A TOOL FOR DESIGNING ROBUST AUTOPILOTS FOR RAMJET MISSILES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

ALPER KAHVECİOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
AEROSPACE ENGINEERING

JANUARY 2006

Approval of the Graduate School of Natural and Applied Sciences

Prof. Dr. Canan ÖZGEN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science

Prof. Dr. Nafiz ALEMDAROĞLU
Head of the Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science

Dr. Volkan NALBANTOĞLU
Co-Supervisor

Prof. Dr. Nafiz ALEMDAROĞLU
Supervisor

Examining Committee Members

Assoc. Prof. Dr. Serkan ÖZGEN (METU-AEE) _____

Prof. Dr. Nafiz ALEMDAROĞLU (METU-AEE) _____

Dr. Volkan NALBANTOGLU (ASELSAN) _____

Prof. Dr. M. Kemal ÖZGÖREN (METU-ME) _____

Asst. Prof. Dr. İlkay YAVRUCUK (METU-AEE) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Alper KAHVECİOĞLU

ABSTRACT

A TOOL FOR DESIGNING ROBUST AUTOPILOTS
FOR RAMJET MISSILES

KAHVECİOĞLU, Alper
M.S., Department of Aerospace Engineering
Supervisor: Prof.Dr. Nafiz ALEMDAROĞLU
Co-Supervisor: Dr. Volkan NALBANTOĞLU

January 2006, 134 pages

The study presented in this thesis comprises the development of the longitudinal autopilot algorithm for a ramjet powered air-to-surface missile. Ramjet Missiles have short time-of-flight, however they suffer from limited angle of attack margins due to poor operational-region characteristics of the ramjet engine. Because of such limitations and presence of uncertainties involved, Robust Control Techniques are used for the controller design. Robust Control Techniques not only provide an easy limitation/uncertainty/performance handling for MIMO systems, but also, robust controllers promise stability and performance even in the presence of uncertainties of a pre-defined class. All the design process is carried out in such a way that at the end of the study a tool has been developed, that can process raw aerodynamic data obtained by Missile DATCOM program, linearize the equations of motion, construct the system structure and design sub-optimal H_∞ controllers to meet the requirements provided by the user. An autopilot which is designed by classical control techniques is used for performance and robustness comparison, and a non-linear simulation is used for validation. It is concluded that the code, which is very easy to modify for the specifications of other missile systems, can be used as a reliable tool in the preliminary design phases where there exists uncertainties/limitations and still can provide satisfactory results while making the design process much faster.

Keywords: Ramjet Missile, Robust Control, Missile Autopilot

ÖZ

RAMJET FÜZELERİN GÜRBÜZ OTOPİLOT TASARIMI İÇİN BİR ARAÇ

KAHVECİOĞLU, Alper
Yüksek Lisans, Havacılık ve Uzay Mühendisliği Bölümü
Tez Yöneticisi: Prof.Dr. Nafiz ALEMDAROĞLU
Ortak Tez Yöneticisi: Dr. Volkan NALBANTOĞLU

Ocak 2006, 134 sayfa

Bu tezde, "ramjet" itki sistemli bir havadan-karaya füzenin yunuslama otopilotunun geliştirilmesi için gerçekleştirilen çalışmalar sunulmaktadır. "Ramjet" itkili sistemler günümüzde kısa hedefe-uçuş süreleri sebebiyle tercih edilmekle birlikte, "Ramjet" itki sistemlerinin sınırlı operasyonel zarflarından kaynaklanan hücum açısı kısıtları sebebi ile sıkıntı yaşamaktadır. Bu kısıtlar ve varolan belirsizlikler sebebiyle tasarımda Gürbüz Kontrol Teknikleri kullanılmıştır. Gürbüz Kontrol Teknikleri Çok-Girdi-Çok-Çıktı (MIMO) sistemler için limitasyon/belirsizlik/performans isteklerinin kolayca ele alınmasını sağlamakla birlikte, gürbüz kontrolcüler, önceden mertebesi verilen bilinmezlik ve belirsizliklerin olduğu durumlarda dahi kararlığı ve performansı garantileyebilmektedir. Tasarım süreci, çalışmaların sonunda ortaya Missile DATCOM çıktısı olan ham aerodinamik veriyi işleyebilen, sistem hareket denklemlerini doğrusallaştıran, sistem yapısını oluşturan ve kullanıcının gireceği performans isteklerini karşılayan optimal-altı H_∞ kontrolcüleri tasarlayan bir program çıkartacak şekilde yürütülmüştür. Karşılaştırma için klasik kontrol yöntemleri ile tasarlanmış bir otopilot, ve doğrulama için doğrusal-olmayan bir benzetim aracı kullanılmıştır. Diğer füze sistemleri için de kolaylıkla değiştirilebilecek olan bu kodun, bilinmezlik ve belirsizliklerin olduğu ön tasarım çalışmaları sırasında tasarım işini hızlandırırken aynı zamanda tatmin edici sonuçlar verebilen güvenilir bir araç olarak kullanılabileceği değerlendirilmektedir.

Keywords: Ramjet İtkili Füze, Gürbüz Kontrol, Füze Otopilotu

ayakta kalmak için yaslandıklarıma

ACKNOWLEDGMENTS

For me, performing a thesis study was more than learning more, improving my knowledge and writing a thesis: The study also required me to keep my motivation alive and to continue working even in the “bad days”. I would like to thank my advisors, Prof. Nafiz ALEMDAROĞLU and Dr. Volkan NALBANTOĞLU, not only for their academic advises, but for also their patience and support in times “I was in trouble”.

I also would like to thank my colleagues in ROKETSAN, Mr. Ercan ÖRÜCÜ and Mrs. Mine ALEMDAROĞLU, for the very inspiring discussions.

Finally, I would like to thank my family, Zafer, Nurten, Caner KAHVECİOĞLU and Dilek SARAÇOĞLU, for their endless love and support. Indisputably, without them, this study would never be possible.

