change in set-point 1. ISE_{21} is the interaction ISE of output 2 due to change in set-point 1. ISE_{22} is the ISE of output 2 for a change in set-point 2, and ISE_{12} is the interaction ISE of output 1 due to change in set-point 2.

The constraint is choosing as the interaction overshot will less than 20% as example, this "20%" is choosing by the designer, in this example, if the overshot in interaction is bigger than 20%, then the fitness of this individual is setting to zero. Then the single cost function GA applied again, and the GA is design the full controller parameters which are $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4}. The Max is chosen as 300, the parameter range movement code is also used, and the cost function minimizes the set point tracking property plus the interaction ISE on the other loop for both channels. The genetic algorithm was run for 5000 generations with a population size of 100. The range of each parameters are:

$k_{p1} = [1.36 \quad 1.76]$

$$k_{p2} = \begin{bmatrix} -1.01 & -0.61 \end{bmatrix}$$

$$k_{p3} = \begin{bmatrix} -1.49 & -1.09 \end{bmatrix}$$

$$k_{p4} = \begin{bmatrix} 0.50 & 0.90 \end{bmatrix}$$

$$k_{i1} = \begin{bmatrix} 1.18 & 1.58 \end{bmatrix}$$

$$k_{i2} = \begin{bmatrix} -0.64 & -0.24 \end{bmatrix}$$

$$k_{i3} = \begin{bmatrix} -1.56 & -1.16 \end{bmatrix}$$

and

$$k_{i4} = \begin{bmatrix} 0.74 & 1.14 \end{bmatrix}$$

When the single cost function GA has applied, the following values were then obtained from the genetic algorithm:

$$K_p = \begin{bmatrix} 1.5376 & -1.0015 \\ -1.4194 & 0.8225 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 1.4520 & -0.3561 \\ -1.3112 & 0.9392 \end{bmatrix}$$

And the output of the system is shown below:

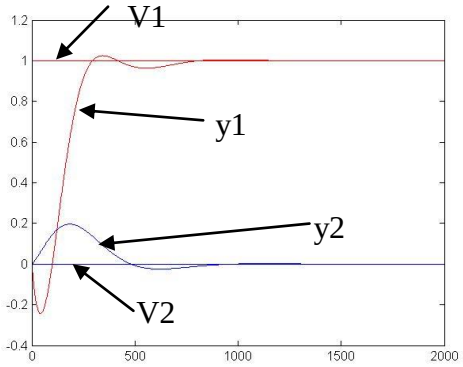

Figure 2.23: the output 1 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction

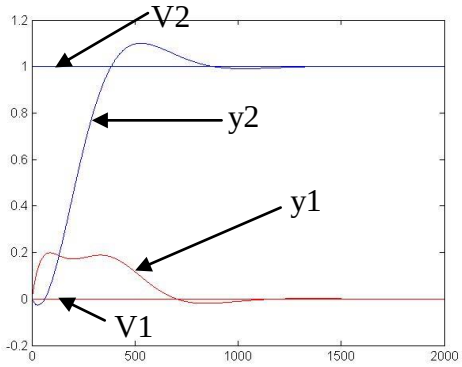

Figure 2.24: the output 2 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction

Where V1 is the set point 1; V2 is the set point 2; y1 is the output 1, y2 is the output 2.

As the above graph shows, the ISE (Integral Square of Error) is considered for each output is calculated by:

$$ISE_1 = ISE_{11} + ISE_{21}$$

And

$$ISE_2 = ISE_{12} + ISE_{22}$$

In this design, ISE_{11} is the ISE of output 1 for a change in set-point 1. ISE_{21} is the interaction ISE of output 2 due to change in set-point 1. ISE_{22} is the ISE of output 2 for a change in set-point 2, and ISE_{12} is the interaction ISE of output 1 due to change in set-point 2. Therefore, the ISE for single GA design controller are

The $ISE_1 = 189.056620022091$

The $ISE_2 = 189.747355101328$

The total cost is equal to

$$ISE_{total} = ISE_1 + ISE_2 = 189.056620022091 + 189.747355101328$$
$$= 378.8039751234190$$

2.3.1.9 Single cost function GA design with weight factor greater than one under constraint

In this design situation, the weight factor added for the interaction will be greater than one. So the two weight factor will be chosen as 10 for example, two cost function become:

$$ISE_1 = ISE_{11} + 10 \times ISE_{21} \quad \text{.....(2.72)}$$

And

$$ISE_2 = 10 \times ISE_{12} + ISE_{22} \quad \text{.....(2.73)}$$

In this design, ISE_{11} is the ISE of output 1 for a change in set-point 1. ISE_{21} is the interaction ISE of output 2 due to change in set-point 1. ISE_{22} is the ISE of output 2 for a change in set-point 2, and ISE_{12} is the interaction ISE of output 1 due to change in set-point 2.

The constraint is choosing as the interaction overshot will less than 20% as example, this “20%” is choosing by the designer, in this example, if the overshot in interaction

is bigger than 20%, then the fitness of this individual is setting to zero. Then the single cost function GA applied again, and the GA is design the full controller parameters which are $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4}. The Max is chosen as 300, the parameter range movement code is also used, and the cost function minimizes the set point tracking property plus the interaction ISE on the other loop for both channels. The genetic algorithm was run for 5000 generations with a population size of 100. The range of each parameters are:

$$k_{p1} = [1.40 \quad 1.80]$$

$$k_{p2} = [-1.31 \quad -0.91]$$

$$k_{p3} = [-1.69 \quad -1.29]$$

$$k_{p4} = [0.74 \quad 1.14]$$

$$k_{i1} = [1.34 \quad 1.74]$$

$$k_{i2} = [-0.87 \quad -0.47]$$

$$k_{i3} = [-1.72 \quad -0.32]$$

and

$$k_{i4} = [0.97 \quad 1.37]$$

When the single cost function GA has applied, the following values were then obtained from the genetic algorithm:

$$K_p = \begin{bmatrix} 1.5009 & -1.0558 \\ -1.4292 & 0.8394 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 1.4402 & -0.7339 \\ -1.4193 & 1.2225 \end{bmatrix}$$

And the output of the system is shown below:

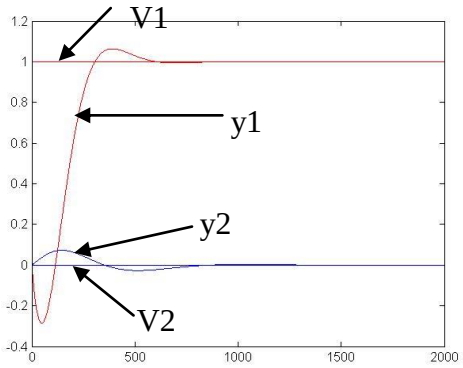

Figure 2.25: the output 1 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction

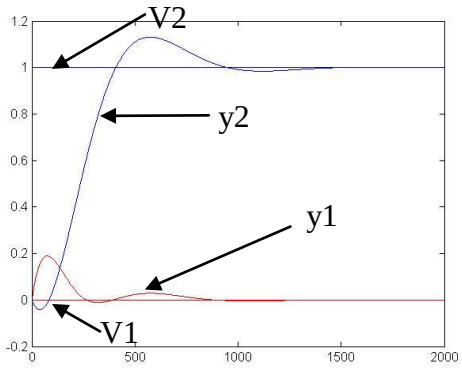

Figure 2.26: the output 2 of GA design $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} with the set-point changing and interaction

Where V1 is the set point 1; V2 is the set point 2; y1 is the output 1, y2 is the output 2.

As the above graph shows, the ISE (Integral Square of Error) is considered for each output is calculated by:

$$ISE_1 = ISE_{11} + ISE_{21}$$

And

$$ISE_2 = ISE_{12} + ISE_{22}$$

In this design, ISE_{11} is the ISE of output 1 for a change in set-point 1. ISE_{21} is the interaction ISE of output 2 due to change in set-point 1. ISE_{22} is the ISE of output 2 for a change in set-point 2, and ISE_{12} is the interaction ISE of output 1 due to change in set-point 2. Therefore, the ISE for single GA design controller are

The $ISE_1 = 212.743171848532$

The $ISE_2 = 203.804732545061$

The total cost is equal to

$$ISE_{total} = ISE_1 + ISE_2 = 212.743171848532 + 203.804732545061$$
$$= 416.5479043935930$$

To compare with those different design methods, the set point tracking and interaction output under different design setting are plot below:

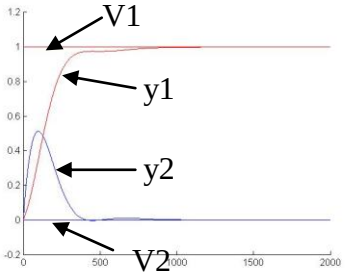

Figure 2.27: the output 1 and 2 of GA design full controller

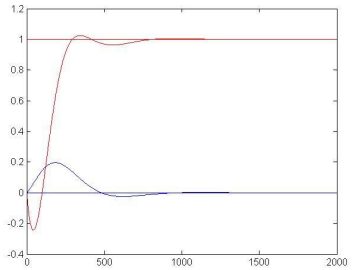

Figure 2.29: the output 1 and 2 of GA design full controller with weight factor=1 and constraint on interaction

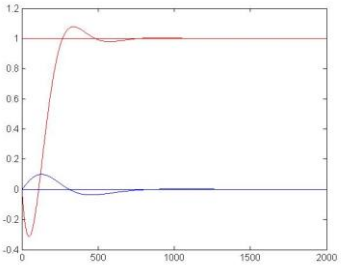

Figure 2.31: the output 1 and 2 of GA design full controller with weight factor=10

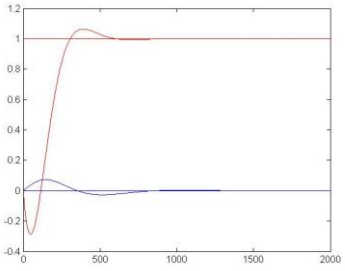

Figure 2.28: the output 1 and 2 of GA design full controller with weight factor=10 and constraint on interaction

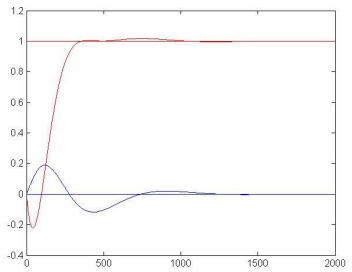

Figure 2.30: the output 1 and 2 of GA design full controller with weight factor=0.1 and constraint on interaction

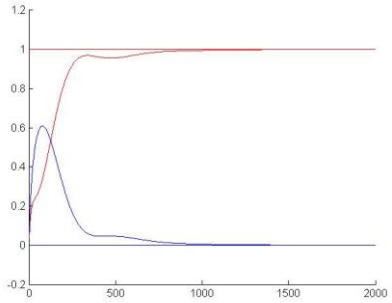

Figure 2.32: the output 1 and 2 of GA design full controller with weight factor=0.1

Where V1 is the set point 1; V2 is the set point 2; y1 is the output 1, y2 is the output 2.

If the results in different design method compare each other, the ISE and overshot are show below:

	ISE1 (Combine ISE for change in set-point 1)	ISE2 (Combine ISE for change in set-point 2)	Overshot in output 2 when set point 1 change	Overshot in output 1 when set point 2 change
GA improve decoupling method	227.5	142.9	0	71%
GA design the full parameters	134.92	119.5	51%	65%
GA design the full parameters with weight factor less than one	125.3	114.8	61%	77%
GA design the full parameters with weight factor greater than one	205.6	183.1	10%	26%
GA design the full parameters with weight factor less than one with constraint of 20%	187.3	195	19%	20%
GA design the full parameters with weight factor equal to one with constraint of 20%	189.1	189.7	20%	20%
GA design the full parameters with weight factor greater than one with constraint of 20%	212.7	203.8	7%	19%

Table 2.2: the ISE comparison and overshoot

As table 2.2 shows, the B. Porter and A.H. Jones design method (system performance baseline) and genetic algorithm design for all the controller parameters, both can design the multivariable square system. However, the B. Porter and A.H. Jones design method only guarantee the system output stable but not the optimal output performance. When compared with the genetic algorithm design where all the controller parameters are selected by the genetic algorithm this system shows much better performance. However, the system output may give rise to large interactions. There are two techniques to reduce the interactions. The first one is adding a weighting factor into the interaction cost function. But the weight factor is very hard

to choose, if the weight factor is choosing correctly, then the system output interaction will reduced to below the designed level. The second technique is adding an interaction constraint. The cost function adjustment with this technique is simple and all ways guarantee the interaction to be less than a designed level with system output showing performance under this interaction constraint. This is achieved by making the fitness equal to zero if the overshot is bigger than a chosen value. With the constraint added into the interaction, the genetic algorithm can take a very long time to get started, because too many individuals have more than the designed maximum interaction.

2.3.2 The asymptotically stable plant when α=3 (functionally uncontrollable)

In some cases the multivariable system to be turned by the genetic algorithm might be functionally uncontrollable. In such cases the genetic algorithm cannot design a suitable controller for the system. In order to demonstrate this the system describable on section 2.3 is consider when $\alpha = 3$.

The asymptotically stable plant when α=3 is chosen, and the two inputs and two outputs closed loop control system has open loop transfer function

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} g_{11}(s) & g_{12}(s) \\ g_{21}(s) & g_{22}(s) \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \dfrac{1}{s+1} & \dfrac{3}{s+3} \\ \dfrac{1}{s+1} & \dfrac{1}{s+1} \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \text{.....(2.74)}$$

Therefore, the steady space equation is

$$\dot{X} = A \times X + B \times u$$

$$\dot{X} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \times X + \begin{bmatrix} 1 & 0 \\ 0 & 3 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \times u \text{.....(2.75)}$$

And

$$y = C \times X = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \times X \text{.....(2.76)}$$

And the system input $u = k_p \times e(t) + k_i \times \int_0^t e(t)dt$ .....(2.77)

Where

$e = v - y$

v is the set point and y is the system output

The controller sign is given by

$K_p = inv(CB) \times \Delta$ .....(2.78)

and

$K_i = (G(0))^{-1} \times \varepsilon$ .....(2.79)

Where

$$\Delta = \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix}$$ .....(2.80)

And

$$\varepsilon = \begin{bmatrix} \varepsilon_1 & 0 \\ 0 & \varepsilon_2 \end{bmatrix}$$ .....(2.81)

And the Simulink as below:

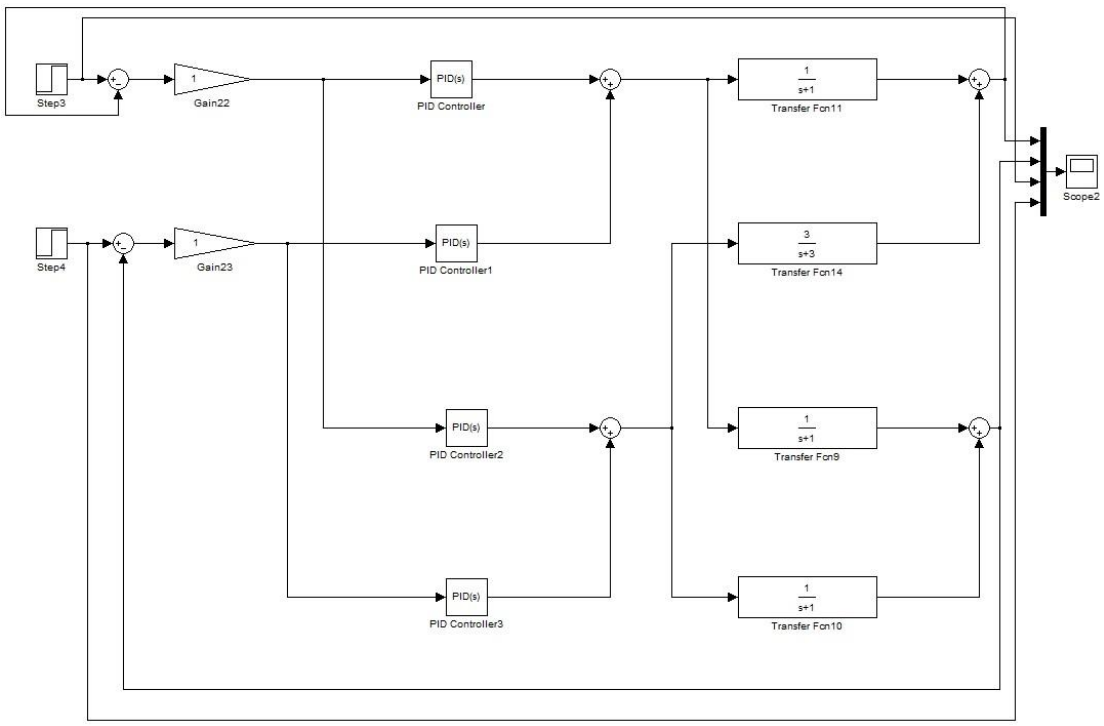

Figure 2.33: Square multivariable control system transfer function

Therefore, the overall transfer function

$$G(s) = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} = \begin{bmatrix} \frac{1}{s+1} & \frac{3}{s+3} \\ \frac{1}{s+1} & \frac{1}{s+1} \end{bmatrix} \quad \text{.....(2.82)}$$

And the system input

$$u = k_p \times e(t) + k_i \times \int_0^t e(t)dt \quad \text{.....(2.83)}$$

where

$$k_p = \begin{bmatrix} k_{p11} & k_{p12} \\ k_{p21} & k_{p22} \end{bmatrix} \quad \text{.....(2.84)}$$

And

$$k_i = \begin{bmatrix} k_{i11} & k_{i12} \\ k_{i21} & k_{i22} \end{bmatrix} \quad \text{.....(2.85)}$$

and $k_{p11}, k_{p12}, k_{p21}, k_{p22}, k_{i11}, k_{i12}, k_{i21}$ and k_{i22} are the parameters need be to design by genetic algorithm.

When the single cost function GA has applied, the Max is chosen as 500, the parameter range movement code is also used, and the cost function minimizes the set point tracking property plus the interaction ISE on the other loop for both channels. The genetic algorithm was run for 5000 generations with a population size of 100. The range of each parameters are:

$k_{p1} = [0.66 \quad 2.66]$

$k_{p2} = [-0.85 \quad 1.15]$

$k_{p3} = [-1.14 \quad 0.86]$

$k_{p4} = [0.12 \quad 2.12]$

$k_{i1} = [-9.74 \quad -7.74]$

$k_{i2} = [-0.10 \quad 1.90]$

$k_{i3} = [7.91 \quad 9.91]$

and

$$k_{i4} = \begin{bmatrix} -0.45 & 1.55 \end{bmatrix}$$

When the single cost function GA has applied, the following values were then obtained from the genetic algorithm:

$$K_p = \begin{bmatrix} 1.6551 & 0.1527 \\ -0.1388 & 1.1181 \end{bmatrix}$$

$$K_i = \begin{bmatrix} -8.7365 & 0.8991 \\ 8.9124 & 0.5529 \end{bmatrix}$$

And the output of the system is shown below:

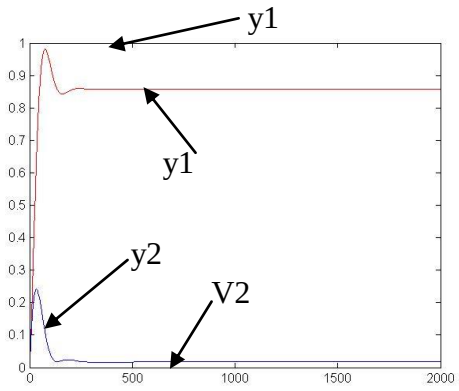

Figure 2.34: the output 1 of GA design with the set-point changing

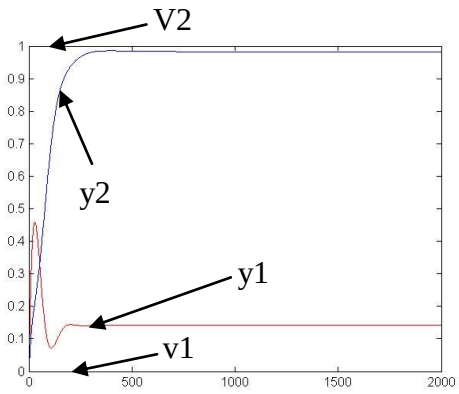

Figure 2.36: the output 2 of GA design with the set-point changing

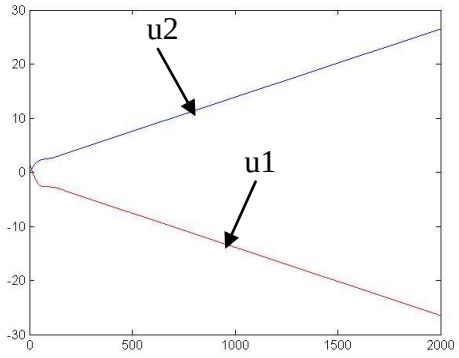

Figure 2.35: the input 1 and input 2 during time

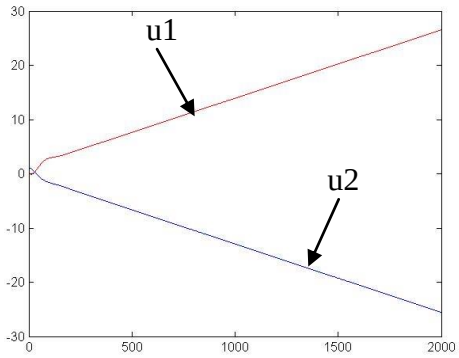

Figure 2.37: the input 1 and input 2 during time

Where V1 is the set point 1; V2 is the set point 2; y1 is the output 1, y2 is the output 2; u1 is the input 1; u2 is the input 2.

As the above graph shows, the ISE (Integral Square of Error) is considered for each output is calculated by:

$$ISE_1 = ISE_{11} + ISE_{21}$$

And

$$ISE_2 = ISE_{12} + ISE_{22}$$

In this design, ISE_{11} is the ISE of output 1 for a change in set-point 1. ISE_{21} is the interaction ISE of output 2 due to change in set-point 1. ISE_{22} is the ISE of output 2 for a change in set-point 2, and ISE_{12} is the interaction ISE of output 1 due to change in set-point 2. Therefore, the ISE for single GA design controller are

The $ISE_1 = 61.4601424560732$

The $ISE_2 = 99.8326687598742$

The total cost is equal to

$$ISE_{total} = ISE_1 + ISE_2 = 61.4601424560732 + 99.8326687598742$$
$$= 416.5479043935930$$

It should be noticed that in this case, the outputs from the system do not track, and the interactions do not goes to zero. In such cases the final value of ISE cost function will converge to a constant value. However, In this case, it will continue to grow as the simulation time is increased. This is because the system is functionally uncontrollable. This is shown by the control input increasing constantly even when the outputs appear to have setting down. This feature should be checked by the simulation and a warning given that the system cannot achieve set point tracking, because it is functionally uncontrollable.

2.3.3 The asymptotically stable plant when α=4 (minimum phase)

The asymptotically stable plant when $\alpha = 4$ is chosen, and the two inputs and two outputs closed loop control system has open loop transfer function

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} g_{11}(s) & g_{12}(s) \\ g_{21}(s) & g_{22}(s) \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{s+1} & \frac{4}{s+3} \\ \frac{1}{s+1} & \frac{1}{s+1} \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \text{.....(2.86)}$$

Therefore, the state space equation is

$$\dot{X} = A \times X + B \times u$$

$$\dot{X} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \times X + \begin{bmatrix} 1 & 0 \\ 0 & 4 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \times u \quad \text{.....(2.87)}$$

and

$$y = C \times X = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \times X \quad \text{.....(2.88)}$$

And the system input $u = k_p \times e(t) + k_i \times \int_0^t e(t)dt$ .....(2.89)

Where

$e = v - y$ .....(2.90)

v is the set point and y is the system output

The controller sign is given by

$K_p = inv(CB) \times \Delta$ .....(2.91)

and

$K_i = (G(0))^{-1} \times \varepsilon$ .....(2.92)

Where

$$\Delta = \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} \quad \text{.....(2.93)}$$

And

$$\varepsilon = \begin{bmatrix} \varepsilon_1 & 0 \\ 0 & \varepsilon_2 \end{bmatrix} \quad \text{.....(2.94)}$$

And the Simulink as below:

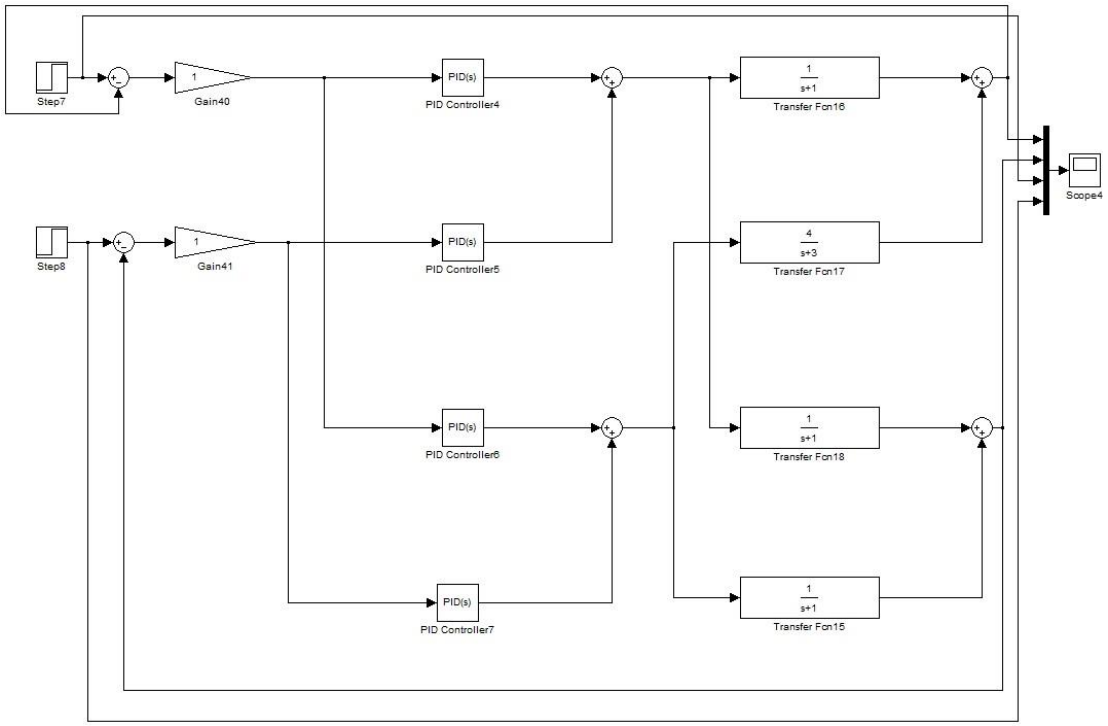

Figure 2.38: Square multivariable control system transfer function

2.3.3.1 Single cost function genetic algorithm design the tuning diagonal matrix

In this case the multivariable controller will be designed by the technique of Porter and Jones. Where the controller structure is chosen from the plan transfer function and set of diagonal controller parameters are tuned.

Therefore, the overall transfer function

$$G(s) = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} = \begin{bmatrix} \dfrac{1}{s+1} & \dfrac{4}{s+3} \\ \dfrac{1}{s+1} & \dfrac{1}{s+1} \end{bmatrix} \quad \text{.....(2.95)}$$

And the system input

$$u = k_p \times e(t) + k_i \times \int_0^t e(t)dt \quad \text{.....(2.96)}$$

where

$$K_p = inv(CA^{-1}(e^{At} - I_n)B) \times \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} \quad \text{.....(2.97)}$$

And

$$K_i = inv(-CA^{-1}B) \times \begin{bmatrix} \varepsilon_1 & 0 \\ 0 & \varepsilon_2 \end{bmatrix} \quad \text{.....(2.98)}$$

and Δ_1, Δ_2, ε_1 and $_2\varepsilon$ are the parameters need be to design by genetic algorithm.

The controller sign is given by

$$u = k_p \times e(t) + k_i \times \int_0^t e(t)dt$$

Where

$$K_p = inv(CA^{-1}(e^{At} - I_n)B) \times \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} \quad \text{.....(2.99)}$$

and

$$K_i = inv(-CA^{-1}B) \times \begin{bmatrix} \varepsilon_1 & 0 \\ 0 & \varepsilon_2 \end{bmatrix} \quad \text{.....(2.100)}$$

Therefore

$$K_p = inv(CA^{-1}(e^{At} - I_n)B) \times \Delta$$

$$K_p = inv\left(\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \times inv\left\{ \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \right\} \times \left(e^{\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \times 0.1} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right) \right) \times \begin{bmatrix} 1 & 0 \\ 0 & 4 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\times \Delta = \begin{bmatrix} 0.1800 & -0.1008 \\ -0.0587 & 0.0317 \end{bmatrix} \times \Delta$$

$$K_p = \begin{bmatrix} 0.1800 & -0.1008 \\ -0.0587 & 0.0317 \end{bmatrix} \times \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} \quad \text{.....(2.101)}$$

and

$$K_i = inv\left(-CA^{-1}B\right) \times \varepsilon$$

$$K_i = inv\left\{ -\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \times inv \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -3 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 4 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \right\}$$

$$\times \begin{bmatrix} \varepsilon_1 & 0 \\ 0 & \varepsilon_2 \end{bmatrix}$$

$$K_i = \begin{bmatrix} -0.0300 & 0.0400 \\ 0.0300 & -0.0300 \end{bmatrix} \times \begin{bmatrix} \varepsilon_1 & 0 \\ 0 & \varepsilon_2 \end{bmatrix} \quad \text{.....(2.102)}$$

So this K_p and K_i are the parameter of the proportional and integral controllers, and this is the start point of the single objective GA running. π_1, π_2, $\varepsilon_1 and \varepsilon_2$ are the four parameters choosing by GA. All the four parameters are positive. All the four parameters ranges are:

$$\Delta_1 = [0 \quad 1]$$

$$\Delta_2 = [0 \quad 1]$$

$$\varepsilon_1 = [0 \quad 1]$$

$$\varepsilon_2 = [0 \quad 1]$$

The constraint is choosing as the interaction overshot will less than 20% as example, this "20%" is choosing by the designer, in this example, if the overshot in interaction is bigger than 20%, then the fitness of this individual is setting to zero. When the single cost function GA has applied, the Max is chosen as 200, the parameter range movement code is used, and the cost function minimizes the set point tracking property plus the interaction ISE on the other loop for both channels. The genetic algorithm was run for 5000 generations with a population size of 100, the following values were then obtained from the genetic algorithm:

The four parameters are:

$$\Delta = \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} = \begin{bmatrix} 1.8201 & 0 \\ 0 & 0.6881 \end{bmatrix} \quad \text{.....(2.103)}$$

And

$$\varepsilon = \begin{bmatrix} \varepsilon_1 & 0 \\ 0 & \varepsilon_2 \end{bmatrix} = \begin{bmatrix} 1.7966 & 0 \\ 0 & 1.9589 \end{bmatrix} \quad \text{.....(2.104)}$$

Therefore, the total PI controllers are:

$$K_p = \begin{bmatrix} 0.1800 & -0.1008 \\ -0.0587 & 0.0317 \end{bmatrix} \times \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix}$$

$$= \begin{bmatrix} 0.1800 & -0.1008 \\ -0.0587 & 0.0317 \end{bmatrix} \times \begin{bmatrix} 1.8201 & 0 \\ 0 & 0.6881 \end{bmatrix} \quad \text{.....(2.105)}$$

$$= \begin{bmatrix} 0.3276 & -0.0693 \\ -0.1068 & 0.0218 \end{bmatrix}$$

$$K_i = \begin{bmatrix} -0.0300 & 0.0400 \\ 0.0300 & -0.0300 \end{bmatrix} \times \begin{bmatrix} \varepsilon_1 & 0 \\ 0 & \varepsilon_2 \end{bmatrix}$$

$$= \begin{bmatrix} -0.0300 & 0.0400 \\ 0.0300 & -0.0300 \end{bmatrix} \times \begin{bmatrix} 1.7966 & 0 \\ 0 & 1.9589 \end{bmatrix} \quad \text{.....(2.106)}$$

$$= \begin{bmatrix} -0.0539 & 0.0783 \\ 0.0539 & -0.0587 \end{bmatrix}$$

And the output of the system is shown below:

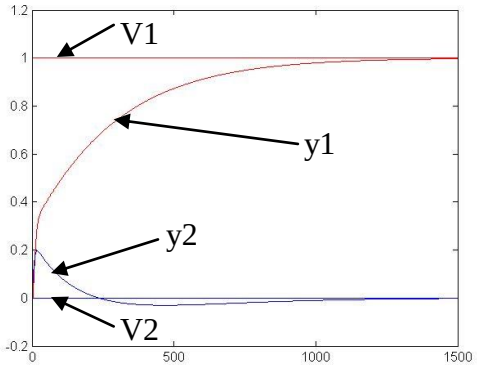

Figure 2.39: the output 1 of GA design the diagonal matrix with the set-point changing

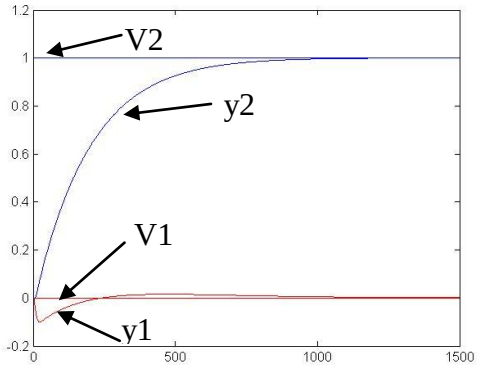

Figure 2.40: the output 2 of GA design the diagonal matrix with the set-point changing

Where V1 is the set point 1; V2 is the set point 2; y1 is the output 1, y2 is the output 2.

As the above graph shows, the ISE (Integral Square of Error) is considered for each output is calculated by:

$$ISE_1 = ISE_{11} + ISE_{21}$$

And

$$ISE_2 = ISE_{12} + ISE_{22}$$

In this design, ISE_{11} is the ISE of output 1 for a change in set-point 1. ISE_{21} is the interaction ISE of output 2 due to change in set-point 1. ISE_{22} is the ISE of output 2 for a change in set-point 2, and ISE_{12} is the interaction ISE of output 1 due to change in set-point 2. Therefore, the ISE for single GA design controller are

The $ISE_1 = 84.062106157534160$

The $ISE_2 = 106.1421321665217$

The total cost is equal to

$$ISE_{total} = ISE_1 + ISE_2 = 84.062106157534160 + 106.1421321665217$$
$$= 190.2042383240559$$

2.3.3.2 Single cost function GA design of all parameters of the controller

Rather than the genetic algorithm design the diagonal matrix, the genetic algorithm can design the entire controller given by:

$$K_p = \begin{bmatrix} k_{p1} & k_{p2} \\ k_{p3} & k_{p4} \end{bmatrix} \quad \text{....(2.107)}$$

And

$$K_i = \begin{bmatrix} k_{i1} & k_{i2} \\ k_{i3} & k_{i4} \end{bmatrix} \quad \text{.....(2.108)}$$

Therefore, $k_{p1}, k_{p2}, k_{p3}, k_{p4}, k_{i1}, k_{i2}, k_{i3}$ and k_{i4} are the parameters need to be design by GA.

The constraint is choosing as the interaction overshot will less than 20% as example, this "20%" is choosing by the designer, in this example, if the overshot in interaction is bigger than 20%, then the fitness of this individual is setting to zero. When the

single cost function GA has applied, the Max is chosen as 50, the parameter range movement code is used, and the cost function minimizes the set point tracking property plus the interaction ISE on the other loop for both channels. The genetic algorithm was run for 5000 generations with a population size of 100. The range of each parameters are:

$$k_{p1} = \left[-0.6973 \quad -1.6973\right]$$

$$k_{p2} = \left[6.4826 \quad 7.4826\right]$$

$$k_{p3} = \left[2.9489 \quad 3.9489\right]$$

$$k_{p4} = \left[-1.4392 \quad -0.4392\right]$$

$$k_{i1} = \left[-3.3571 \quad -2.3571\right]$$

$$k_{i2} = \left[6.2343 \quad 7.2343\right]$$

$$k_{i3} = \left[4.2568 \quad 5.2568\right]$$

and

$$k_{i4} = \left[-4.3609 \quad -3.3609\right]$$

When the single cost function GA has applied, the following values were then obtained from the genetic algorithm:

$$K_p = \begin{bmatrix} -1.1973 & 6.9825 \\ 3.4489 & -0.9392 \end{bmatrix}$$

$$K_i = \begin{bmatrix} -2.8571 & 6.7343 \\ 4.7568 & -3.8609 \end{bmatrix}$$

And the output of the system is shown below:

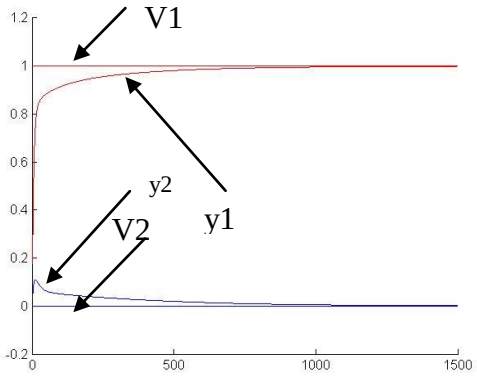

Figure 2.41: the output 1 of GA design the full controller with the set-point changing

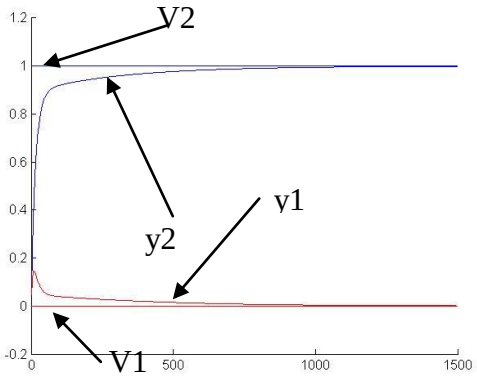

Figure 2.42: the output 2 of GA design the full controller with the set-point changing

Where V1 is the set point 1; V2 is the set point 2; y1 is the output 1, y2 is the output 2.

As the above graph shows, the ISE (Integral Square of Error) is considered for each output is calculated by:

$$ISE_1 = ISE_{11} + ISE_{21}$$

And

$$ISE_2 = ISE_{12} + ISE_{22}$$

In this design, ISE_{11} is the ISE of output 1 for a change in set-point 1. ISE_{21} is the interaction ISE of output 2 due to change in set-point 1. ISE_{22} is the ISE of output 2 for a change in set-point 2, and ISE_{12} is the interaction ISE of output 1 due to change in set-point 2. Therefore, the ISE for single GA design controller are

The $ISE_1 = 8.374956493648856$

The $ISE_2 = 12.717224713386496$

The total cost is equal to

$$ISE_{total} = ISE_1 + ISE_2 = 8.374956493648856 + 12.717224713386496 = 21.092181207035352$$

If this result compared with the diagonal matrix GA design, the ISE and overshot are show below:

	ISE1 (Combine ISE for change in set-point 1)	ISE2 (Combine ISE for change in set-point 2)	ISE_1+ISE_2
GA design with diagonal matrix	84.06210616	106.1421322	190.2042383
GA design with full parameter	8.374956493	12.71722471	21.09218120

Table 2.3: the ISE comparison and overshoot

As the result show, both designs have good system output performance, and full parameter genetic algorithm design provides a better solution. Because this system is minimum phase system, the system output does not have large interaction.

2.4 Multi objective GA

For the single objective GA the weighting factor ϖ_i has to be chosen. If the output units are the same for the all outputs, the ISEs can be added together. If the output units are not the same each ISE cannot be directly added with each other because of scaling issues. This could appear where one output was measured in ℃(temperature) and another output was measured in bar (pressure) or Newton (force). This means the sum of ISE and all outputs is difficult to assess. However, multi objective GA can solve this problem.

Compared with single objective GA, the multi objective GA searches for a family of optimal solution with each individual part of the multi-objective function is optimised in its own right. Therefore, there is no single best set of parameters to be found, but a set of Pareto solutions in the parameter range set. By using non-dominated solution the multi-objective genetic algorithm optimised an individual cost function, but all other cost function are also considered by the dominance condition.

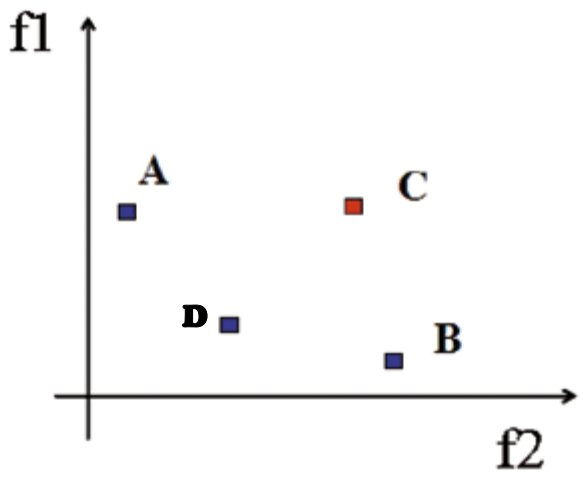

Figure 2.43: the concept of Pareto dominance.

As figure 2.40 shows, Point C is dominated by points A and B. Points A, B and D are better than point C both for objective f1 and objective f2. Point A does not dominate point B and point B does not dominate point A because point A is the best point with respect to the objective f2 and point B is the best one with respect to objective f1. Point A dominate point D in objective f2 only, point B dominate point D in objective f1 only. Hence points A, B and D are non-dominated and efficient solutions.

2.4.1.1 Multi-objective genetic algorithm design for the asymptotically stable plant α =2 (Non-minimum phase)

For the multi objective Pareto front method, this example chosen was the same as that in chapter 2.3, the two inputs and two outputs example are chosen as below:

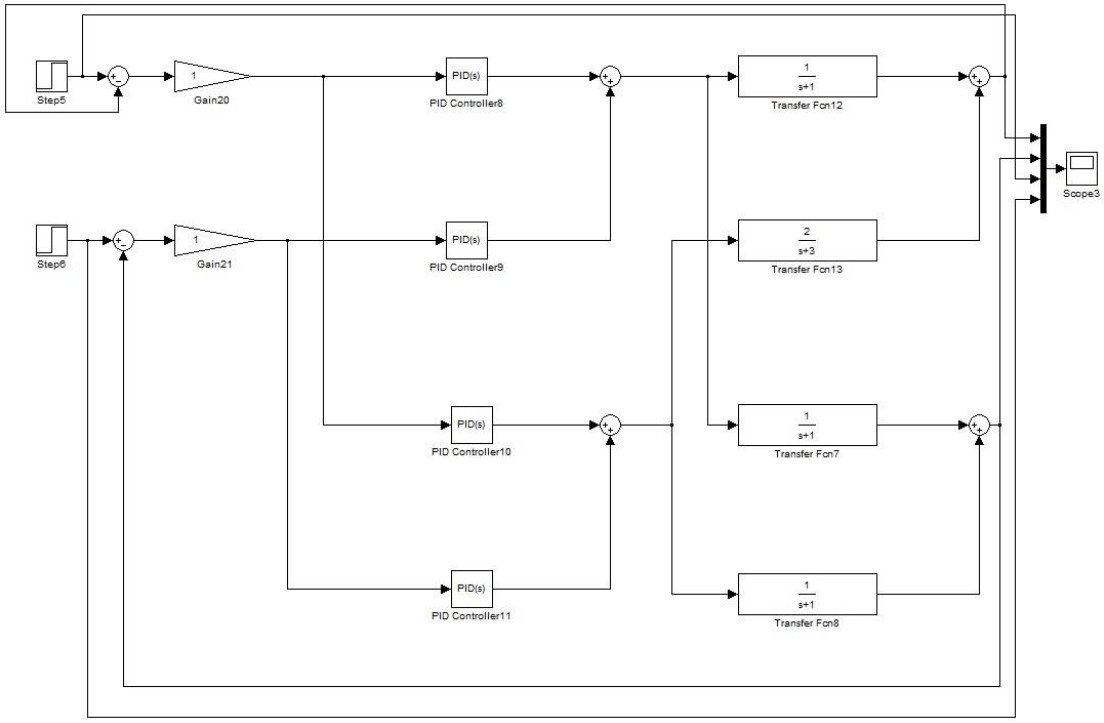

Figure 2.44: Square multivariable control system transfer function

Therefore, the overall transfer function

$$G(s)=\begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix}=\begin{bmatrix} \dfrac{1}{s+1} & \dfrac{2}{s+3} \\ \dfrac{1}{s+1} & \dfrac{1}{s+1} \end{bmatrix} \quad \text{.....(2.109)}$$

And the system input

$$u=k_p\times e(t)+k_i\times\int_0^t e(t)dt \quad \text{.....(2.110)}$$

where

$$k_p=\begin{bmatrix} k_{p11} & k_{p12} \\ k_{p21} & k_{p22} \end{bmatrix} \quad \text{.....(2.111)}$$

And

$$k_i=\begin{bmatrix} k_{i11} & k_{i12} \\ k_{i21} & k_{i22} \end{bmatrix} \quad \text{.....(2.112)}$$

and $k_{p11}, k_{p12}, k_{p21}, k_{p22}, k_{i11}, k_{i12}, k_{i21}$ and k_{i22} are the parameters need be to design by genetic algorithm.

When the multi-objective GA has applied, the two objectives are the set point tracking property plus the interaction ISE on the other loop for both channels. The genetic algorithm was run for 1000 generations with a population size of 1000. The range of each parameters are:

$$k_{p1} = [2.01 \quad 2.41]$$

$$k_{p2} = [-0.81 \quad -0.41]$$

$$k_{p3} = [-1.38 \quad -0.98]$$

$$k_{p4} = [1.56 \quad 1.96]$$

$$k_{i1} = [1.74 \quad 2.14]$$

$$k_{i2} = [-0.24 \quad 0.16]$$

$$k_{i3} = [-0.99 \quad -0.59]$$

And

$$k_{i4} = [1.44 \quad 1.84]$$

However, there are two objectives which are the two outputs ISE:

$$Objective_1 = ISE_1 = ISE_{11} + ISE_{21}$$

And

$$Objective_2 = ISE_2 = ISE_{12} + ISE_{22}$$

In this design, ISE_{11} is the ISE of output 1 for a change in set-point 1. ISE_{21} is the interaction ISE of output 2 due to change in set-point 1. ISE_{22} is the ISE of output 2 for a change in set-point 2, and ISE_{12} is the interaction ISE of output 1 due to change in set-point 2.

The system will have big interaction with normal GA design (as chapter 2.3.1shows), and this large interaction is not desirable for the system performance. Therefore, the interaction constraint has been included in the multi-objective GA as well. The idea with the constraint is if the interaction is bigger than 20% (this percentage is choosing by the designer) this individual will not put into the population.

The Pareto front plot could be drawn in graph below:

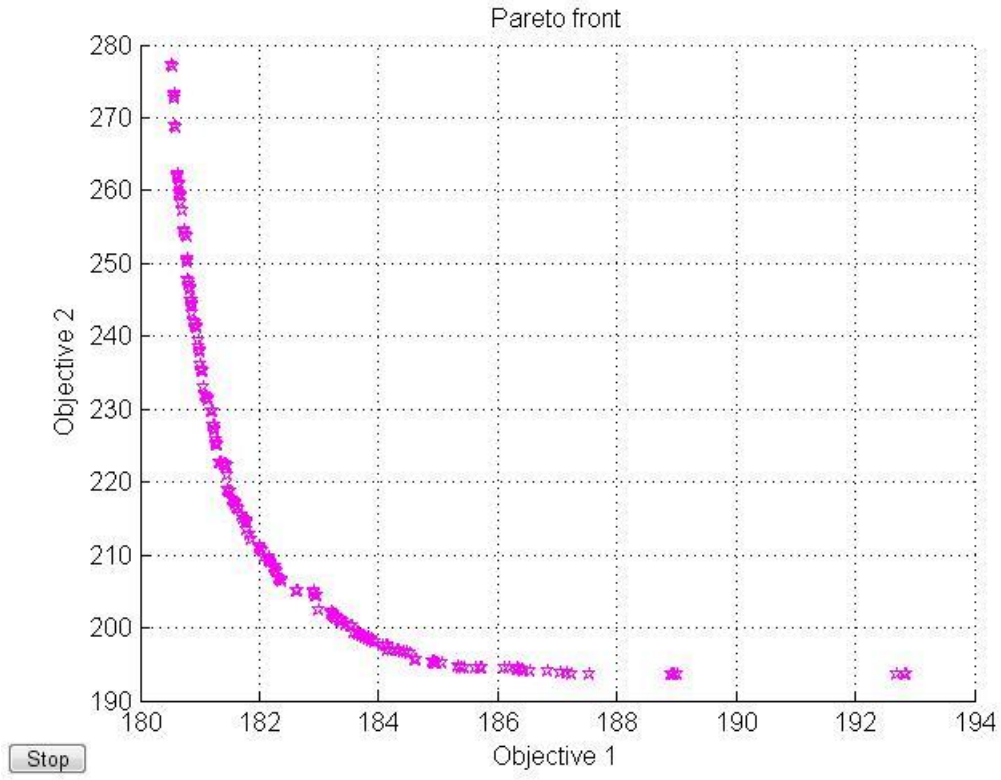

Figure 2.45: the Pareto front plot for example in chapter 2.4

As the Pareto front plot shows, the single objective GA result from chapter 2.3.1.8: the $ISE_1 = 189.1$ and the $ISE_2 = 189.7$ is very closed to one of the optimal solutions. The benefit of the Pareto front method is that the decision maker can choose the different objective combinations for different requirements on ISE1 and ISE2.

The three interesting results are selected from the family of non-dominated Pareto front solutions which are the minimum ISE_1, ISE_2 and the sum of ISE are shown in table 2.4:

	Objective 1 for Combine ISE for change in set-point 1	Objective 2 for Combine ISE for change in set-point 2	sum of two objective
1	182.2758165	253.3179327	435.5937492
2	187.1730229	198.0417338	385.2147568

Table 2.4: the part results from the non-dominated Pareto front solutions

The result for the minimum objective$_1$

The ISE$_1$ value from multi-objective genetic algorithm is little smaller than the single cost function genetic algorithm, ISE$_2$ and the sum of ISE from the multi-objective genetic algorithm are similar with single cost function genetic algorithm, the multi-objective genetic algorithm could provide the optimal performance in ISE$_1$ and others are still good.

As the table show above, the red marked ISE$_1$ channel value is the lowest one, then the controller for this design are:

$$K_p = \begin{bmatrix} 1.5460 & -0.8559 \\ -1.3502 & 0.7237 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 1.3607 & -0.4990 \\ -1.3050 & 0.9481 \end{bmatrix}$$

And the output of the system is shown below:

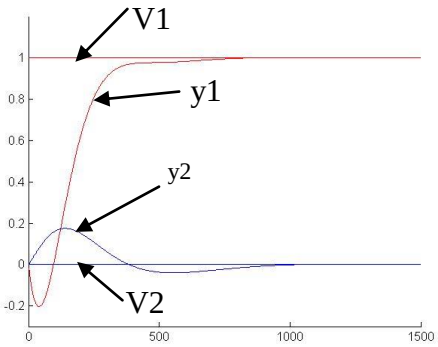

Figure 2.46: the output 1 of GA design the full controller with the set-point changing

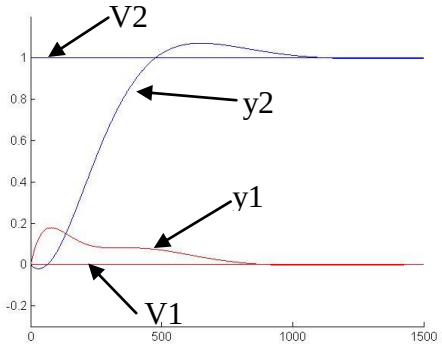

Figure 2.47: the output 2 of GA design the full controller with the set-point changing

Where V1 is the set point 1; V2 is the set point 2; y1 is the output 1, y2 is the output 2.

The result for the minimum objective$_2$

The ISE$_2$ value from multi-objective genetic algorithm is little smaller than the single cost function genetic algorithm, ISE$_1$ and the sum of ISE from the multi-objective genetic algorithm are similar with single cost function genetic algorithm, the

multi-objective genetic algorithm could provide the optimal performance in ISE_2 and others are still good.

As the table show above, the red marked ISE_2 channel value is the lowest one, then the controller for this design are:

$$K_p = \begin{bmatrix} 1.5421 & -0.8537 \\ -1.3493 & 0.6819 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 1.3605 & -0.4875 \\ 1.3041 & 1.0088 \end{bmatrix}$$

And the output of the system is shown below:

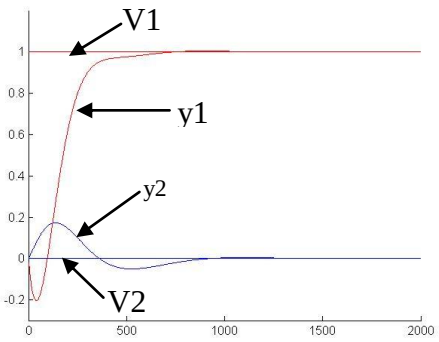

Figure 2.48: the output 1 of GA design the full controller with the set-point changing

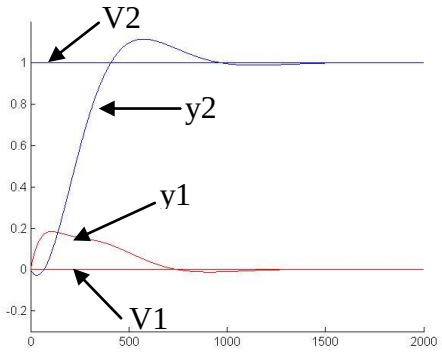

Figure 2.49: the output 2 of GA design the full controller with the set-point changing

Where V1 is the set point 1; V2 is the set point 2; y1 is the output 1, y2 is the output 2.

The result for the minimum sum of two objectives

The sum of ISE value from multi-objective genetic algorithm is closed to the single cost function genetic algorithm, ISE_1 and ISE_2 from the multi-objective genetic algorithm are similar with single cost function genetic algorithm, the multi-objective genetic algorithm could provide the optimal performance in sum of ISE and others are still good.

As the table show above, the red marked ISE1 channel value is the lowest one, then the controller for this design are:

$$K_p = \begin{bmatrix} 1.5421 & -0.8537 \\ -1.3493 & 0.6819 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 1.3605 & -0.4875 \\ 1.3041 & 1.0088 \end{bmatrix}$$

And the output of the system is shown below:

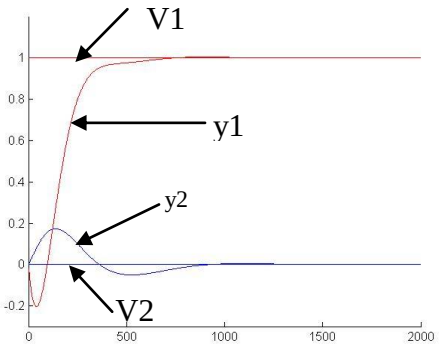

Figure 2.50: the output 1 of GA design the full controller with the set-point changing

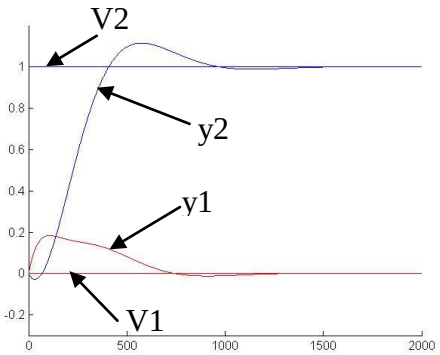

Figure 2.51: the output 2 of GA design the full controller with the set-point changing

Where V1 is the set point 1; V2 is the set point 2; y1 is the output 1, y2 is the output 2.

2.4.1.2 Multi-objective genetic algorithm design for the asymptotically stable plant when α =4 (minimum phase)

For the multi objective Pareto front method, this example has chosen the same as chapter 2.3, the two inputs and two outputs example are chosen as below:

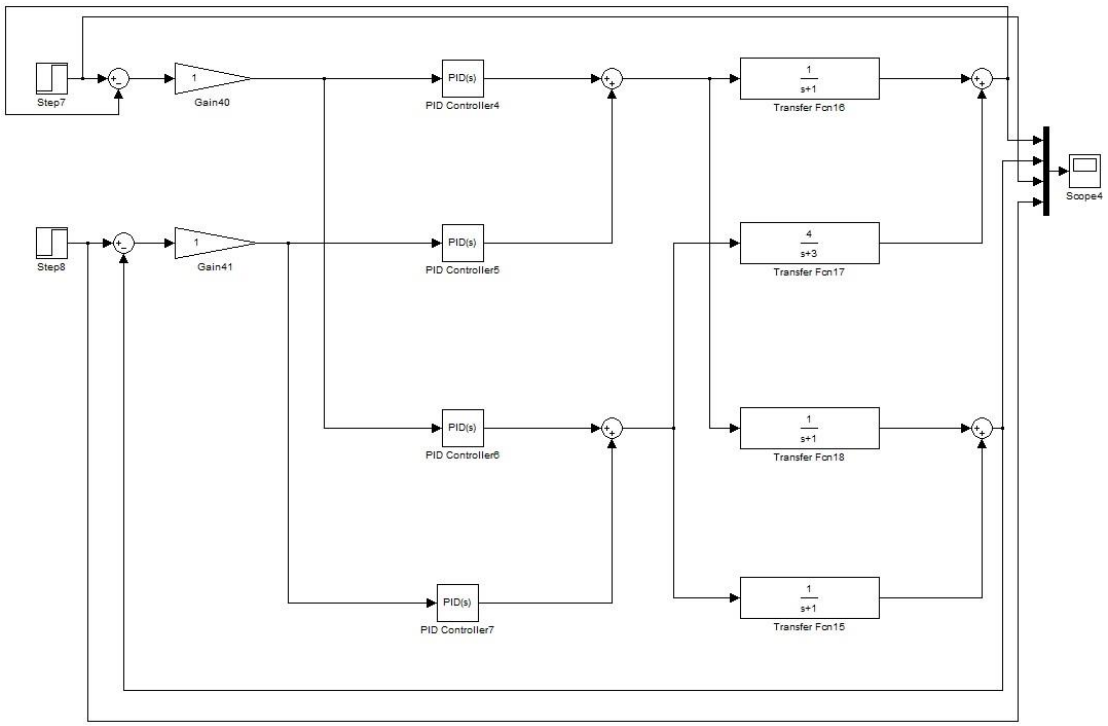

Figure 2.52: Square multivariable control system transfer function

Therefore, the overall transfer function

$$G(s) = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} = \begin{bmatrix} \dfrac{1}{s+1} & \dfrac{4}{s+3} \\ \dfrac{1}{s+1} & \dfrac{1}{s+1} \end{bmatrix} \quad \text{.....(2.113)}$$

And the system input

$$u = k_p \times e(t) + k_i \times \int_0^t e(t)dt \quad \text{.....(2.114)}$$

where

$$k_p = \begin{bmatrix} k_{p11} & k_{p12} \\ k_{p21} & k_{p22} \end{bmatrix} \quad \text{.....(2.115)}$$

$$\text{and } k_i = \begin{bmatrix} k_{i11} & k_{i12} \\ k_{i21} & k_{i22} \end{bmatrix} \quad \text{.....(2.116)}$$

$k_{p11}, k_{p12}, k_{p21}, k_{p22}, k_{i11}, k_{i12}, k_{i21}$ and k_{i22} are the parameters need be to design by genetic algorithm.

When the multi-objective GA has applied, the two objectives are the set point tracking property plus the interaction ISE on the other loop for both channels. The genetic algorithm was run for 1000 generations with a population size of 1000. The range of each parameters are:

$$k_{p1} = \begin{bmatrix} -0.6973 & -1.6973 \end{bmatrix}$$

$$k_{p2} = \begin{bmatrix} 6.4826 & 7.4826 \end{bmatrix}$$

$$k_{p3} = \begin{bmatrix} 2.9489 & 3.9489 \end{bmatrix}$$

$$k_{p4} = \begin{bmatrix} -1.4392 & -0.4392 \end{bmatrix}$$

$$k_{i1} = \begin{bmatrix} -3.3571 & -2.3571 \end{bmatrix}$$

$$k_{i2} = \begin{bmatrix} 6.2343 & 7.2343 \end{bmatrix}$$

$$k_{i3} = \begin{bmatrix} 4.2568 & 5.2568 \end{bmatrix}$$

and

$$k_{i4} = \begin{bmatrix} -4.3609 & -3.3609 \end{bmatrix}$$

There are two objectives which are the two outputs ISE:

$$Objective_1 = ISE_1 = ISE_{11} + ISE_{21}$$

And

$$Objective_2 = ISE_2 = ISE_{12} + ISE_{22}$$

In this design, ISE_{11} is the ISE of output 1 for a change in set-point 1. ISE_{21} is the interaction ISE of output 2 due to change in set-point 1. ISE_{22} is the ISE of output 2 for a change in set-point 2, and ISE_{12} is the interaction ISE of output 1 due to change in set-point 2.

The constraint is choosing as the interaction overshot will less than 20% as example, this "20%" is choosing by the designer, in this example, if the overshot in interaction is bigger than 20%, then the fitness of this individual is setting to zero.

The Pareto front plot could be drawn in graph below:

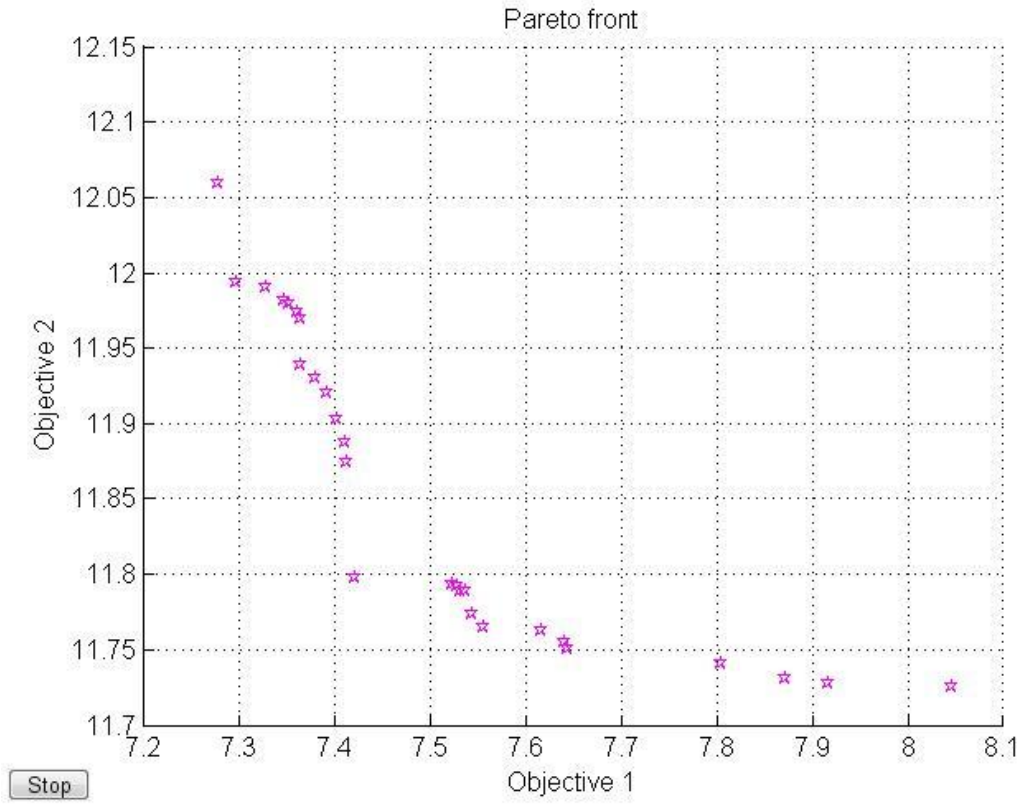

Figure 2.53: the Pareto front plot for example in chapter 2.3

As the Pareto front plot shows, the single objective genetic algorithm result from chapter 2.3: the objective1 ISE_1=8.37 and the objective2 ISE_2=12.71 is very closed to the Pareto front solution, so the single cost function genetic algorithm result is one of the optimal solutions and the multi-objective genetic algorithm Pareto front solutions have a family of Pareto front solutions. All the solutions are optimised solution.

The three interesting results are selected from the family of non-dominated Pareto front solutions which are the minimum ISE_1, ISE_2 and the sum of ISE are shown in table 2.5:

	Objective 1 for Combine ISE for change in set-point 1	Objective 2 for Combine ISE for change in set-point 2	sum of two objective
1	7.643273237	12.20019586	19.84346909
2	8.353239818	12.04718814	20.40042796
3	7.705758179	12.08395408	19.78971226

Table 2.5: the part results from the non-dominated Pareto front solutions

The result for the minimum objective$_1$

The ISE_1 value from multi-objective genetic algorithm is little smaller than the single cost function genetic algorithm, ISE_2 and the sum of ISE from the multi-objective genetic algorithm are similar with single cost function genetic algorithm, the multi-objective genetic algorithm could provide the optimal performance in ISE_1 and others are still good.

As the table show above, the red marked ISE_1 channel value is the lowest one, then the controller for this design are:

$$K_p = \begin{bmatrix} -1.1426 & 7.2303 \\ 3.6566 & -1.1493 \end{bmatrix}$$

$$K_i = \begin{bmatrix} -2.856 & 7.1134 \\ 5.2260 & -4.0332 \end{bmatrix}$$

And the output of the system is shown below:

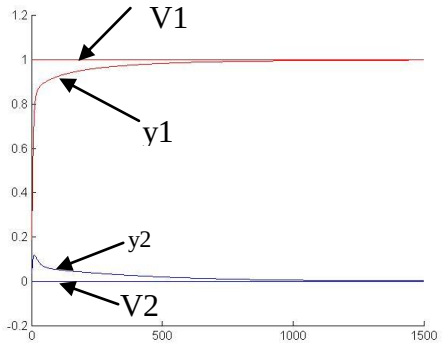

Figure 2.54: the output 1 of GA design the full controller with the set-point changing

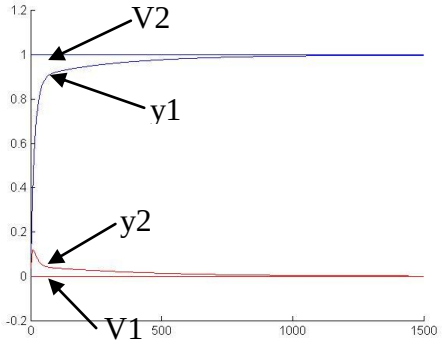

Figure 2.55: the output 2 of GA design the full controller with the set-point changing

Where V1 is the set point 1; V2 is the set point 2; y1 is the output 1, y2 is the output 2.

The result for the minimum objective$_2$

The ISE$_2$ value from multi-objective genetic algorithm is little smaller than the single cost function genetic algorithm, ISE$_1$ and the sum of ISE from the multi-objective genetic algorithm are similar with single cost function genetic algorithm, the multi-objective genetic algorithm could provide the optimal performance in ISE$_2$ and others are still good.

As the table show above, the red marked ISE$_2$ channel value is the lowest one, then the controller for this design are:

$$K_p = \begin{bmatrix} -1.6472 & 7.2655 \\ 3.2823 & -1.1019 \end{bmatrix}$$

$$K_i = \begin{bmatrix} -2.8209 & 7.1895 \\ 5.1107 & -3.9601 \end{bmatrix}$$

And the output of the system is shown below:

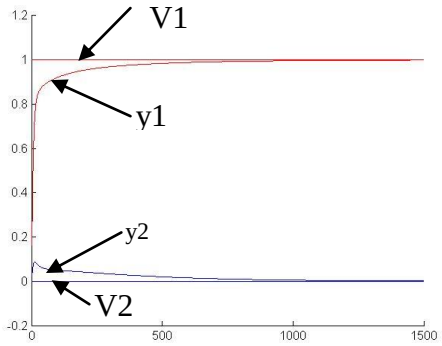

Figure 2.56: the output 1 of GA design the full controller with the set-point changing

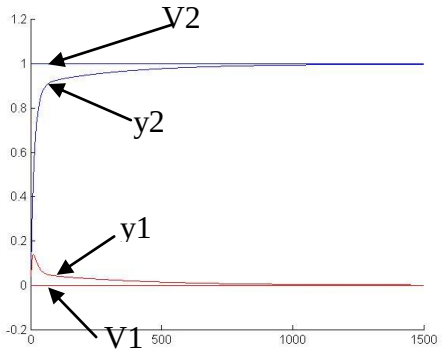

Figure 2.57: the output 2 of GA design the full controller with the set-point changing

The result for the minimum sum of two objectives

The sum of ISE value from multi-objective genetic algorithm is little smaller than the single cost function genetic algorithm, ISE_1 and ISE_2 from the multi-objective genetic algorithm are similar with single cost function genetic algorithm, the multi-objective genetic algorithm could provide the optimal performance in sum of ISE and others are still good.

As the table show above, the red marked ISE1 channel value is the lowest one, then the controller for this design are:

$$K_p = \begin{bmatrix} -1.5333 & 7.2290 \\ 3.6648 & -1.2328 \end{bmatrix}$$

$$K_i = \begin{bmatrix} -3.1106 & 7.2068 \\ 5.2302 & -3.9511 \end{bmatrix}$$

And the output of the system is shown below:

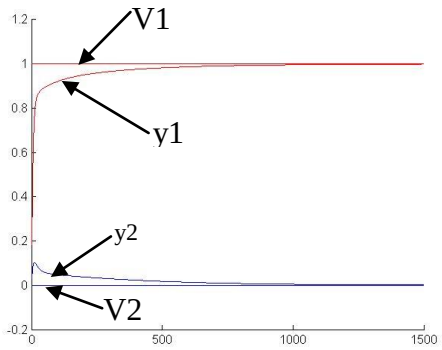

Figure 2.58: the output 1 of GA design the full controller with the set-point changing

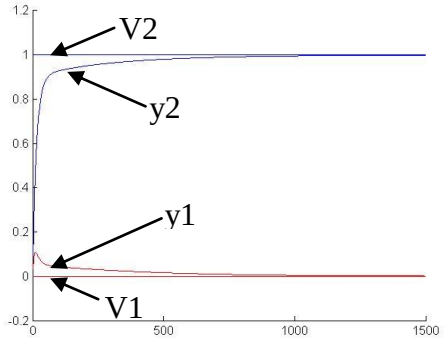

Figure 2.59: the output 2 of GA design the full controller with the set-point changing

Where V1 is the set point 1; V2 is the set point 2; y1 is the output 1, y2 is the output 2.

2.5 Conclusion

In this chapter, the design of decoupling controllers for square multivariable systems for non-minimum phase, minimum phase and functionally uncontrollable systems have been considered. Both single cost function genetic algorithm and multi objective genetic algorithm have been used when applied to the same square multivariable systems.

There are a number of issues which are researched in this chapter. The main conclusions which can be drawn are:

- Evolving a non-diagonal controller is more effective than evolving the tuning parameters associated with a decoupling controller. The system's ISE performance for genetic algorithm evolve non-diagonal controller improved about 84% compared with genetic algorithm evolve the diagonal controller, and it was significant as shown in table 2.2.
- With evolving multivariable controllers against a cost function involving the ISE due to set-point tracking and the ISE due to interaction, there is no direct control

over the amount of interaction. Like the figure 2.15 and 2.16 shows, as the system ISE performance improve the interaction over 50%.

- To control the amount the interaction a weighting factor can be introduced into the cost function. However, the weighting factor is hard to choose to achieve an exact design goal. Like the figure 2.17 and 2.18 shows, if the weighting factor decrease, the interaction will increase. Like the figure 2.19 and 2.20 shows, if the weighting factor increase, the interaction will decrease.
- In order to avoid choosing the weighting factor to control interaction a more effective technique is to add an interaction constraint into the cost function. This can control exactly the maximum level of interaction. As shown in figure 2.17, 2.18, 2.19 and 2.20 where the weighting factor is very hard to choose, because the weighting factor does affect the interaction, but the amount of interaction is not known until the end of the design process. However, as the figure 2.23 and 2.24 shows the constraint is easy to achieve the design goal.

According to these conclusions, the recommendations are:

- The single objective cost function genetic algorithm technique is very effective in designing multivariable controllers if the design trade-off weighting function are known. However, if a constraint is added into the genetic algorithm cost function, then a designed level of interaction is assured.
- The multi objective Pareto front GA provides a set of Pareto optimal solutions for the system. The set of optimal solution are all focused on satisfying set point tracking plus interaction constraint. In the Pareto optimal solutions, the designer can easily select the optimal solution for a specific design requirement such that a trade off the in the performance between each control loop objectives can be achieved easily.

Chapter Three

GENETIC DESIGN OF MULTIVARIABLE FAIL SAFE CONTROL SYSTEM

3.1 Introduction

The fail safe multivariable control system is for non-square multivariable control system when the control system has more inputs than outputs.

These systems have more actuators than system outputs. If one of the actuator fails, and there are still have enough actuators to control the system outputs, and keep the system outputs stable (Chakraborty, 2009). The concept this is called fail-safe control.

There are two kinds of methods to design fail-safe controllers: the passive technique and the active technique. The passive technique is one of the simplest fail-safe forms which can deal with actuator failures. The design of the passive fail-safe needs an underlying controller to be hyper-robust (Looze et al, 1985). Because there is a fixed controller to deal with both non-actuator failure and actuator failure situations, the system stability at both situations has to be guaranteed, but the system performance when an actuator fails is not guaranteed. One "passive" fail-safe control system design method for multivariable control system was introduced in 1988 by R.N.Fripp and solved the single failure fail-safe control design problems (Fripp, 1988). However, this design method was for single failure fail-safe control system, but was not guaranteed to works for multiple failures. The active design technique requires that each actuator have an additional sensor to detect if it has failed. If an actuator has

failed then a new control system can be deployed which has been specifically designed to take into account the actuator failure. Such techniques require additional sensor and switching software to fully implement the controller (Nicolaidis, 1998).

The aircraft pitch roll control is the one of the classic example for fail safe control system design. The aircraft pitch roll control uses the elevators, inboard ailerons, outboard ailerons and canard to control the pitch and roll angle. The control system is used to control each elevators, inboard ailerons, outboard ailerons and canard angle to control the aircraft pitch and roll angle. If one of the elevators, ailerons or canard fails the results could cause the system to become unstable. Normally, the control structure of the aircraft pitch roll can be described like figure 3.1 (Bosworth, 2012):

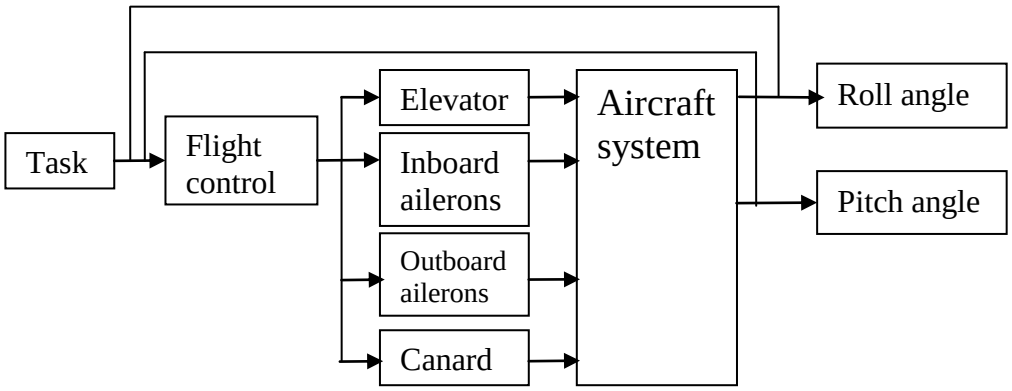

Figure 3.1: Control architecture

In this architecture, the system has four inputs: the angle of elevator, inboard ailerons, outboard ailerons and canard, and two system outputs: roll angle and pitch angle. Because there are two more actuators than the system outputs, then the system could should be able to cope with two actuator failures. Therefore, when looking into the design of aircraft roll and pitch angle control system the fail safe control theory becomes one of the options to consider.

3.1.1 Analysis

3.1.1.1The pseudo inverse design method:

This work was original carried out by R.N.Fripp (Fripp, 1988), and it shows the mathematical way to prove the multivariable system should be stable in single actuator failure.

The plants under consideration are governed on the continuous-time set $T = (0,+\infty]$ by state and output equations of the respective forms

$$\dot{x}(t) = Ax(t) + Bu(t) \quad \text{(3.1)}$$

and

$$y(t) = Cx(t) \quad \text{(3.2)}$$

where the state vector $x(t) \in R^n$, the input vector $u(t) \in R^m$, the output vector $y(t) \in R^l$, $l \le m$. Where n is the number of the state vector, m is the number of the input vector and p is the number of output vector. It is assumed that all the eigenvalues of the plant matrix $A \in R^{n\times n}$ lie in the open left half-plane C^-, and thatA, B, C and n in equations (3.1) and (3.2) are unknown. However, it is assumed that the steady-state transfer function matrix

$$G = G(0) = -CA^{-1}B \in R^{l\times m} \quad \text{(3.3)}$$

is known from open-loop tests where the plant transfer function matrix

$$G(s) = C(sI_n - A)^{-1}B \quad \text{(3.4)}$$

And

$$H(t) = CA^{-1}\left[e^{At} - I_n\right]B \in R^{l\times m} \quad \text{(3.5)}$$

Then

$$k_p = H^T(T)\left[H(T)H^T(T)\right]^{-1} \wedge (T)\pi \quad \text{(3.6)}$$

where

$$\pi = diag\{\pi_1, \pi_2, \ldots, \pi_l\}, \pi_i \in R^+(i = 1,2,\ldots,l) \quad \text{(3.7)}$$

and

$$k_i = G^T\left[GG^T\right]^{-1}\Sigma \quad \text{(3.8)}$$

where

$$\Sigma = diag\{\sigma_1, \sigma_2, \ldots, \sigma_l\}, \sigma_i \in R^+(i = 1,2, \ldots, l) \quad \ldots\ldots(3.9)$$

This design method guarantees that if a single actuator failure occurs then original controller will remain stable. However, there was a requirement to manually tune the controller to obtain a good transient response.

3.1.1.2The Genetic Algorithm design method

R.N.Fripp's design method only provides the system with stability guarantees for non-failure and single failure situations, but not multiple failures. Furthermore, optimal system performance is not guaranteed. Rather than design the controller using the passive technique of R.N. Fripp, genetic algorithm can be used to design and optimise the controller parameters.

Because Genetic algorithm can design any number of parameters, the Genetic algorithm can design the whole controller matrix which is

$$u_k = K_p e_k + K_i z_k \quad \ldots\ldots(3.10)$$

Where

$$z_k = z_{k-1} + Te_{k-1} \in R^m \quad \ldots\ldots(3.11)$$

$$K_p = \begin{bmatrix} k_{p11} & \cdots & k_{p1l} \\ \vdots & \ddots & \vdots \\ k_{pm1} & \cdots & k_{pml} \end{bmatrix} \quad \ldots\ldots(3.12)$$

and

$$K_i = \begin{bmatrix} k_{i11} & \cdots & k_{i1l} \\ \vdots & \ddots & \vdots \\ k_{im1} & \cdots & k_{iml} \end{bmatrix} \quad \ldots\ldots(3.13)$$

where $k_{p11} \cdots k_{pml}$ and $k_{i11} \cdots k_{iml}$ are the parameters for proportional gain integral gain and the derivative gain need to be chosen, and m and n are the number of system input and system output. Therefore, for this fail safe system, the number of inputs is greater than the number of outputs, then m>l, so the total number of parameters is $m \times l + m \times l = 2ml$.

This chapter is going to use the GA to find the optimal solution using a cost function. Therefore, the cost function is very important. The cost function selection is also a measure of the controlled system's performance. These measures are used to compare the system's performance between different control situation or different controller parameters.

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} g_{11}(s) & g_{12}(s) & g_{13}(s) \\ g_{21}(s) & g_{22}(s) & g_{23}(s) \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

Where y_1 and y_2 are the system outputs, u_1, u_2 and u_3 are the system inputs, g(s) are the system transfer function.