TABLE OF CONTENTS

PLAGIARISM iii
ABSTRACT iv
ÖZ v
ACKNOWLEDGMENTS vii
TABLE of CONTENTS viii
LIST OF SYMBOLS AND ABBREVIATIONS xiii
CHAPTERS
1 INTRODUCTION 1
2 PROBLEM DEFINITION and PRELIMINARY INFORMATION ABOUT THE SYSTEM 4
2.1 Expected Trajectory 4
2.1.1 Flight Phases 5
2.1.1.1 Booster Phase 5
2.1.1.2 Climbing Phase: 5
2.1.1.3 Cruise Phase 7
2.1.1.4 Terminal Phase 8
2.1.2 General Issues about the Flight Phases: 8
2.2 Design Points 9
2.2.1 Design Point 1 (DP.1) 9
2.2.2 Design Point 2 (DP.2) 10
2.2.3 Design Point 3 (DP.3) 11
2.2.4 Design Point 4 (DP.4) 11
2.3 Requirements for the Autopilot 12
3 MODELING THE SYSTEM 13
3.1 Aerodynamic Model 13
3.1.1 Obtaining Aerodynamic Data 14

3.1.2 Expressing Aerodynamic Data 15
3.2 Mass Model 15
3.3 Inertial Measurement Unit (IMU) Model 15
3.4 Control Actuation System (CAS) Model 16
3.5 Thrust and Velocity Model 17
3.6 Reference Parameters 17
3.7 Missile and Autopilot Axes 17
3.8 Equations of Motion 18
3.9 Non-Linear Simulation 20
4 CLASSICAL CONTROLLER DESIGN 21
4.1 Obtaining Transfer Functions 22
4.2 Obtaining Necessary Data for Design 22
4.3 Inspection of System Dynamics and Deciding on Controller Configuration 23
4.4 Determining Autopilot Coefficients 25
4.4.1 Closing Rate Damping Loop 25
4.4.2 Closing Synthetic Stabilization Loop 26
4.4.3 Closing Acceleration Loop 26
4.4.4 Response of a System Designed Accordingly 27
4.5 Controller Gains 28
4.6 Testing the Design in Linear and Non-Linear Environments 29
4.6.1 Testing in Linear Simulation 29
4.6.2 Testing in Non-Linear Simulation 31
5 ROBUST CONTROLLER DESIGN: THEORY 32
5.1 Historical Background 32
5.2 Theory 33
5.2.1 Problem Statement 34
5.2.2 Norms of Systems 34
5.2.3 Linear Fractional Transformations (LFTs) 36
5.2.4 Transfer Functions Defined on a Feedback Structure 37
5.2.5 Model Uncertainty and Stability/Performance 38
5.2.5.1 Definitions for Stability Functions 38

5.2.5.1.1 Multiplicative Uncertainty 39
5.2.5.1.2 Additive Uncertainty 39
5.2.5.2 Stability/Performance Tests Defined for Sensitivity Functions (SISO case) 40
5.2.5.2.1 Nominal Stability (NS) 41
5.2.5.2.2 Nominal Performance (NP) 41
5.2.5.2.3 Robust Stability (RS) 41
5.2.5.2.4 Robust Performance (RP) 42
5.2.5.3 Stability/Performance Tests Defined for Sensitivity Functions (MIMO case) 42
5.2.5.3.1 Nominal Stability (NS) 43
5.2.5.3.2 Nominal Performance (NP) 43
5.2.5.3.3 Robust Stability (RS) 43
5.2.5.3.4 Robust Performance (RP) 43
5.2.6 Solution to H_∞ problem 44
5.2.7 Structured Singular Value (μ) 48
5.2.7.1 Definition 48
5.2.7.2 Robust Stability (RS) 49
5.2.7.3 Robust Performance (RP) 49
6 ROBUST CONTROLLER DESIGN: APPLICATION 50
6.1 Obtaining State-Space Realization 50
6.2 Formulating the Problem 51
6.2.1 Constructing the Structure for Use with LFT Framework 52
6.2.2 From Performance Specs to Weights 54
6.2.2.1 Uncertainty Weighting Functions W1 to W9 54
6.2.2.2 Uncertainty Weighting Functions W10 to W12 59
6.2.2.3 Performance Weighting Function “W_an_per” 60
6.2.2.4 Performance Weighting Function “W_alpha_per” 61
6.2.2.5 Performance Weighting Function “W_acc_noise” and ”W_gyro_noise” 62
6.2.2.6 Performance Weighting Functions “W_act_pper” and “W_act_sper” 62

6.3 Implementation on MATLAB 63
6.4 Obtaining Controllers 64
6.5 Controller Order Reduction 67
7 RESULTS and CONCLUSION 68
7.1 Results 68
7.1.1 Performance Issues 68
7.1.1.1 Design Point 1 69
7.1.1.2 Design Point 2 71
7.1.1.3 Design Point 3 74
7.1.1.4 Design Point 4 77
7.1.2 Stability Issues 79
7.2 Conclusion 82
REFERENCES 85
APPENDICES
A GRAPHS FOR THE AERODYNAMIC COEFFICIENTS 88
B OPEN-LOOP RESPONSES 93
C OBTAINING TRANSFER FUNCTIONS FOR THE CLASSICAL CONTROLLER 95
D OBTAINING TRANSFER FUNCTIONS FOR THE ROBUST CONTROLLER 98
D.1 Theory 98
D.2 Obtaining f_i 99
D.3 Expressing Aerodynamic Data as Polynomials 100
D.4 Calculating Trim Conditions 102
D.5 Obtaining Linearized Equations 102
D.6 Supplementary Codes 103
D.6.1 Code for “Implementing Nth Order Fitting for an 3D Database“ Algorithm 103
D.6.2 Code for Simplified Algorithm 104
D.6.3 Analytical Derivatives for State Equations 104
E CODE FOR ROBUST CONTROLLER DESIGN 107
E.1 Main Function “main.m” 107

E.2 Sub-function “robust_synthesize.m” 108
E.3 Sub-function “robust_initialize.m” 111
E.4 Sub-function “construct_components.m” 113
E.5 Sub-function “linearize.m” 116
E.6 Sub-function “rp_analysis.m” 116
E.7 Sub-function “polynomialize.m” 117
E.8 Sub-function “analytic_soln.m” 118