The cost function is the sum of all ISE under all three non-failure and single failure. The individual is calculated as:

- $ISE_t = \Sigma ISE_{i,j,k}$

where

- t is the actuator fault number, t=0 means non-actuator failure; t=1 means actuator 1 failure, t=2 means actuator 2 failure
- i is the actuator fault number
- j is the set point change number
- k is the output number

There are three individual ISE_i

$ISE_0 = (ISE_{0,1,1} + ISE_{0,1,2} + ISE_{0,2,1} + ISE_{0,2,2})$;

$ISE_1 = (ISE_{1,1,1} + ISE_{1,1,2} + ISE_{1,2,1} + ISE_{1,2,2})$;

$ISE_2 = (ISE_{2,1,1} + ISE_{2,1,2} + ISE_{2,2,1} + ISE_{2,2,2})$;

Therefore, for the single cost function GA, the cost function ISE (Integral Square of Error) is calculated by:

$$e = \int (\varepsilon)^2 dt = (\omega_0 \times ISE_0 + \omega_1 \times ISE_1 + \varpi_2 \times ISE_2)$$

where ϖ_i is the weight factor of each ISE.

The objective function for a single cost function genetic algorithm used in this chapter could be one of two types. The first type objective is global optimisation: this kind of objective is calculated by the ISE of set point tracking plus the ISE of interaction for all situations (Solihin et al, 2008). In this objective, the genetic algorithm will design a controller which will make the total sum of all ISE minimum. The second type objective is worst case failure optimisation: this type of objective is calculated by the largest ISE of individual set point tracking plus the ISE of interaction for all the outputs. In this objective, the genetic algorithm will design a controller which will make the largest of ISE out of all ISE as small as possible. To compare those two objective functions, the first one will give the minimum total ISE, but maybe some of the individual loop ISE will have a large value; the second objective will make sure there are no single large ISE value for a single loop, by focusing the algorithm on brings the worst one down. However, this will be at the cost of a slightly higher overall performance cost.

For the multi-objective genetic algorithm, the objectives are the Individual failure mode optimisation: this type of objective is a trade-off of the ISE for non-failure and all different failures, and multi-objective genetic algorithm will find a family solutions using a non-dominated Pareto front method. In this design, the multi-objective genetic algorithm will minimise the ISE for non-failure and failures situations simultaneously.

In genetic algorithm, the parameter range is very important, because it is related to the genetic algorithm will finds out the optimal solution or not, or how quick the genetic algorithm finds out optimal solutions. Moreover, the parameter range normally is very hard to choose. Therefore, the parameter range movement has been included into the genetic algorithm program. The parameter range movement method is when the best individual becomes close to the upper or the lower range limit, and then the whole range moves up or down. Normally the “close to the upper or the lower range limit” means 10% of the total range size, and how much percentage is chosen by the designer. Normally if the ranges need to be moved, the range will move up or down by 20% of the total range, and how much percentage to move is chosen by the designer as well. After the range movement, the GA needs to re-decode and re-scale the parameter range again, to make sure the GA binary number and the real parameter

number are matched. Without adopted this technique, the parameter range is not guaranteed to coverage to the optimal solution.

In the single cost function genetic algorithm and the multi-objective genetic algorithm, the interaction constrain are also included. However, the constraint should be chosen carefully, if the constraint is too small, the genetic algorithm design method will not be able to find a solution. As chapter two shows, the interaction constrain works well with genetic algorithm.

3.2 Genetic Algorithm design of single actuator failure fail-safe control system design

In this section, a two input one output non-square system is introduced. Because the system has one more inputs than outputs, so the system will still be controllable if one of the inputs fails. Therefore, there are three different control systems to be controlled in the design process:

- Non-failure
- Actuator 1 failure
- Actuator 2 failure

In this chapter, there are two design techniques have been used: the first one is the Pseudo inverse design method which is introduced by R.N.Fripp (1988), in this design method, genetic algorithm will be used to find the diagonal matrices to tune the controller. The second one is the full parameter Genetic algorithm design method, in this design method, the genetic algorithm is going to find all the controller parameter. It cannot guarantee there is one such controller to make the system stable and track in all conditions, but if it can find one, the genetic algorithm will optimise the design of the controller and will be better than the Pseudo inverse design method of Fripp.

In the Pseudo inverse design method, the cost function chosen is the sum of ISE under non-failure situation, ISE under actuator 1 failure and ISE under actuator 2 failure. In the full Genetic algorithm design method, there are two cost functions: Global

Optimisation and Worst case failure Optimisation. The Global optimisation cost function is the sum of ISE under non-failure situation, ISE under actuator 1 failure and ISE under actuator 2 failure. The cost function is calculated as:

$ISE_{global\ optimisation}=ISE_0 + ISE_1 + ISE_2$

Where ISE_0 is the total ISE for the system with no actuator failure, ISE_1 is the total ISE for the system with actuator 1 failure, ISE_2 is the total ISE for the system with actuator 2 failure.

The Worst case failure Optimisation is the largest ISE from the three of the ISE under non-failure situation, ISE under actuator 1 failure and ISE under actuator 2 failure. The cost function is calculated as:

$ISE_{worst\ failure\ case\ optimisation}$=Max {$ISE_0$, ISE_1 and ISE_2}

Where ISE_0 is the total ISE for the system with no actuator failure, ISE_1 is the total ISE for the system with actuator 1 failure, ISE_2 is the total ISE for the system with actuator 2 failure.

In this example a two-input one-output multivariable fail-safe control system is considered. The open loop transfer function is given by

$$[y] = \begin{bmatrix} \frac{16}{s^3 + 6s^2 + 12s + 8} & \frac{81}{s^3 + 9s^2 + 27s + 27} \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \text{.....(3.14)}$$

And the simulation block design is show below:

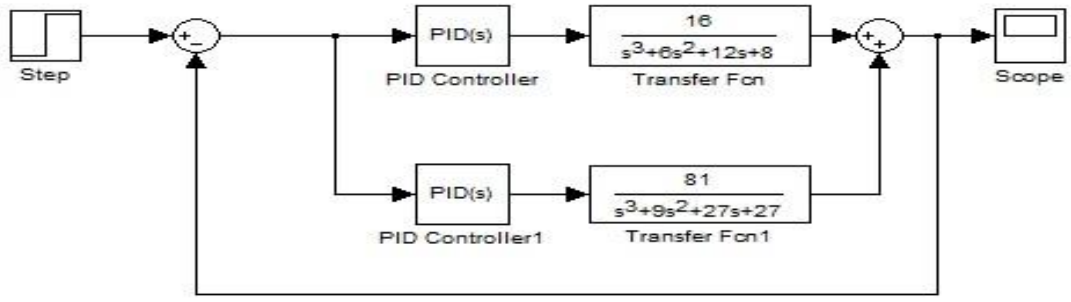

Figure 3.2 Transfer function

3.2.1 The Pseudo Inverse design method with Global Optimisation:

The Pseudo Inverse design method has introduced by R.N.Fripp (1988), with this design method the Genetic algorithm is going to search for the two diagonal matrixes to optimise the controller.

The cost function is chosen as $ISE_{global\ optimisation}=ISE_0 + ISE_1 + ISE_2$
Where ISE_0 is the ISE under non-failure situation, ISE_1 is the ISE under actuator 1 failure and ISE_2 is the ISE under actuator 2 failure.

The controller is given by

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = K_p \times [e_1] + K_i \times \left[\int e_1\right] \quad \text{.....(3.15)}$$

Where

$$K_p = B_1' \times inv(B_1 \times B_1') \times \Delta \quad \text{.....(3.16)}$$

where

$$\Delta = diag\{\Delta_1, \Delta_2, \ldots, \Delta_I\},\ \Delta_i \in R^+(i = 1,2,\ldots, I) \quad \text{.....(3.17)}$$

and

$$K_i = G' \times inv(G \times G') \times \Sigma \quad \text{.....(3.18)}$$

where

$$\Sigma = diag\{\sigma_1, \sigma_2, \ldots, \sigma_I\},\ \sigma_i \in R^+(i = 1,2,\ldots, I) \quad \text{.....(3.19)}$$

And

B_1 is the numerator of $G(S)$, and G is equal to $G(S)$ when $s = 0$

So

$$B_1 = [16 \quad 81] \quad \text{.....(3.20)}$$

and

$$G = \begin{bmatrix} \frac{16}{8} & \frac{81}{27} \end{bmatrix} = [2 \quad 3] \quad \text{.....(3.21)}$$

Therefore

$$K_p = B_1' \times inv\left(B_1 \times B_1'\right) \times \Delta$$

$$K_p = \begin{bmatrix}16\\81\end{bmatrix} \times inv\left\{\begin{bmatrix}16 & 81\end{bmatrix} \times \begin{bmatrix}16\\81\end{bmatrix}\right\} \times \Delta_1$$

$$K_p = \begin{bmatrix}0.002347\\0.011882\end{bmatrix} \times \Delta_1 \quad \text{.....(3.22)}$$

and

$$K_i = G' \times inv\left(G \times G'\right) \times \Sigma$$

$$K_i = \begin{bmatrix}2\\3\end{bmatrix} \times inv\left\{\begin{bmatrix}2 & 3\end{bmatrix} \times \begin{bmatrix}2\\3\end{bmatrix}\right\} \times \Sigma_1$$

$$K_i = \begin{bmatrix}0.153846\\0.230769\end{bmatrix} \times \Sigma_1 \quad \text{.....(3.23)}$$

In this controller the parameter of the proportional and integral controller structures are fixed and there are two tuning parameters $\Delta_1 and \varepsilon_1$ to be chosen by GA. All the two parameters ranges are:

$$\Delta_1 = \begin{bmatrix}0 & 2\end{bmatrix}$$

$$\varepsilon_1 = \begin{bmatrix}0 & 2\end{bmatrix}$$

When the single cost function GA has applied, the Max is chosen as 1000, the parameter range movement code is used, and the cost function minimizes the set point tracking property plus the interaction ISE on the other loop for both channels. The genetic algorithm was run for 5000 generations with a population size of 100, the sampling time is 0.01. The following values were then obtained from the genetic algorithm:

The two parameters are:

$\Delta_1 = 3.270967$ and $\Sigma_1 = 1.479765$

Then the final PI controllers are:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = K_p \times \Delta_1 \times \begin{bmatrix} e_1 \end{bmatrix} + K_i \times \Sigma_1 \times \begin{bmatrix} \int e_1 \end{bmatrix}$$

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 0.002347 \\ 0.011882 \end{bmatrix} \times \Delta_1 \times \begin{bmatrix} e_1 \end{bmatrix} + \begin{bmatrix} 0.153846 \\ 0.230769 \end{bmatrix} \times \Sigma_1 \times \begin{bmatrix} \int e_1 \end{bmatrix}$$

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 0.002347 \\ 0.011882 \end{bmatrix} \times 3.270967 \times \begin{bmatrix} e_1 \end{bmatrix} + \begin{bmatrix} 0.153846 \\ 0.230769 \end{bmatrix} \times 1.479765 \times \begin{bmatrix} \int e_1 \end{bmatrix} \quad \ldots.(3.24)$$

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 0.007677 \\ 0.038865 \end{bmatrix} \times \begin{bmatrix} e_1 \end{bmatrix} + \begin{bmatrix} 0.227656 \\ 0.341484 \end{bmatrix} \times \begin{bmatrix} \int e_1 \end{bmatrix}$$

And the outputs are shown below:

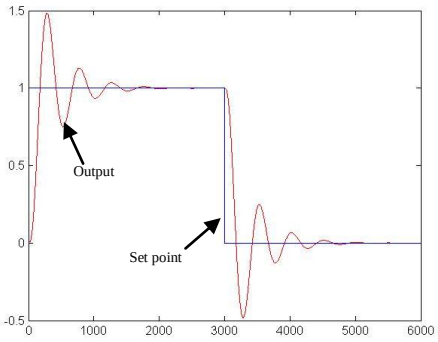

Figure 3.3: The output with non-failure

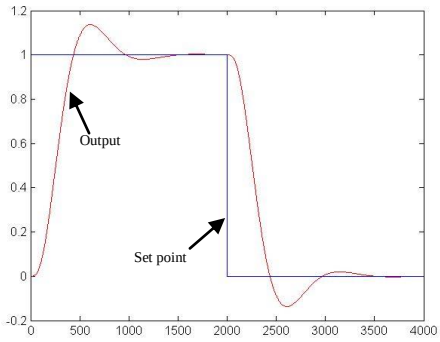

Figure 3.5: The output with Actuator 2 failure

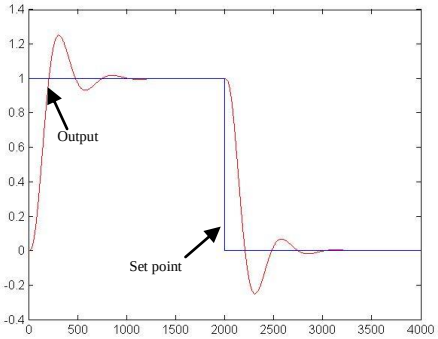

Figure 3.4: The output with Actuator 1 failure

As the above graph shows, the ISE (Integral Square of Error) is considered for output is calculated by:

ISE_0 under non-failure situation

ISE_1 under actuator 1 failure situation

ISE_2 under actuator 2 failure situation

Then:

The ISE of non-failure $ISE_0 = 268.272849319583$

The ISE of actuator 1 failure $ISE_1 = 225.130818983552$

The ISE of actuator 2 failure $ISE_2 = 412.61525368618$

The total three ISE $ISE_{total} = 906.018922$

3.2.2 The Genetic algorithm design method with single cost function full parameters Optimisation:

Rather than using the genetic algorithm to optimise the diagonal tuning matrixes to improve the Pseudo inverse design, the genetic algorithm can design the complete controller. Because the genetic algorithm is a search process, it is not guarantee to find a solution. However, if it finds one, the solution should have close to optimal performance.

In this design method, the cost function is chosen as $ISE_{global\ optimisation}=ISE_0 + ISE_1 + ISE_2$

Where ISE_0 is the ISE under non-failure situation, ISE_1 is the ISE under actuator 1 failure and ISE_2 is the ISE under actuator 2 failure.

And when the single objective GA has applied, the system input is:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = K_p \times \begin{bmatrix} e_1 \end{bmatrix} + K_i \times \begin{bmatrix} \int e_1 \end{bmatrix} \quad \text{.....(3.25)}$$

where $K_p = \begin{bmatrix} kp_1 \\ kp_2 \end{bmatrix}$ .....(3.26)

and

$$K_i = \begin{bmatrix} ki_1 \\ ki_2 \end{bmatrix} \quad \text{.....(3.27)}$$

Therefore, the kp_1 , kp_2 , ki_1 and ki_2 are the four parameters to be designed. The parameter range are

$$kp_1 = [-0.25 \quad 0.25]$$

$$kp_2 = [0.55 \quad 0.95]$$

$$ki_1 = [-0.15 \quad 0.84]$$

$$ki_2 = [-0.15 \quad 0.92]$$

When the single cost function GA has applied, the Max is chosen as 600, the parameter range movement code is used, and the cost function minimizes the set point tracking property plus the interaction ISE on the other loop for both channels. The genetic algorithm was run for 5000 generations with a population size of 100, the sampling time is 0.01. The following values were then obtained from the genetic algorithm:

The four parameters are:

$$K_p = \begin{bmatrix} 0.0144144850031482 \\ 0.753376325400707 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 0.342665805086368 \\ 0.42193936323745 \end{bmatrix}$$

And the single objective GA designed outputs are shown below:

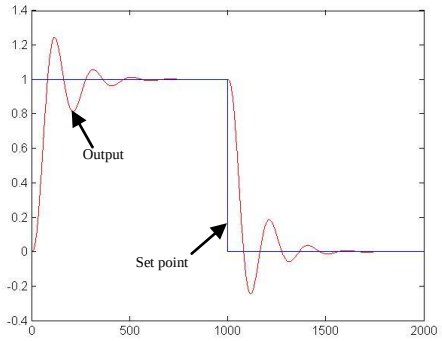

Figure 3.6: The output with non-failure

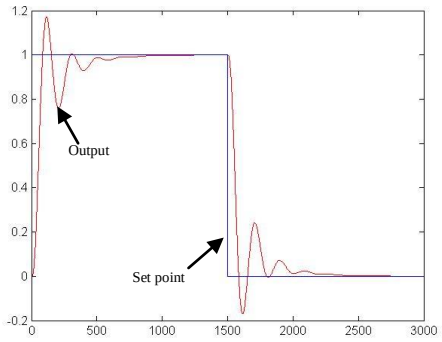

Figure 3.7: The output with Actuator 1 failure

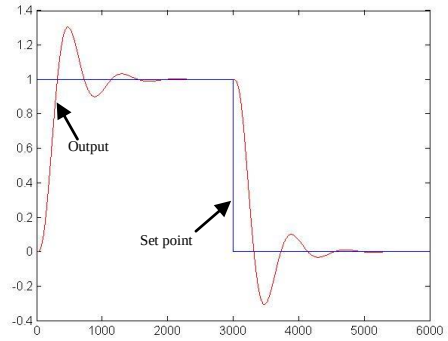

Figure 3.8: The output with Actuator 2 failure

As the above graph shows, the ISE (Integral Square of Error) is considered for output is calculated by;

- ISE_0 under non-failure situation
- ISE_1 under actuator 1 failure situation
- ISE_2 under actuator 2 failure situation

Then:

The ISE of non-failure $ISE_0 = 93.5633248219005$

The ISE of actuator 1 failure $ISE_1 = 95.4885836071419$

The ISE of actuator 2 failure $ISE_2 = 372.113734070826$

The total three ISE $ISE_{total} = 561.165642$

3.2.3 The genetic algorithm design method with Worst Case Failure Optimisation:

In this design method, the cost function is chosen as $ISE_{\text{worst case failure optimisation}}$=Max {$ISE_0$, ISE_1 and ISE_2}

Where ISE_0 is the ISE under non-failure situation, ISE_1 is the ISE under actuator 1 failure and ISE_2 is the ISE under actuator 2 failure.

When the single objective genetic algorithm is applied, the controller is defined as:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = K_p \times \begin{bmatrix} e_1 \end{bmatrix} + K_i \times \begin{bmatrix} \int e_1 \end{bmatrix} \quad \text{.....(3.28)}$$

where

$$K_p = \begin{bmatrix} kp_1 \\ kp_2 \end{bmatrix} \quad \text{.....(3.29)}$$

and

$$K_i = \begin{bmatrix} ki_1 \\ ki_2 \end{bmatrix} \quad \text{.....(3.30)}$$

Therefore, the kp_1 , kp_2 , ki_1 and ki_2 are the four parameters to be designed. The parameter range are

$$kp_1 = \begin{bmatrix} -0.25 & 0.25 \end{bmatrix}$$

$$kp_2 = \begin{bmatrix} -0.25 & 0.45 \end{bmatrix}$$

$$ki_1 = \begin{bmatrix} -0.15 & 0.54 \end{bmatrix}$$

$$ki_2 = \begin{bmatrix} -0.15 & 0.62 \end{bmatrix}$$

When the single cost function GA has applied, the Max is chosen as 400, the parameter range movement code is used, and the cost function minimizes the set point tracking property plus the interaction ISE on the other loop for both channels. The genetic algorithm was run for 5000 generations with a population size of 100, the sampling time is 0.01. The following values were then obtained from the genetic algorithm:

The four parameters are:

$$K_p = \begin{bmatrix} 0.041725 \\ 0.204563 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 0.356801 \\ 0.247633 \end{bmatrix}$$

And the single objective GA designed outputs are shown below:

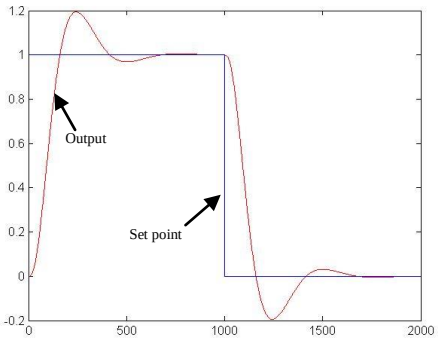

Figure 3.9: The output with non-failure

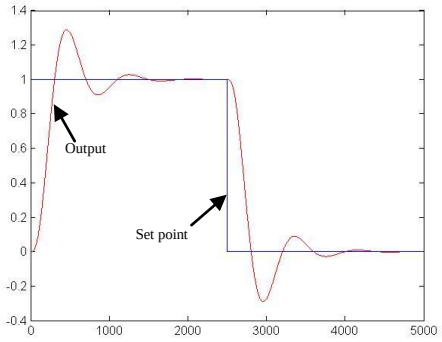

Figure 3.11: The output with Actuator 2 failure

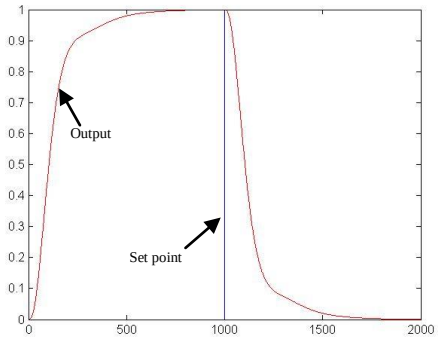

Figure 3.10: The output with Actuator 1 failure

As the above graph shows, the ISE (Integral Square of Error) is considered for output is calculated by:

- ISE_0 under non-failure situation
- ISE_1 under actuator 1 failure situation
- ISE_2 under actuator 2 failure situation

Then:

The ISE of non failure $ISE_0 = 153.593259147372$

The ISE of actuator 1 failure $ISE_1 = 166.643147105488$

The ISE of actuator 2 failure $ISE_2 = 347.368571924136$

The total three ISE $ISE_{total} = 667.604978$

And compare those two designs, the ISE comparison graph is shown below:

	ISE for non-failure	ISE for actuator 1 failure	ISE for actuator 2 failure	Total ISE
GA improve pseudo inverse design	268.272849	225.130819	412.615254	906.018922
Single objective GA design full parameters Optimization	93.5633248	95.4885836	372.113734	561.165642
Single objective GA design Worst Case Failure Optimization	153.593259	166.643147	347.368572	667.604978

Table 3.1: the comparesion bewtween normal inverse design and single objective GA design

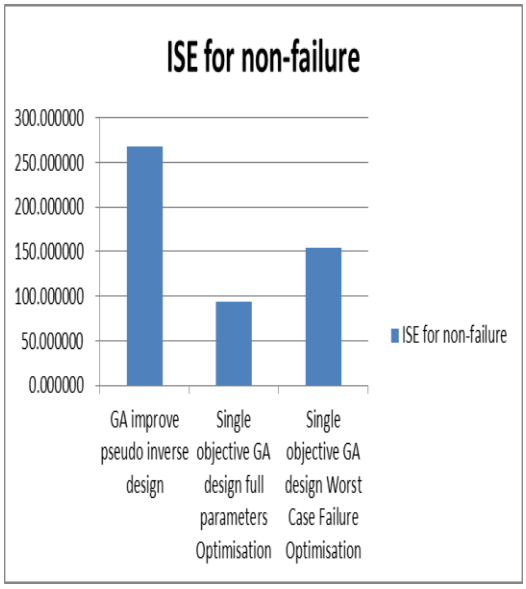

Figure 3.12: the comparesion bewtween normal inverse design and single objective GA design

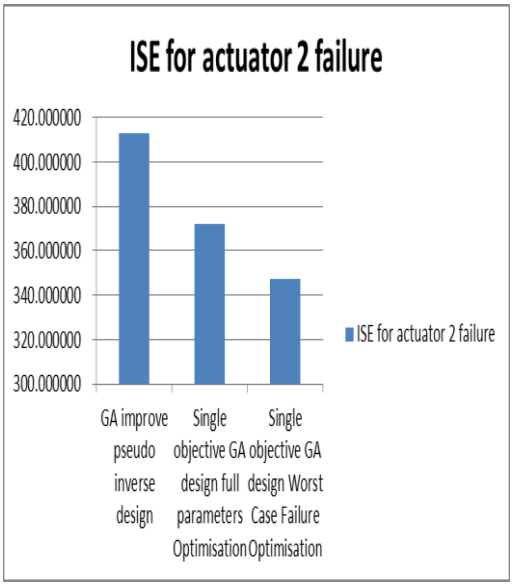

Figure 3.14: the comparesion bewtween normal inverse design and single objective GA design

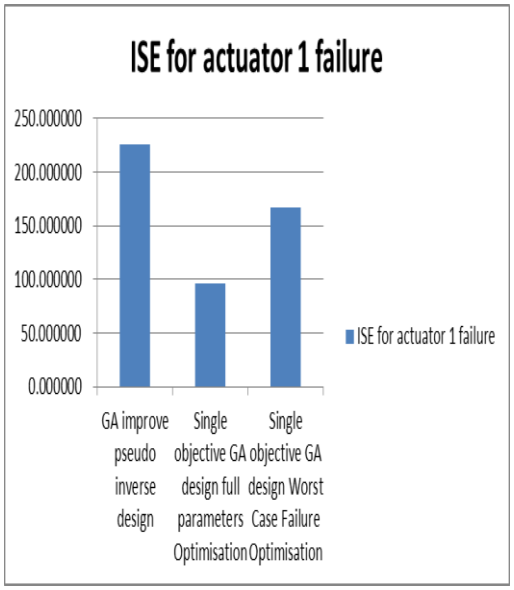

Figure 3.13: the comparesion bewtween normal inverse design and single objective GA design

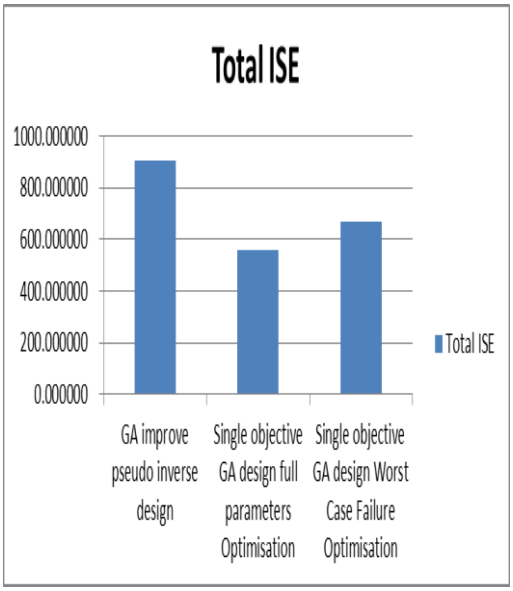

Figure 3.15: the comparesion bewtween normal inverse design and single objective GA design

As the table and figure show above, both the single objective GA design has considerably improved the system performance. The global optimisation method

provides the smallest total ISE, but the individual ISE are larger than the worst case failure optimisation method. The worst case failure optimisation method ensure the individual ISE is at a minimum but the total ISE is larger than the global optimisation.

3.3 Pareto front design of single actuator failure fail-safe control system design

For the single objective GA the weight factor α is chosen equal to one, this means the output with no failure and the output with failures all have same importance. However, in the real world each output may not be equally important, so the weight factor should be chosen by the designer. The weight factor chosen is a trade-off choice, if the situation has changed the weight factor should change as well.

However, the multi objective GA can find optimal solutions for multi objective functions simultaneously so there is no need to choose the weight factor. Therefore, there is no single best parameter that needs to be found, but a set of non-dominated Pareto solutions need to be found. Therefore, in each generation only the non-dominated solutions have been kept.

For the multi objective GA design, the example of 3.2 has chosen.
The controller is given by

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = K_p \times \begin{bmatrix} e_1 \end{bmatrix} + K_i \times \begin{bmatrix} \int e_1 \end{bmatrix} \quad \text{....(3.31)}$$

where

$$K_p = \begin{bmatrix} kp_1 \\ kp_2 \end{bmatrix} \quad \text{.....(3.32)}$$

and

$$K_i = \begin{bmatrix} ki_1 \\ ki_2 \end{bmatrix} \quad \text{.....(3.33)}$$

Therefore, the kp_1, kp_2 ki_1 ki_2 are the four parameters need to be designed by multivariable objective GA. The Pseudo inverse designed parameter is the start point of multivariable objective GA design.

By using the non-dominated Pareto front method, the ISE of non-failure, actuator 1 failure and actuator 2 failure are optimised individually. Some of the solutions from the Pareto front are shown in the table:

	ISE for non-failure	ISE for actuator 1 failure	ISE for actuator 2 failure	min x+y	min √x^2+y^2
1	95.151370	96.226528	372.617757	563.995655	396.430727
2	96.571247	95.409124	324.713641	516.694012	351.948654
3	100.894461	151.407560	324.523562	576.825584	372.047691

Table 3.2: the pareto front plot of single actuator failure

The multi-objective genetic algorithm with non-dominated Pareto front method tunes the three ISE and keeps the non-dominated Pareto front solutions. The red values in "ISE for non-failure", "ISE for actuator 1 failure" and "ISE for actuator 2 failure" are the minimum value in each individual channel. The "sum of all ISE" channel is calculated by the "ISE for non-failure" plus "ISE for actuator 1 failure" plus "ISE for actuator 2 failure". The red value in this channel is the minimum value. The "square root of sum of all squared ISE" channel is calculated by the square "ISE for non-failure" plus square "ISE for actuator 1 failure" plus square "ISE for actuator 2 failure". The red value in this channel is the minimum value.

3.3.1 *The minimum of "ISE for non-actuator failure" channel:*

Because the minimum value of sum of ISE for non-actuator failure, the GA finds the proportional and integral controllers as:

$$K_p = \begin{bmatrix} 0.013839 \\ 0.697844 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 0.347291 \\ 0.441134 \end{bmatrix}$$

And the outputs are shown below:

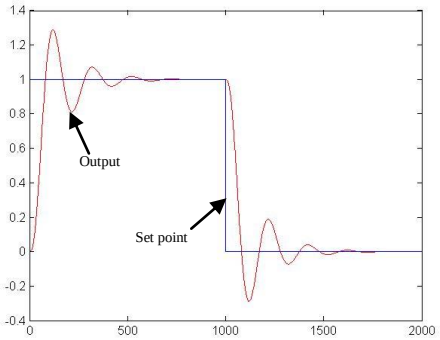

Figure 3.16:The output with non-failure

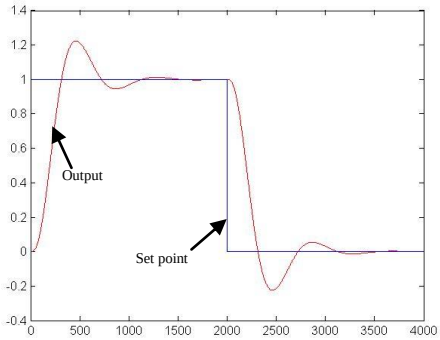

Figure 3.18 The output with Actuator 2 failure

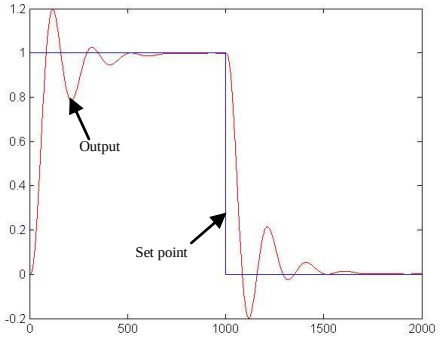

Figure 3.17: The output with Actuator 1 failure

3.3.2 The minimum of "ISE for actuator 2 failure" channel:

Because the ISE for actuator 2 failure is the largest ISE, this is the case of worst case failure optimization in the single cost function GA, then the GA finds the proportional and integral controllers as:

$$K_p = \begin{bmatrix} 0.078669 \\ 0.556464 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 0.439868 \\ 0.139174 \end{bmatrix}$$

And the outputs are shown below:

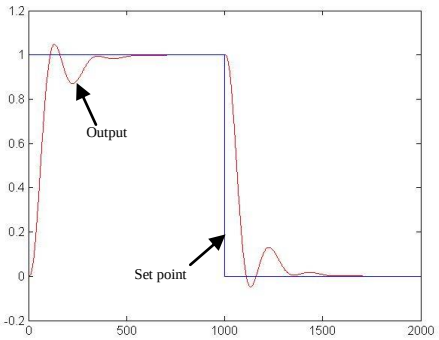

Figure 3.19: The output with non-failure

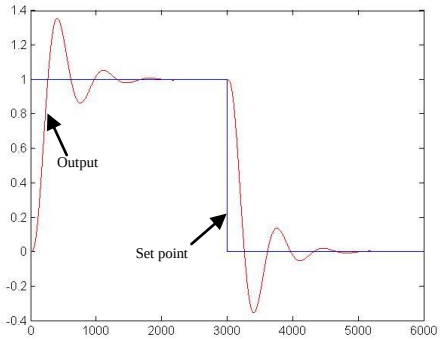

Figure 3.21: The output with Actuator 2 failure

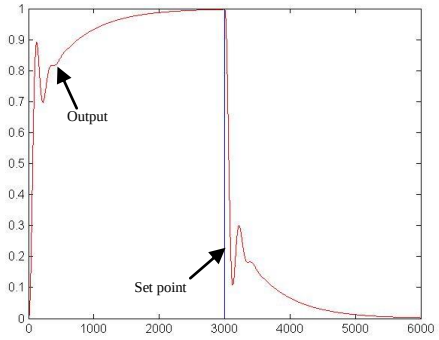

Figure 3.20: The output with Actuator 1 failure

3.3.3 The minimum of "sum of all ISE" channel "ISE for actuator 1 failure" channel and "square root of sum of all squared ISE" channel:

Because the minimum value of sum of all ISE is the same as the minimum of square root of sum of square ISE, the GA finds the proportional and integral controllers as:

$$K_p = \begin{bmatrix} 0.073025 \\ 0.719806 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 0.337844 \\ 0.493270 \end{bmatrix}$$

And the outputs are shown below:

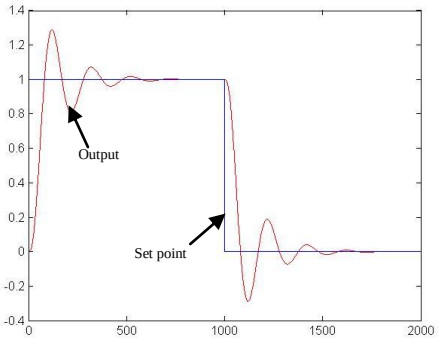

Figure 3.22:The output with non-failure

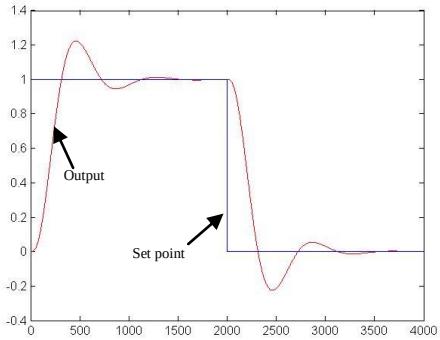

Figure 3.24: The output with Actuator 2 failure

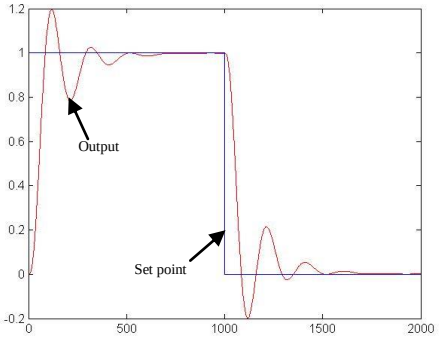

Figure 3.23: The output with Actuator 1 failure

As the multi-objective genetic algorithm with non-dominated Pareto front method results shows, the multi-objective genetic algorithm can find a set of optimised solutions for the control system. This can be compared with the single cost function genetic algorithm result, the single cost function genetic algorithm with full parameter optimisation has similar system performance to the multi-objective genetic algorithm minimum sum of all ISE result. Moreover, the single cost function genetic algorithm with worst failure case optimisation result is not as good as multi-objective genetic algorithm minimum of ISE for actuator 2 failure result. This is because the single cost function genetic algorithm only deals with the largest cost function with worst failure case optimisation of this cost function. However, the multi-objective genetic algorithm not only optimises the worst failure case but also optimise the other cases simultaneously. Moreover, the multi-objective genetic algorithm also provide more optimised solutions for different requires, such as the best performance on non-actuator failure case and other different non-failure and failure combinations.

Therefore, the multi-objective genetic algorithm with non-dominated Pareto front method is preferred.

3.4 Genetic algorithm design of single actuator and multiple actuator failure fail-safe control system

Because the pseudo inverse design method introduced by Fripp can only deal with a single actuator failure, but the genetic algorithm can search for solutions which can also deal with multiple actuator failures. The multiple actuator failures design is investigated.

In this example a four-input two-output multivariable fail-safe control system is considered. The open loop transfer function is given by

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{s+1} & \frac{2}{s+3} & \frac{4}{s+5} & \frac{6}{s+7} \\ \frac{-4}{s+2} & \frac{-3}{s+1} & \frac{2}{s+4} & \frac{8}{s+10} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \quad \text{.....(3.34)}$$

Because the system has two more inputs than outputs, so the system can still function stable if one or two of actuators have failed. If all single or double actuator failure conditions are considered, there are eleven different control systems to be controlled those are:

- Non-failure
- Actuator 1 failure
- Actuator 2 failure
- Actuator 3 failure
- Actuator 4 failure
- Actuator 1 and 2 failure
- Actuator 1 and 3 failure
- Actuator 1 and 4 failure
- Actuator 2 and 3 failure
- Actuator 2 and 4 failure
- Actuator 3 and 4 failure

In this chapter, there are two design techniques have been used: the first one is the Pseudo inverse design method which is introduced by R.N.Fripp (1988), in this design method, the genetic algorithm is going to find the diagonal tuning matrices to tune the controller; the second one is the full parameters genetic algorithm design method, in this design method, the genetic algorithm is going to find all the controller parameters. Tt cannot guarantee there is one such controller to make the system stable and track in all conditions, but the genetic algorithm can evolve an optimum solution to the design problem.

In the Pseudo inverse design method, the cost function is chosen as the sum of ISE under all eleven non-failure and single failure and multiple failures. In pure Genetic algorithm design method, there are two cost functions: Global Optimisation and Worst case failure Optimisation. The Global optimisation cost function is the sum of all ISE under all eleven non-failure and single failure and multiple failures.