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol	Definition	Unit
α	Angle of Attack in Pitch Axis	rad
β	Angle of Attack in Yaw Axis	rad
δ	Control Surface Deflection	rad
C_n	Force Coefficient in z_b	-
C_m	Moment Coefficient in y_b	-
C_a	Force (drag) Coefficient in x_b	-
C_{nq}	Damping Coefficient in z_b	$1/rad$
C_{mq}	Damping Coefficient in y_b	$1/rad$
C_{aq}	Damping Coefficient in x_b	$1/rad$
C_{n_α}	Derivative of C_n w.r.t α	$1/rad$
C_{m_α}	Derivative of C_m w.r.t α	$1/rad$
C_{n_δ}	Derivative of C_n w.r.t δ	$1/rad$
C_{m_δ}	Derivative of C_m w.r.t δ	$1/rad$
l_{ref}	Reference Length (missile diameter)	m
S_{ref}	Reference Area	m^2
m	Mass	kg
I_{ij}	Moment of Inertia on Missile Body Axis C_{ij}	kgm^2
J	Moment of Inertia on Missile Body Axis C_{yy} $(= I_{yy})$	kgm^2
T	Thrust	N

Symbol	Definition	Unit
V	Total Velocity in Body Axis	m/s
Q	Dynamic Pressure	Pa
u, v, w	Velocities in Body Axes x_b, y_b, z_b	m/s
p, q, r	Angular Velocities in Body Axes x_b, y_b, z_b	rad/s
ϕ, θ, ψ	Euler Angles in Inertial Axes x_i, y_i, z_i	rad
L, M, N	Net Moment acting on Missile on Body Axes x_b, y_b, z_b	Nm
F_x, F_y, F_z	Net Force acting on Missile on Body Axes x_b, y_b, z_b	N
ω_{rgu}	Natural Frequency of Rate Gyroscope	Hz
ζ_{rgu}	Damping Ratio of Rate Gyroscope	-
ω_{acc}	Natural Frequency of Accelerometer	Hz
ζ_{acc}	Damping Ratio of Accelerometer	-

Abbreviation	Definition
$\mathbb{R}$	Set of Real Numbers
$\mathbb{C}$	Set of Complex Numbers
$K1$, $K2$, $K3$, K_f	Classical Controller Coefficients
ACC	Accelerometer
CAS	Control Actuation System
CCT	Classical Control Techniques
CL	Closed-Loop
DP	Design Point
FDLTI	Finite Dimensional Liner Time Invariant
GM	Gain Margin
IMU	Inertial Measurement Unit

Abbreviation	Definition
LFT	Linear Fractional Transformations
MIMO	Multi Input Multi Output
NP	Nominal Performance
NS	Nominal Stability
OL	Open-Loop
PM	Phase Margin
RCT	Robust Control Techniques
RGU	Rate Gyroscopic Unit
RS	Robust Stability
RP	Robust Performance
SISO	Single Input Single Output

Throughout the text,

Numbers in brackets	DENOTE	References
Numbers in parenthesis		Equations

CHAPTER 1

INTRODUCTION

The “autopilot” is an algorithm in the flight control software of a flight system. It has three major tasks:

1. To provide stability if the missile is inherently unstable.
2. To produce control surface deflection commands in order to apply guidance algorithm orders.
3. To assure that the first and second tasks are fulfilled in the possible presence of disturbances like gusts and (continuous) wind.

The theory for designing such controllers, namely the Feedback Control Theory, is well developed. There are various techniques like classical control, modern control, robust control, adaptive control, non-linear control... The common feature of all these methods is that, regardless of the method used, designing an autopilot requires detailed information about the system, such as

- Aerodynamic behavior of the body,
- Mass and inertia properties,
- Parameters related to flight trajectory.
- Information on sensors on-board.
- Information on actuators on-board.

Another common feature is that, given all this data, designing an autopilot requires considerable time and effort. Moreover, in case any of these parameters listed above changes, the whole design process has to be repeated.

On the other hand, as a matter of fact, conceptual and/or preliminary design phases of a flight system involve unknowns and uncertainties. As a result of

continuous trade-off studies, parameters continuously change. Once it is decided that a parameter must change, all the calculations have to be repeated over again to see the effect on the overall system.

The motivation of this thesis arose due to these facts. A conceptual design process for a ramjet powered air-to-surface missile was being performed, and as a part of this study an autopilot had to be designed[1]. The idea was to perform the design studies such that, a code, which will minimize the effort required for the iteration, will be developed along with the design process. Therefore, the study presented in this thesis comprises of

- The results of an autopilot design for one of the alternative configuration in the preliminary design.
- A code for an easier "next iteration".

As one might easily guess, such a code can also be used in detailed design phases. Moreover, it can also be modified quite easily and be used as a standalone design tool in a different design problem.

In this study, all steps taken for the design of a ramjet autopilot and additional steps for automating the process and verifying the tool has been presented. Robust Control Theory has been used in the code because of its efficiency in handling such systems and due to its ease for automatization. For evaluation of controller performance, a different autopilot has been used. This autopilot has been designed by classical control techniques using root-locus and bode plot. These methods are well-known and proved to be useful especially in SISO problems.

The design problem in hand is a ramjet powered air-to-surface missile. This type of missile is very popular because of its short time-to-target properties. This feature is surely due to the ramjet propulsion system, which enables a

[1] It must be mentioned that a very frequent approach is to perform preliminary design calculations without the autopilot: Without any simulation that represent the dynamics, angle of attack is determined such that the required normal acceleration is obtained. Control surface deflection is then determined such that, at that angle of attack total pitch moment on the missile is zero. However, current problem is a special case that basic trajectory planning calculations are very much dependent on the autopilot performance because of the strict angle of attack limitations. These limitations are due to shut-off problem of ramjet propulsion systems. This issue will be discussed in detail in the following sections.

flight velocity around 3.5 Mach. However, ramjet propulsion involves its unique challenges in a wide-range of areas. One of these is presented in this study. Another is about the design of inlets and is presented in [13].

This study will be used in the preliminary design phase of the missile, where a repetitive design studies of an autopilot is needed for trajectory planning. However, it may be argued that, the code developed in this thesis may be used in more complicated studies that will be performed in detailed design phases.

CHAPTER 2

PROBLEM DEFINITION and PRELIMINARY INFORMATION ABOUT THE SYSTEM

This chapter presents the information about the trajectory and the flight parameters of the missile. Data given in this chapter is provided by the System Designer at the end of some preliminary calculations and is the basis for the conceptual design.

2.1 Expected Trajectory

Schematic figure given in Figure 2-1 shows the conceptual trajectory of the missile.

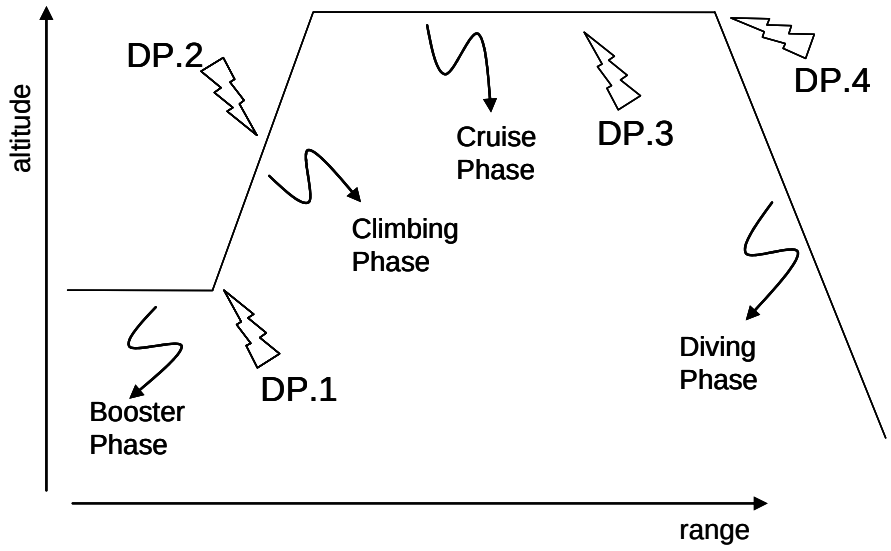


Figure 2-1 Conjectural Trajectory

2.1.1 Flight Phases

2.1.1.1 Booster Phase

During this phase the missile is accelerated as quickly as possible such that the ramjet can be started.

Flight during the booster phase is described as accelerating level flight; hence, the flight path angle is set to be zero in the entire phase.

The launching altitude for the missile is given as 5000m. Minimum launching speed is given as M=0.5. Booster phase will end when the booster grain is exhausted. Velocity at the end of this phase depends on the launching speed, but booster grain will be designed such that it can accelerate a missile launched from 5000m at M=0.5 to a speed of M=2.5. Simulations show that such a flight takes only a few seconds. It is a fact that, in this phase, dynamic pressure will not be enough for aerodynamic control. Also, maneuvering during the booster phase may not be feasible. Because of such concerns, no guidance commands other than gravity compensation will be given during these phase. Similarly, no acceleration due to steering is expected; the missile only tries to keep the direction it is launched toward.

Launching weight of the missile is 800kg, and during the booster phase 200kg of booster is used. Booster provides a constant thrust of 80000N.

2.1.1.2 Climbing Phase:

Climbing phase comprises the maneuver that conveys the missile to its cruising altitude.

The phase will start with a minimum velocity of M=2.5 and a minimum altitude of 5000m. Some trajectory planning simulations need to be performed for determining the conditions at different launching altitudes and velocities.

Since the missile uses lift to increase its flight path angle in order to climb, angle of attack is the variable to be controlled. In terms of flight path angle and angle of attack, climbing phase can be divided into two as shown in Figure 2-2.

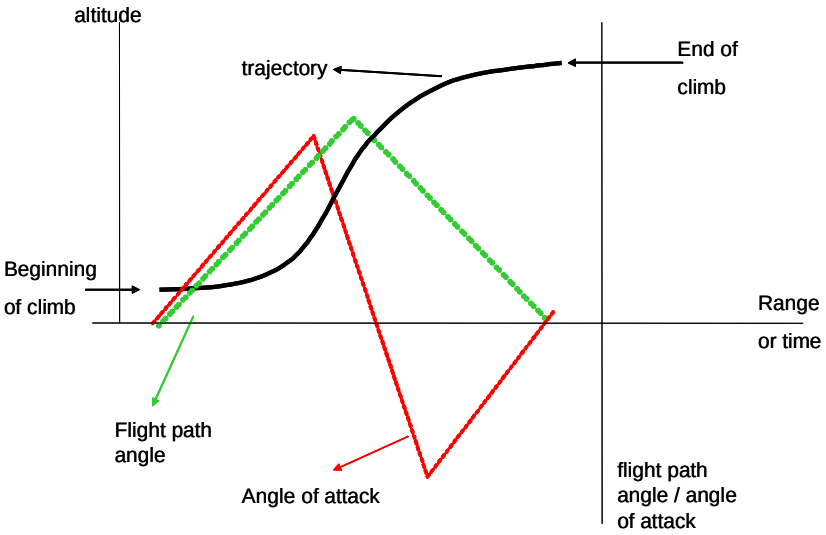


Figure 2-2 Change of Angle of Attack and Flight Path Angle during Climbing Phase

Figure 2-2 shows the variation of angle of attack and flight path angle during the climb phase, and is not to scale. The exact values of these variables will be determined by trajectory planning. The crucial thing for guidance in this phase is the angle of attack limit for the ramjet, since the only way to obtain such a flight path angle behavior is having such an angle of attack profile. It is given that the ramjet can operate in [-3°, 6°] range. In order to decrease the flight path angle in the second part of the climb, the missile must pull negative g's, thus negative angles of attack, which can result in an engine shut-off. During design,

- Ramjet will need to be redesigned so that it can operate at higher negative angles of attack
- The trajectory planning algorithm will have to find a way to pitch-down the missile with lower (in magnitude) angles of attack, also incorporating gravity to help the pull-down maneuver.

These considerations on guidance algorithm and the answer for the problem will affect the design of the autopilot. However, for the time being, it may be expected that during the initial phase of the climb the autopilot will have to ensure a maximum angle of attack of 6°, whereas a minimum of -3° during the last phase. Therefore, although quantities in requirements may change, a drastic change in autopilot structure is not expected.