The individual ISE is calculated as:

- $ISE_t = \Sigma ISE_{i,j,k}$

where

- t is the actuator fault number, t=0 means non-actuator failure, t=1 means actuator 1 failure, t=2 means actuator 2 failure, t=3 means actuator 3 failure, t=4 means actuator 4 failure, t=12 means actuator 1 and 2 failures, t=13 means actuator 1 and 3 failures, t=14 means actuator 1 and 4 failures, t=23 means actuator 2 and 3 failures, t=24 means actuator 2 and 4 failures, t=34 means actuator 3 and 4 failures
- i is the actuator fault number
- j is the set point change number
- k is the output number

There are eleven individual ISE_i where i is the actuator fault number:

- $ISE_0 = (ISE_{0,1,1} + ISE_{0,1,2} + ISE_{0,2,1} + ISE_{0,2,2})$;
- $ISE_1 = (ISE_{1,1,1} + ISE_{1,1,2} + ISE_{1,2,1} + ISE_{1,2,2})$;
- $ISE_2 = (ISE_{2,1,1} + ISE_{2,1,2} + ISE_{2,2,1} + ISE_{2,2,2})$;

- $ISE_3 = (ISE_{3,1,1} + ISE_{3,1,2} + ISE_{3,2,1} + ISE_{3,2,2})$;
- $ISE_4 = (ISE_{4,1,1} + ISE_{4,1,2} + ISE_{4,2,1} + ISE_{4,2,2})$;
- $ISE_{12} = (ISE_{12,1,1} + ISE_{12,1,2} + ISE_{12,2,1} + ISE_{12,2,2})$;
- $ISE_{13} = (ISE_{13,1,1} + ISE_{13,1,2} + ISE_{13,2,1} + ISE_{13,2,2})$;
- $ISE_{14} = (ISE_{14,1,1} + ISE_{14,1,2} + ISE_{14,2,1} + ISE_{14,2,2})$;
- $ISE_{23} = (ISE_{23,1,1} + ISE_{23,1,2} + ISE_{23,2,1} + ISE_{23,2,2})$;
- $ISE_{24} = (ISE_{24,1,1} + ISE_{24,1,2} + ISE_{24,2,1} + ISE_{24,2,2})$;
- $ISE_{34} = (ISE_{34,1,1} + ISE_{34,1,2} + ISE_{34,2,1} + ISE_{34,2,2})$;

The cost function is calculated as:

$ISE_{\text{global optimisation}} = ISE_0 + ISE_1 + ISE_2 + ISE_3 + ISE_4 + ISE_{12} + ISE_{13} + ISE_{14} + ISE_{23} + ISE_{24} + ISE_{34}$

The Worst case failure Optimisation is the largest ISE from the eleven non-failure and single failure and multiple failures. The cost function is calculated as:

$ISE_{\text{worst case failure optimisation}} = \text{Max}\ \{ISE_0 + ISE_1 + ISE_2 + ISE_3 + ISE_4 + ISE_{12} + ISE_{13} + ISE_{14} + ISE_{23} + ISE_{24} + ISE_{34}\}$

And the simulation block design is show below:

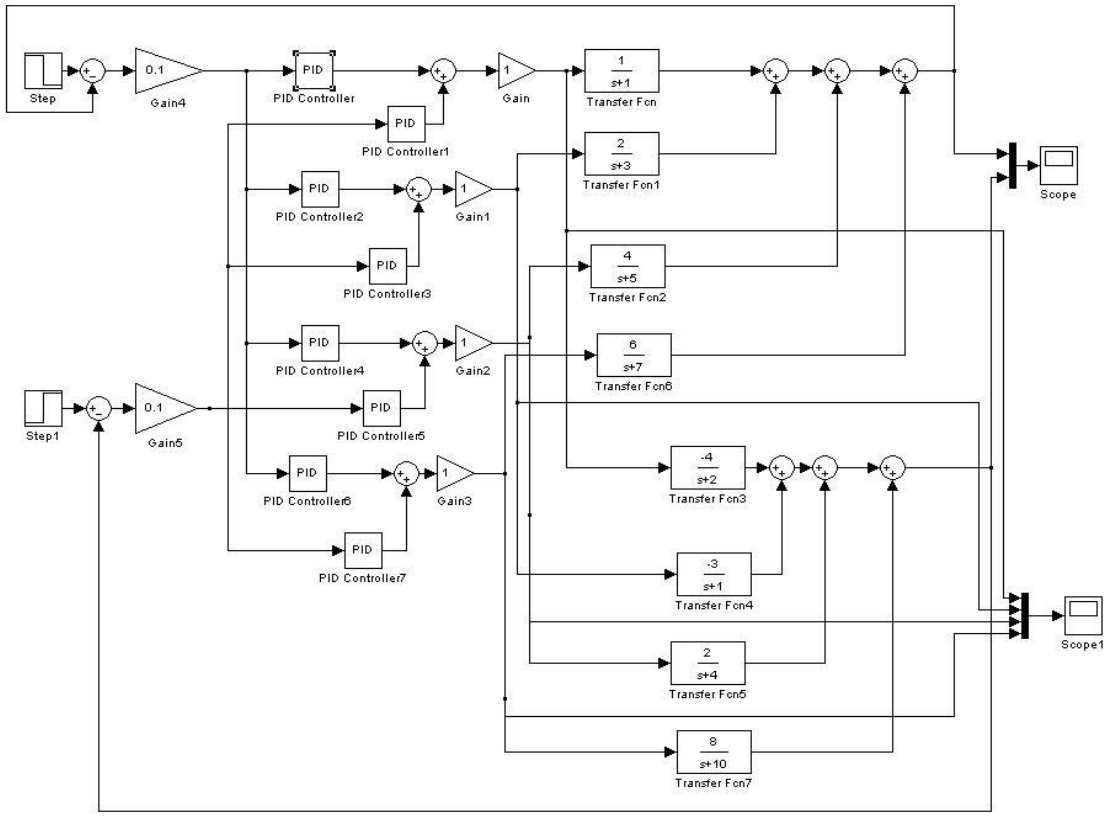

Figure 3.25 Transfer function

3.4.1 The Pseudo inverse design method with full parameters Optimisation:

This work was original carried out by R.N.Fripp (1988), and now extended to the multiple actuator failures.

The plants under consideration are governed on the continuous-time set $T = [0,+\infty]$ by state and output equations of the respective forms

$$\dot{x}(t) = Ax(t) + Bu(t)$$

and

$$y(t) = Cx(t)$$

where the state vector $x(t) \in R^n$, the input vector $u(t) \in R^m$, the output vector $y(t) \in R^l$, $l \leq m$. Where n is the number of the state vector, m is the number of the input vector and p is the number of output vector. It is assumed that all the eigenvalues of the plant matrix $A \in R^{n \times n}$ lie in the open left half-plane C^-, and

that A, B, C and n are unknown. However, it is assumed that the steady-state transfer function matrix

$$G = G(0) = -CA^{-1}B \in R^{l\times m}$$

is known from open-loop tests where the plant transfer function matrix

$$G(s) = C(sI_n - A)^{-1}B$$

And

$$H(t) = CA^{-1}\left[e^{At} - I_n\right]B \in R^{l\times m}$$

Then

$$k_p = H^T(T)\left[H(T)H^T(T)\right]^{-1} \wedge (T)\Delta$$

where

$$\Delta = diag\{\Delta_1, \Delta_2, \ldots, \Delta_l\}, \Delta_i \in R^+(i = 1,2,\ldots, l)$$

and

$$k_i = G^T\left[GG^T\right]^{-1}\Sigma$$

where

$$\Sigma = diag\{\sigma_1, \sigma_2, \ldots, \sigma_l\}, \sigma_i \in R^+(i = 1,2,\ldots, l)$$

This design method guarantees that if a single actuator failure occurs then original controller will remain stable. However, there was a requirement to manually tune the controller to obtain a good transient response.

The controller is given by

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = K_p \times \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + K_i \times \begin{bmatrix} \int e_1 \\ \int e_2 \end{bmatrix} \quad \text{.....(3.35)}$$

$$\text{where } K_p = \begin{bmatrix} k_{p11} & k_{p12} \\ k_{p21} & k_{p22} \\ k_{p31} & k_{p32} \\ k_{p41} & k_{p42} \end{bmatrix} \times \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} \text{ and } K_i = \begin{bmatrix} k_{i11} & k_{i12} \\ k_{i21} & k_{i22} \\ k_{i31} & k_{i32} \\ k_{i41} & k_{i42} \end{bmatrix} \times \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \quad \text{.....(3.36)}$$

and

$$\Delta = diag\{\Delta_1, \Delta_2, \ldots, \Delta_p\}, \Delta_i \in R^+(i = 1,2,\ldots, p)$$

and

$$\Sigma = diag\{\sigma_1, \sigma_2, ..., \sigma_p\}, \sigma_i \in R^+ (i = 1,2,..., p)$$

So

$$B_1 = \begin{bmatrix} 1 & 2 & 4 & 6 \\ -4 & -3 & 2 & 8 \end{bmatrix} \quad \text{.....(3.37)}$$

and

$$G = \begin{bmatrix} 1 & \frac{2}{3} & \frac{4}{5} & \frac{6}{7} \\ -2 & -3 & \frac{1}{2} & \frac{8}{10} \end{bmatrix} \quad \text{.....(3.38)}$$

Therefore

$$K_p = B_1' \times inv(B_1 \times B_1') \times \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} \quad \text{.....(3.39)}$$

$$K_p = \begin{bmatrix} 1 & -4 \\ 2 & -3 \\ 4 & 2 \\ 6 & 8 \end{bmatrix} \times inv\left\{ \begin{bmatrix} 1 & 2 & 4 & 6 \\ -4 & -3 & 2 & 8 \end{bmatrix} \times \begin{bmatrix} 1 & -4 \\ 2 & -3 \\ 4 & 2 \\ 6 & 8 \end{bmatrix} \right\} \times \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} \quad \text{.....(3.40)}$$

$$K_p = \begin{bmatrix} 0.08697 & -0.086028 \\ 0.10173 & -0.082575 \\ 0.087912 & -0.021978 \\ 0.059655 & 0.056515 \end{bmatrix} \times \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} \quad \text{.....(3.41)}$$

and

$$K_i = G' \times inv(G \times G') \times \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \quad \text{.....(3.42)}$$

$$K_i = \begin{bmatrix} 1 & -2 \\ 0.6667 & -3 \\ 0.8 & 0.5 \\ 0.85715 & 0.8 \end{bmatrix} \times inv\left\{ \begin{bmatrix} 1 & 0.6667 & 0.8 & 0.85714 \\ -2 & -3 & 0.5 & 0.8 \end{bmatrix} \times \begin{bmatrix} 1 & -2 \\ 0.6667 & -3 \\ 0.8 & 0.5 \\ 0.85714 & 0.8 \end{bmatrix} \right\} \times \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 0.26289 & -0.088831 \\ 0.016864 & -0.21244 \\ 0.40989 & 0.122 \\ 0.46428 & 0.15501 \end{bmatrix} \times \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \quad \text{.....(3.43)}$$

In this technique the K_p and K_i are the proportional and integral controllers. The $\Delta_1, \Delta_2, \Sigma_1$ and Σ_2 are the four tuning parameters. All the four parameters ranges are:

$$\Delta_1 = [0 \quad 2]$$

$$\Delta_2 = [0 \quad 2]$$

$$\varepsilon_1 = [0 \quad 2]$$

$$\varepsilon_2 = [0 \quad 2]$$

When the single cost function GA has applied, the Max is chosen as 500, the length of bit is 10, the parameter range movement code is used, and the cost function minimizes the global optimisation which is the sum of all eleven individual ISEs. The genetic algorithm was run for 5000 generations with a population size of 100, the sampling time is 0.1. The following values were then obtained from the genetic algorithm:

The four parameters are:

$$\Delta_1 = 4.9569, \Delta_2 = 5.6069, \Sigma_1 = 6.1587 \text{ and } \Sigma_2 = 12.7458$$

Then the final PI controller are

$$K_p = \begin{bmatrix} 0.08697 & -0.086028 \\ 0.10173 & -0.082575 \\ 0.087912 & -0.021978 \\ 0.059655 & 0.056515 \end{bmatrix} \times \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix}$$

$$= \begin{bmatrix} 0.08697 & -0.086028 \\ 0.10173 & -0.082575 \\ 0.087912 & -0.021978 \\ 0.059655 & 0.056515 \end{bmatrix} \times \begin{bmatrix} 4.9569 & 0 \\ 0 & 5.6069 \end{bmatrix} = \begin{bmatrix} 0.431110 & -0.482359 \\ 0.504258 & -0.462994 \\ 0.435779 & -0.123231 \\ 0.295707 & 0.316879 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 0.26289 & -0.088831 \\ 0.016864 & -0.21244 \\ 0.40989 & 0.122 \\ 0.46428 & 0.15501 \end{bmatrix} \times \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix}$$

$$= \begin{bmatrix} 0.26289 & -0.088831 \\ 0.016864 & -0.21244 \\ 0.40989 & 0.122 \\ 0.46428 & 0.15501 \end{bmatrix} \times \begin{bmatrix} 6.1587 & 0 \\ 0 & 12.7458 \end{bmatrix} = \begin{bmatrix} 1.619057 & -1.132226 \\ 0.103862 & -2.707774 \\ 2.524385 & 1.554945 \\ 2.859386 & 1.975696 \end{bmatrix}$$

And under the **Pseudo inverse design PI controller**, the outputs for various conditions are considered:

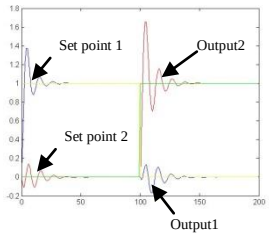

Figure 3.26: The output with non-failure

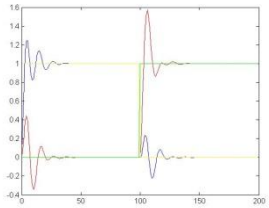

Figure 3.27: The output with Actuator 1 failure

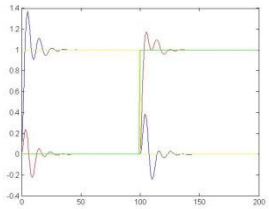

Figure 3.28: The output with actuator 2 failure

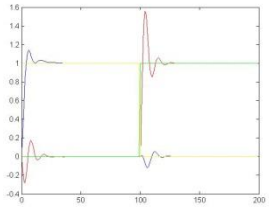

Figure 3.29: The output with actuator 3 failure

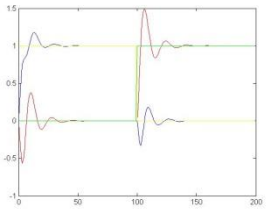

Figure 3.30: The output with actuator 4 failure

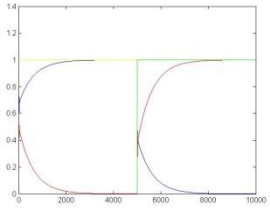

Figure 3.31: The output with actuator 1 and 2 failure

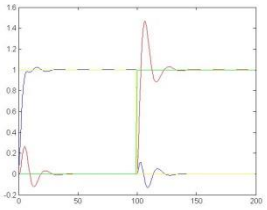

Figure 3.32: The output with actuator 1 and 3 failure

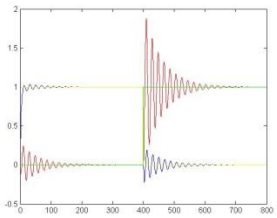

Figure 3.33: The output with actuator 1 and 4 failure

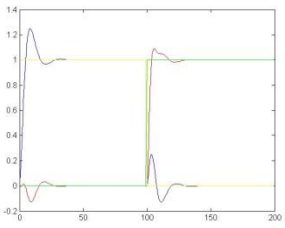

Figure 3.34: The output with actuator 2 and 3 failure

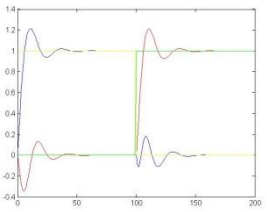

Figure 3.35: The output with actuator 2 and 4 failure

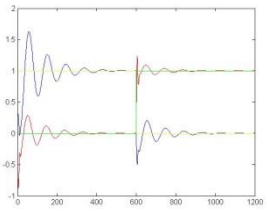

Figure 3.36: The output with actuator 3 and 4 failure

Then the ISE for each non-failure and failure situation is shown below:

	Pseudo inverse design ISE	Interaction caused by the set point 1 change	Interaction caused by the set point 2 change
Non-failure	7.098678172	17%	19%
Act 1 failure	8.024912371	45%	20%
Act 2 failure	6.680891951	21%	39%
Act 3 failure	6.427709778	25%	10%
Act 4 failure	8.980376217	51%	43%
Act 1 and 2 failure	291.4435739	70%	50%
Act 1 and 3 failure	7.140127165	23%	16%
Act 1 and 4 failure	21.60520139	30%	20%
Act 2 and 3 failure	6.487653917	15%	21%
Act 2 and 4 failure	8.397330686	33%	25%
Act 3 and 4 failure	52.59184059	70%	50%
Total ISE	424.8782962		

Table 3.3: The ISE for MIMO fail-safe control system under all situations (design method: Pseudo inverse)

As the above table shows, the ISE (Integral Square of Error) is considered for each output is calculated by the set point tracking ISE plus the interaction ISE for each channel. In this design, the weight factors are all equal to one.

As all the figures for system output and ISE table shows, the genetic algorithm could tune the passive design method of Fripp and provide a good system output performance. However, the interaction and overshot in some of the situations is very big. Therefore, the interaction's overshot constraint is applied.

3.4.2 The full parameter Genetic algorithm design method with constraint:

In this section, the same four inputs two outputs example is used. However, the genetic algorithm design is tuning the whole controller parameters. The controller is:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = K_p \times \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + K_i \times \begin{bmatrix} \int e_1 \\ \int e_2 \end{bmatrix} \quad \text{.....(3.44)}$$

where

$$K_p = \begin{bmatrix} k_{p11} & k_{p12} \\ k_{p21} & k_{p22} \\ k_{p31} & k_{p32} \\ k_{p41} & k_{p42} \end{bmatrix} \text{and} \quad K_i = \begin{bmatrix} k_{i11} & k_{i12} \\ k_{i21} & k_{i22} \\ k_{i31} & k_{i32} \\ k_{i41} & k_{i42} \end{bmatrix} \quad \text{.....(3.45)}$$

Therefore, the parameters $k_{p11}, k_{p12}, k_{p21}, k_{p22}, k_{p31}, k_{p32}, k_{p41}, k_{p42}, k_{i11}, k_{i12}, k_{i21}, k_{i22}, k_{i31}, k_{i32}, k_{i41}$ and k_{i42} are the sixteen parameters need to be searched for using the genetic algorithm. The parameter ranges are:

$$k_{p11} = [0.42 \quad 1.22]$$

$$k_{p12} = [-0.99 \quad -0.09]$$

$$k_{p21} = [-0.33 \quad 0.33]$$

$$k_{p22} = [-1.51 \quad -0.51]$$

$$k_{p31} = [0.56 \quad 1.76]$$

$$k_{p32} = [-0.45 \quad 0.35]$$

$$k_{p41} = [-0.17 \quad 0.77]$$

$$k_{p42} = [-0.18 \quad 0.98]$$

$$k_{i11} = [0.39 \quad 1.59]$$

$$k_{i12} = \begin{bmatrix} -1.71 & 0.61 \end{bmatrix}$$

$$k_{i21} = \begin{bmatrix} -0.51 & 0.61 \end{bmatrix}$$

$$k_{i22} = \begin{bmatrix} -2.14 & -1.14 \end{bmatrix}$$

$$k_{i31} = \begin{bmatrix} 2.17 & 3.47 \end{bmatrix}$$

$$k_{i32} = \begin{bmatrix} -1.54 & -0.14 \end{bmatrix}$$

$$k_{i41} = \begin{bmatrix} 0.88 & 1.98 \end{bmatrix}$$

$$k_{i42} = \begin{bmatrix} 1.98 & 2.78 \end{bmatrix}$$

When the single cost function GA has applied, the Max is chosen as 150, the length of bit is 10, the parameter range movement code is used, and the cost function minimizes the global optimisation which is the sum of all eleven individual ISEs. The genetic algorithm was run for 5000 generations with a population size of 100, the sampling time is 0.1. However, the interaction constraint is as 70% interaction. Therefore, if the overshot in interaction is bigger than 70%, then this individual will be removed from the population. The following values were then obtained from the genetic algorithm: Therefore, the GA chooses the proportional and integral controllers are:

$$K_p = \begin{bmatrix} 0.817178 & -0.498479 \\ 0.017260 & -1.107765 \\ 1.161402 & 0.092743 \\ 0.379626 & 0.483751 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 1.094133 & -1.113796 \\ 0.017981 & -1.543940 \\ 2.875684 & -0.745697 \\ 1.386035 & 2.489517 \end{bmatrix}$$

So this K_p and K_i are the proportional and integral controllers.

And under the **Genetic algorithm design method with constraint and Global Optimisation** PI controller, the outputs for various conditions are considered:

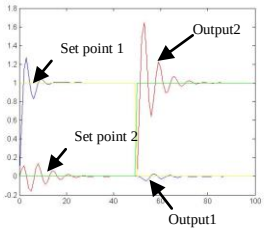

Figure 3.37: The output with non-failure

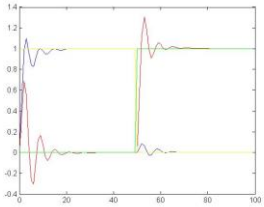

Figure 3.38: The output with Actuator 1 failure

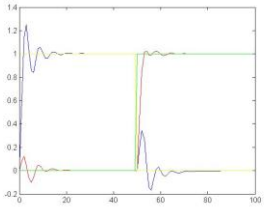

Figure 3.39: The output with actuator 2 failure

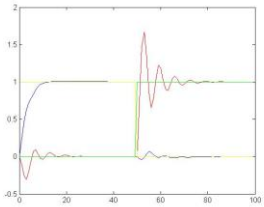

Figure 3.40: The output with actuator 3 failure

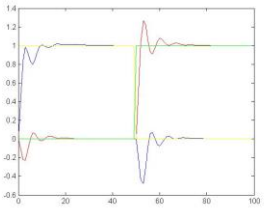

Figure 3.41: The output with actuator 4 failure

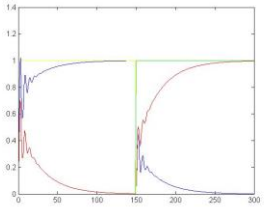

Figure 3.42: The output with actuator 1 and 2 failure

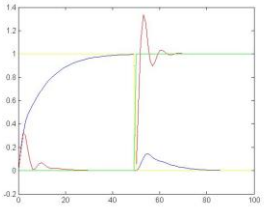

Figure 3.43: The output with actuator 1 and 3 failure

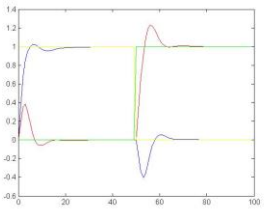

Figure 3.44: The output with actuator 1 and 4 failure

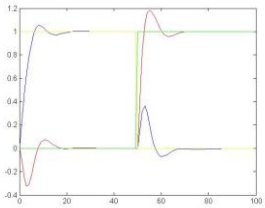

Figure 3.45: The output with actuator 2 and 3 failure

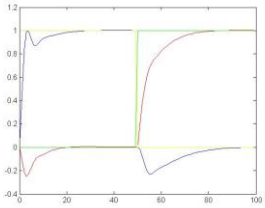

Figure 3.46: The output with actuator 2 and 4 failure

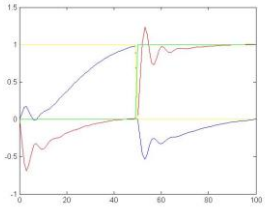

Figure 3.47: The output with actuator 3 and 4 failure

	Pseudo inverse design	Interaction constraint caused by the set point 1 change	Interaction constraint caused by the set point 2 change	GA design full parameter	Interaction constraint caused by the set point 1 change	Interaction constraint caused by the set point 2 change
Non-failure	7.098678	17%	19%	5.54068289	15%	5%
Act 1 failure	8.024912	45%	20%	5.82147900	70%	8%
Act 2 failure	6.680891	21%	39%	4.97143395	10%	35%
Act 3 failure	6.427709	25%	10%	6.57286370	30%	8%
Act 4 failure	8.980376	51%	43%	5.55097959	23%	43%
Act 1 and 2 failure	291.4435	70%	50%	16.7692166	70%	45%
Act 1 and 3 failure	7.140127	23%	16%	7.96761836	33%	15%
Act 1 and 4 failure	21.60520	30%	20%	6.27402722	40%	40%
Act 2 and 3 failure	6.487653	15%	21%	6.35837581	33%	37%
Act 2 and 4 failure	8.397330	33%	25%	7.00380591	25%	23%
Act 3 and 4 failure	52.59184	70%	50%	27.0285635	70%	53%
Total ISE	424.87			99.8590466		

Table 3.4: the ISE for MIMO fail-safe control system under all situations compared between Pseudo Inverse design and GA design Global Optimization

As the above table shows, the ISE (Integral Square of Error) is considered for each output is calculated by the set point tracking ISE plus the interaction ISE for each channel. In this design, the weight factors are all equal to one.

As the figures for the system output and ISE table shows, the genetic algorithm can design a controller for the system with a 70% constraint and optimise the system output performance under this constraint. However, the system performance will be slightly worse than if there was no constraint.

3.4.3 The full parameter Genetic algorithm design method with constraint and Worst Case Failure Optimisation:

In this section, the same four inputs two outputs example is used. However, the genetic algorithm design is tuning the whole controller parameters. The controller is:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = K_p \times \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + K_i \times \begin{bmatrix} \int e_1 \\ \int e_2 \end{bmatrix} \quad \text{.....(3.46)}$$

where

$$K_p = \begin{bmatrix} k_{p11} & k_{p12} \\ k_{p21} & k_{p22} \\ k_{p31} & k_{p32} \\ k_{p41} & k_{p42} \end{bmatrix} \quad \text{.....(3.47)}$$

and

$$K_i = \begin{bmatrix} k_{i11} & k_{i12} \\ k_{i21} & k_{i22} \\ k_{i31} & k_{i32} \\ k_{i41} & k_{i42} \end{bmatrix} \quad \text{.....(3.48)}$$

Therefore, the parameters $k_{p11}, k_{p12}, k_{p21}, k_{p22}, k_{p31}, k_{p32}, k_{p41}, k_{p42}, k_{i11}, k_{i12}, k_{i21}, k_{i22}, k_{i31}, k_{i32}, k_{i41}$ and k_{i42} are the sixteen parameters need to be searched for using the genetic algorithm. The parameter ranges are:

$$k_{p11} = [0.42 \quad 1.22]$$

$$k_{p12} = [-0.99 \quad -0.09]$$

$$k_{p21} = [-0.33 \quad 0.33]$$

$$k_{p22} = [-1.51 \quad -0.51]$$

$$k_{p31} = [0.56 \quad 1.76]$$

$$k_{p32} = [-0.45 \quad 0.35]$$

$$k_{p41} = [-0.17 \quad 0.77]$$

$$k_{p42} = [-0.18 \quad 0.98]$$

$$k_{i11} = [0.39 \quad 1.59]$$

$$k_{i12} = [-1.71 \quad 0.61]$$

$$k_{i21} = [-0.51 \quad 0.61]$$

$$k_{i22} = [-2.14 \quad -1.14]$$

$$k_{i31} = [2.17 \quad 3.47]$$

$$k_{i32} = [-1.54 \quad -0.14]$$

$$k_{i41} = [0.88 \quad 1.98]$$

$$k_{i42} = [1.98 \quad 2.78]$$

When the single cost function GA has applied, the Max is chosen as 30, the length of bit is 10, the parameter range movement code is used, and the cost function minimizes the worst case failure optimisation which is the largest ISE from all eleven individual ISEs. The genetic algorithm was run for 5000 generations with a population size of 100, the sampling time is 0.1. However, the interaction constraint is as 70% interaction. Therefore, if the overshot in interaction is bigger than 70%, then this individual will be removed from the population. The following values were then obtained from the genetic algorithm:

Therefore, the GA chooses the proportional and integral controllers are:

$$K_p = \begin{bmatrix} 0.803092 & -0.594833 \\ -0.417695 & -0.880747 \\ 0.600278 & -0.261772 \\ 0.284142 & 0.338844 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 1.006577 & -0.847911 \\ -0.164051 & -1.179238 \\ 1.519614 & -0.724984 \\ 0.522369 & 1.304453 \end{bmatrix}$$

So this K_p and K_i are the proportional and integral controllers.

And under the **Genetic algorithm design method with constraint and Worst Case Failure Optimisation** PI controller, the outputs for various conditions are considered:

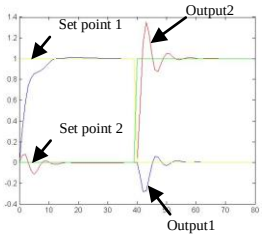

Figure 3.48: The output with non-failure

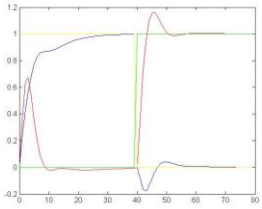

Figure 3.49: The output with Actuator 1 failure

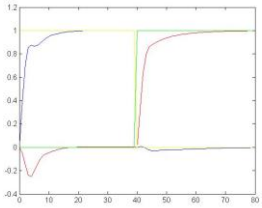

Figure 3.50: The output with actuator 2 failure

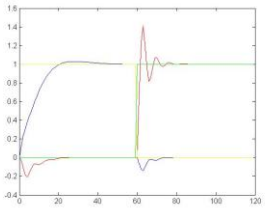

Figure 3.51: The output with actuator 3 failure

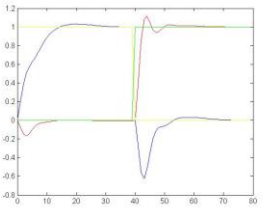

Figure 3.52: The output with actuator 4 failure

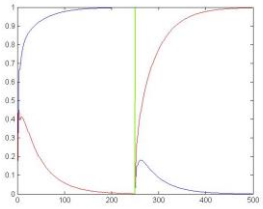

Figure 3.53: The output with actuator 1 and 2 failure

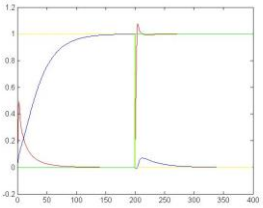

Figure 3.54: The output with actuator 1 and 3 failure

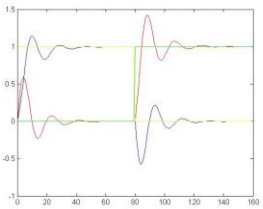

Figure 3.55: The output with actuator 1 and 4 failure

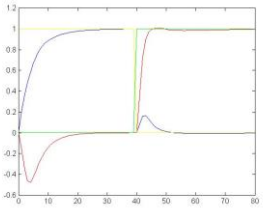

Figure 3.56: The output with actuator 2 and 3 failure

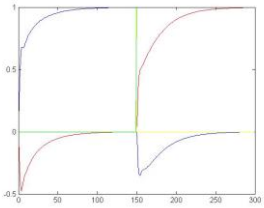

Figure 3.57: The output with actuator 2 and 4 failure

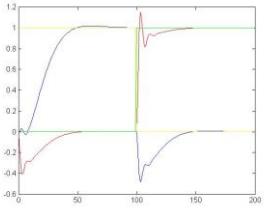

Figure 3.58: The output with actuator 3 and 4 failure

	Pseudo inverse design	Interaction constraint caused by the set point 1 change	Interaction constraint caused by the set point 2 change	GA design Global Optimisation	Interaction constraint caused by the set point 1 change	Interaction constraint caused by the set point 2 change	GA design Worst Case Failure Optimisation	Interaction constraint caused by the set point 1 change	Interaction constraint caused by the set point 2 change
Non-failure	7.098678172	17%	19%	5.540682897	15%	5%	5.465616381	11%	28%
Act 1 failure	8.024912371	45%	20%	5.821479009	70%	8%	7.522806982	68%	19%
Act 2 failure	6.680891951	21%	39%	4.971433955	10%	35%	5.48971439	25%	3%
Act 3 failure	6.427709778	25%	10%	6.572863705	30%	8%	8.177865043	21%	13%
Act 4 failure	8.980376217	51%	43%	5.550979595	23%	43%	7.622928879	17%	62%
Act 1 and 2 failure	291.4435739	70%	50%	16.76921666	70%	45%	25.02327771	45%	18%
Act 1 and 3 failure	7.140127165	23%	16%	7.967618363	33%	15%	24.64919438	50%	7%
Act 1 and 4 failure	21.60520139	30%	20%	6.274027229	40%	40%	12.18662064	60%	59%
Act 2 and 3 failure	6.487653917	15%	21%	6.358375816	33%	37%	7.48301353	47%	17%
Act 2 and 4 failure	8.397330686	33%	25%	7.00380591	25%	23%	16.65310972	47%	35%
Act 3 and 4 failure	52.59184059	70%	50%	27.02856353	70%	53%	25.00139174	40%	48%
Total ISE	424.8782962			99.85904667			145.2755394		

Table 3.5: the ISE for MIMO fail-safe control system under all situations compared between Pseudo Inverse design and GA design Global Optimization and GA design Worst Case Failure Optimization

As the above table shows, the ISE (Integral Square of Error) is considered for each output is calculated by the set point tracking ISE plus the interaction ISE for each channel. In this design, the weight factors are all equal to one.

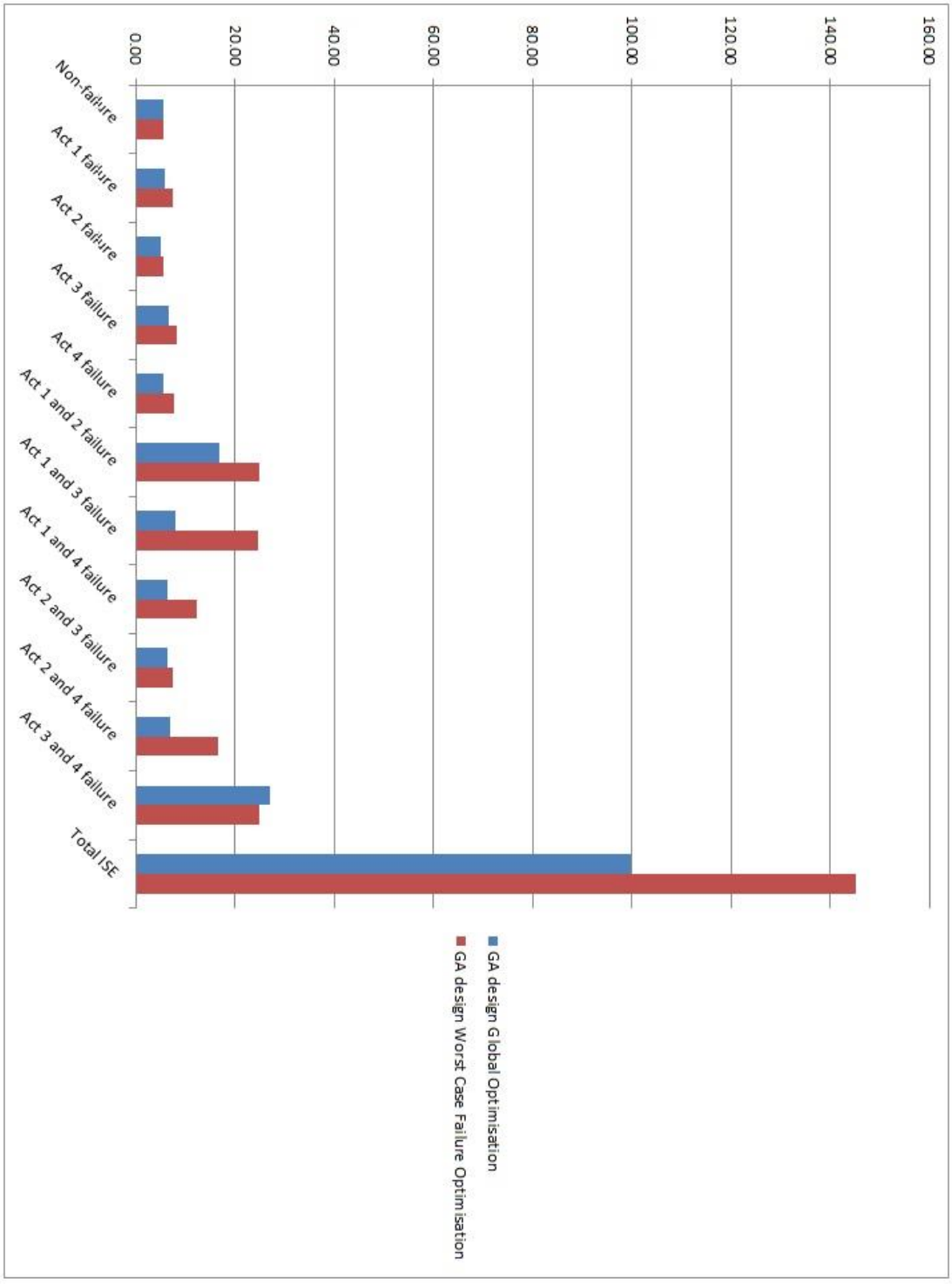

Figure 3.59: the ISE compression between Pseudo inverse design and GA design

As the table 3.3 and figure 3.4 shows, both GA designs are much better than Pseudo inverse design. Both GA design have reduced the total ISE and individual ISE together. With the Global Optimisation genetic algorithm design, it provides the smallest ISE but the worst case failure ISE is not as small as the Worst Case Failure Optimisation genetic algorithm design. Therefore, if the worst case failure ISE is acceptable with Global Optimisation genetic algorithm design, then the cost function is chosen Global Optimisation. If the worst case failure ISE is not acceptable, then the cost function chosen with Worst Case Failure Optimisation, because this cost function will reduce the worst case failure ISE and ensure the other ISEs remain with acceptable limits.

3.5 Pareto front design of single and multiple actuator failure fail-safe control systems

For a single objective GA the weight factor α is chosen equal to one, this means the output with no failure and the output with failure are same importance. However, in the real world each output may not be equally important, so the weighting factor can be introduced. The weight factor chosen is a trade-off choice, if the situation has changed the weight factor should change as well.

However, when compared with the single objective GA, the multi objective GA deals with each sub-set of the optimisation simultaneously so there is no need to choose the weighting factor. Therefore, there is no single best set of parameter that can be found, but a set of Pareto solutions can be found in the parameter range. In this method, a family of solutions are found witch would improve one objective and at the same time does not make the other objectives worse. The set of Pareto optimal solution are called non-dominated solution, because those solutions are not dominated by other solutions. Therefore, in each generation only the non-dominated solutions have been kept.

For the multi objective GA design, the example of 3.4 has chosen.
The controller is given by

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = K_p \times \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + K_i \times \begin{bmatrix} \int e_1 \\ \int e_2 \end{bmatrix} \quad \text{.....(3.49)}$$

Where

$$K_p = \begin{bmatrix} k_{p11} & k_{p12} \\ k_{p21} & k_{p22} \\ k_{p31} & k_{p32} \\ k_{p41} & k_{p42} \end{bmatrix} \quad \text{.....(3.50)}$$

and

$$K_i = \begin{bmatrix} k_{i11} & k_{i12} \\ k_{i21} & k_{i22} \\ k_{i31} & k_{i32} \\ k_{i41} & k_{i42} \end{bmatrix} \quad \text{.....(3.51)}$$

Therefore, the parameters $k_{p11}, k_{p12}, k_{p21}, k_{p22}, k_{p31}, k_{p32}, k_{p41}, k_{p42}, k_{i11}, k_{i12}, k_{i21}, k_{i22}, k_{i31}, k_{i32}, k_{i41}$ and k_{i42} are the sixteen parameters need to be found by multi-objective genetic algorithm. The parameter ranges are:

$k_{p11} = [0.58 \quad 1.032]$

$k_{p12} = [-0.89 \quad -0.20]$

$k_{p21} = [-1.05 \quad 0.65]$

$k_{p22} = [-1.13 \quad -0.85]$

$k_{p31} = [-0.16 \quad 1.92]$

$k_{p32} = [-0.82 \quad 0.64]$

$k_{p41} = [-0.01 \quad 0.67]$

$k_{p42} = [-0.01 \quad 0.82]$

$k_{i11} = [0.72 \quad 1.28]$

$k_{i12} = [-1.18 \quad -0.78]$

$k_{i21} = [-0.55 \quad 0.40]$

$k_{i22} = [-1.71 \quad -1.10]$

$k_{i31} = [-0.04 \quad 4.43]$

$$k_{i32} = \begin{bmatrix} -0.90 & -0.57 \end{bmatrix}$$
$$k_{i41} = \begin{bmatrix} -0.54 & 2.44 \end{bmatrix}$$
$$k_{i42} = \begin{bmatrix} -0.08 & 3.87 \end{bmatrix}$$

The Pseudo inverse designed controller parameters are used as the start point of genetic algorithm design, the parameter range is plus and minus 20% of the parameter, the length of bit is 10, and multi-objective genetic algorithm is going to minimize the each individual ISE simultaneously. However, the interaction constraint is as 70% interaction. Therefore, if the overshot in interaction is bigger than 70%, then this individual will be removed from the population. The genetic algorithm was run for 1000 generations with a population size of 1000.

By using the Pareto front method, the ISE of non-failure, single actuator failure and multiple failures are optimised on a non-dominated Pareto front, and because there are 11 ISE's, this is difficult to shown in graphical form, for this reason it is shown as a table:

ISE for Non-failure	ISE Actuator 1 failure	ISE Actuator 2 failure	ISE Actuator 3 failure	ISE Actuator 4 failure	ISE Actuator 1 and 2 failure	ISE Actuator 1 and 3 failure	ISE Actuator 1 and 4 failure	ISE Actuator 2 and 3 failure	ISE Actuator 2 and 4 failure	ISE Actuator 3 and 4 failure	sum of all ISE	square root of sum of all ISE squared
4.86741047	5.83292	6.04514	6.42971	5.62688	39.4965	7.86749	6.90758	7.31049	4.87741	28.6022	123.8	54.90
5.90248173	5.43828	5.14423	8.05902	5.56315	14.6469	13.8119	6.16129	7.67863	5.90248	33.0734	111.3	44.87
5.08036927	6.03914	4.83759	6.55467	7.34642	23.8646	9.06997	11.8845	6.02631	5.08036	26.5746	112.3	57.28
5.03959306	6.04893	5.21476	5.53743	7.86150	22.6783	8.38254	9.74834	5.63939	5.03959	27.0840	108.2	49.83
5.50154603	6.03713	5.34302	9.01758	4.99539	13.5450	238.232	5.94752	10.6594	5.50154	26.6641	331.4	241.26
4.97854987	5.45074	5.14093	7.90074	5.81270	12.0275	19.2741	7.30808	9.34123	4.97855	27.5964	109.8	44.24
5.04044535	5.65463	5.20454	5.65824	7.26237	51.7966	6.67999	7.85575	5.62231	5.04044	30.1085	135.9	65.92
7.44995519	6.21542	5.49906	7.23220	5.40385	16.505	8.22628	5.61233	6.59461	7.44995	28.1283	104.3	40.16
6.59858113	6.18788	5.16209	6.97362	7.39676	24.3588	8.00883	8.36731	5.56007	6.59858	28.4806	113.6	48.38
4.87118928	5.47506	5.91179	7.06976	5.86799	30.4313	8.63030	7.21415	7.84317	4.86118	27.6368	115.8	48.97
5.31316623	5.76389	4.99119	8.02596	6.55231	20.6011	10.1098	7.33483	7.86141	5.31316	26.1751	108.0	43.48
5.45917643	5.83454	5.15027	5.95498	6.29413	18.7978	7.17041	8.18072	5.77164	5.45917	30.2449	104.3	45.78

Table 3.60: The Pareto front solutions for MIMO fail safe control system

As the above table shows, the ISE (Integral Square of Error) is considered for each output is calculated by the set point tracking ISE plus the interaction ISE for each channel.