Actually, most significant outcome of the trade-off in the design process is the point where the cruise phase starts: the less the g's pulled[2], the later the cruise phase starts. Whereas, the missile is intended to start its cruise phase as earlier as possible, since the ramjet is optimized to work in that region.

The aim at the end of climb is to reach the cruise altitude, which is 16000m, with zero flight path angle and if possible, with a flight speed of M=3.5. Whether the missile will be able to reach M=3.5 flight speed at the end of the climb phase is a major problem that has to be found-out after the trajectory planning and the propulsion system design. If it can not, the missile will have to accelerate in the early phases of the cruise. However, it is not always possible to accelerate while trying to climb. Also, in the light of this discussion, it is intended that the ramjet will provide maximum thrust available during the whole climbing phase.

During climbing, ramjet provides maximum thrust (7500N), and uses 35 kg of liquid fuel.

As in boost phase, it is intended that there will be no guidance command for steering. This is an issue that will be clarified after the launch envelope of the missile is determined. In case the missile is launched with an off-set azimuth angle, if the missile diverges considerably before driving it to the cruise phase, a yaw acceleration in climb will be necessary. But as mentioned above, to begin steering in cruise, where velocity is high and no complex maneuvers in other axes are present, is preferable.

2.1.1.3 Cruise Phase

During the cruise phase, a steady, level flight is performed. Also, velocity is kept constant at M=3.5. Cruise is performed at an altitude of 16000m.

As explained above, cruise phase starts with condition that flight path angle is zero, altitude is 16000m and velocity is M=3.5. If this condition is not satisfied, a transition phase is needed in which the ramjet again gives maximum thrust to

[2] so the angle of attack is low

accelerate the missile. In order to maintain altitude, an altitude hold algorithm will be employed by the guidance algorithm.

As velocity has to be kept constant, a velocity control system has to be used. This can be achieved by controlling the thrust: Assuming that the pitch and the yaw dynamics are much slower than the dynamics of the ramjet engine, a separate controller can be used to adjust the amount of fuel fed to the combustion chamber.

Cruise phase ends with the command from the guidance.

In cruise phase, ramjet provides 4500N average thrust, and uses 60kg of liquid fuel. At the end of the phase, weight of the missile reduces to 500kg.

Flight regime in cruise phase is an altitude which involves relatively small disturbances. Therefore, autopilot that controls the maneuvers in the pitch plane is not forced a lot. Consequently, control authority can be used to perform maneuvers in yaw plane primarily.

2.1.1.4 Terminal Phase

Terminal phase comprises the maneuvers that guide the missile to the target and ensures the angle of impact is a pre-determined one.

Terminal phase starts at a velocity of 3.5 M, and 16000m altitude, with the engine shut-down. During this unpowered flight, relatively large accelerations and high angle of attack values are experienced due the guidance algorithm.

At the terminal conditions, the missile will have a velocity of M=1.5.

2.1.2 General Issues about the Flight Phases:

- During the whole flight, roll stabilization will be performed.
- During the whole flight, any of the maneuvers has to cause motions less than 5g's due to structural limitations.
- During the whole flight, no maneuvers has to cause motions more than 60°/s, due to accuracy limitations of the inertial measurement units.
- During the whole flight, no actuator commands must cause deflections more than 20° and deflection rates more than 250°/s, due to structural limitations of the control actuator.

2.2 Design Points

A design point defines local altitude, angle of attack, velocity, mass and thrust. Design points must be chosen such that they represent the entire flight regime. In this thesis, design points are chosen such that they represent the most important phases of the flight. Autopilot will be tested for performance and robustness at these selected flight conditions:

- Ramjet Powered, (+) "g" command given by the guidance algorithm.
- Ramjet Powered, (-) "g" command given by the guidance algorithm.
- Ramjet Powered, level flight.
- Unpowered, maximum maneuvering required.

As mentioned above, results will be given for the above cases. However, for a full-envelope missile autopilot,

- Controllers have to be designed at some other points on the trajectory. With the code developed, it is easy to obtain the controllers for some other design points.
- Throughout the flight, deflection commands have to be computed based on controllers gains determined at these design points. Such an algorithm is called "gain scheduling". Gain-scheduling may depend on different parameters; most common of which are velocity, dynamic pressure, and angle of attack.

2.2.1 Design Point 1 (DP.1)

This design point defines the flight condition at the end of boost phase, where the angle of attack increases as much as possible in order to provide an acceleration that will result in a pitch-up maneuver. Provided maximum thrust, the velocity will increase immediately, however the design point shall consider the velocity is still M=2.5.

Typical parameter values for the design point 1 are given in Table 2-1.

At the end of first part of climbing phase, the velocity is expected to be around M=3. The variation in mass during this phase is also small. Therefore, for gain-

scheduling, a combined approach depending on both the dynamic pressure and the angle of attack might be necessary.

Table 2-1 Data for DP.1

Property	**Value**
Altitude	5000m
Velocity	M=2.5
Thrust	7500N (Ramjet on)
Mass / cg from nose	600kg / 275cm
Expected angle of attack range	[0°, +6°] Design will be performed for (+3°)
Expected g-envelope	[-1 4]g

2.2.2 Design Point 2 (DP.2)

This design point defines the flight condition at the end of first part of climbing phase, where the missile has 20°-30° flight path angle, and wants to pitch-down. This is the crucial phase for the ramjet, as explained in section 2.1.1.2. Typical parameter values for the DP.2 are given in Table 2-2.

Table 2-2 Data for DP.2

Property	**Value**
Altitude	12000m
Velocity	M=3
Thrust	7500N (Ramjet on)
Mass / cg from nose	575kg / 275cm
Expected angle of attack range	[-3°, 0°] Design will be performed for (0°)
Expected g-envelope	[-2 0]g

About the gain-scheduling, same argument with the one in DP.1 applies here.

2.2.3 Design Point 3 (DP.3)

This design point defines the flight condition in the cruise phase.
Typical characteristics of the design point are given in Table 2-3.

Table 2-3 Data for DP3

Property	**Value**
Altitude	16000m
Velocity	M=3.5
Thrust	4500N (Ramjet on)
Mass / cg from nose	560kg / 273cm (values at the beginning of cruise)
Expected angle of attack range	[0°, 6°] Design will be performed for (+3°)
Expected g-envelope	[-1 2]g

During cruise phase, dynamic pressure is nearly constant during the whole flight, so it is impossible to schedule the autopilot gains depending on dynamic pressure. The only parameter that is changing is the mass; therefore the scheduling must be based on mass.

2.2.4 Design Point 4 (DP.4)

This design point defines the flight condition at the end of cruise phase, where the ramjet is off. After this point the missile starts the diving maneuver, so the altitude and velocity conditions are taken the same as in the cruise phase.
Typical characteristics of the design point are given in Table 2-4.

Table 2-4 Data for DP.4

Property	**Value**
Altitude	16000m
Velocity	M=3.5
Thrust	0N (Ramjet off)
Mass / cg from nose	500kg / 270cm
Expected angle of attack range	[-15°, 15°] Design will be performed for (-5°)
Expected g-envelope	[-4 +2]g

In the diving phase, it is gain-scheduling based on both dynamic pressure and angle of attack might be needed.

2.3 Requirements for the Autopilot

Following are the requirements for the autopilot as declared by the System Designer. The values are determined by the trajectory planning studies and are considered to be of crucial importance for the overall performance of the system. Values given in brackets denote they are subject to change.

1. Steady state error in acceleration response shall be less than 10%.
2. Angle of attack shall be in the [-3°, 6°] range in ramjet powered flight.
3. Rise time shall be less than [1] seconds.
4. Response shall settle in [1.5] seconds.
5. Overshoot in response to step input shall be less than [15%].
6. Maximum tail deflection command shall be less than 15°.
7. Maximum tail deflection rate shall be less than 200°/s.

CHAPTER 3

MODELING THE SYSTEM

Autopilot design requires inputs from almost all features of the system. Almost independent of the control methodology used, modeling of the components of the missile and definition of notation has to be made. Also, if not already present, an environment simulating tool for testing/validating the controller is required. This chapter comprises

- The modeling of external (aerodynamic) surface of the missile, sensors and control actuation system so that design inputs for controller design are generated.
- Missile axes and equations of motion.
- Preparation of a non-linear simulation, in which the controller, designed for a linear plant, will be tested in an environment that is closer to the real-world.

3.1 Aerodynamic Model

Most important feature of a system is its aerodynamic characteristics as they primarily determine the response of the system. Therefore, an aerodynamic model that gives aerodynamic data is needed.

3.1.1 Obtaining Aerodynamic Data

Since this thesis aims to be a part of a study in conceptual design phase where final configuration of the system is not frozen yet, one shouldn't expect accurate aerodynamic data. Instead, simple tools that can provide reasonable information are used to design the autopilot and the autopilot is expected to "make the system work" until more accurate data is available. Hence, the available software, Missile DATCOM[3] is used to obtain the aerodynamic data.
The work "obtaining data" consists of preparing an input file for the Missile DATCOM software, where the necessary data for the program is provided to let the program run. The input file for DATCOM has been prepared by ROKETSAN Missile Industries.
In this study two sets of aerodynamic data are obtained from DATCOM. The first set of data is for the design points and the second set is for the non-linear simulation, which in turn includes the first. At conditions other than the design points, aerodynamic coefficients are calculated in order to obtain the aerodynamic database used in nonlinear simulation
Table below reveals the parameter space that the databases are constructed for.

Table 3-1 Database Parameter Space

	α points (deg)	**δ points (deg)**	**Mach values**
DP1	-1,0,2,3*,4,6,7	-15,-10,-5,0,5,10,15	2.3,2.4,2.5*,2.6,2.7
DP2	-4,-3,-2,-1,0*,1,2	-15,-10,-5,0,5,10,15	2.8,2.9,3*,3.1,3.2
DP3	-1,0,2,3*,4,6,7	-15,-10,-5,0,5,10,15	3.3,3.4,3.5*,3.6,3.7
DP4	-15,-10,-5*,0,5,10,15	-15,-10,-5,0,5,10,15	2.8,2.9,3*,3.1,3.2

* denotes design points

All the aerodynamic coefficients are calculated within this parameter space, and the graphs for the aerodynamic coefficients are given in appendix A.

[3] Missile DATCOM is a semi-empiric tool for predicting aerodynamic coefficients of missiles. It is primarily used an aerodynamic design tool, which has the predictive accuracy suitable for preliminary design.

3.1.2 Expressing Aerodynamic Data

Missile DATCOM calculates the parameters for specific points that represent the flight conditions. In order to perform the controller design, the tabulated aerodynamic data which is expressed in functional (continuous) form. at each of these design point[4]. Thus, each aerodynamic coefficient is expressed in terms of a polynomial as follows:

$$\begin{aligned} C_i \triangleq & (b_{22}\cdot\delta^2 + b_{21}\cdot\delta + b_{20})\cdot\alpha^2 \\ & +(b_{12}\cdot\delta^2 + b_{11}\cdot\delta + b_{10})\cdot\alpha \\ & +(b_{02}\cdot\delta^2 + b_{01}\cdot\delta + b_{00}) \end{aligned} \qquad (3.1)$$

Here, "i" must be replaced with "n" for normal force coefficient, "m" for pitch moment coefficient, "a" for drag coefficient, "q" for pitch damping coefficient. For calculating each (sub) coefficient given here, a method has been developed, and is given in appendix D.3.

3.2 Mass Model

No functional (continuous) mass model is applied; at each design point, a single mass value which is given in section 2.2 is used and kept constant (for a given DP) in all studies.

3.3 Inertial Measurement Unit (IMU) Model

IMU is the unit that comprises the accelerometer and rate gyroscope. These sensors measure the motions in autopilot coordinate system (see Figure 3-1)
IMU can not be chosen in conceptual design phase; rather, guidance and navigation studies will generate the requirements for the IMU to be used. In controller design, it is very common to assume that the IMU is perfect for preliminary studies, which is also the case in this thesis. Dynamics of IMU will only be considered in robustness analyses. Therefore, an IMU model that is

[4] This is a fairly standard application. Especially wind tunnel data can be found in polynomial form. For an example of polynomial representation of aerodynamic coefficients, see [21].

thought to be sufficient (fast enough) for the missile is used. The transfer functions for the IMU are as follows:

For the Rate Gyroscopic Unit (RGU)

$$G(s)_{RGU} \triangleq \frac{\omega_{RGU}^{\ 2}}{s^2 + 2 \cdot \zeta_{RGU} \cdot \omega_{RGU} \cdot s + \omega_{RGU}^{\ 2}} \qquad (3.2)$$

where

$$\begin{aligned} \omega_{RGU} &= 45Hz \\ \zeta_{RGU} &= 0.7 \end{aligned} \qquad (3.3)$$

The gyroscope is capable to sense turn rates as fast as 60°/s.