As table 3.60 shows, all the results are non-dominated Pareto front solutions, the red number is the minimum value for the eleven different situations considered. The results are similar to the single cost function genetic algorithm design with Global Optimisation where the ISE value was 99.85.

These results permit the design to choose a solution which could be based on certain actuators failing or the non-failed case. In order to demonstrate the various solutions from the Pareto front three solutions are reviewed:

- ISE for non-failure
- ISE for minimum sum of all ISE
- ISE for actuator 3 and 4 failure

As the red value marked in the table, this is the minimum value of non failure, so this is the one of the Pareto front solution which might be chosen as a final design.
The PI controller at this particularly Pareto front situation is:

$$K_p = \begin{bmatrix} 0.64347 & -0.2351 \\ 0.0022 & -0.9460 \\ 0.8940 & 0.2423 \\ 0.3147 & 0.3880 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 0.8498 & -0.8772 \\ 0.0504 & -1.5607 \\ 3.5669 & -0.7004 \\ 1.1515 & 0.7506 \end{bmatrix}$$

And under the **minimum value of square root of sum of all ISE square** PI controller, the outputs for various conditions are considered:

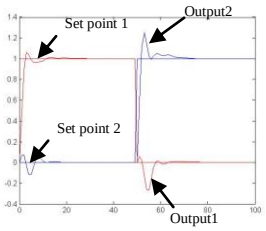

Figure 3.60: The output with non-failure

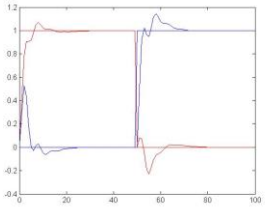

Figure 3.61: The output with Actuator 1 failure

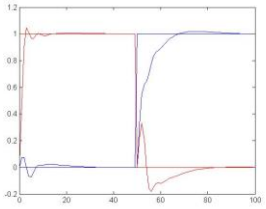

Figure 3.62: The output with actuator 2 failure

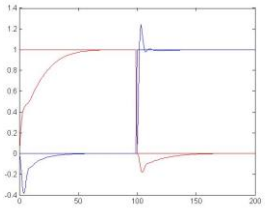

Figure 3.63: The output with actuator 3 failure

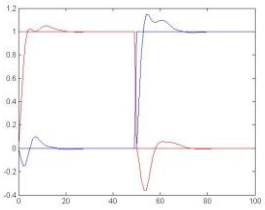

Figure 3.64: The output with actuator 4 failure

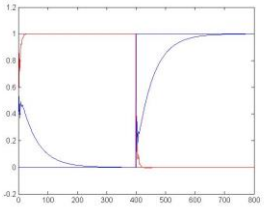

Figure 3.65: The output with actuator 1 and 2 failure

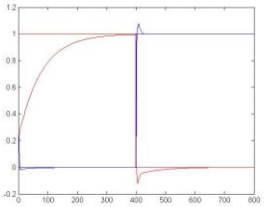

Figure 3.66: The output with actuator 1 and 3 failure

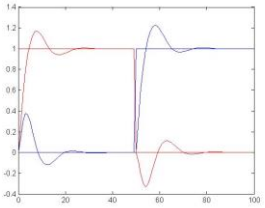

Figure 3.67: The output with actuator 1 and 4 failure

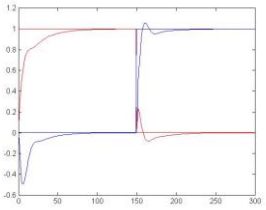

Figure 3.68: The output with actuator 2 and 3 failure

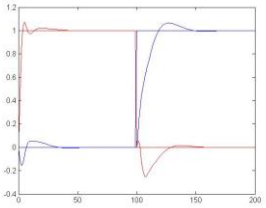

Figure 3.69: The output with actuator 2 and 4 failure

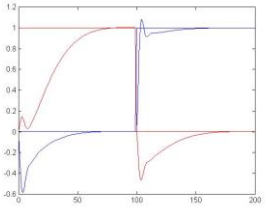

Figure 3.70: The output with actuator 3 and 4 failure

As the red value marked in the table, this is the minimum value of sum of all ISE, so this is the one of the Pareto front solution which might be chosen as a final design. The PI controller at this particularly Pareto front situation is:

$$K_p = \begin{bmatrix} 0.664229 & -0.538236 \\ 0.028931 & -0.855651 \\ 0.592670 & 0.105045 \\ 0.481856 & 0.671179 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 0.825255 & -0.988822 \\ 0.110543 & -1.678788 \\ 3.007867 & -0.633454 \\ 1.505329 & 2.161824 \end{bmatrix}$$

And under the **minimum value of sum of all ISE** PI controller, the outputs for various conditions are considered:

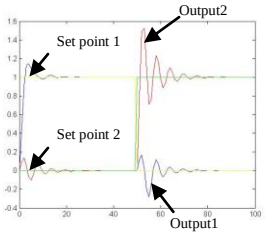

Figure 3.71: The output with non-failure

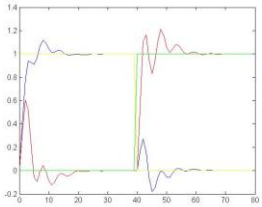

Figure 3.72: The output with Actuator 1 failure

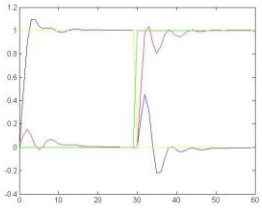

Figure 3.73: The output with actuator 2 failure

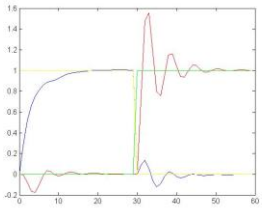

Figure 3.74: The output with actuator 3 failure

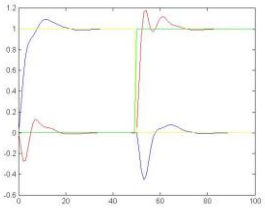

Figure 3.75: The output with actuator 4 failure

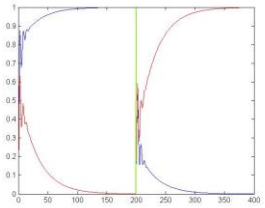

Figure 3.76: The output with actuator 1 and 2 failure

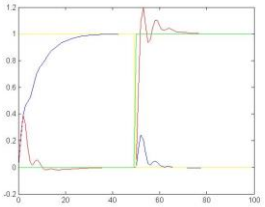

Figure 3.77: The output with actuator 1 and 3 failure

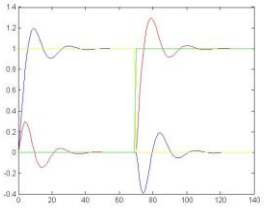

Figure 3.78: The output with actuator 1 and 4 failure

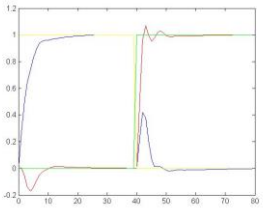

Figure 3.79: The output with actuator 2 and 3 failure

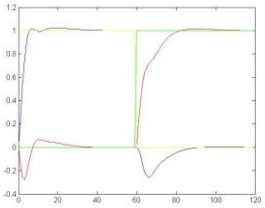

Figure 3.80: The output with actuator 2 and 4 failure

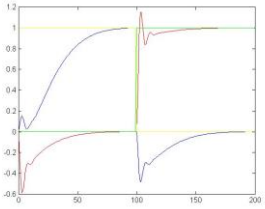

Figure 3.81: The output with actuator 3 and 4 failure

As the red value marked in the table, this is the minimum value of ISE for actuator 3 and 4 failure, so this is the one of the Pareto front solution which might be chosen as a final design.

The PI controller at this particularly Pareto front situation is:

$$K_p = \begin{bmatrix} 0.798679 & -0.772639 \\ -0.094392 & -0.863069 \\ 1.063109 & 0.036374 \\ 0.114809 & 0.166919 \end{bmatrix}$$

$$K_i = \begin{bmatrix} 1.047440 & -0.883092 \\ 0.061527 & -1.367356 \\ 2.064433 & -0.661527 \\ 0.891341 & 2.169557 \end{bmatrix}$$

And under the **minimum value of ISE for actuator 3 and 4 failure** PI controller, the outputs for various conditions are considered:

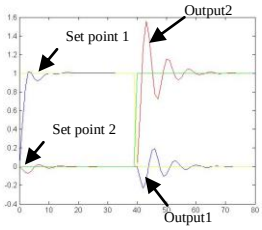

Figure 3.82: The output with non-failure

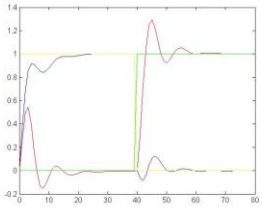

Figure 3.83: The output with Actuator 1 failure

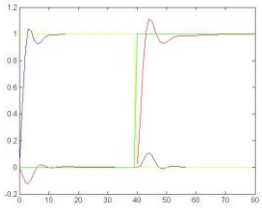

Figure 3.84: The output with actuator 2 failure

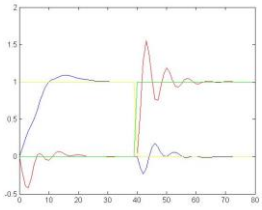

Figure 3.85: The output with actuator 3 failure

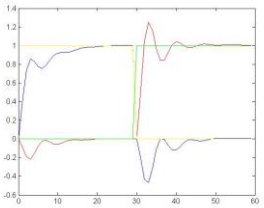

Figure 3.86: The output with actuator 4 failure

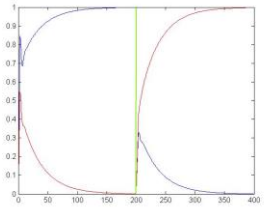

Figure 3.87: The output with actuator 1 and 2 failure

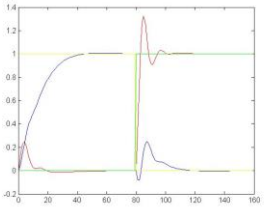

Figure 3.88: The output with actuator 1 and 3 failure

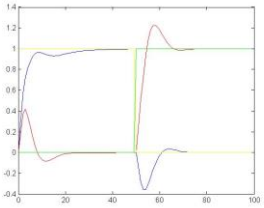

Figure 3.89: The output with actuator 1 and 4 failure

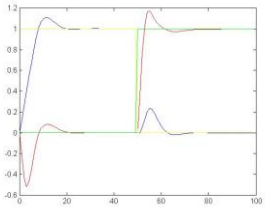

Figure 3.90: The output with actuator 2 and 3 failure

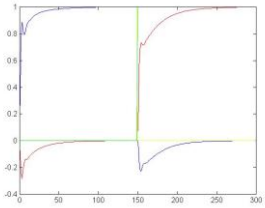

Figure 3.91: The output with actuator 2 and 4 failure

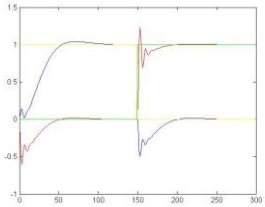

Figure 3.92: The output with actuator 3 and 4 failure

3.6 Conclusion

In this chapter, the design of multivariable fail-safe control system under non-actuator failure, single actuator failure and multiple actuator failures has been considered. Both single cost function genetic algorithm and multi-objective genetic algorithm have been used when applied to the same multivariable fail-safe system.

As the results shown in this chapter, the conclusions are:

- The genetic algorithm approach to the design of fail-safe controller was very successful. The genetic algorithm can evolve the controller for fail-safe system has good system performance under all non-actuator failure and single actuator failure and multiple actuator failures. Which there is no formal design method could do. Indeed, a full set of fail-safe controller was evolved by the genetic algorithm and optimal controller performance was achieved. Like the table 3.4 shows the single cost function genetic algorithm could improve the system ISE performance about 99% compare with Pseudo inverse design.
- Using a single cost function for the genetic algorithm resulted in a minimum cost performance for the overall system. However, individual cost function cannot be optimised. As the table 3.5 shows the sum of ISE is equal to 99.85 which is minimum compare with Pseudo Inverse design but the ISE for actuator 3 and 4 failure is 27 still larger than the rest of ISE.
- By using a worst case cost function, the worst case can be optimised. However, the cost for the overall solution is not globally optimal. As the table 3.5 shows, the ISE for actuator 3 and 4 is 25 which is smaller compare with global optimisation, but the sum of ISE is 145 which is larger than the global optimisation result.
- The multi-objective genetic algorithm was able to design a family of solutions. The family of solutions include the minimisation of the sum of cost function. As the table 3.6 shows the minimum ISE in ‘sum of all ISE column’ 104 is close to the ISE 99.85 in the single cost function genetic algorithm design. This family of solutions present all of the interesting and different useable design, like the minimum ISEs in each column in table 3.6 shows there are more than one design in one multi-objective genetic algorithm run,

for this reason the multi-objective genetic algorithm is used as an extremely effective design tool. As the table 3.6 shows the multi-objective genetic algorithm evolve a family of solutions.

According to these conclusions, the recommendations are:

- To use the Pseudo Inverse design method to design the controller parameters first; then use this design result as the start point for the genetic algorithm. This will guarantee the genetic algorithm will evolve a solution and it will have a good chance to evolve a better solution.
- To use the global optimisation cost function if the designer is looking for the optimal total cost. To use the worst failure case optimisation cost function if the designer is looking for the optimise the worst failure case.
- To use single cost function genetic algorithm if the designer knows the trade-off performance weighting factor in each non-actuator failure and all actuator failure combinations and the designer looking for a single optimal solution.

To use the multi-objective genetic algorithm if the designer is looking for the all combination of trade-off performance in each non-actuator failure and all actuator failure combinations. Like table 3.6 shows, the designer could choose any combination of trade-off performance design depends on their need.

Chapter Four

GENETIC DESIGN OF MULTIVARIABLE OVERRIDE CONTROL SYSTEM

4.1 Introduction

In order to guarantee set-point tracking in a multivariable control system, there needs to be at least as many inputs as outputs (Acikmese, 2001). In the case where there are more outputs than inputs not all the outputs can be controlled simultaneously. Rather a sub-set can be controlled. In many systems with more outputs than inputs high output values are to avoided, these could be high speeds, high pressures or high temperatures. In such cases the worst sub-set has to be controlled. This leads to the concept of override control. In override control there are a number of controllers that could be used but only the controller which control the worst variables are used. If the worst variable changes the controller must switch to new worst case set.

In 1971 Buckley P.S. first introduced the override control concept with feedforward control systems (Buckley, 1971). Since then the override control has become popular. After his research, lots of researchers have used override control as a tool to design switching control systems but there has been little published results on how to improve the theory to deal with the override control system for both single input multi output and multi input multi output systems. Moreover, most of the research has been on single input multi output override control system, as Buzzard W.S. only used the single input multi output override strategies for analog control (Buzzard, 1978). Additionally, as the override control system cannot control all outputs the main analysis of override control system design is around avoiding limit cycles. Furthermore, all the designs for override control system is for a specific

situations, and there are no general theory for the design of override control system. However, genetic algorithm can be used to design override control system.

One of the classic example of override control system is jet engine control system. This type of system can be shown like figure 4.1:

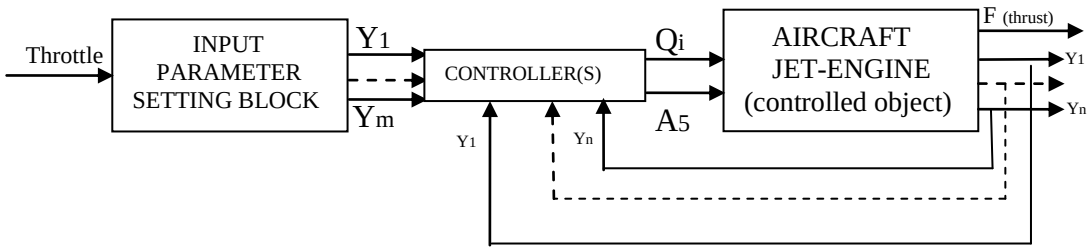

Figure 4.1: Aircraft engine's automatic control system

Mostly, aircraft engines are using fuel flow rate and inlet guide vanes to control engine's spool speed and the engine's burned gas temperature and the total thrust (Tudosie, 2011). Only two of three variables can be controlled at any time. If engine speeds and temperature become too high, the engine thrust is not controlled as the control system switches to control the two variables which are too high.

The classic override control system can be drawn like figure 4.2 and 4.3 shows (Alejandro and Joseph, 2003):

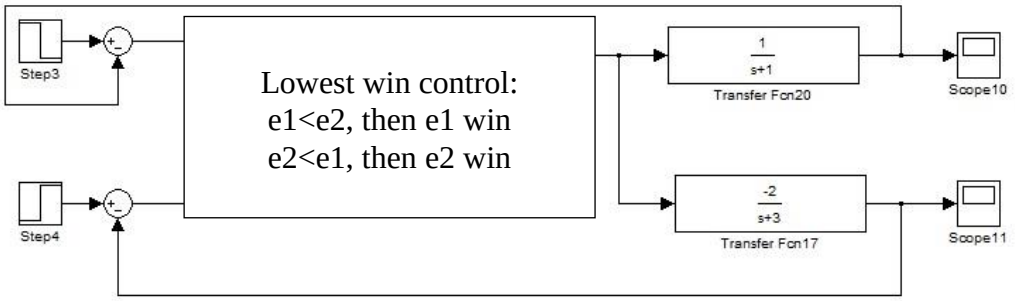

Figure 4.2: override control when loop one under control

Like the figures shown in figure 4.2, override control could switch between each loop if the switch condition has satisfied. The switch most commonly used switches between a PI controller using a 'lowest win control' strategy to ensure the most negative error is controlled.

When the control system switches from one loop to another the integrator value has to be carefully dealt with during the switch. If it is not done correctly a bump in the output will occur. For this reason all override controller need a bumpless transfer mechanism.

4.1.1 Absolute input and incremental input

There are two Proportional-integral controller forms absolute and incremental. The absolute controller is given by:

$$u_k = K_p e_k + K_i z_k \quad \text{.....(4.1)}$$

Where

$$z_k = z_{k-1} + T e_{k-1} \in R^m \quad \text{.....(4.2)}$$

And the incremental controller is given by:

$$u_k = u_{k-1} + K_p\left(e_k - e_{k-1}\right) + K_i e_k \quad \text{.....(4.3)}$$

4.1.2 Bumpless transfer:

If an incremental form of PI control is used the bumpless transfer of the integrator is avoided as the incremental form does not have an integrator state in the algorithm. Therefore, the bumpless transfer need to be carried out only with absolute form of PI controller (Peng et al, 1996). This means when switch occur the integral value z_k should be revaluated as:

$$z_k = inv\left(k_i\right) \times \left(u_k - k_p \times e_k\right) \quad \text{.....(4.4)}$$

$Z_k = inv(k_i) \times (u_k - k_p \times e_k)$

4.1.3 Genetic algorithm for override control system design

This chapter is going to use the genetic algorithm to design a multivariable override control system using a cost function. Therefore, the cost function is very important. The cost function selection is also a measure of the controlled system's performance. These measures are used to compare the system's performance between different control situation or different controller parameters. This chapter is going to use the: Integral Squared Error (ISE).

The total ISE is equal to the sum of ISE for each output, and each output ISE is equal to the ISE calculated by the set point tracking plus the ISE calculated by the interaction. For example, there is a two inputs and three output system:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} g_{11}(s) & g_{12}(s) \\ g_{21}(s) & g_{22}(s) \\ g_{32}(s) & g_{33}(s) \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Because this is two inputs three outputs override control system, there are three ISEs for three different system control situations.

The individual ISE is calculated as:

- $ISE_t = \Sigma ISE_{i,j}$

where

- t is the control loop number, t=12 means loop 1 and 2 under control, t=13 means loop 1 and 3 under control, t=23 means loop 2 and 3 under control
- i is the output number
- j is the set point change number

If the system is under loop 1 and 2 control, the total ISE for the override control system when loop 1 and 2 under control:

$ISE_{12} = (ISE_{1,1} + ISE_{1,2} + ISE_{2,1} + ISE_{2,2})$

If the system is under loop 1 and 3 control, the total ISE for the override control system when loop 1 and 3 under control:

$ISE_{13} = (ISE_{1,1} + ISE_{1,3} + ISE_{3,1} + ISE_{3,3})$

If the system is under loop 2 and 3 control, the total ISE for the override control system when loop 2 and 3 under control:

$ISE_{23} = (ISE_{2,2} + ISE_{2,3} + ISE_{3,2} + ISE_{3,3})$

Therefore, for the single cost function GA, the cost function ISE (Integral Square of Error) is calculated by:

$$e = \int (\varepsilon)^2 dt = (\omega_1 \times ISE_{12} + \omega_2 \times ISE_{13} + \varpi_3 \times ISE_{23})$$

where ϖ_i is the weight factor of each ISE.

The objective function for a single cost function genetic algorithm used in this chapter could be one of two types. The first type of objective is global optimisation: this kind of objective is calculated by the ISE of set point tracking plus the ISE of interaction for all situations (Solihin et al, 2008). In this objective, the genetic algorithm will design a controller which will minimise the total sum of all the ISE. The second type objective is worst case optimisation: this type of objective is calculated by the largest ISE of set point tracking plus the ISE of interaction in all situations. In this objective, the genetic algorithm will design a controller which will make the largest of ISE out of all ISE as small as possible. To compare those two objective functions, the first one will give the minimum total ISE, but maybe some of the individual loop ISE will have a large value; the second objective will make sure there are no single large ISE value for a single loop, by focusing the algorithm on brings the worst one down. However, this will be at the cost of a slightly higher overall performance cost.

For the multi-objective genetic algorithm, the objectives are the individual loop cost functions. The multi-objective genetic algorithm will find a family of solutions using a non-dominated Pareto front method. In this design, the multi-objective genetic algorithm will minimise the ISE for all loop combinations situations simultaneously.

In genetic algorithm, the parameter range is very important, because it is related to the genetic algorithm will finds out the optimal solution or not, or how quick the genetic algorithm finds out optimal solutions. Moreover, the parameter range normally is very hard to choose. Therefore, the parameter range movement has been included into the genetic algorithm program. The parameter range movement method is when the best individual becomes close to the upper or the lower range limit, and then the whole range moves up or down. Normally the "close to the upper or the lower range limit" means 10% of the total range size, and how much percentage is chosen by the designer. Normally if the ranges need to be moved, the range will move up or down by 20% of the total range, and how much percentage to move is chosen by the designer as well. After the range movement, the GA needs to re-decode and re-scale the parameter range again, to make sure the GA binary number and the real parameter number matched. Without adopted this technique, the parameter range is not grantee to coverage the optimal solution.

4.1.4 Limit cycle in override control system

In 1981, Foss A.M. has drawn attention to a major concern with override control namely that limit cycle can occur. Indeed the system chosen by Foss is shown below (Foss, 1981):

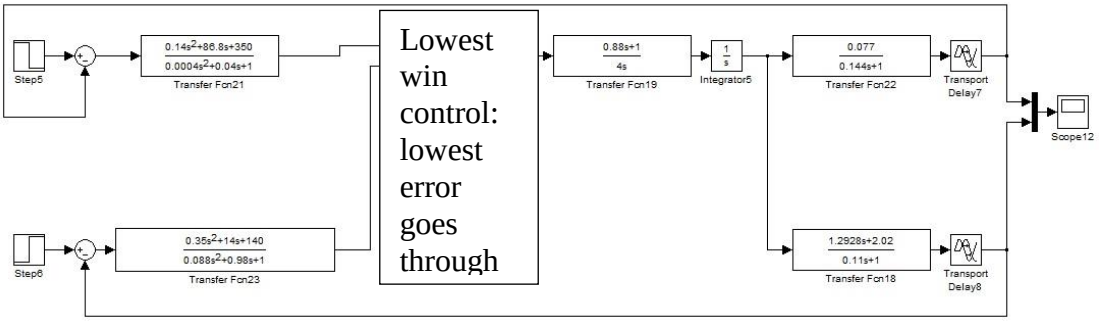

Figure 4.3: Foss's override control system

The system Foss's proposes shows that limit cycle could happen in certain ranges of the set point. He also explains that limit cycle can occur when a limit is lowered. Thus an override control system could exhibit good transient responses for a range of set-point changes.

However, for another range of set points the system could start to limit cycle. For this reason it is important to know the exact range which the override system will not limit cycle.

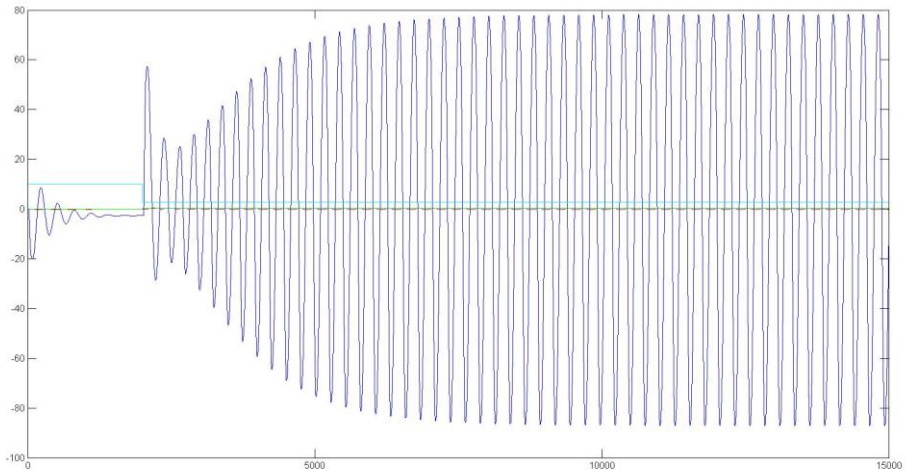

Figure 4.4: Foss's override control system without and with limit cycle when set point changing

As the figure show above, when the set point are v1=-0.1 v2=10 at the beginning, there is no limit cycle. However, when the set point changes to v1=0.2 v2=2.7, then the system limit cycles. The boundary of set point over which limit cycle will not occur involves a search on each set point. The solution space is the same as a non-dominated Pareto front. This problem of detecting when a limit cycle will occur can thus be solved using a multi-objective genetic algorithm.

4.2 Genetic design of single input multi output override control systems

In this section, a one input two outputs override control system is introduced. This override control system has two controllers. The controllers will switch if the errors on the control loops change.

In this example, there are two different control systems to be designed:

- Loop one under control
- Loop two under control

Because the override control system is focused on the lowest error, the cost function for the override control will be the control loops that is active.

When investigating override control system the outputs will react to the controller inputs. Systems where one output goes positive and another goes negative for the same input are problematic in override situations. This is because when a limit is lowered the controlled output goes down, but the other output goes up and force a switching to take place which can in some cause result in limit cycles.

The one input two outputs override control system is shown in figure 4.3

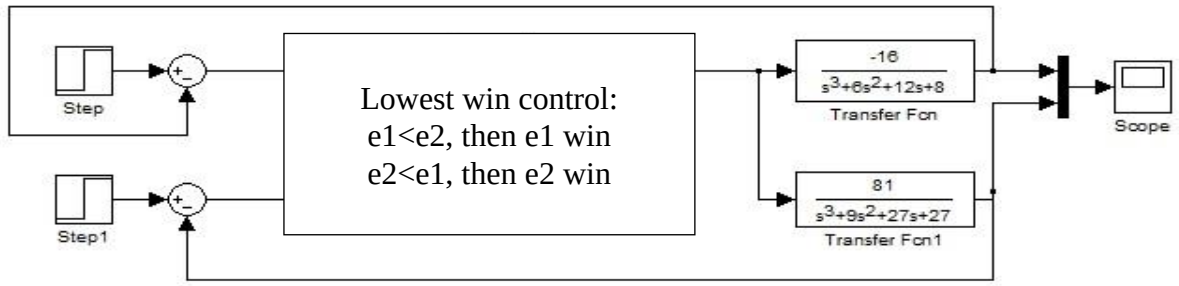

Figure 4.5: One input two output override control system with negative sign in steady-state transfer function matrix

As figure 4.3 shows, the two transfer functions are

$$g_1(s) = \frac{-16}{s^3 + 6s^2 + 12s + 8} \quad \text{.....(4.5)}$$

and

$$g_2(s) = \frac{81}{s^3 + 9s^2 + 27s + 27} \quad \text{.....(4.6)}$$

Because there are two error e_1 and e_2, if e_1 wins, the $\Delta u_k = K_p \Delta e_1 + K_i e_1$; if e_1 wins, the $\Delta u_k = K_p \Delta e_2 + K_i e_2$; then the control system is updated $u_k = u_{k-1} + \Delta u_k$. So there are two PI controller K_{p1} and K_{i1} for loop one under control and K_{p2} and K_{i2} for loop two under

control, and kp_1, kp_2, ki_1 and ki_2 are the four parameter need to be designed by the genetic algorithm.

The parameter ranges are:

$K_{p1} = [-0.02952 \quad -0.01968]$ and $K_{i1} = [-0.1998 \quad -0.1332]$

$K_{p2} = [0.01968 \quad 0.02952]$ and $K_{i2} = [0.1332 \quad 0.1998]$

The Global Optimisation cost function is calculated by adding the ISE for loop one when under control plus the ISE for the loop two when under control. The single cost function genetic algorithm is going to minimum this cost function. When the single cost function GA has applied, the Max is chosen as 50, the parameter range movement code is used. The genetic algorithm was run for 5000 generations with a population size of 100, the sampling time is 0.1. The following values were then obtained from the genetic algorithm:

The genetic algorithm results are:

$K_{p1} = -0.0489$ and $K_{i1} = -0.3310$ for loop one under control.

$K_{p2} = 0.0472$ and $K_{i2} = 0.319$ for loop two under control

The set point for loop one V1 is set at 1.0 and changes to 2 after 600 sample time. The set point for loop two V2 is set at 2 and changes to -2 at 600 sample time.

Therefore, the output of the system is as figure below shows

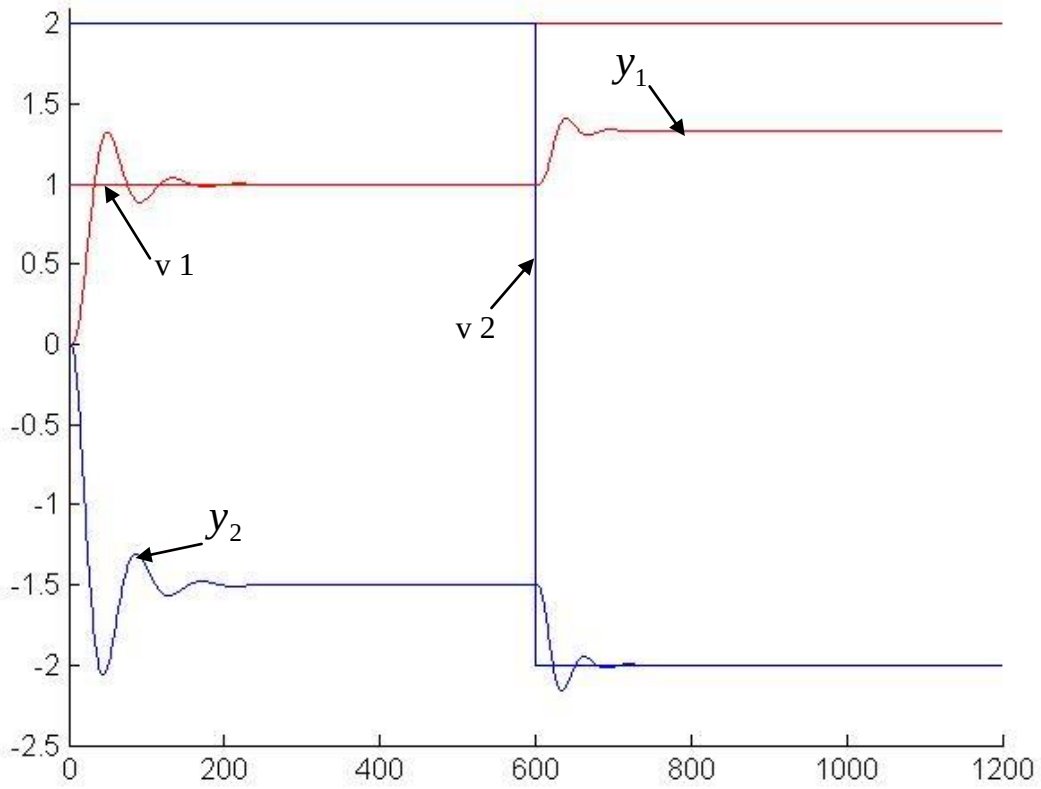

Figure 4.6: System output of the one input two output override control system with negative sign in steady-state transfer function matrix

The output for system

$$g_1(s) = \frac{-16}{s^3 + 6s^2 + 12s + 8} \text{ is } y_1;$$

the output for system

$$g_2(s) = \frac{81}{s^3 + 9s^2 + 27s + 27} \text{ is } y_2.$$

As the figure 4.6 shows, for the first 600 seconds G_1 is under control. However, at 600 seconds, the set point one V1 has changed from 1 to 2 and the set point two V2 has changed to from 2 to -2, then the system G_2 is under control. In this Simulation, the override control system ensures one output is tracking and the other is below its set point.

Then the ISE for the override control system are:

The ISE of loop one under control $e_1 = 21.3870986798534$

The ISE of loop two under control $e_2 = 3.76516329574505$

4.2.1 Limit Cycle override control system

In some situations a control system can be designed which appear to be stable. However, if the set points are changed to other values a limit cycle will occur. For example if after 600 seconds the set point for loop 1 is set at V_1= 1.0, then a limit cycle starts. The output of the system is shown in figure 4.7

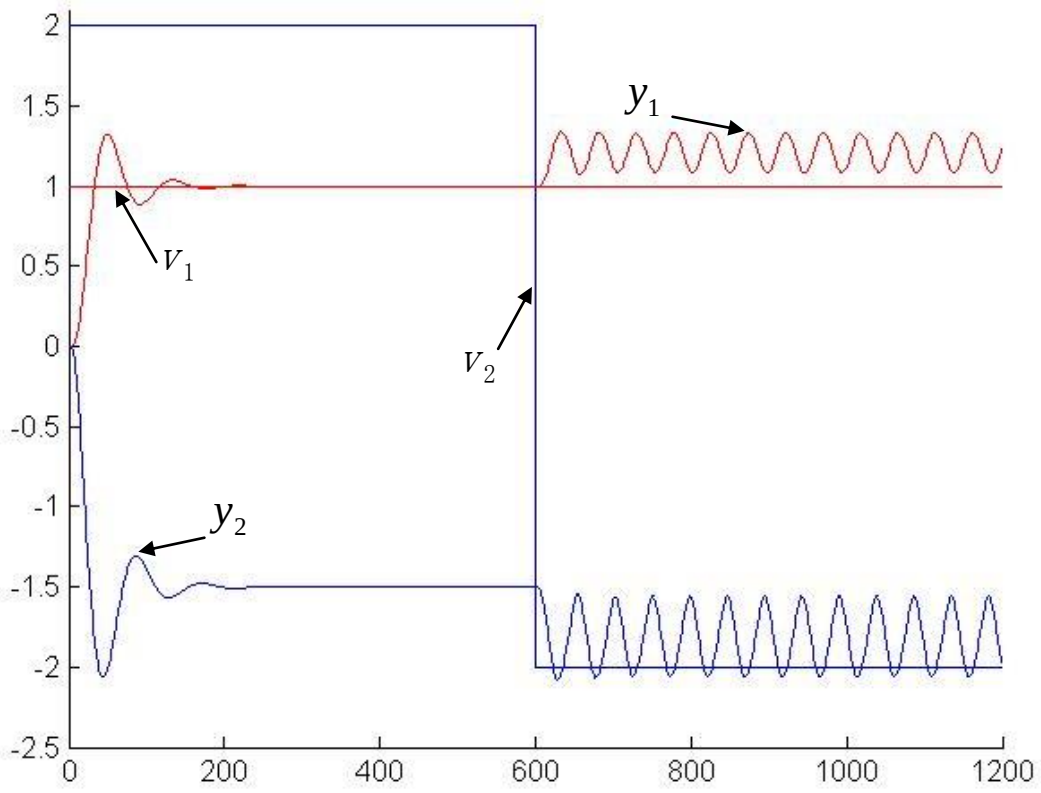

Figure 4.7: System output of the one input two output override control system with negative sign in steady-state transfer function matrix

If the set point returned to V2 equal to 2.0 then the limit cycle will stop. Thus when designing an override control system it is also important to know what range of set point make sure the system remain stable and not limit cycle.

4.3 Multi-objective Genetic Algorithm search for the limit cycle boundary for single input multi outputs override control systems

In this section, a one input two outputs override control system is introduced. This override control system has two controllers. The controllers will switch if the errors on the control loops change.

In this example, there are two different control systems to be designed:

- Loop one under control
- Loop two under control

Once an override control system has been designed. It is important to find out the range of set point when it will not limit cycle. In order to determine the set point values a multi-objective genetic algorithm can be deployed. This is because the search involves looking at each set point in turn and looking for the boundary between stability and limit cycling. The non-dominated Pareto front approach is well suited to solve such a problem.

As figure 4.17 shows, the one input two output override control system shown below.

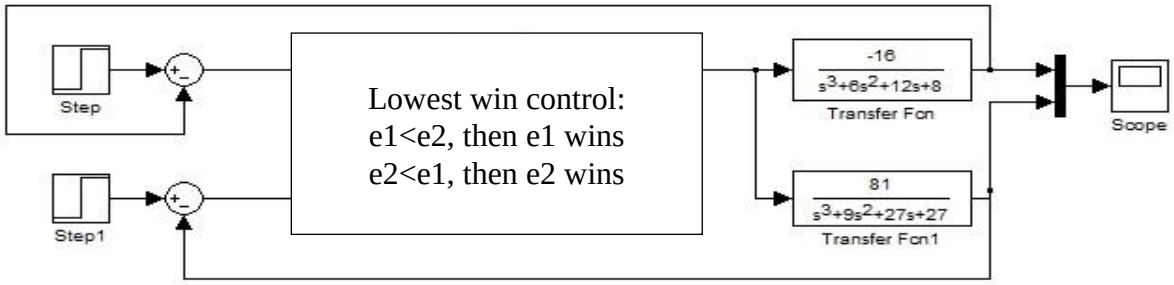

Figure 4.8: One input two output override control system

As figure 4.8shows, the two transfer functions are

$$g_1(s) = \frac{-16}{s^3 + 6s^2 + 12s + 8} \quad \text{.....(4.7)}$$

and

$$g_2(s) = \frac{81}{s^3 + 9s^2 + 27s + 27} \quad \text{.....(4.8)}$$

As the pervious result shows in this chapter, this system is more likely have limit cycle if the set point is in a specific range. Because there are two set points and the limit cycle will happen with the combination of reduce those two set points. The system loop will switch when other loop's error become lower. So the limit cycle boundary is very hard to find. Therefore, it is necessary to search for the boundary of the combination of two set points such that the system will not limit cycle. This type of objective is a Pareto front.

To find the boundary between set point V_1 and set point V_2 when the system will start to limit cycle the multi-objective genetic algorithm is deployed, the parameters used by the multi-objective genetic algorithm designs are the two set points V_1 and V_2. The parameter range is chosen from the set point values when the override control system has limit cycles. The parameter ranges are:

$V_1 = [-1.5 \quad 2]$

$V_2 = [-3 \quad 2]$

As the multi-objective genetic algorithm is searching only the non limit cycle set point value will be kept in the population non-dominated solution will be given a higher fitness.. The controller is fixed as

$K_{p1} = -0.0246$ and $K_{i1} = -0.1665$ for loop two under control

$K_{p2} = 0.0246$ and $K_{i2} = 0.1665$ for loop one under control

The length of bit is 10, and multi-objective genetic algorithm is going to search the two set points simultaneously. If the set points make the override control system limit cycle, then this individual will be removed from the population. The genetic algorithm was run for 1000 generations with a population size of 1000.