For the Accelerometer (ACC)

$$G(s)_{ACC} \triangleq \frac{\omega_{ACC}^{\ 2}}{s^2 + 2 \cdot \zeta_{ACC} \cdot \omega_{ACC} \cdot s + \omega_{ACC}^{\ 2}} \qquad (3.4)$$

where

$$\begin{aligned} \omega_{ACC} &= 25Hz \\ \zeta_{ACC} &= 0.7 \end{aligned} \qquad (3.5)$$

The accelerometer is able to sense accelerations as large as 10g's.

3.4 Control Actuation System (CAS) Model

CAS is one of the most crucial components in design of an autopilot, since it is the component which has dynamics closest to that of the missile. It is evident that the specifications for the CAS can not be known exactly beforehand during the conceptual design phase, but if dynamics of the CAS are not assumed properly, it is probable that one will has to redesign the autopilot. CAS is modeled as a second order system. Transfer function for the CAS is as follows:

$$G(s)_{CAS} \triangleq \frac{\omega_{CAS}^{\ 2}}{s^2 + 2 \cdot \zeta_{CAS} \cdot \omega_{CAS} \cdot s + \omega_{CAS}^{\ 2}} \qquad (3.6)$$

where

$$\begin{aligned} \omega_{CAS} &= 10Hz \\ \zeta_{CAS} &= 0.7 \end{aligned} \qquad (3.7)$$

CAS is accepted to provide 20° of control surface deflections with 250°/s turn rates. However, as mentioned in section 2.3, the autopilot is allowed to use 15° control surface deflections and 200°/s turn rates.

3.5 Thrust and Velocity Model

Throughout the study no dynamic model for thrust is considered. This means that the thrust is not a control parameter for pitch, yaw and roll autopilots. For all powered phases of flight except the cruise, thrust is assumed to be maximum available thrust. For cruise phase, it is assumed that the velocity is kept constant at the ramjet design Mach number by a separate algorithm.

3.6 Reference Parameters

Some reference parameters that will be used throughout the study are as follows:

$$l_{ref} \triangleq 0.38m \qquad (3.8)$$

l_{ref} is the diameter of the missile.

$$S_{ref} \triangleq \pi \frac{{l_{ref}}^2}{4} \qquad (3.9)$$

3.7 Missile and Autopilot Axes

Definitions for the coordinates and control surface deflections that will be used throughout the study are shown in Figure 3-1. There are two basic coordinate systems: first is a right-handed coordinate system which is fixed on the missile body. Other one is the autopilot coordinate system, which also obeys the right-hand rule, and is coincident with the body coordinate system but axes are pointing in the opposite directions with the first. The reason there exists two different coordinate systems is the desire to assign (+) to the upwards direction for autopilot.

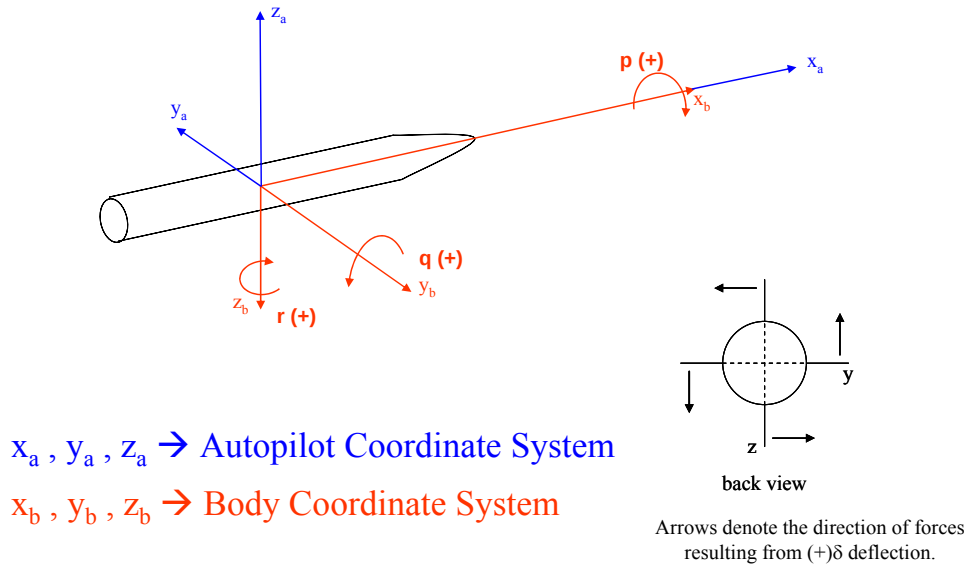
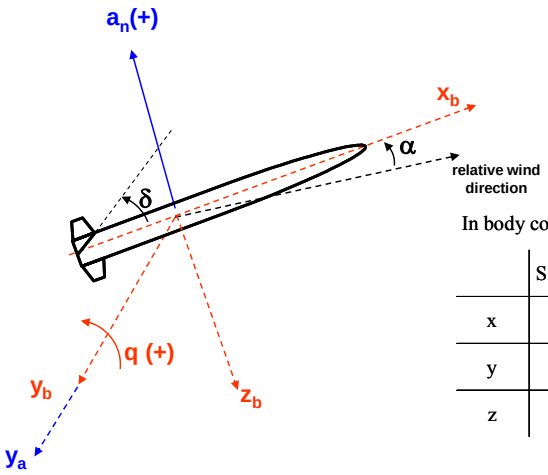


Figure 3-1 Coordinate Systems used

Definitions for other related parameters are given in Figure 3-2.