The multi-objective genetic algorithm was then used, and the plot of two set points which ensure the override control system does not limit cycle is shown in figure 4.9

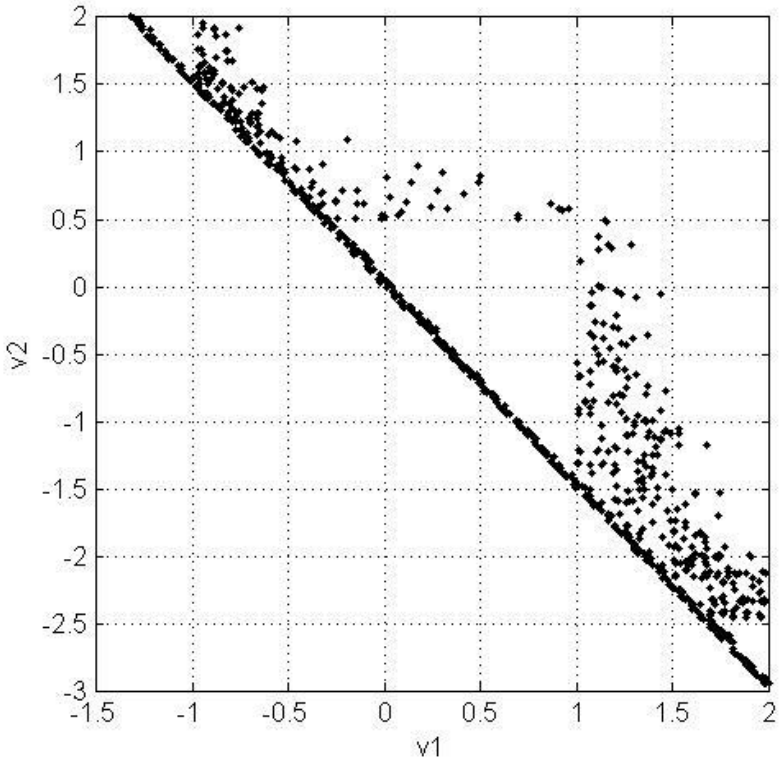

Figure 4.9: set points for no limit cycle Pareto front plot

The plot in figure 4.9 shows the set point boundary V1 and V2 when limit cycles will occur. Any combination of the two set points which are below the boundary will result in the override control system limit cycling.

When the set points are chosen above the boundary means the override control system will not limit cycle, the two set point value are V1=0 andV2=0.1. The set point one V1 is equal to 1 at start of simulates, and set point twoV2 equal to 2; when the two of system outputs are stable, at 200 seconds, the set point one V1 changes to 0 and set point two V2 changes to 0.1.

Therefore, the output of the system is as figure 4.10 shows

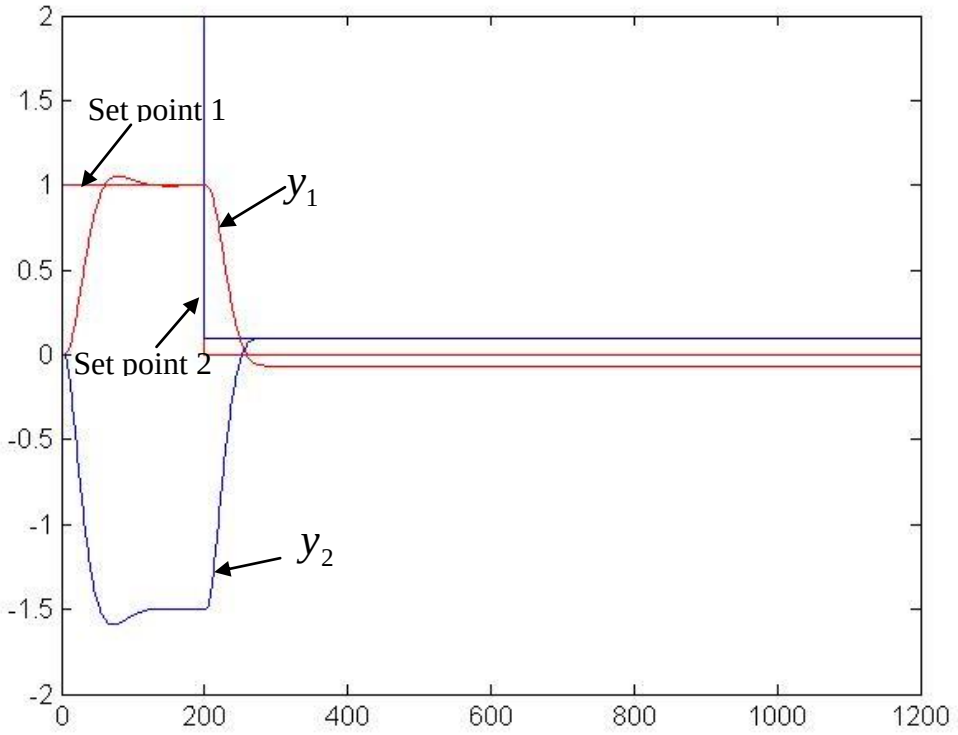

Figure 4.10: System output of the one input two output override control system with negative sign in steady-state transfer function matrix

As figure show above, there is no limit cycle in the override control system.

However, When the set point is chosen below the boundary the override control system will limit cycle, the two set point value are $V_1 = 0$ and $V_2 = -0.5$. The set point $V_1 = 1$ at start of the simulation, and the set point $V_2 = 2$. Both system outputs are stable, at 200 seconds, the set point is lowered such that $V_1 = 0$ and set point $V_2 = -0.5$.

Therefore, the output of the system is as figure 4.11 shows

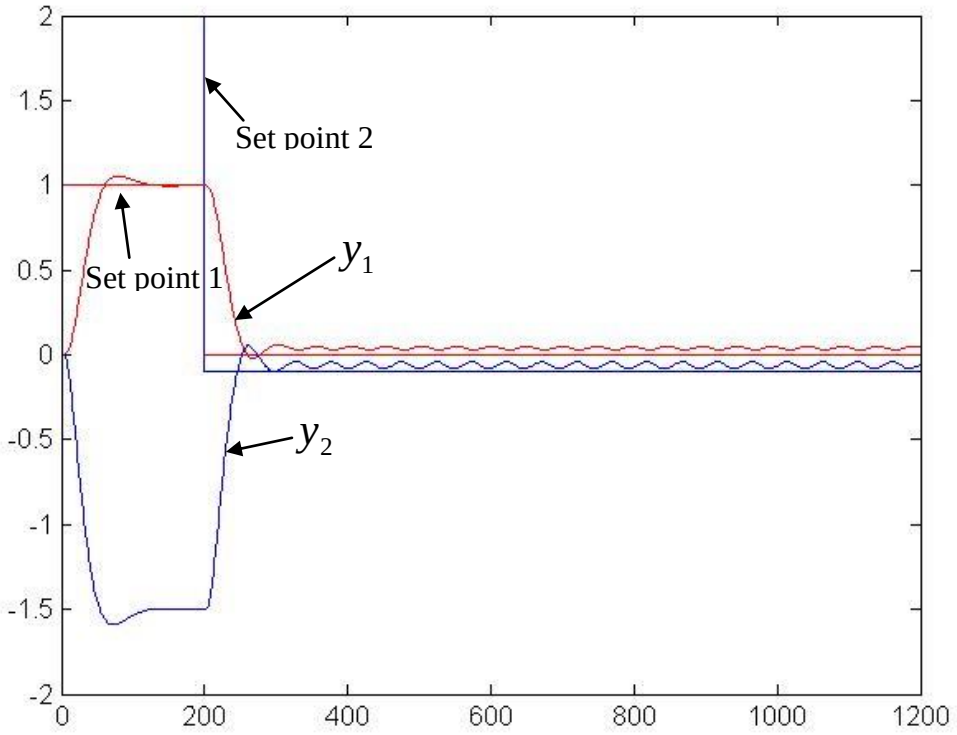

Figure 4.11: System output of the one input two output override control system with negative sign in steady-state transfer function matrix

As figure shows in figure 4.11, there is limit cycle in the override control system. This feature of the multi-objective genetic algorithm to find out the boundary of limit cycling is a very useful result.

4.4 Multi-objective Genetic Algorithm design of single input multi outputs override control systems

In this section, a one input two outputs override control system previously introduced is considered. This override control system has two controllers. The controllers will switch if the errors on the control loops change.

In this example, there are two different control systems to be designed:

- Loop one under control
- Loop two under control

There are two ways to trade-off the performance between loops. The first technique is adding weighting factor on all loops. However, the value of weighting factor is very hard to choose. However, if the multi-objective genetic algorithm is used to search for the controllers, the use of weighting factors can be avoided, because the multi-objective genetic algorithm uses a non-dominated Pareto front method to provide a family of solutions.

As figure 4.12 shows, the one input two output override control system shown below.

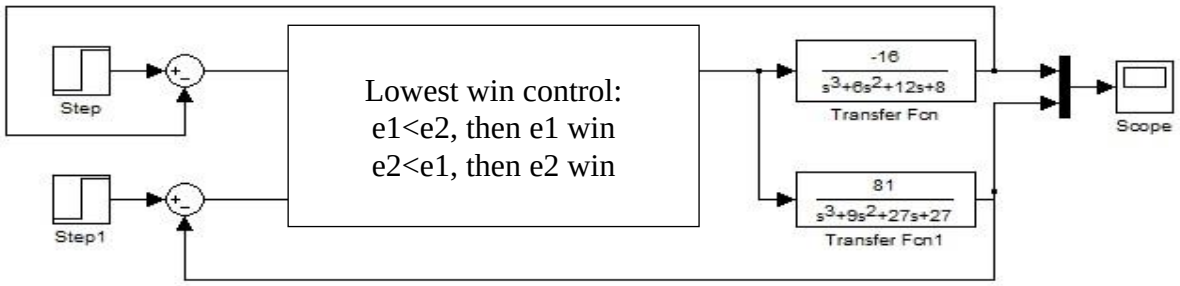

Figure 4.12: One input two output override control system

As figure 4.12 shows, the two transfer functions are

$$G_1 = \frac{-16}{s^3 + 6s^2 + 12s + 8} \quad \text{.....(4.9)}$$

and

$$G_2 = \frac{81}{s^3 + 9s^2 + 27s + 27} \quad \text{.....(4.10)}$$

Multi-objective genetic algorithm also could design a family of controller that trade-off the two system loop performances. With this multi-objective genetic algorithm, the parameters to be found by the multi-objective genetic algorithm are the controllers for each loop under control. The parameter range is chosen as same as single cost function genetic algorithm Global Optimisation situation. The population will be selected based on the non-dominated Pareto front method.

When using the multi-objective genetic algorithm the two cost function are ISE_1 which is the ISE for loop 1 under control and ISE_2 which is the ISE for loop 2 under control. A trade-off between the performance in each loop can then be made by using the data from figure 4.13

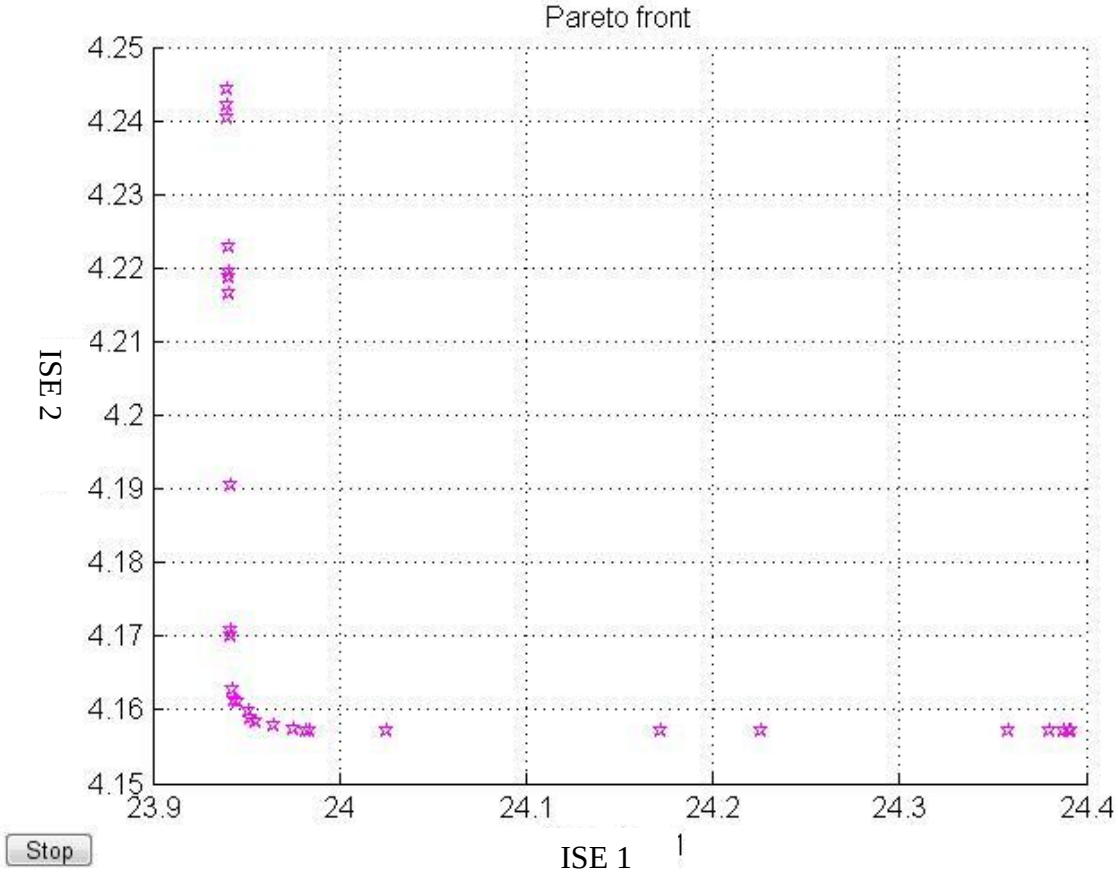

Figure 4.13: Pareto front plot for the trade-off performance

The plot shows the points for the Pareto front of the two control loop ISE's for the override control system.

4.5 Multi inputs multi outputs override control system

In this section, a two inputs three outputs override control system is introduced, this override control system has three controllers which will switch if the errors in the control loop change.

The design of multiple inputs multiple outputs override control systems is make more complex, because of the possibility of the system limit cycling under certain set point values. In this example, there are three different control systems to be designed:

If e1<e3 and e2<e3, then loop one and two under control:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_{p12} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + \begin{bmatrix} k_{i12} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

where $z_1 = \int e_1$ and $z_2 = \int e_2$

The cost function for the period of system is $ISE_{\text{loop 1 and 2 under control}} = ISE_{\text{set-point tracking}} + ISE_{\text{interaction}}$ for both loops have same size set point changing.

If e1<e2 and e3<e2, then loop one and three under control

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_{p13} \end{bmatrix} \begin{bmatrix} e_1 \\ e_3 \end{bmatrix} + \begin{bmatrix} k_{i13} \end{bmatrix} \begin{bmatrix} z_1 \\ z_3 \end{bmatrix}$$

where $z_1 = \int e_1$ and $z_3 = \int e_3$

The cost function for the period of system is $ISE_{\text{loop 1 and 3 under control}} = ISE_{\text{set-point tracking}} + ISE_{\text{interaction}}$ for both loops have same size set point changing.

If e2<e1 and e3<e1, then loop two and three under control

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_{p23} \end{bmatrix} \begin{bmatrix} e_2 \\ e_3 \end{bmatrix} + \begin{bmatrix} k_{i23} \end{bmatrix} \begin{bmatrix} z_2 \\ z_3 \end{bmatrix}$$

where $z_2 = \int e_2$ and $z_3 = \int e_3$

The cost function for the period of system is $ISE_{\text{loop 2 and 3 under control}} = ISE_{\text{set-point tracking}} + ISE_{\text{interaction}}$ for both loops have same size set point changing.

Therefore, the cost function for the single cost function genetic algorithm is $ISE_{\text{single cost function}} = ISE_{\text{loop 1 and 2 under control}} + ISE_{\text{loop 1 and 3 under control}} + ISE_{\text{loop 2 and 3 under control}}$.

This section is going to show that the multi-inputs multi-outputs override control system can sometime be stable for some set point values and limit cycle for other set point values. This is demonstrated through two examples. The first example the multi-inputs multi-outputs override control system does not have limit cycles. The second multi-inputs multi-outputs override control system does exhibit limit cycles.

4.5.1 No limit cycle override control:

The block diagram for the multi input multi output override control system is shown in figure 4.14:

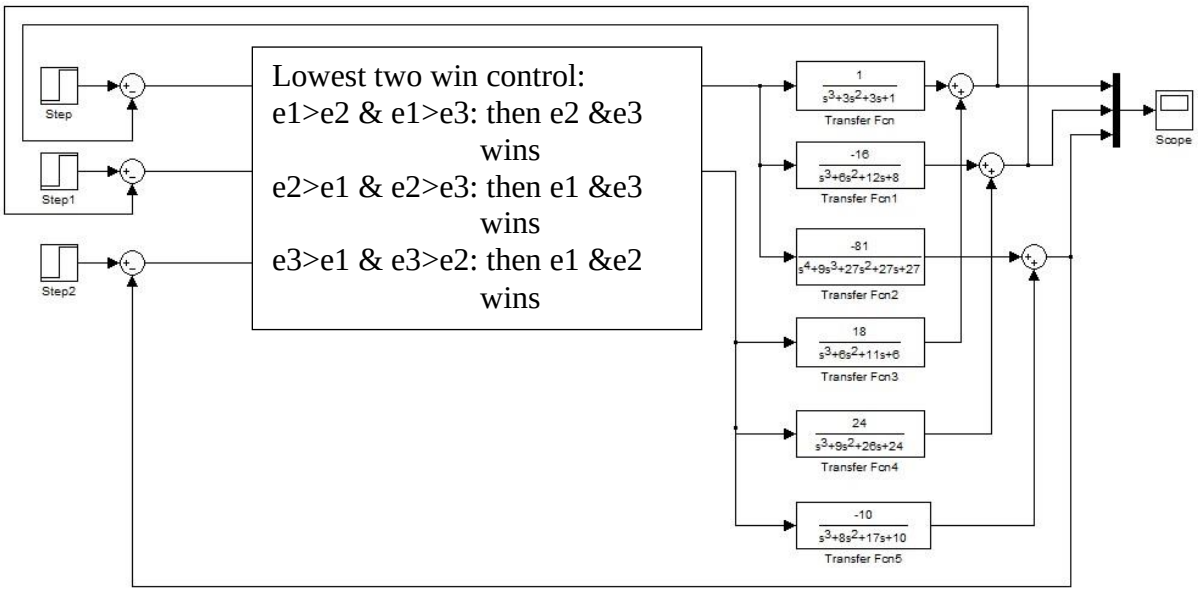

Figure 4.14: Multi input multi output override control system block diagram

As figure above shows, the six transfer functions are

$$G_1 = \frac{1}{s^3 + 3s^2 + 3s + 1} \quad \text{.....(4.11)}$$

$$G_2 = \frac{-16}{s^3 + 6s^2 + 12s + 8} \quad \text{.....(4.12)}$$

$$G_3 = \frac{81}{s^3 + 9s^2 + 27s + 27} \quad \text{.....(4.13)}$$

$$G_4 = \frac{-18}{s^3 + 6s^2 + 11s + 6} \quad \text{.....(4.14)}$$

$$G_5 = \frac{24}{s^3 + 9s^2 + 26s + 24} \quad \text{.....(4.15)}$$

and

$$G_6 = \frac{-10}{s^3 + 8s^2 + 17s + 10} \quad \text{.....(4.16)}$$

The controllers are:

If e1<e3 and e2<e3, then loop one and two under control:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_{p12} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + \begin{bmatrix} k_{i12} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

where $z_1 = \int e_1$ and $z_2 = \int e_2$

The cost function for the period of system is $ISE_{\text{loop 1 and 2 under control}} = ISE_{\text{set-point tracking}} + ISE_{\text{interaction}}$ for both loops have same size set point changing.

If e1<e2 and e3<e2, then loop one and three under control

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_{p13} \end{bmatrix} \begin{bmatrix} e_1 \\ e_3 \end{bmatrix} + \begin{bmatrix} k_{i13} \end{bmatrix} \begin{bmatrix} z_1 \\ z_3 \end{bmatrix}$$

where $z_1 = \int e_1$ and $z_3 = \int e_3$

The cost function for the period of system is $ISE_{\text{loop 1 and 3 under control}} = ISE_{\text{set-point tracking}} + ISE_{\text{interaction}}$ for both loops have same size set point changing.

If e2<e1 and e3<e1, then loop two and three under control

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_{p23} \end{bmatrix} \begin{bmatrix} e_2 \\ e_3 \end{bmatrix} + \begin{bmatrix} k_{i23} \end{bmatrix} \begin{bmatrix} z_2 \\ z_3 \end{bmatrix}$$

where $z_2 = \int e_2$ and $z_3 = \int e_3$

The cost function for the period of system is $ISE_{\text{loop 2 and 3 under control}} = ISE_{\text{set-point tracking}} + ISE_{\text{interaction}}$ for both loops have same size set point changing.

The cost function for this design is $ISE_{\text{single cost function}} = ISE_{\text{loop 1 and 2 under control}} + ISE_{\text{loop 1 and 3 under control}} + ISE_{\text{loop 2 and 3 under control}}$.

The design of single cost function genetic algorithm the cost function is the sum of set point tracking ISE plus the interaction ISE for the loop under control. The genetic algorithm was run for 5000 generation with a population size of 100, the following values were obtained from the genetic algorithm.

The PI controllers are

$$K_{p12} = \begin{bmatrix} -0.2635 & -0.4925 \\ -0.4376 & -0.0024 \end{bmatrix} K_{i12} = \begin{bmatrix} -0.0936 & -0.4687 \\ -0.2899 & -0.1351 \end{bmatrix}$$

And

$$K_{p23} = \begin{bmatrix} 0.0001 & 0.3207 \\ 0.9921 & 0.0021 \end{bmatrix} K_{i23} = \begin{bmatrix} 0.5290 & 0.5213 \\ 1.7916 & 0.8320 \end{bmatrix}$$

and

$$K_{p13} = \begin{bmatrix} -0.0002 & 0.2886 \\ -0.6579 & -0.0003 \end{bmatrix} K_{i13} = \begin{bmatrix} 0 & 0.3521 \\ -0.2505 & 0.0933 \end{bmatrix}$$

The set point one $V_1 = 1$ at the start of the simulation, and set point two $V_2 = 2$ and set point three $V_3 = 3$. When the three of system outputs are stable, the set point two is changed to $V_2 = 3$ and set point three is changed to $V_3 = -4$ and set point V_1 is remain constant, and when it is stable again, the set point one is changed to $V_1=5$ and set point two is changed to $V_2=2$, set point three V_3 is remain constant.

Therefore, the output of the system is like the figures shows

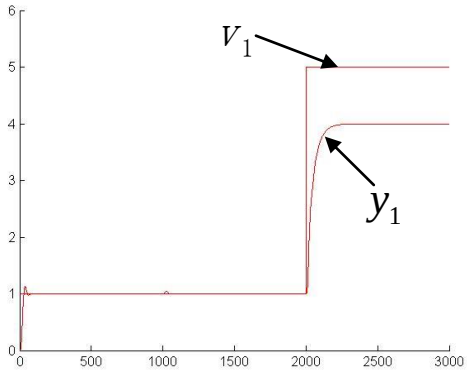

Figure 4.15: Output 1 for Multi input multi output override control system

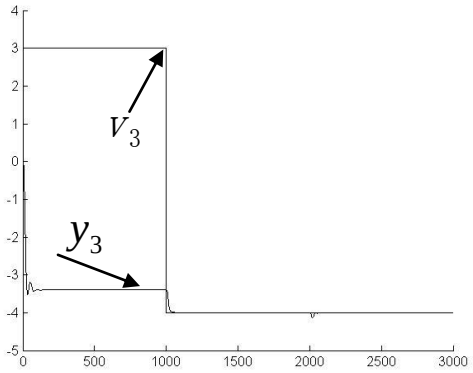

Figure 4.17: Output 3 for Multi input multi output override control system

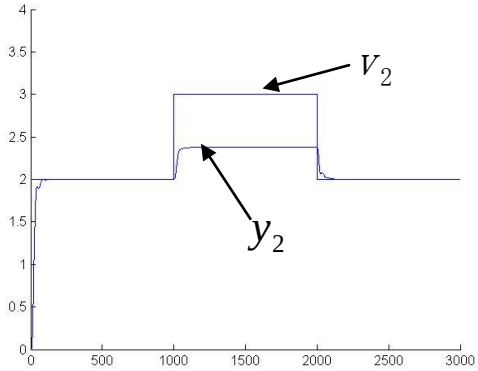

Figure 4.16: Output 2 for Multi input multi output override control system

As figure 4.15, 4.16 and 4.17 shows, the multi input multi output override control system is tracking and stable. Up to 1000 second, the output y_1 and y_2 are under control. Between 1000 and 2000 second, the output y_1 and y_3 are taking over control. And at the last part time period, the output y_2 and y_3 are under control.

4.5.2 Limit cycle override control:

However, for the same multi-inputs multi-outputs override control system, with the same controllers, if the set points changes to another range the system will exhibit limit cycle. For example, if the set point one V1= 1 at start of the simulation, and set point two V2= 20 and set point three V3= 10; and then the set point one V_1 is changed to V_1= -100 and the set point two V_2 is changed to V2= -14.2 and set point three V3 is kept constant, then the system will limit cycle. But if the set points are changed back to the initial values, the system will stop limit cycling.

Therefore, the output of the system is like the figures shows

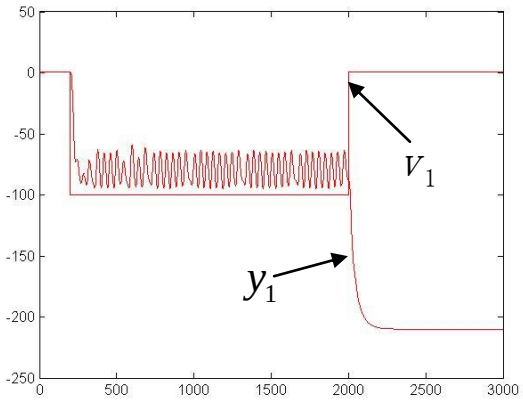

Figure 4.19: Output 1 for Multi input multi output override control system

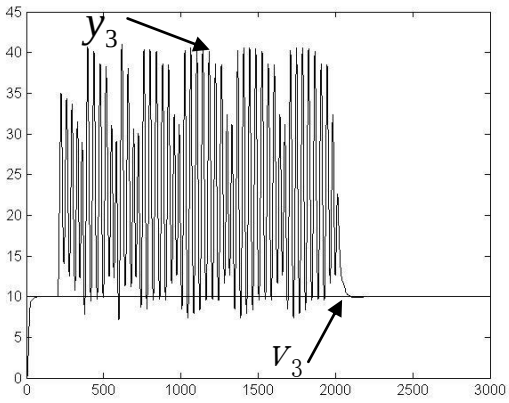

Figure 4.21: Output 3 for Multi input multi output override control system

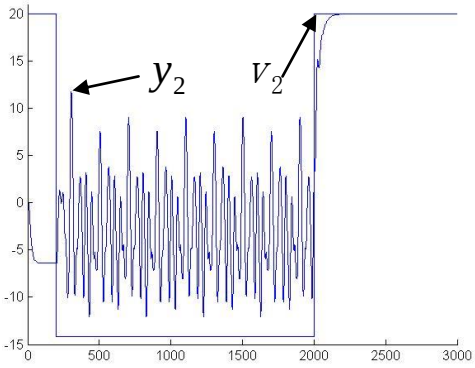

Figure 4.20: Output 2 for Multi input multi output override control system

As figure 4.19, 4.20 and 4.21 shows, the system at the start of the simulation is stable and tracking. But at the 200 to 2000 seconds the three outputs start to limit cycle. Finally it can be seen that all of them stop limit cycling when the set points are changed back to initial values.

4.6 Genetic design of multi inputs multi outputs override control system

The design of self-selecting multivariable controllers has been addressed by Jones A.H., Porter B. and Chrysanthou A (Jones, Porter and Chrysanthou, 1988). However, such controllers used a multivariable decoupling approach to design the controller structure, and the final controller had to be tuned manually. The multivariable override control system could be designed using genetic algorithms and this technique could design the controller by using an appropriate cost function for the genetic search.

The total ISE is equal to the sum of ISE for each output, and each output ISE is equal to the ISE calculated by the set point tracking plus the ISE calculated by the interaction. For example, there is a two inputs and three output system:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} g_{11}(s) & g_{12}(s) \\ g_{21}(s) & g_{22}(s) \\ g_{32}(s) & g_{33}(s) \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Then ISE for each individual outputs will be like below:

$$\begin{bmatrix} ISE_{11} & ISE_{12} \\ ISE_{21} & ISE_{22} \\ ISE_{31} & ISE_{32} \end{bmatrix}$$

Because this is two inputs three outputs override control system, there are three ISEs for three different system control situations.

If the system is under loop 1 and 2 control, the total ISE for the override control system when loop 1 and 2 under control:

$ISE_1 = [(ISE_{11} + ISE_{12})_{\text{set point tracking}} + (ISE_{21} + ISE_{22})_{\text{interaction}}] + [(ISE_{11} + ISE_{12})_{\text{interaction}} + (ISE_{21} + ISE_{22})_{\text{set point tracking}}]$

If the system is under loop 1 and 3 control, the total ISE for the override control system when loop 1 and 3 under control:

$ISE_2 = [(ISE_{11} + ISE_{12})_{\text{set point tracking}} + (ISE_{31} + ISE_{32})_{\text{interaction}}] + [(ISE_{11} + ISE_{12})_{\text{interaction}} + (ISE_{31} + ISE_{32})_{\text{set point tracking}}]$

If the system is under loop 2 and 3 control, the total ISE for the override control system when loop 2 and 3 under control:

$ISE_3 = [(ISE_{21} + ISE_{22})_{\text{set point tracking}} + (ISE_{31} + ISE_{32})_{\text{interaction}}] + [(ISE_{21} + ISE_{22})_{\text{interaction}} + (ISE_{31} + ISE_{32})_{\text{set point tracking}}]$

Therefore, for the single cost function GA, the cost function ISE (Integral Square of Error) is calculated by:

$$e = \int (\varepsilon)^2 dt = (\omega_1 \times ISE_1 + \omega_2 \times ISE_2 + \varpi_3 \times ISE_3)$$

where ϖ_i is the weight factor of each ISE.

In this section, a two inputs three outputs override control system is introduced, this override control system has three controllers which will switch if the errors in the control loop change.

The design of multiple inputs multiple outputs override control systems is make more complex, because of the possibility of the system limit cycling under certain set point values. In this example, there are three different control systems to be designed:

If e1<e3 and e2<e3, then loop one and two under control:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_{p12} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + \begin{bmatrix} k_{i12} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

where $z_1 = \int e_1$ and $z_2 = \int e_2$

If e1<e2 and e3<e2, then loop one and three under control

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_{p13} \end{bmatrix} \begin{bmatrix} e_1 \\ e_3 \end{bmatrix} + \begin{bmatrix} k_{i13} \end{bmatrix} \begin{bmatrix} z_1 \\ z_3 \end{bmatrix}$$

where $z_1 = \int e_1$ and $z_3 = \int e_3$

If e2<e1 and e3<e1, then loop two and three under control

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_{p23} \end{bmatrix} \begin{bmatrix} e_2 \\ e_3 \end{bmatrix} + \begin{bmatrix} k_{i23} \end{bmatrix} \begin{bmatrix} z_2 \\ z_3 \end{bmatrix}$$

where $z_2 = \int e_2$ and $z_3 = \int e_3$

In this chapter, the controllers for the multi-inputs multi-outputs override control system will be designed and tuned by the single cost function genetic algorithm.

The block diagram for the multi input multi output override control system is shown figure 4.18:.

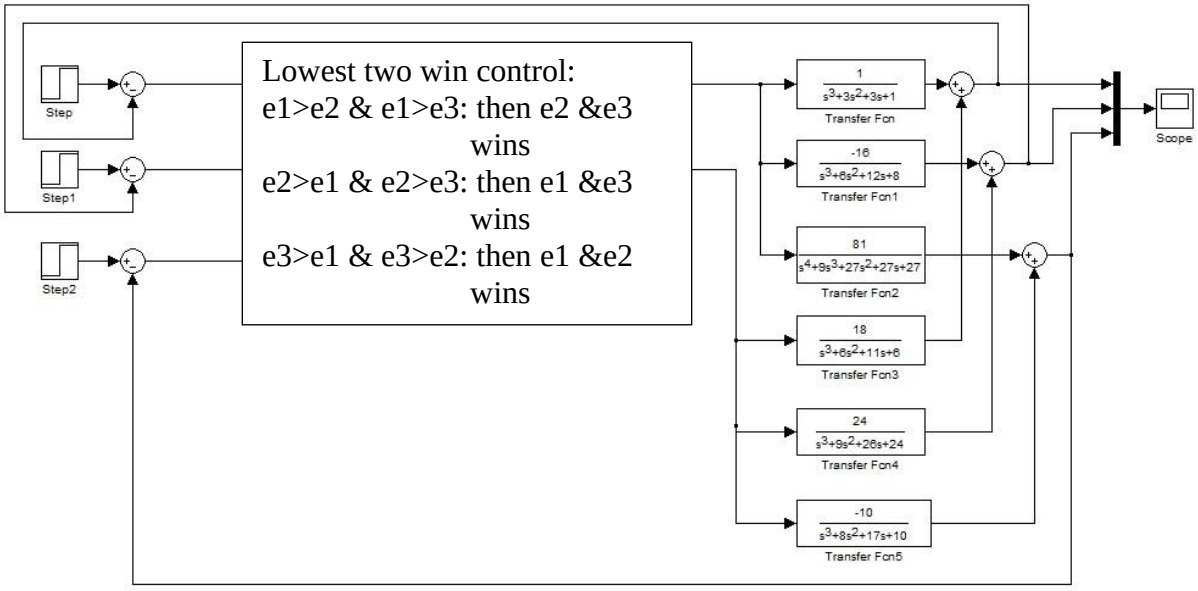

Figure 4.22: Multi input multi output override control system block diagram

As figure above shows, the six transfer functions are

$$G_1 = \frac{1}{s^3 + 3s^2 + 3s + 1} \quad \text{.....(4.17)}$$

$$G_2 = \frac{-16}{s^3 + 6s^2 + 12s + 8} \quad \text{.....(4.18)}$$

$$G_3 = \frac{81}{s^3 + 9s^2 + 27s + 27} \quad \text{.....(4.19)}$$

$$G_4 = \frac{-18}{s^3 + 6s^2 + 11s + 6} \quad \text{.....(4.20)}$$

$$G_5 = \frac{24}{s^3 + 9s^2 + 26s + 24} \quad \text{.....(4.21)}$$

And

$$G_6 = \frac{-10}{s^3 + 8s^2 + 17s + 10} \quad \text{.....(4.22)}$$

Hence, there are three individual controllers for three different control loop combinations, the control loops switch when the errors in the control loop change. So the system input:

If e1<e3 and e2<e3, then loop one and two under control:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_{p12} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + \begin{bmatrix} k_{i12} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

where $z_1 = \int e_1$ and $z_2 = \int e_2$

If e1<e2 and e3<e2, then loop one and three under control

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_{p13} \end{bmatrix} \begin{bmatrix} e_1 \\ e_3 \end{bmatrix} + \begin{bmatrix} k_{i13} \end{bmatrix} \begin{bmatrix} z_1 \\ z_3 \end{bmatrix}$$

where $z_1 = \int e_1$ and $z_3 = \int e_3$

If e2<e1 and e3<e1, then loop two and three under control

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_{p23} \end{bmatrix} \begin{bmatrix} e_2 \\ e_3 \end{bmatrix} + \begin{bmatrix} k_{i23} \end{bmatrix} \begin{bmatrix} z_2 \\ z_3 \end{bmatrix}$$

where $z_2 = \int e_2$ and $z_3 = \int e_3$

where three different PI controller parameters are:

$$k_{p1} = \begin{bmatrix} k_{p11} & k_{p12} \\ k_{p21} & k_{p22} \end{bmatrix} k_{i1} = \begin{bmatrix} k_{i11} & k_{i12} \\ k_{i21} & k_{i22} \end{bmatrix}$$

And

$$k_{p2} = \begin{bmatrix} k_{p31} & k_{p32} \\ k_{p41} & k_{p42} \end{bmatrix} k_{i2} = \begin{bmatrix} k_{i31} & k_{i32} \\ k_{i41} & k_{i42} \end{bmatrix}$$

And

$$k_{p3} = \begin{bmatrix} k_{p51} & k_{p52} \\ k_{p61} & k_{p62} \end{bmatrix} k_{i3} = \begin{bmatrix} k_{i51} & k_{i52} \\ k_{i61} & k_{i62} \end{bmatrix}$$

and $k_{p11}, k_{p12}, k_{p21}, k_{p22}, k_{p31}, k_{p32}, k_{p41}, k_{p42}, k_{p51}, k_{p52}, k_{p61}, k_{p62}, k_{i11}, k_{i12}, k_{i21}, k_{i22}, k_{i31}, k_{i32}$, $k_{i41}, k_{i42}, k_{i51}, k_{i52}, k_{i61}$ and k_{i62} are the parameters need be to design by genetic algorithm. The design of single cost function genetic algorithm the cost function is the sum of set point tracking ISE plus the interaction ISE for the loop under control. The genetic algorithm was run for 5000 generation with a population size of 100, the following values were obtained from the genetic algorithm.

The PI controllers are:

$$K_{p1} = \begin{bmatrix} -0.2635 & -0.4925 \\ -0.4376 & -0.0024 \end{bmatrix} K_{i1} = \begin{bmatrix} -0.0936 & -0.4687 \\ -0.2899 & -0.1351 \end{bmatrix}$$

and

$$K_{p2} = \begin{bmatrix} 0.0001 & 0.3207 \\ 0.9921 & 0.0021 \end{bmatrix} K_{i2} = \begin{bmatrix} 0.5290 & 0.5216 \\ 1.7916 & 0.8320 \end{bmatrix}$$

and

$$K_{p3} = \begin{bmatrix} -0.0002 & 0.2886 \\ -0.6579 & -0.0003 \end{bmatrix} K_{i3} = \begin{bmatrix} 0.0001 & 0.3521 \\ -0.2505 & 0.0933 \end{bmatrix}.$$

At the start of the simulation the set points V_1 is set at V_1=1 and V_2=2 and V_3=3. After 200 seconds the set point V_1 is changed to V_1=0 with the others kept constant. At 400 seconds the set point V_2 is changed to V_2=1 with the other set points kept constant. At 600 seconds the set

point V_2 and V_3 are changed to V_2=1 and V_3=-2 with V_1 kept constant. At 800 seconds the set point V_1 is changed to V_1=-1 with the other set points kept constant. At 1000 seconds the set point V_3 is changed to V_3=-3 with the other two kept constant. At 1200 seconds the set point V_1 and V_2 are changed to V_1=13 and V_2=2 with set point V_3 kept constant. At 1400 seconds the set point V_2 is changed to V_2=1 with the other set points kept constant. At 1600 seconds the set point V_3 is changed to V_3=-4 with the other set points kept constant.

Therefore, the output of the system is like the figures shows below:

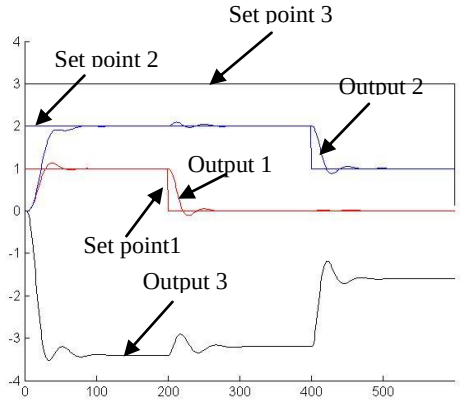

Figure 4.23: The output 1 and 2 under control for multi input multi output override control system

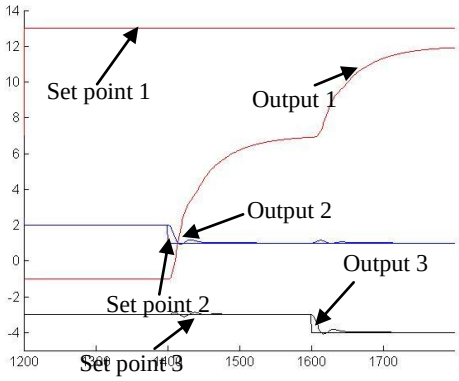

Figure 4.25: The output 2 and 3 for multi input multi output override control system

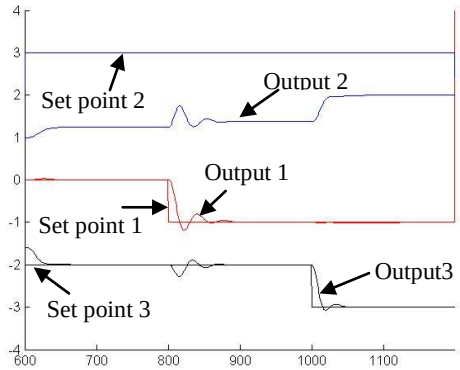

Figure 4.24: The output 1 and 3 under control for multi input multi output override control system

As figures 4.19, 4.20 and 4.21 shows, the multi input multi output override control system is stable and set point tracking. First the output y1 and y2 are under control. Then set point change in both outputs and then switch to output y1 and y3 under control. Then set point change in both outputs and then switch to output y2 and y3 under control. Then set point change in both outputs.

In this section, the cost function is the sum of the set point tracking ISE and the interaction ISE of the loops under control. The weight factors are all equal to 1. To use the single cost function genetic algorithm to design the multi-inputs multi-outputs override control system will provide the controllers with good system performance.

The ISE of genetic algorithm design override control system
The ISE1 for loop 1 and 2 under control $ISE1 = 21.7541013493978$
The ISE2 for loop 1 and 3 under control $ISE2 = 18.6213017605196$
The ISE3 for loop 2 and 3 under control $ISE3 = 17.0289700665691$
The sum of all three $ISE = 57.4043731764865$

4.7 Multi-objective Genetic algorithm search for the limit cycle boundary for multi inputs multi outputs override control systems

In this section, the two inputs three outputs override control system is previously used in considered. This override control system has three controllers which will switch if the errors in the control loop change.

The design of multiple inputs multiple outputs override control systems is made more complex, because of the possibility of the system limit cycling under certain set point values. So in this example, the three controllers are fixed and the parameters searched for by multi-objective genetic algorithm are the three set points when the system begins to limit cycle. As there are a family of set points and not just a single point, the multi-objective

genetic algorithm can search for the non-dominated Pareto front solution for V1, V2 and V3 for the multivariable override control system has no limit cycle.

- Set point V1 for the loop one
- Set point V2 for the loop two
- Set point V3 for the loop three

In this section, the multi-objective genetic algorithm is used to determine boundary between the family set points that make the multi-inputs multi-outputs override control system limit cycle or not under the fixed controllers. All the set points values are selected by the non-dominated Pareto front method.

The block diagram for the multi input multi output override control system is shown in figure 4.22:

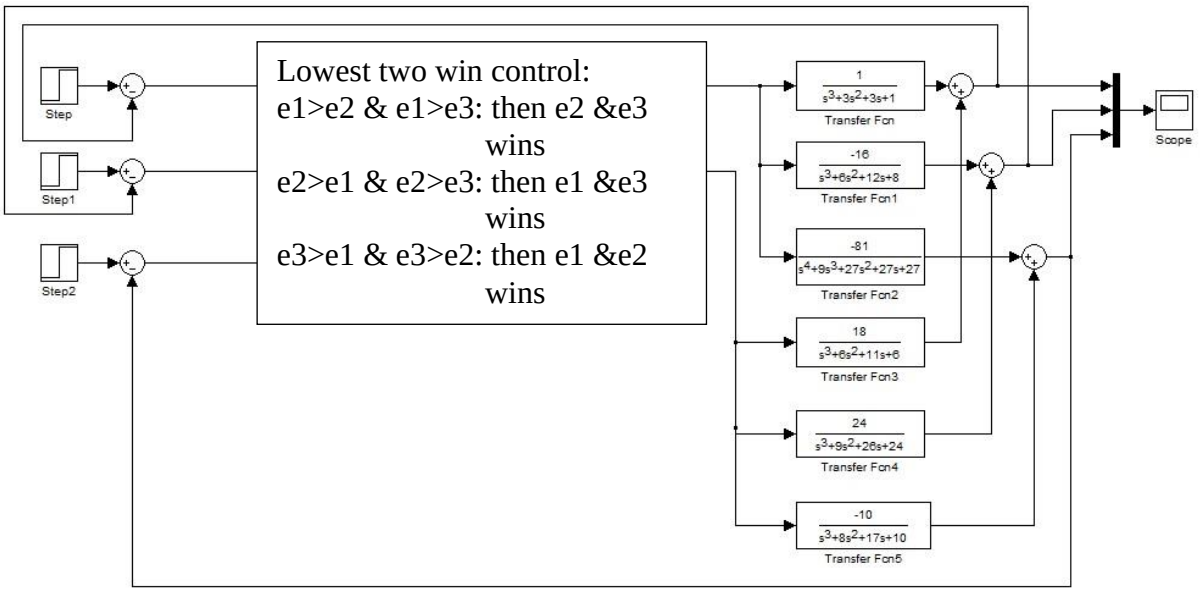

Figure 4.26: Multi input multi output override control system block diagram

As figure above shows, the six transfer functions are

$$G_1 = \frac{1}{s^3 + 3s^2 + 3s + 1} \quad \text{.....(4.23)}$$

$$G_2 = \frac{-16}{s^3 + 6s^2 + 12s + 8} \quad \text{.....(4.24)}$$

$$G_3 = \frac{-81}{s^3 + 9s^2 + 27s + 27} \quad \text{.....(4.25)}$$

$$G_4 = \frac{-18}{s^3 + 6s^2 + 11s + 6} \quad \text{.....(4.26)}$$

$$G_5 = \frac{24}{s^3 + 9s^2 + 26s + 24} \quad \text{.....(4.27)}$$

And

$$G_6 = \frac{-10}{s^3 + 8s^2 + 17s + 10} \quad \text{.....(4.28)}$$

The multi-objective genetic algorithm was used to determine the Pareto front which defines where the system will limit cycle or not. When the multi-objective genetic algorithm has applied, the number of population is 1000, the Pareto front population size is 250, the parameters range V_1=[20 30], V_2=[95 110] and V_3=[15 30], the genetic algorithm run for 1000 generations.

The family of solutions which emerge from the genetic algorithm solutions are shown in figure 4.24:

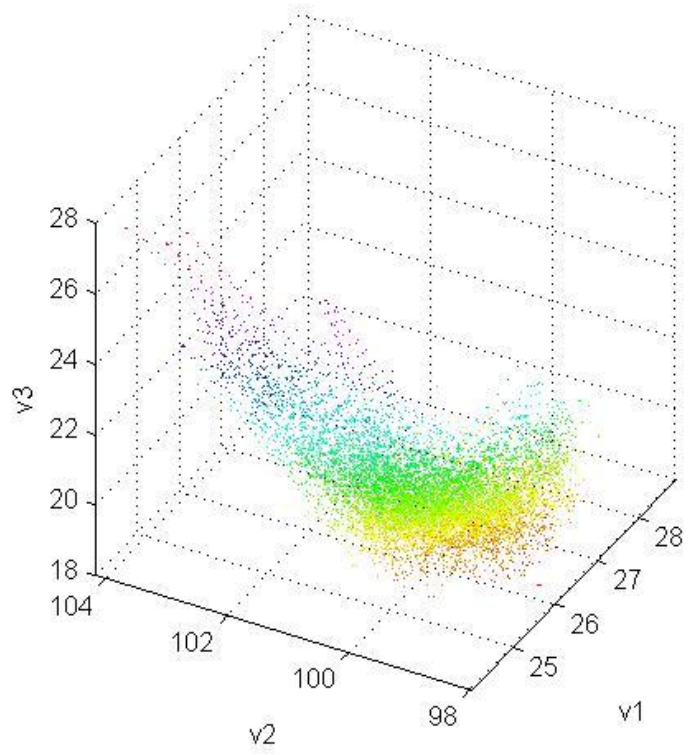

Figure 4.27: Multi input multi output override control system

The results from the Pareto front are in a 3 dimensional spaces and relate a surface in the set-point space V_1, V_2 and V_3. Value of V_1, V_2 and V_3 chosen below this boundary will result in the system limit cycle. However, value of V_1, V_2 and V_3 chosen above the boundary will result in the system not limit cycling.

4.8 Multi-objective Genetic algorithm design of multi input multi output override control systems

In this section, the two inputs three outputs override control system previously used is considered, this override control system has three controllers which will switch if the errors in the control loop change.

The design of multiple inputs multiple outputs override control systems is make more complex, because of the possibility that the system mat limit cycle under certain set point values.

The total ISE is equal to the sum of ISE for each output, and each output ISE is equal to the ISE calculated by the set point tracking plus the ISE calculated by the interaction. For example, there is a two inputs and three output system:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} g_{11}(s) & g_{12}(s) \\ g_{21}(s) & g_{22}(s) \\ g_{32}(s) & g_{33}(s) \end{bmatrix} \times \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Then ISE for each individual outputs will be like below:

$$\begin{bmatrix} ISE_{11} & ISE_{12} \\ ISE_{21} & ISE_{22} \\ ISE_{31} & ISE_{32} \end{bmatrix}$$

Because this is two inputs three outputs override control system, there are three ISEs for three different system control situations.

If the system is under loop 1 and 2 control, the total ISE for the override control system when loop 1 and 2 under control:

$ISE_1 = [(ISE_{11} + ISE_{12})_{\text{set point tracking}} + (ISE_{21} + ISE_{22})_{\text{interaction}}] + [(ISE_{11} + ISE_{12})_{\text{interaction}} + (ISE_{21} + ISE_{22})_{\text{set point tracking}}]$

If the system is under loop 1 and 3 control, the total ISE for the override control system when loop 1 and 3 under control:

$ISE_2 = [(ISE_{11} + ISE_{12})_{\text{set point tracking}} + (ISE_{31} + ISE_{32})_{\text{interaction}}] + [(ISE_{11} + ISE_{12})_{\text{interaction}} + (ISE_{31} + ISE_{32})_{\text{set point tracking}}]$

If the system is under loop 2 and 3 control, the total ISE for the override control system when loop 2 and 3 under control:

$ISE_3 = [(ISE_{21} + ISE_{22})_{\text{set point tracking}} + (ISE_{31} + ISE_{32})_{\text{interaction}}] + [(ISE_{21} + ISE_{22})_{\text{interaction}} + (ISE_{31} + ISE_{32})_{\text{set point tracking}}]$

Therefore, for the single cost function GA, the cost function ISE (Integral Square of Error) is calculated by:

$$e = \int (\varepsilon)^2 dt = (\omega_1 \times ISE_1 + \omega_2 \times ISE_2 + \varpi_3 \times ISE_3)$$

where ϖ_i is the weight factor of each ISE.

In this example, there are three different control systems to be designed:

If e1<e3 and e2<e3, then loop one and two under control:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_{p12} \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} + \begin{bmatrix} k_{i12} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

where $z_1 = \int e_1$ and $z_2 = \int e_2$

If e1<e2 and e3<e2, then loop one and three under control

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_{p13} \end{bmatrix} \begin{bmatrix} e_1 \\ e_3 \end{bmatrix} + \begin{bmatrix} k_{i13} \end{bmatrix} \begin{bmatrix} z_1 \\ z_3 \end{bmatrix}$$

where $z_1 = \int e_1$ and $z_3 = \int e_3$

If e2<e1 and e3<e1, then loop two and three under control

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} k_{p23} \end{bmatrix} \begin{bmatrix} e_2 \\ e_3 \end{bmatrix} + \begin{bmatrix} k_{i23} \end{bmatrix} \begin{bmatrix} z_2 \\ z_3 \end{bmatrix}$$

where $z_2 = \int e_2$ and $z_3 = \int e_3$

where three different PI controller parameters are:

$$k_{p1} = \begin{bmatrix} k_{p11} & k_{p12} \\ k_{p21} & k_{p22} \end{bmatrix} k_{i1} = \begin{bmatrix} k_{i11} & k_{i12} \\ k_{i21} & k_{i22} \end{bmatrix}$$

And

$$k_{p2} = \begin{bmatrix} k_{p31} & k_{p32} \\ k_{p41} & k_{p42} \end{bmatrix} k_{i2} = \begin{bmatrix} k_{i31} & k_{i32} \\ k_{i41} & k_{i42} \end{bmatrix}$$

And

$$k_{p3} = \begin{bmatrix} k_{p51} & k_{p52} \\ k_{p61} & k_{p62} \end{bmatrix} k_{i3} = \begin{bmatrix} k_{i51} & k_{i52} \\ k_{i61} & k_{i62} \end{bmatrix}$$

and $k_{p11}, k_{p12}, k_{p21}, k_{p22}, k_{p31}, k_{p32}, k_{p41}, k_{p42}, k_{p51}, k_{p52}, k_{p61}, k_{p62}, k_{i11}, k_{i12}, k_{i21}, k_{i22}, k_{i31}, k_{i32}$, $k_{i41}, k_{i42}, k_{i51}, k_{i52}, k_{i61}$ and k_{i62} are the parameters need be to design by genetic algorithm.

In this section, the controllers for the multi-inputs multi-outputs override control system will be designed by the multi-objective genetic algorithm. Because there are three different control systems, then there are three trade-off ISE in this design method: the ISE for the loop one and two under control, the ISE for the loop one and three under control and the ISE for the loop two and three under control.

The block diagram for the multi input multi output override control system can be designed like figure 4.19 shows:

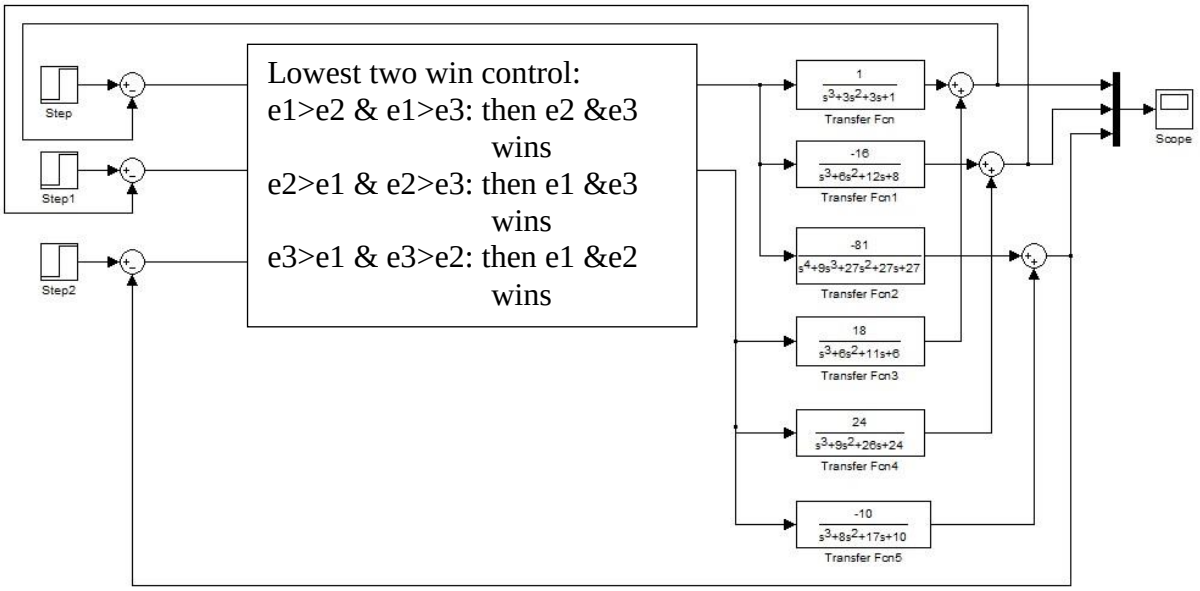

Figure 4.28: Multi input multi output override control system block diagram

As figure 4.24 shows, the six transfer functions are

$$G_1 = \frac{1}{s^3 + 3s^2 + 3s + 1} \quad \text{.....(4.29)}$$

$$G_2 = \frac{-16}{s^3 + 6s^2 + 12s + 8} \quad \text{.....(4.30)}$$

$$G_3 = \frac{81}{s^3 + 9s^2 + 27s + 27} \quad \text{.....(4.31)}$$

$$G_4 = \frac{-18}{s^3 + 6s^2 + 11s + 6} \quad \text{.....(4.32)}$$

$$G_5 = \frac{24}{s^3 + 9s^2 + 26s + 24} \quad \text{.....(4.33)}$$

and

$$G_6 = \frac{-10}{s^3 + 8s^2 + 17s + 10} \quad \text{.....(4.34)}$$

In the Pareto optimization method, the multivariable objective GA is searching for three multivariable PI controllers which minimum the cost function. In this multi-objective design, the simulation is running with fixed set points changes and all of the set-points are changed such that all three control loops are active during the simulation. This ensures that the three cost functions can be evaluated correctly. The three performance cost function are ISE_1 for

the loop 1 and 2 under control, ISE_2 for the loop 1 and 3 under control, ISE_3 for the loop 2 and 3 under control. The multi-objective genetic algorithm will provide a family of non-dominate Pareto front solutions are with each controller being individual optimised.

By using the Pareto front method, the ISE for loop 1 and 2 under control, loop 1 and 3 under control and loop 2 and 3 under control are optimised on a non-dominated Pareto front method. In order to demonstrate the solutions that is produced. Four specific non-dominated Pareto front solutions are reviewed, they are the minimum value for ISE1 for loop 1 and 2 under control, the minimum value for ISE2 for loop 1 and 3 under control, the minimum value for ISE3 for loop 2 and 3 under control and the minimum value for sum of all ISE:

	ISE1	ISE2	ISE3	sum of ISE
1	22.41792725	22.54515627	17.753888	62.71697152
2	23.71336706	18.58923116	18.00910795	60.31170617
3	22.51944439	19.30182201	17.15953263	58.98079904

Table 4.1: the Pareto front solutions of three ISE

Minimum value for loop 1 and 2 under control

As the result shows in table 4.1, ISE_1, ISE_2, ISE_3 and sum of ISE results compare with the result in section 4.6:

The ISE1 for loop 1 and 2 under control ISE_1=21.75

The ISE2 for loop 1 and 3 under control ISE_2=18.62

The ISE3 for loop 2 and 3 under control ISE_3=17.03

The sum of all three ISE=57.40

Only the ISE_1 value is similar with the section 4.6 result, other ISE are all larger. This means the ISE for loop 1 and 2 under control might cannot optimise any future.

As table 4.1 shows above, the red marked sum of three ISE is the lowest one, then the controllers for this design are:

$$K_{p12} = \begin{bmatrix} -0.3142 & -0.5121 \\ -0.3858 & 0.0624 \end{bmatrix} K_{i12} = \begin{bmatrix} -0.0870 & -0.4531 \\ -0.2614 & -0.1799 \end{bmatrix}$$

And

$$K_{p23} = \begin{bmatrix} -0.0652 & 0.3923 \\ 0.9794 & 0.0192 \end{bmatrix} K_{i23} = \begin{bmatrix} 0.5437 & 0.4923 \\ 1.8427 & 0.8393 \end{bmatrix}$$

and

$$K_{p13} = \begin{bmatrix} 0.0646 & 0.3579 \\ -0.7469 & -0.0668 \end{bmatrix} K_{i13} = \begin{bmatrix} 0.0671 & 0.3512 \\ -0.2939 & 0.0937 \end{bmatrix}.$$

Therefore, the output of the system is like the figures shows below:

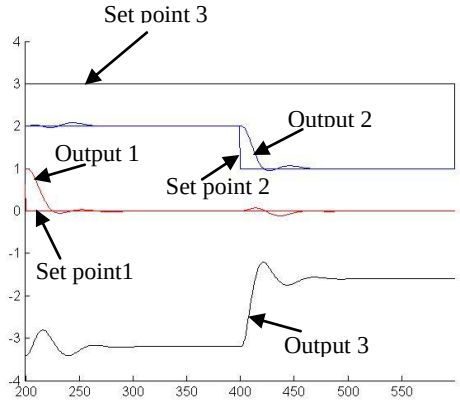

Figure 4.29: The output 1 and 2 under control for multi input multi output override control system

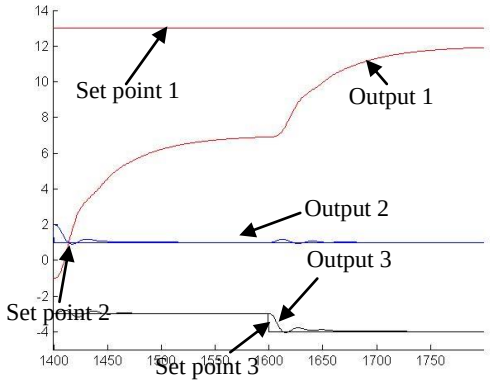

Figure 4.31: The output 2 and 3 for multi input multi output override control system

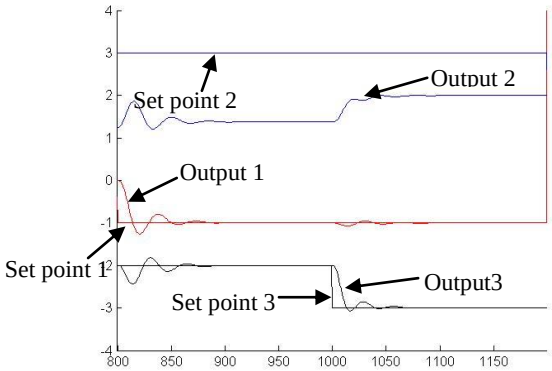

Figure 4.30: The output 1 and 3 under control for multi input multi output override control system

Minimum value for loop 1 and 3 under control

As the result shows in table 4.1, ISE_1, ISE_2, ISE_3 and sum of ISE results compare with the results in section 4.6:

The ISE1 for loop 1 and 2 under control ISE_1=21.75

The ISE2 for loop 1 and 3 under control ISE_2=18.62

The ISE3 for loop 2 and 3 under control ISE_3=17.03

The sum of all three ISE=57.40

The ISE_2 value is smaller than the section 4.6 result. Other ISE is a little larger than the section 4.6 result. This shows multi-objective genetic algorithm could optimise this channel better.

As table 4.1 shows, the red marked sum of three ISE is the lowest one, then the controllers for this design are:

$$K_{p12} = \begin{bmatrix} -0.3208 & -0.4140 \\ -0.5016 & -0.0453 \end{bmatrix} K_{i12} = \begin{bmatrix} -0.0061 & -0.5344 \\ -0.3003 & -0.0505 \end{bmatrix}$$

And

$$K_{p23} = \begin{bmatrix} 0.0832 & 0.2285 \\ 0.9094 & 0.0319 \end{bmatrix} K_{i23} = \begin{bmatrix} 0.5152 & 0.5310 \\ 1.8128 & 0.8557 \end{bmatrix}$$

and

$$K_{p13} = \begin{bmatrix} -0.0788 & 0.2529 \\ -0.7373 & -0.0394 \end{bmatrix} K_{i13} = \begin{bmatrix} -0.0083 & 0.3594 \\ -0.2369 & 0.0501 \end{bmatrix}.$$

Therefore, the output of the system is like the figures shows below:

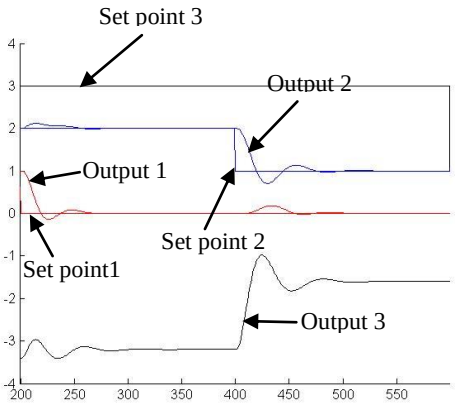

Figure 4.32: The output 1 and 2 under control for multi input multi output override control system

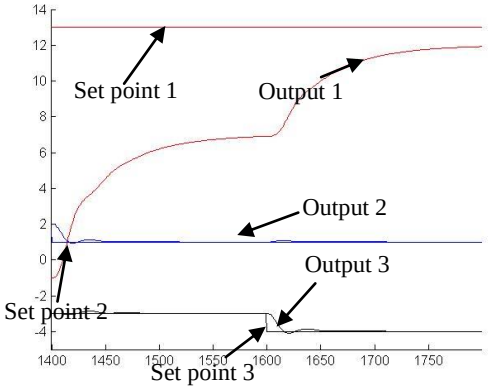

Figure 4.34: The output 2 and 3 for multi input multi output override control system

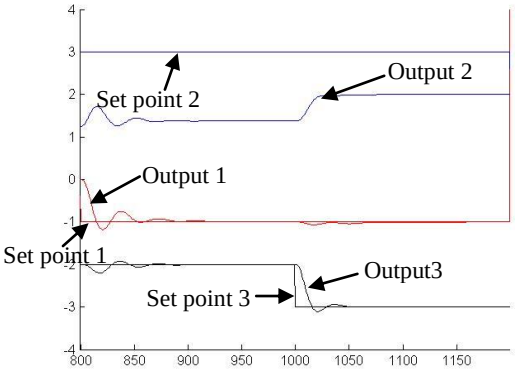

Figure 4.33: The output 1 and 3 under control for multi input multi output override control system

Minimum value for loop 2 and 3 under control

As the result shows in table 4.1, ISE_1, ISE_2, ISE_3 and sum of ISE results compare with the results in section 4.6:

The ISE1 for loop 1 and 2 under control ISE_1=21.75

The ISE2 for loop 1 and 3 under control ISE_2=18.62

The ISE3 for loop 2 and 3 under control ISE_3=17.03

The sum of all three ISE=57.40

The ISE_3 value is very similar with the section 4.6 result. Other ISE is a little larger than the section 4.6 result. This shows multi-objective genetic algorithm might not able to improve this channel any further.

As table shows above, the red marked sum of three ISE is the lowest one, then the controllers for this design are:

$$K_{p12} = \begin{bmatrix} -0.2935 & -0.4812 \\ -0.5272 & 0.0725 \end{bmatrix} K_{i12} = \begin{bmatrix} -0.1445 & -0.5397 \\ -0.2114 & -0.2292 \end{bmatrix}$$

And

$$K_{p23} = \begin{bmatrix} 0.0369 & 0.2713 \\ 1.0183 & -0.0652 \end{bmatrix} K_{i23} = \begin{bmatrix} 0.5222 & 0.4714 \\ 1.6917 & 0.8557 \end{bmatrix}$$

and

$$K_{p13} = \begin{bmatrix} 0.0949 & 0.2314 \\ -0.6924 & -0.0011 \end{bmatrix} K_{i13} = \begin{bmatrix} 0.0634 & 0.4098 \\ -0.2789 & 0.0599 \end{bmatrix}.$$

Therefore, the output of the system is like the figures shows below:

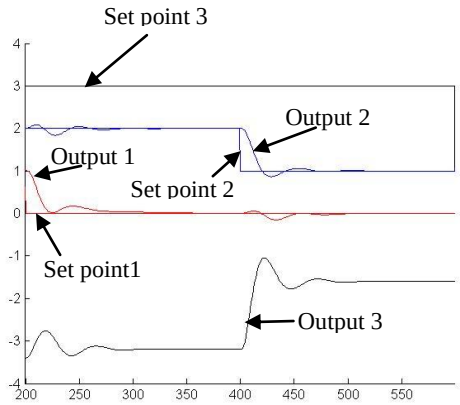

Figure 4.35: The output 1 and 2 under control for multi input multi output override control system

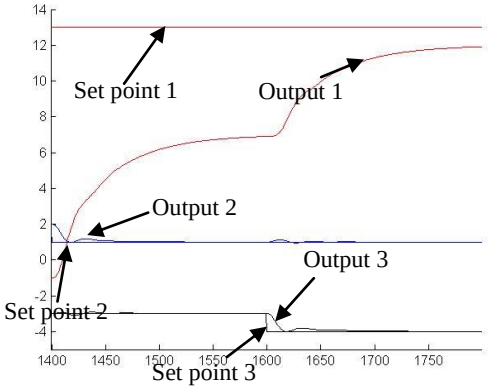

Figure 4.37: The output 2 and 3 for multi input multi output override control system

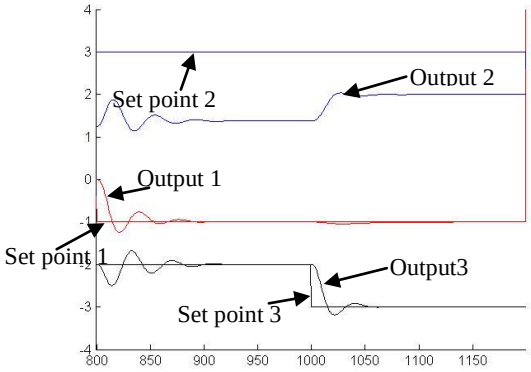

Figure 4.36: The output 1 and 3 under control for multi input multi output override control system

Minimum value for sum of ISE:

As the result shows in table 4.1, ISE_1, ISE_2, ISE_3 and sum of ISE results compare with the results in section 4.6:

The ISE1 for loop 1 and 2 under control ISE_1=21.75

The ISE2 for loop 1 and 3 under control ISE_2=18.62

The ISE3 for loop 2 and 3 under control ISE_3=17.03

The sum of all three ISE=57.40

The sum of ISE value is very similar with the section 4.6 result. Other ISE is a little larger than the section 4.6 result. This shows multi-objective genetic algorithm might not able to improve this channel any further.

As table shows above, the red marked sum of three ISE is the lowest one, then the controllers for this design are:

$$K_{p12} = \begin{bmatrix} -0.3142 & -0.5121 \\ -0.3858 & 0.0624 \end{bmatrix} K_{i12} = \begin{bmatrix} -0.0870 & -0.4531 \\ -0.2614 & -0.1799 \end{bmatrix}$$

And

$$K_{p23} = \begin{bmatrix} -0.0652 & 0.3923 \\ 0.9794 & 0.0192 \end{bmatrix} K_{i23} = \begin{bmatrix} 0.5437 & 0.4923 \\ 1.8427 & 0.8393 \end{bmatrix}$$

and

$$K_{p13} = \begin{bmatrix} 0.0646 & 0.3579 \\ -0.7469 & -0.0668 \end{bmatrix} K_{i13} = \begin{bmatrix} 0.0671 & 0.3512 \\ -0.2939 & 0.0937 \end{bmatrix}.$$

Therefore, the output of the system is like the figures shows below:

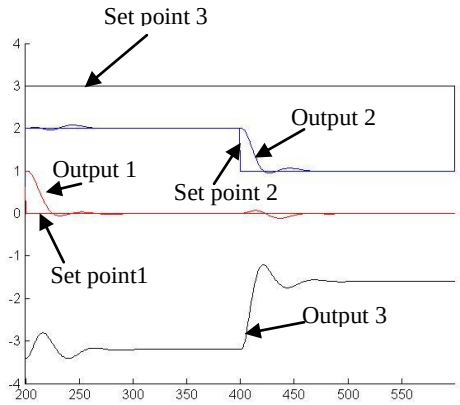

Figure 4.38: The output 1 and 2 under control for multi input multi output override control system

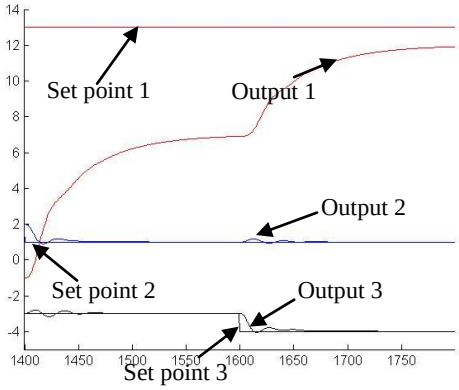

Figure 4.40: The output 2 and 3 for multi input multi output override control system

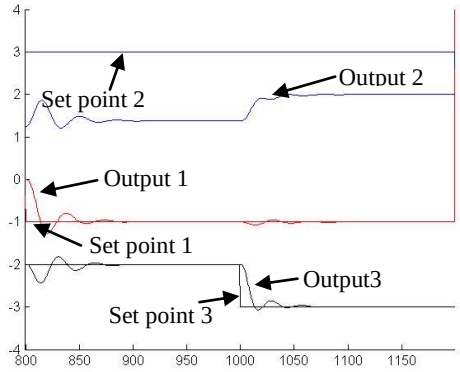

Figure 4.39: The output 1 and 3 under control for multi input multi output override control system

4.9 Conclusion

In this chapter, the both single cost function genetic algorithm and multi-objective genetic algorithm are used to design both single input multi-output and multi-input multi-output override control system. Moreover, the multi-objective genetic algorithm has been used to find out the set point change boundary which will makes the override control system limit cycle.

The design and optimisation of multivariable override control system has recovered little attention as a research topic.

In this thesis:

- Genetic algorithm has been shown to be an ideal tool for both the design and optimisation of multivariable override controllers. As figure 4.23, 4.24 and 4.25 shows, whichever outputs are selected to be controlled, the corresponding set-point tracking is good.
- There is an advantage in design the multivariable override controller using a multi-objective genetic algorithm as this can individually optimise the combination of outputs that be controlled in one run of the genetic algorithm. As table 4.1 shows the family of Pareto front solutions are evolved. Those design include all combination of outputs are selected to be controlled and all optimal. The designer could choose any design which satisfy their need.
- The issue of where a multivariable override control system limit cycles is clearly very important. The multi-objective genetic algorithm has been shown to be an ideal tool to accurately define the set-point boundary beyond which the override control system will limit cycle (see example in figure 4.27).

According to these conclusions, the recommendations are:

- To use the genetic algorithm to evolve the controllers for the override control system to ensure the system has good performance whichever outputs are selected to be controlled, the corresponding set-point tracking is good.
- To use single cost function genetic algorithm if the designer knows the trade-off performance weighting factor in each control groups and the designer looking for a single optimal solution.

- To use the multi-objective genetic algorithm if the designer is looking for the all combination of trade-off performance whichever outputs are selected to be controlled.
- To use the multi-objective genetic algorithm to find out the set point boundary beyond which the override control system will limit cycle.

CHAPTER FIVE

CONCLUSION

5.1 Overview

In this thesis the design and tuning of multivariable control system has been addressed. There are two main types of system: the first one is number of inputs is equal to the number of outputs which is defined as the square multivariable control systems, the second one is where the number of inputs is not equal to the number of outputs. They are called non-square multivariable control systems. Non-square multivariable control system has two types, firstly the systems which have more inputs than outputs and secondly systems which have more outputs than inputs. In the case of more inputs than outputs, the design involves fail-safe control. In the case of more outputs than inputs, the design involves an override control system.

Because of the nature of the multivariable control system design problem, where there are a group of optimisation objectives associate with each set point tracking loop and interactions. A single cost function can result in an optimal solution, but this optimal solution can contain poor performance in a sub-set of control loops. In order to avoid this issue, multi-objective genetic algorithm has been adopted to solve the multivariable control system design problem. The multi-objective genetic algorithm design procedure results in all the individual control loops simultaneously to be optimised. Therefore, with multi-objective genetic algorithm the design procedure will provide the optimal individual control loop objectives and optimal the global objectives.

There are only limited techniques of design procedures for multivariable control system design for system which there are more inputs than outputs. One of the main design method was introduced by R.N.Fripp(1988) who developed a fail-safe design procedure for single failure situation. In this thesis the genetic design of fail-safe controller for multiple failures has been addressed and in the case where solutions exist the genetic algorithm has been used

to design and tune optimal fail-safe controllers. In addition the multi-objective genetic algorithm has been applied to the fail-safe multivariable control system problems, the multi-objective genetic algorithm design procedure individually optimise the different failure situations and provide a family of solutions involving individual loops, and any individual loop solution can be chosen.

There has only been a limited amount of work done on override control system design for multi-inputs and multi-outputs system. The design of override control was introduced by Alejandro A.L. and Joseph A.M. Furthermore, no design method has been proposed whilst can optimize the controllers. In this thesis the genetic algorithm has been used to design and optimal such override controller for multivariable systems. In addition, the multi-objective genetic algorithm has also been used to tune each controller configuration to be individually optimal. Due to the non-linear behavior of override control system, limit cycle can easily take place if the set points are changed from the original design situations. An important task of design override control system is to find out range of the set points which the override control system will not limit cycle. In this thesis the multi-objective genetic algorithm has been used to find the range of set points which no limit cycle can take place. This boundary of set points is the boundary of the override control system which will be stable or limit cycle.

Computational methods for the design of multivariable control system have been developed in this thesis. The design methods can be used to design high performance stable controller for both multivariable square control system and multivariable non-square control system. In addition, the use multi-objective Genetic Algorithm to optimised and design the controllers has been presented and it has been shown be able to design very effective multivariable control system which can look at each loop individually without using any weighting factors. The following sections summarise the achievements under the three main headings.

5.2 Square multivariable control system

In the case of square multivariable control system design the decoupling design has been introduced by many researchers. This classic decoupling design only provides the system with stability and set point tracking, but those controllers have a set of tuning parameters that

have to be chosen by on-line tuning method. The tuning of such controllers has been carried out by using the single objective GA method. In the genetic algorithm tuning method, each set point tracking ISE and interaction ISE is one individual optimisation objective. If the designer needs to weight one of the outputs more than other this means the more important output needs better performance than the others. In this situation the single objective GA has to introduce weighting factor in front of each ISE in the cost function. However, the single objective GA only can deal with one final cost function, even with the weighting factors it cannot deal with more than one objective. In this thesis, there are two type of cost function have been used to optimised in the square multivariable control system: the first cost function is a Global optimisation, and the second type is called worst case optimisation. The Global optimisation cost function is calculated by adding all objective cost functions together with weighting factors; The Global optimisation cost function will provide the minimum sum of ISE for all objectives, but some of the individual ISE might have poor system output response, but this algorithm can only reduce the poor individual ISE by increasing the weighting factor. Alternately, the worst case optimisation cost function is the largest cost from the all individual objective cost. the worst case optimisation cost function cannot provide the minimum sum of ISE for all objectives compared with Global optimisation cost function, but will focused on the worst ISE objective only and reduced it. This cost function will result in the worst set point tracking ISE and interaction ISE smaller. However, the single cost function genetic algorithm cannot combine the minimum sum of ISE and optimal individual ISE together, because it cannot improve individual objectives simultaneously. In this thesis, the multi-objective genetic algorithm has been used to address this problem. The multi-objective genetic algorithm uses the non-dominated Pareto front method to find out a family of Pareto solutions. And those set of solutions is going to optimise all individual cost simultaneously.

The final conclusion for genetic design of square system are:

- Evolving a non-diagonal controller is more effective than evolving the tuning parameters associated with a decoupling controller. The system's ISE performance for genetic algorithm evolve non-diagonal controller improved a lot compared with genetic algorithm evolve the diagonal controller.

- To control the amount the interaction a weighting factor can be introduced into the cost function. However, the weighting factor is hard to choose to archive an exact design goal. In order to avoid choosing the weighting factor to control interaction a more effective technique is to add an interaction constraint into the cost function. This can control exactly the maximum level of interaction.

The limitations are:

- The single objective cost function genetic algorithm technique is very effect in designing multivariable controllers if the design trade-off weighting function are known.
- The multi-objective cost function genetic algorithm is very effect at generating a family of optimal solutions which present system performance in individual channels to be taken into account. However, the multi-objective genetic algorithm need much more time to converge.

5.3 Fail-safe multivariable control system

In the case of multivariable control system where the number of inputs is greater than the number of outputs, this type of system is defined as a fail-safe control system. In this case Genetic Algorithm have been introduced to design and optimize the performance of the fail-safe multivariable controller. The original fail-safe control system with single actuator failure situation design method was introduced by R.N.Fripp (1988), his method is the called pseudo inverse design method. This design method only provides one controller that makes the control system into fail-safe system which is stable and exhibits set point tracking for an un-failed situation or any single actuator failure situation, but the system performance might be poor. In this thesis, the single cost function genetic algorithm with global optimisation cost function has been introduced to design and tune the diagonal matrixes parameters described in R.N.Fripp design to improve the system performance. Rather than tune the diagonal matrixes, the single cost function genetic algorithm could also complete design and tune the controller straight away with this design technique the speed of response and system performance can be improved. In the genetic algorithm design method, each output ISE is one individual objective. For single cost function genetic algorithm, there are two type of cost function have been used in the fail safe multivariable control system design. The first type of

cost function is called Global optimisation, the second one called Worst case failure optimisation. The Global optimisation cost function is calculated by adding all objective cost together with weight factor; The Global optimisation cost function will provide the minimum sum of ISE for all objectives, but some of the individual ISE might have a poor transmit response; alternately, the Worst case failure optimisation cost function is the largest cost from the all individual objective costs. The worst case failure optimisation cost function cannot provide the minimum sum of ISE for all objectives compared with Global optimisation cost function, but will focused on the worst ISE objective only and reduced it. This cost function will results in a smaller worst case ISE and more equal ISE values across all the objectives. However, the single cost function genetic algorithm cannot combine the minimum sum of ISE and optimal individual ISE together, because it cannot improve individual objectives simultaneously. As the results shows in chapter three, Genetic algorithm design method is much better than the Pseudo Inverse design which introduced by R.N.Fripp. Additionally, there is no formal design method for multiple failures in fail-safe multivariable control system, but if a solution exists the controllers can be found with the genetic algorithm technique. The genetic algorithm is used in a similar way as single failure case, but in this case it is choosing all the controller parameters and looking at each failure case including multiple failures, and results shows the speed of response and system performance has improved. The single cost function genetic algorithm cannot improve all objectives simultaneously but multi-objective genetic algorithm can. The multi-objective genetic algorithm with non-dominated Pareto front design method has been introduced. The results are even better than the single cost function genetic algorithm solutions. Because the multi-objective genetic algorithm with non-dominated Pareto front design technique is going to optimise the all objectives simultaneously, and provide a family of solutions. The specific solution could be chosen from the family of solutions which could satisfy the designer needs. For example, if the specific channel needs to be minimised, then the controller combined with this minimum ISE values in this channel should be selected. Moreover, if sum of the each individual ISE together and select the minimum value in this new channel, it is the optimal global situation.

The contributions for genetic design for fail-safe system are:

- The genetic algorithm approach to the design of fail-safe controller was very successful. The genetic algorithm can evolve the controller for fail-safe system has good system performance under all non-actuator failure and single actuator failure and multiple actuator failures. Which there is no formal design method could do. Indeed, a full set of fail-safe controller was evolved by the genetic algorithm and optimal controller performance was achieved.
- The multi-objective genetic algorithm was able to design a family of solutions. The family of solutions include the minimisation of the sum of cost function. This family of solutions present all of the interesting and different useable design, for this reason the multi-objective genetic algorithm is used as an extremely effective design tool.

The limitations are:

- The genetic algorithm cannot guarantee to evolve a solution.
- To use the global optimisation cost function if the designer is looking for the optimal total cost. To use the worst failure case optimisation cost function if the designer is looking for the optimise the worst failure case.
- To use the single cost function genetic algorithm if the designer knows the trade-off performance weighting factor in each non-actuator failure and all actuator failure combinations and the designer looking for a single optimal solution.
- To use the multi-objective genetic algorithm if the designer is looking for the all combination of trade-off performance in each non-actuator failure and all actuator failure combinations. However, multi-objective genetic algorithm need much more time to converge.

5.4 Override multivariable control system

For the case of multivariable control system where the number of outputs is greater than the number of inputs, this kind of system is defined as override control system. because the number of inputs is less than the number of outputs, not all the outputs can be under control. This means the override control will control the worst output which is normally defined as the output above its set point or the output which has the most negative error. Since 1971 Buckley P.S. has first introduced the override control with feed forward control systems, this

override control has become popular. After his research, lots of researchers used override control as a main tool to design the control systems but have not improved the theory for either single input multi outputs or multi inputs multi outputs override control system. However, most researchers have focused on single input multi output override control system, as Buzzard W.S. In addition, as the override control system cannot control all outputs, the main analysis of override control system design has focused on avoiding limit cycles.

Because there is no formal design method for override control system and genetic algorithm method could be used. However, single input multiple outputs override control system will control the largest negative error, which is the worst case, so there is only one cost function in genetic algorithm design method for this kind of override control system. In this thesis the single cost function genetic algorithm has used to design and tune the controller parameters, the cost function is calculated by the ISE which the loop under control. With this design method, the override control system will be stable, track and optimised the system output performance, and ensure there is not limit cycling. Moreover, the multiple inputs multiple outputs override control system have more than one inputs, the cost function is calculating by the set point tracking ISE and interaction ISE. Therefore, there are two type of cost function have been used to optimised in the multivariable override control system: the first cost function is a Global optimisation, and the second type is called worst case optimisation. The Global optimisation cost function is calculated by adding all under controlled objective cost functions together with weighting factors; The Global optimisation cost function will provide the minimum sum of ISE for all controlled objectives, but some of the individual ISE might have poor system output response, but this algorithm can only reduce the poor individual ISE by increasing the weighting factor; alternately, the worst case optimisation cost function is the largest cost from the all individual controlled objective cost. the worst case optimisation cost function cannot provide the minimum sum of ISE for all controlled objectives compared with Global optimisation cost function, but will focused on the worst ISE objective only and reduced it, this cost function will result in the worst set point tracking ISE and interaction ISE smaller. However, the single cost function genetic algorithm cannot combine the minimum sum of ISE and optimal individual ISE together, because it cannot improve individual objectives simultaneously. In this thesis, the multi-objective genetic algorithm has been used

to address this problem. The multi-objective genetic algorithm uses the non-dominated Pareto front method to find a family of Pareto solutions. The set of solutions is going to optimise all individual cost simultaneously. As result shows in chapter four, there are lots of Pareto front solutions, and the designer can choose the specific results which satisfy their need. The designer could choose the optimal solution in each individual channel or the sum of all channels. Like the results shows in chapter four, the controller for the minimum sum of all individual ISE provides good system performance. Moreover, the controller for the minimum ISE from the worst ISE of all individual ISE provides the smallest ISE in this channel and the sum of all individual ISE still smaller than the single cost function genetic algorithm worst case optimisation.

The override control system has a non-linear switch in the control loops, this can lead to limit cycle. For both single input multi outputs and multi inputs multi outputs override control system, as chapter four shows, the override control system may limit cycle or not limit cycle, and this is related to the set points. It is hard to analyse when the override control system will limit cycle. In this thesis the multi-objective Genetic algorithm method has been shown to be able to find the boundary of when the set points of the override control system could not limit cycle in a specific controller. These boundaries are easily found by using a multi-objective genetic algorithm.

The contributions for genetic design for override system are:

- Genetic algorithm has been shown to be an ideal tool for both the design and optimisation of multivariable override controllers, whichever outputs are selected to be controlled, the corresponding set-point tracking is good.
- There is an advantage in design the multivariable override controller using a multi-objective genetic algorithm as this can individually optimise the combination of outputs that be controlled in one run of the genetic algorithm. The designer could choose any designs which satisfy their need.
- The issue of where a multivariable override control system limit cycles is clearly very important. The multi-objective genetic algorithm has been shown to be an ideal tool to accurately define the set-point boundary beyond which the override control system will limit cycle.

The limitations are:

- To use single cost function genetic algorithm if the designer knows the trade-off performance weighting factor in each control groups and the designer looking for a single optimal solution.
- To use the multi-objective genetic algorithm if the designer is looking for the all combination of trade-off performance whichever outputs are selected to be controlled. However, multi-objective genetic algorithm need much more time to converge.
- To use the multi-objective genetic algorithm to find out the set point boundary beyond which the override control system will limit cycle.

FURTHER WORK

In this thesis it has been shown that genetic algorithm can be used to design fail safe multivariable control system. However, this design procedure is going to tuning one fixed controller to deal with all non-failure and failure situations. However, if the specific actuator failure could be detected, then to use the pre-designed controllers to deal with the specific failure situation and this will provide even better output performance. To implement this technique requires a fault detection method. Fault detection can be done in a variety of ways but one popular method is to carry out real-time modelling of the system dynamics. Genetic algorithm have been used successfully to identify system dynamic off-line. One research direction could to be investigating that to improve the genetic algorithm off-line system identification into on-line system identification.

For multivariable override control system, the limit cycle is take place because of the nonlinearity. Therefore, it may be possible to use the describing function. The describing function method required coverage this switching mechanism into a non-linear function. Then it may be possible to apply describing function method to analysis the override control system design problem.

REFERENCE

Aceves Lopez Alejandro, Aguilar Martin Joseph: "On the stability of override control systems", LAAS - CNRS, 7 avenue du Colonel Roche, 31077 Toulouse Cedex 4, France, 2003

Acikmese A.B.: "Robust constant output tracking for uncertain/nonlinear systems with PI controllers", American Control Conference, 2001. Proceedings of the 2001 (Volume: 5)

Alejandro A.L. and Joseph A.M.: "On the stability of override control systems", LAAS - CNRS, 7 avenue du Colonel Roche, 31077 Toulouse Cedex 4, France CONACYT, Beca No. 112409, México D.F.

Alexandru-Nicolae Tudosie: "Aircraft Gas-Turbine Engine's Control Based on the Fuel Injection Control", Aeronautics and Astronautics, Prof. Max Mulder (Ed.), ISBN: 978-953-307-473-3, InTech, 2011

Amir M Yasir, Abbas Vali uddin: "Modeling and neural control of quadrotor helicopter", Yanbu journal of Engineering and Science, ISSN: 1658-5321, 2011

Aron, I.; Tudosie, A.: "Jet Engine Exhaust Nozzle's Automatic Control System", Proceedings of the 17th International Symposium on Naval and Marine Education, pp. 36- 45, section III, Constanta, Romania, May 24-26, 2001

Ayman A. Aly: "PID Parameters Optimization Using Genetic Algorithm Technique for Electrohydraulic Servo Control System", Intelligent Control and Automation, 2011, 2, 69-76

Back and Schwefel H.P.: "An Overview of Evolutionary Algorithms for Parameter Optimisation", Evol. Comput., Vol.1, 1993

Back T.: “Evolutionary Algorithms in Theory and Practice”, Oxford University Press, ISBN 1-19-509971-0, 1996

Back T. and Schwefel H.P.: “A survey of evolution strategies”, in Genetic Algorithm: Proc. 4th Int. Conf., R.K. Belew and L.B. Booker, Eds. San mateo, CA: Morgan Kaufmann, 1991

Bandyopadhyay.S and Saha.S,:“Some Single- and Multiobjective Optimization Techniques”, Unsupervised Classification, DOI 10.1007/978-3-642-32451-2_2, © Springer-Verlag Berlin Heidelberg 2013

Basile, G. and Marro, G.: “Controlled Invariants and Conditioned Invariants in Linear System Theory”. Prentice-Hall. Englewood Cliffs, NJ., 1992

Benjamin C. Kuo & Farid Golnaraghi,: “Automatic Control Systems”, John Wiley & Sons, Inc. New York, NY, USA, 2002

Bennet, S.: “A history of control engineering 1800-1930”. IEE Control Engineering Series 8, Peter Peregrinus Ltd., London, 1979

Bosworth, John T.: “Linearized aerodynamic and control law models of the X-29A airplane and comparison with flight data”, NASA technical memorandum 4356, February 1992

Bousbaine A. Wu M.H. and Poyi G.T.: “Modelling and simulation of a quad-rotor helicopter”, Power Electronics, Machines and Drives (PEMD 2012), 6th IET International Conference on, pp. 1-6, 2012

Bradley E., Easley M. and Stolle R.: “Reasoning about nonlinear system identification”, Tech.rep.CU-CS-894-99, University of Colorado, 2000

Bresciani Tommaso: “Modelling, Identification and Control of a Quadrotor Helicopter”, Department of Automatic Control Lund University, ISSN 0280-5316,2008

Campbell, D. P.: “Process Dynamics”. Wiley, New York, 1958

Carlos M. Fonseca, Peter J. Fleming,: “An Overview of Evolutionary Algorithms in Multiobjective Optimization”, Evolutionary, Computation,3(1):1{16, Spring 1995

Carolyn A. B. Fiebig et al,: “Real time, fail safe process control system and method”, International Business Machines Corporation, Armonk, N.Y., 1989

Ceaglske, N. H.: “Automatic Process Control for Chemical Engineers”. Wiley, 1956

Chakraborty A.: “Fault tolerant fail safe system for railway signalling”, Proceedings of the World Congress on Engineering and Computer Science 2009 Vol II WCECS 2009, October 20-22, 2009, San Francisco, USA

Charles L. Phillips & Royce D. Harbor: “Feedback Control Systems”, LAVOISIER S.A.S. 1999

Chein, I.L.: “IMC-PID Controller Design-An Extension,” IFAC Proceeding Series, 6, 147–152, 1988.

Chen X.S., Zhai J.Y., Li Q. and Fei S.M.: “Override and model predictive control of particle size and feed rate in grinding process”, Proceedings of the 26thChinese Control Conference July 26-31, 2007, Zhangjiajie, Hunan, China

Chipperfield A. and Fleming P.: “Gas turbine engine controller design using multi-objective genetic algorithms”, Genetic algorithms in engineering systems: Innovations and applications, 12-14 September 1995, Conference Publication No.414

Clark D.W. and Mohtadi C. and Tuffs,: “Generalized predictive control”, Automatica, vol.23, 1987

Clarke. D.W., C. Mohtadi and P.S. Tuffs: "Generalized Predictive Control Part I. The Basic Algorithm, " Automatica, 23(2), 137–148, 1987

Coit D.W., Jin T. and Wattanapongsakorn N.: "System Optimization With Component Reliability Estimation Uncertainty: A Multi-Criteria Approach", IEEE TRANSACTIONS ON RELIABILITY, VOL. 53, NO. 3, SEPTEMBER 2004

Coradini,M.L. & Orlando, G,: "Actuator Failure Identification and Compensation Through Sliding Modes", Dipt. di Matematica e Informatica, Univ. di Camerino, 2007

Corne, D. W., Knowles J. D., Oates M. J.: "The Pareto Envelope-Based Selection Algorithm for Multi-Objective Optimization". − In: PPSN 2000. LNCS, Vol. 1917, Springer, Heidelberg (K. Deb, et al., Eds.), 839-848, 2000

Coughanowr, D. R. and Koppel, L. B.: "Process System Analysis and Control". McGraw-Hill,1965

Dacison, E.J.: "Multivariable tuning regulators: The feedforward and robust control of a general servomechanism problem", IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL. AC-21, NO. 1, FEBRUARY 1976

Damien Koenig and Saïd Mammar,: "Design of Proportional-Integral Observer for Unknown Input Descriptor Systems", IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL. 47, NO. 12, DECEMBER 2002

Dave Misira & Heidar A. Malki & Guanrong Chena,: "Design and analysis of a fuzzy proportional-integral-derivative controller", Department of Electrical Engineering, University of Houston, Houston, TX 77204, USA, 1996

Davison E.J.: “Multivariable tuning regulators: The feed forward and robust control of general servo mechanism problem”, IEEE Trans. Auto. Control., 21, 35-21, 1976

Deb, K., Pratap A., Agarwal S., Meyarivan T.: “A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II”. IEEE Transactions on Evolutionary Computation, Vol. 6, No 2, 182-197, Apr. 2002

Dinh T.V., Freeman C.T., Lewin P. and Tan Y.: “Assessment of Gradient-based Point-to-Point ILC for MIMO Systems with Varying Interaction”, 2012 IEEE International Symposium on Intelligent Control (ISIC) Part of 2012 IEEE Multi-Conference on Systems and Control October 3-5, 2012. Dubrovnik, Croatia

Donald E. Kirk: “Optimal Control Theory: An Introduction”, Dover Publications inc. mineola, New York, 2004

Donald W. Marquardt,: “An algorithm for Least-Squares Estimation of Nonlinear Parameters”, J.Soc.Indust.Appi, 1963

Dzeroski S.: “Learning qualitative models with inductive logic programming”, Informatica, 16(4), 30-41

Dzeroski S. and Todorovski L.: “Discovering dynamics: from inductive logic programming to machine discovery”, J. Intell. Information syst., 4, 89-108, 1995

Ebert, Christof & Jones, Capers.: "Embedded Software: Facts, Figures, and Future". IEEE Computer Society Press. 2009

Echtle, Klaus Kimmeskamp, Thorsten,: “Fault-Tolerant and Fail-Safe Control Systems - Using Remote Redundancy”, in IEEE, ISBN: 978-3-8007-3133-6 , 2009

Eckart Zitzler, Lothar Thiele,: “Multiobjective Evolutionary Altorithms: A Comparative Case Study and the Strength Pareto Approach”, IEEE Transactions on Evolutionary Computation, VOL. 3, NO. 4, November 1999

Eckman, D.: “Automatic Process Control”. Wiley, New York, 1958

Emanuel Todorov: “Optimal Control Theory”, University of California San Diego, 2006

Eslinger R. A. and D’Azzo J. J.: “Multivariable control law design for the AFTI/F-16 with a failed control surface,” in IEEE National Aerospace and Electronics Conference, Dayton, OH, 1985

Feuer, A. & Morse, A.: “Adaptive control of single-input, single-output linear systems，Yale University, New Haven, CT, USA， 1978

Fonseca C. M. and Fleming P. J.: “Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization,” in Proceedings of the Fifth International Conference on Genetic Algorithms,S. Forrest, Ed. San Mateo, CA: Morgan Kauffman, 1993, pp. 416–423

Fonseca,C.M., FlemingP.J.: “Genetic Algorithms for Muitiobjective Optimization: Formulation, Discussion and Generalization”. In: Proc. of the 5th International Conference on Genetic Algorithms, San Mateo, California, University of Illinois at Urbana-Champaign (Stephanie Forrest, Ed.), Morgan Kauffman Publishers, 416-423, 1993

Forouraghi B.,: "A Genetic Algorithm for Multiobjective Robust Design," Journal of Applied Intelligence, Kluwer Academic Publishers, Vol. 12, No. 3, pp. 151-161, 2000.

Foss A.M.: “Criterion to assess stability of a ‘lowest wins’ control strategy”, IEEPROC, Vol. 128, Pt. D, No. 1, JANUARY 1981

Fripp R.N.: “Design of Digital Controllers for Multivariable Plants with Actuator Failures”, The University of Salford, (1988)

George M. Coghill, Ashwin Srinivasan and Ross D. King,: “ Qualitative system identification from imperfect data”, Journal of artificial intelligence research 32, 2008

Gilbert E.G. and Tan K.T.: “Linear systems with state and control constraints: The theory and application of Maximal output admissible sets”, IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL. 36, NO. 9, SEPTEMBER 1991

Giorgos Vasiliadis, Spiros Antonatos, Michalis Polychronakis, Evangelos P. Markatos and Sotiris Ioannidis,: "Gnort: High Performance Network Intrusion Detection Using Graphics Processors", Boston, MA, USA, 2008

Glattfelder A.H., Schaufelberger W.: “Stability of discrete override and cascade-limiter single-loop control systems”, IEEE Transactions on Automatic Control; 33 (6):532–540, 1988

Glattfelder A.H., Schaufelberger W., Fassler H.P.: “Stability of override control systems”, International Journal of Control;37(5):1023–1037,1983

Gold S.: “Fault tolerance with control systems”, Engineering Designer. Vol. 22, no. 5, 1996

Goldberg D. P.: “Manual of the General Health Questionnaire. Slough”, National Foundation for Education Research. (1978)

Goldberg D.E.: “Genetic algorithms in search, optimization and machine learning”, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA ©1989

Goldberg D.E. and Richardson J.: "Genetic algorithms with sharing for multimodal function optimization", In Proceedings of the second international conference on genetic algorithms (148-154), san Mateo, CA: Morgan Kaufmann, 1987

Goldberg,D.E. Deb.K..&Clark.J.H.: "Genetic algorithms, noise, and the sizing of populations", complex systems, 1992

Grebe, J. J., Boundy, R. H., and Cermak, R. W.: "The control of chemical processes". Trans. Am. Inst. Chem. Eng., 1933,29: 211

Grefenstette J.J.: "Optimization of control parameters for genetic algorithms", In Sage, A.P.(Ed.), IEEE transactions on systems, Man, and Cybernetics, 1986

Guili Yuan, Yan-guang Xue, and Jizhen Liu: "Adaptive Immune Genetic Algorithm and Its Application in PID Parameter Optimization for Main Steam Temperature Control System", Third International Workshop on Advanced Computational Intelligence Suzhou, Jiangsu, China, August 25-27,2010

Håkan Hjalmarsson,: "Efficient tuning of linear multivariable controllers using iterative feedback tuning", John Wiley & Sons, Ltd, 1999

Håkan Hjalmarsson,: "From experiment design to closed-loop control", Department of Signals, Sensors and Systems, Royal Institute of Technology, Sweden, 2005

Harriott, P.: "Process Control". McGraw-Hill, New York, 1964

Harry G. Kwatny: "Introduction to Optimal Control System", Department of Mechanical Engineering & Mechanics Drexel University

Hendrik Bode,: "Network analysis and Feedback Amplifier Design", New York, D. Van Nostrand Co.,1945

Himmelstoss, F.A., Kolar, J.W. and Zach, F.C.: “Analysis of a Smith-predictor-based-control concept eliminating the right-half plane zero of continuous mode boost and buck-boost DC/DC converters”, Industrial Electronics, Control and Instrumentation, 1991. Proceedings. IECON '91., 1991 International Conference on

Horn J., Nafploitis N., and Goldberg D. E.: “A niched Pareto genetic algorithm for multiobjective optimization,” in Proceedings of the First IEEE Conference on Evolutionary Computation, Z. Michalewicz,. Piscataway, NJ: IEEE Press, 1994, pp. 82–87

Horn, J., Nafpliotis N., Goldberg D. E.: “A Niched Pareto Genetic Algorithm for Multiobjective Optimization”. – In: Proc. of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, Vol. I, Piscataway, New Jersey, IEEE Service Center, 82-87, June 1994

Hubert Maxwell James, Nathaniel B. Nicholas, Ralph Saul Phillips,: “Theory of servomechanisms”, McGraw-Hill Book Co., 1947

Hwang, K & Faye, A,: “Computer architecture and parallel processing”, McGraw-Hill, New York, NY, USA, 1984

Ishibuchi H., Nojima Y. and Doi T.: “Comparison between Single-Objective and Multi-Objective Genetic Algorithms: Performance Comparison and Performance Measures”, 2006 IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, July 16-21, 2006

Ishibuchi H., Nojima Y. and Doi T.: “Comparison between Single-Objective and Multi-Objective Genetic Algorithms: Performance Comparison and Performance Measures”, IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada July 16-21, 2006

Ishibuchi H., Osaka Prefecture Univ. Osak, Nojima Yusuke and Doi, T.: "Comparison between Single-Objective and Multi-Objective Genetic Algorithms: Performance Comparison and Performance Measures", Evolutionary Computation, CEC 2006. IEEE Congress on, 2006

Ivo F. Sbalzariniy, Sibylle Mullery ,Petros Koumoutsakosyz,: "Multiobjective optimization using evolutionary algorithms", Center for Turbulence Research, Proceedings of the Summer Program, 2000

Jones A.H., Porter B and Chrysanthou A.: "Design of digital self-selective multivariable controllers", Proc 3rd IEEE International Symposium on Intelligent Control, Arlington, 1988

Jorge Miguel Brito Domingues,: "Quadrotor prototype", Dissertação para obtenção do Grau de Mestre em, 2009

Kailath.T.: "Linear Systems", Prentic-Hall, Englewood Cliffs,NJ. 1980

Kanchan Rani and Vikas Kumar: "Solving travelling salesman problem using genetic algorithm based on heuristic crossover and mutation operator", International Journal of Research in Engineering & Technology (IMPACT: IJRET) ISSN(E): 2321-8843; ISSN(P): 2347-4599 Vol. 2, Issue 2, Feb 2014, 27-34

Karl Johan Astrom & Bjorn Wittenmark: "Adaptive Control", Addison-Wesley Publishing Company, 1989

Katebi, M.R. and Moradi M.H.: " Predictive PID Controllers", IEE Proc. Control Theory Application, 148(6), 478–487,2001

Knowles J. and Corne D.: "The Pareto archived evolution strategy: A new baseline algorithm for multiobjective optimization," in Proceedings of the 1999 Congress on Evolutionary Computation. Piscataway, NJ: IEEE Press, 1999, pp. 98–105.

Knowles, J. D., Corne D. W.: " Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy". Evolutionary Computation, Vol. 8, No 2, 149-172, 2000.

Kok, L. T., McAvoy, T. J.: "Biological studies of Ceutorhynchus trimaculatus F. (Col.:Curculionidae) a thistle rosette weevil. Can. Ent. 115: 927-932, 1983

Korn, U. and H. H. Wilfert,: "Mehrgr¨ossenregelungen. Moderne Entwurfsprinzipien imZeit- und Frequenzbereich". Verlag Technik, Berlin. (1982).

Latawiec, K. J. & Hunek, W. P.: "Control zeros for continuous-time LTI MIMO systems", Proceedings of 8th IEEE International Conference on Methods and Models in Automation and Robotics (MMAR'2002), pp. 411-416, Szczecin, Poland, September 2002.

Latawiec, K. J.; Bańka, S. & Tokarzewski, J.: "Control zeros and nonminimum phase LTI MIMO systems." Annual Reviews in Control, Vol. 24, 2000, pp. 105-112; also in Proceedings of the IFAC World Congress, Vol. D, pp. 397-404, Beijing, P.R. China, 1999.

Lennart Ljung,: "Eiley Encyclopedia of Elevtrical and Electronics Engineering", John Wiley & Sons,Inc. 1999

Liang Zhang, Lindsay B. Jack and Asoke K. Nandi,: "Fault detection using genetic programming", Mechanical Systems and Signal Processing, Volume 19, Issue 2, p. 271-289,2004

Liao, F., Wang, J.L. and Yang, G.H.: "Reliable robust flight tracking control: an LMI approach," IEEE Transactions on control systems technology, 10(1),76-89,2002

Looze D.P., Weiss J.L., Eterno J.S. and Barrett N.M.: "An automatic redesign approach for restructurable control systems", Control Systems Magazine, IEEE (Volume:5 , Issue: 2), pp. 16-22, May 1985

Lungu, R.; Tudosie, A.: “Single Jet Engine Speed Control System Based on Fuel Flow Rate Control”, Proceedings of the XXVIIth International Conference of Technical Military Academy in Bucuresti, pp. 74-80, section 4, Bucuresti, Romania, Nov. 13-14, 1997

Lunze, Jan,: “Robust multivariable feedback control”, Prentice Hall (New York), 1988

Luo Y.: “Synthesis of robust PID controllers design with complete information on prespecifications for the FOPTD systems”, American Control Conference (ACC), 2011

Luyben,W.L.: “Process Modeling Simulation and Control for Chemical Engineers”. McGraw-Hill, New York, 1973

Maciejowski, J.M.: “Multivariable feedback design”, Addison-Wesley (Wokingham, England and Reading, Mass.), 1989

Manavski, Svetlin A. & Giorgio Valle,: "CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment", BMC Bioinformatics, 2008.

Marcos A. N. Guimar˜aes, Carlos A. Castro and Rub´en Romero,: “Reconfiguration of distribution systems by a modified genetic algorithm”, University of Campinas (UNICAMP), Brazil, 2007

Moradi. M.H., M.R. Katebi, and M.A. Johnson: “The MIMO Predictive PID Controller Design, ” Asian Journal of Control, 4(4), 452–463.2002

Morari, M., & Zafiriou, E.: “Robust process control”. Englewood Cliffs, NJ: Prentice-Hall. 1989

Mukherjee, S., and Mishra, R.N.: ‘Order reduction of linear systems using an error minimisation technique’, J. Frank. Inst., 1987, 323, pp. 23–32

Murrill, P. W.: "Automatic Control of Processes". International Textbook Co., 1967

Neeraj and Kumar A.: "Efficient hierarchical hybrids parallel genetic algorithm for shortest path routing", Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference -, pp. 257-261, 2014

Nicolaidis M.: "Fail-safe interfaces for VLSI: Theoretical foundations and implementation", IEEE transactions on computers, VOL. 47, NO. 1, January 1998

Oliveira, P., Sequeira, J., and Sentieiro, J: "Selection of Controller Parameters using Genetic Algorithms", Engineering Systems with Intelligence. Concepts, Tools, and Applications, Kluwer Academic Publishers, Dordrecht, Netherlands, pp431-438. 1991

Onnen, C., Babuška, R., Kaymak, U., Sousa, J. M., Verbruggen, H. B., and Isermann, R.: "Genetic algorithms for optimization in predictive control", Control Engineering Practice, Vol. 5, Iss. 10, pp1363-1372. 1997

Osyczka A.: "Multicritera optimization for engineering design", in john S. Gero. Editor. Design Optimization, pages 193-227. Academic Press, 1985

Pareto V.: "Cours d'Economie Politique". Droz, Geneva, 1896.

Parzen, G: "The linear parameters and the decoupling matrix for linearly coupled motion in 6 dimensional phase space", Particle Accelerator Conference, 1997. Proceedings of the 1997 (Volume:2)

Patton R.J. & Kambhampati C. & Casavola A. & Zhang P. & Ding S. & Sauter D. : "A generic strategy for fault-tolerance in control systems distributed over a network", Lavoisier, Cachan, FRANCE , 1995

Penttinen J. and Koivo H.N.: "Multivariable tuning regulators for unknown systems", Department of Electrical Engineering, Tampere University of Technology, P.O. Box 527, SF-33101 Tampere 10, Finland, 1976

Porter B.: "Issues in the design of intelligent control systems", Control Systems Magazine, IEEE (Volume:9 , Issue: 1) 1989

Porter B. and Jone A.H.s: "Genetic tuning of digital PID controllers", Electronics Letter, Vol.28, 1992.

Porter B. and Jones A.H.: "Genetic tuning of digital PID controllers", Electronic letters, Vol.28 No.9, 23rd April 1992

Porter, B. and Bradshaw A.: "High-gain error-actuated controllers for a class of linear multivariable plants", Decision and Control including the Symposium on Adaptive Processes, 1979 18th IEEE Conference on (Volume:2)

Porter, B. and Jones, A.H.: "Time-domain identification of transmission zero locations of linear multivariable plants", Decision and Control, 1984. The 23rd IEEE Conference on

Porter, B. and Jones, A.H.: "Design of tunable digital set-point tracking PID controllers for linear multivariable plants using step-response matrices", Decision and Control, 1986 25th IEEE Conference on

Postlethwaite, Ian & MacFarlane, A. G. J.: "A complex variable approach to the analysis of linear multivariable feedback systems", Springer-Verlag (Berlin and New York), 1979

Qamar Saeed, VAli Uddin And Reza Katebi: "Multivariable predictive PID control for quadruple tank, " International science index Vol:4, No:7, 2010

Raisch, D.W.: "Barriers to providing cognitive services", American Pharmacy, 1993

Reeves, D. E., & Arkun, Y.: “Interaction measures for nonsquare decentralized control structures”, A.I.Ch.E. Journal, 35(4), 603., 1989

Rivera, D.E., S. Skogestad and M. Morari: “Internal Model Control 4.PID Controller Design,”Ind. Eng Chem. Proc. Design and Development, 25, 252–265, 1986

Rosenbrock, H.H.: “State-space and Multivariable Theory”, Thomas Nelson and Sons LTD, 1970

Rosenbrock,H.H.: “computer-aided control system design”, Academic press, London and New York, 1974

Rudolph G.: “Evolutionary search under partially ordered sets,” Dept. Comput. Sci./LS11, Univ. Dortmund, Dortmund, Germany, Tech. Rep. CI-67/99, 1999

Rusnak I.: “The generalized PID controller for stochastic systems”, Electrical and ELectronic Engineers in Israel, 2000

Samanta B., AI-Balushi K. R., AI-Araimi S. A.: "Artificial neural networks and genetic algorithm for bearing fault detection", Soft Coput., 2006.

Sarma K.L.N. and Chidambaram M.: “Centralized PI/PIDcontrollers for non-square systems with RHP zeros”, J.Indian Inst. Sci., (2005) pp. 201-215

Satish Nagarajaiah Prof.: “System Identification”, CEVE & MEMS, Rice, Unconventional Wisdom, 2009

Schaffer J. David: “Multiple Objective Optimization with Vector Evaluated Genetic Algorithms”, Proceedings of the 1st International Conference on Genetic Algorithms in July, 1985

Senecal Peter Kelly: “Numerical optimization using the GEN4 Micro-Genetic Algorithm code”, Unicersity of Wisconsin-Madison, 2000

Shi-Ning Ju, Cheng-Liang Chen, Chuei-Tin Chang, Shi-Ning Jua, Cheng-Liang Chena, Chuei-Tin Changb,: “Reliability Engineering and System Safety” 81 163–181, 2003

Shinskey, F. G.: “Process Control Systems”. McGraw-Hill, 1967

Sigurd Skogestad,: “Control structure design for complete chemical plants”, Department of Chemical Engineering, Norwegian University of Science and Technology (NTNU) , Norway, 2003

Singh S. and Mitra R. : “Comparative analysis of robustness of optimally offline tuned PID controller and Fuzzy supervised PID controller”, Proceedings of 2014 RAECS UIET Panjab University Chandigarh, 06-08 March, 2014

Skogestad S. and Postlethwaite I.: “Multivariable Feedback Control”, John Wiley & Sons, Chichester, 1996

Soderstrom T and Stoica P,: “System Identification”, Prentice Hall, 1989

Solihin M.I. , Wahyudi , Kamal, M.A.S. and Legowo, A.: “Objective function selection of GA-based PID control optimization for automatic gantry crane”, Computer and Communication Engineering,. International Conference on, pp. 883-887, 2008. ICCCE 2008

Song H., Hu C.Q., Wang L.F. and Hou E.: “Design of a new high accuracy incremental PID controller with dual temperature sensors”, Control and Decision Conference (CCDC), 2012 24th Chinese

Srinivas N. and Deb K.: “Multiobjective function optimization using nondominated sorting genetic algorithms,” Evol. Comput. vol. 2, no. 3, pp. 221–248, Fall 1995.

Srinivas N., Deb K.: “Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms”. Evolutionary Computation, Vol. 2, No 3, 221-248. 1994.

Svaricek F.: “Zuverlässige numerische Analyse linearer Regelungssysteme. Habilitationsschrift”. Stuttgart: Teubner. 1995

Tan, K.K., S.N. Huang, and T.H. Lee: “Development of a GPC-based PID Controller for Unstable System with Dead Times, ” ISA Transaction, 39, 57–70, 2000

Tan, K.K., T.H. Lee, S.N. Huang, and F.M. Leu: “PID Controller Design Based on a GPC Approach, ” Ind. Eng. Chem. Res., 41, 2013–2022, 2002

Tanttu J.T. and Lieslehto J.: “A comparative study of some multivariable PI controller tuning methods”, in Intelligent tuning and adaptive control, Pergamon Press, pp. 357-362. 1991

Todorovski L., Srinivasan A., Whiteley J. and Gavaghan D.: “Discovering the structure of partial differential equations from example behaviour”, In Proceedings of the Seventeenth international Conference on Machine Learning, pp. 991-998, San Francisco, 2000

Tolle, J. E. : “Understanding patrons use of online catalogs: Transaction log analysis of the search method”. White Plains, NY: Knowledge Industry. (1983)

Tolle, J.E. and Hah, S.: “Online search patterns: NLM CATLINE database”, Journal of the American Society for Information Science. (1985)

Tran T.: “Overriding control for stability with manifest variables”, 2010 11th Int. Conf. Control, Automation, Robotics and Vision Singapore, 7-10th December 2010

Tsay Ruey S.: “Lecture 2: ARMA Models”, Bus 41910, Autumn Quarter 2008

Vardulakis A.I.G.: “Linear multivariable control: Algebraic analysis and synthesis methods”, Chichester, West Sussex, England and New York, 1991

Veillette, R.J. , Medanic, J.V. and Perkins, W.R.: “Design of reliable control systems,” IEEE Transactions on Automatic Control, 37(3),290-304, 1992

Vlachos, C., Williams, D., and Gomm, J. B.: “Genetic approach to decentralised PI controller tuning for multivariable processes”, IEE Proceedings – Control Theory and Applications, Vol. 146, No. 1, pp58-64, January 1999

Wang, H., and Wang, Y.: “Neural-network-based fault-tolerant control of unknown nonlinear systems”, IEE Proc., Control Theory Appl, 1999

Wang, P. and Kwok, D. P.: “Autotuning of Classical PID Controllers Using an Advanced Genetic Algorithm”, International Conference on Industrial Electronics, Control, Instrumentation and Automation (IECON 92), Vol. 3, pp1224-1229. 1992

Wang, Q.G., C.C. Hang and X.P. Yang: “ Single Loop Controller Design Via IMC Principles, ” In Proceeding Asian Control Conference, Shanghai, P.R.China, (2001)

Weile D.S., Michiellsen E. and Goldberg D.E.: “Genetic algorithm design of Pareto optimal broadband microwave absorbers”, IEEE transactions on electromagnetic compatibility, VOL. 38, NO. 3, August 1996

Wiley : “Multivariable Feedback Control-Analysis and Design”, Sigurd Skogestad and IAN Postlethwaite, 2005

William F.Punch,: “How Effective are multiple populations in Genetic Programming”, Michigan State University GARAGe, 1998

Wolovich W.A. : “Linear Multivariable Systems.”, Springer-Verlag, New York. 1974

Wonham, W. Murray: “Linear multivariable control: A geometric approach”, Springer-Verlag (New York), 1979

Yang Y. & Yang G.H. & Soh Y.C.: “Reliable control of discrete-time systems with actuator failure”, Control Theory and Applications, IEE Proceedings - , 2000

Yang, G.H., Wang, J.L. and Soh, Y.C.: “Reliable H_∞ controller design for linear system,” Automatica, 37(5), 717-725, 2001

Yang, J. & Roy, S.: “Joint transmitter-receiver optimization for multi-input multi-output systems with decision feedback”, Dept. of Electr. Eng., Pennsylvania Univ., Philadelphia, PA, 1994

Yoon–Jun Kim and Jamshid Ghaboussi: “A New Genetic Algorithm Based Control Method Using State Space Reconstruction”, Department of Civil Engineering, University of Illinois at Urbana–Champaign, Urbana, IL, USA,1998

Youbin Peng, Damir Vrancic, and Raymond Hanus.: “Anti-windup, bumpless, and conditioned transfer techniques for PID controllers”, Control Systems, IEEE (Volume:16 , Issue: 4), Aug 1996

Young, A. J.: “Process Control”. Instruments Publishing Company, Pittsburgh,1954

YU Jue, ZHUANG Jian, YU Dehong: “Parameter optimization of PID controller based on Complex System Genetic Algorithm in Electro-hydraulic Servo Control System”, Proceedings of the 10th World Congress on Intelligent Control and Automation Beijing, China, July 6-8, 2012.

Zadeh L. A.: “On the Identification Problem,” IRE Transactions on Circuit Theory, 3, pp. 277-281., 1956

Zhao, Q and Jiang, J.: “Reliable state feedback control system design against actuator failures,” Automatica, 34(10), 1267-1272, 1998

Zhuxin J. Lu & Glendale & Ariz,: “Method of Optimal Controller Design for Multivariable Predictive Control Utilizing Range Congrol”, Honeywell Inc, Minneapolis, Minn, 1995

Ziegler, J.G and Nichols, N. B.: “ Optimum settings for automatic controllers”, Transactions of the ASME 64. pp. 759–768. 1942

Zitzler E. and Thiele L.: “Multiobjective optimization using evolutionary algorithms—A comparative case study,” in Parallel Problem Solving From Nature, V, A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, Eds. Berlin, Germany: Springer-Verlag, 1998, pp. 292–301

Zitzler E., Deb K., and Thiele L.: “Comparison of multiobjective evolutionary algorithms: Empirical results,” Evol. Comput., vol. 8, no. 2, pp. 173–195, Summer 2000

Zitzler, E., Laumanns M., Thiele L.: “SPEA2: Improving the Strength Pareto Evolutionary Algorithm”. EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, Athens, Greece (K. Giannakoglou et al., Eds.), 95-100, 2002

Zitzler, E., Thiele L.: “Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach”. IEEE Transactions on Evolutionary Computation, Vol. 3, No 4, 257-271, November 1999