In body coordinates

	Speed	Angular Speed	Net Force	Net Moment	Inertia
x	u	p	F_x	L	I_{xx}
y	v	q	F_y	M	I_{yy}
z	w	r	F_z	N	I_{zz}

Figure 3-2 Other Definitions and Notation

3.8 Equations of Motion

For the completeness of the study, the equations of motion are given below, which can also be found in the literature, [1], [15] and [17].

$$
\begin{aligned}
F_x &= m\cdot(\dot{u}+q\cdot w-r\cdot v)\\
F_y &= m\cdot(\dot{v}+r\cdot u-p\cdot w)\\
F_z &= m\cdot(\dot{w}+p\cdot v-q\cdot u)\\
L &= I_{xx}\cdot\dot{p}-I_{yz}\cdot(q^2-r^2)-I_{zx}\cdot(\dot{r}+p\cdot q)-I_{xy}\cdot(\dot{q}-r\cdot p)-(I_{yy}-I_{zz})\cdot q\cdot r\\
M &= I_{yy}\cdot\dot{q}-I_{zx}\cdot(r^2-p^2)-I_{xy}\cdot(\dot{p}+q\cdot r)-I_{yz}\cdot(\dot{r}-p\cdot q)-(I_{zz}-I_{xx})\cdot r\cdot p\\
N &= I_{zz}\cdot\dot{r}-I_{xy}\cdot(p^2-q^2)-I_{yz}\cdot(\dot{q}+r\cdot p)-I_{zx}\cdot(\dot{p}-q\cdot r)-(I_{xx}-I_{yy})\cdot p\cdot q
\end{aligned}
\qquad (3.10)
$$

where F_x, F_y, F_z, L, M and N are the resulting aerodynamic, gravitational and thrust forces acting on the missile, on the relevant axis.

These equations are valid in an inertial coordinate frame and it is assumed that the missile is rigid.

Further assumptions can be made

- C_{xz} is a plane of symmetry so that $I_{yz}=I_{xy}=0$. (3.11)
- Axes are principal (coordinate system is located at the center of gravity) so that $I_{xz} = 0$. (3.12)

Then (3.10) become

$$
\begin{aligned}
F_x &= m\cdot(\dot{u}+q\cdot w-r\cdot v)\\
F_y &= m\cdot(\dot{v}+r\cdot u-p\cdot w)\\
F_z &= m\cdot(\dot{w}+p\cdot v-q\cdot u)\\
L &= I_{xx}\cdot\dot{p}-(I_{yy}-I_{zz})\cdot q\cdot r\\
M &= I_{yy}\cdot\dot{q}-(I_{zz}-I_{xx})\cdot r\cdot p\\
N &= I_{zz}\cdot\dot{r}-(I_{xx}-I_{yy})\cdot p\cdot q
\end{aligned}
\qquad (3.13)
$$

Just for the sake of completeness, other equations defining Euler angles are given in (3.14).

$$
\begin{aligned}
\dot{\phi} &= p+q\cdot\sin(\phi)\cdot\tan(\theta)+r\cdot\cos(\phi)\cdot\tan(\theta)\\
\dot{\theta} &= q\cdot\cos(\phi)-r\cdot\sin(\phi)\\
\dot{\psi} &= \big(q\cdot\sin(\phi)+r\cdot\cos(\phi)\big)\cdot\sec(\theta)
\end{aligned}
\qquad (3.14)
$$

This study assumes that the axes are “completely uncoupled”. This assumption can be realized by the roll autopilot. The compensated roll dynamics is supposed to be very fast as compared to missile dynamics in rest of the axes, which is usually the case. With this result, one can assume that roll angle and roll rate are zero. Then (3.13) become

$$
\begin{aligned}
F_x &= m \cdot (\dot{u} + q \cdot w - r \cdot v) \\
F_y &= m \cdot (\dot{v} + r \cdot u) \\
F_z &= m \cdot (\dot{w} - q \cdot u) \\
L &= I_{xx} \cdot \dot{p} - (I_{yy} - I_{zz}) \cdot q \cdot r \\
M &= I_{yy} \cdot \dot{q} \\
N &= I_{zz} \cdot \dot{r}
\end{aligned}
\qquad (3.15)
$$

Finally, the equations for the angle of attack and the sideslip angle are

$$\alpha = \tan^{-1}(\frac{w}{u}) \qquad (3.16)$$

$$\beta = \tan^{-1}(\frac{v}{u}) \qquad (3.17)$$

3.9 Non-Linear Simulation

A code for simulating both the open-loop and the closed-loop responses is prepared. Open-loop responses are used for investigating the plant behavior and closed-loop responses are used to validate the classical and the robust controllers. The code basically involves

- A part for initializing the system parameters given in chapter 2 and chapter 3.
- A part for solving (3.16) and equations related to pitch axis in (3.15), which fulfills the “flight mechanics” part.
- A part for simulating the CAS dynamics.
- A part for simulating the controller dynamics.
- A part for managing and plotting the data.

CHAPTER 4

CLASSICAL CONTROLLER DESIGN

Classical Control Techniques (CCT) based on frequency domain is a good way to start designing an autopilot. Such a study not only results in a controller, but also (and perhaps more importantly) allows the designer to feel the system dynamics. In this section a controller will be designed with the classical control techniques[5]. This controller will be used to compare the results of robust controller synthesis.

The classical control techniques are simple yet powerful analysis and design tools for SISO systems. They work on the frequency domain and have corresponding performance and robustness measures, like phase margin, gain margin, cross-over frequency, bandwidth, etc. Detailed information on classical control techniques can be found in [15] and [16]. Moreover, [17], [18] contain examples of autopilots designed by CCT.

Designing a classical controller involves several steps:

- Obtaining the transfer functions.
- Obtaining the necessary data for the design.
- Inspection of system dynamics so as to decide on the controller configuration.
- Determining the autopilot coefficients.
- Testing the design in linear and/or non-linear environments.

[5] Hereafter, this controller will be called the “classical controller” .

4.1 Obtaining Transfer Functions

As mentioned in earlier sections, transfer functions which describe the response of the system to control actions[6] have to be derived.

Since the equations of motion given in (3.15) are non-linear, they have to be linearized for designing the controller. To linearize these equations, there are a number of ways, two of which will be used in this study. First one will employ a regular procedure; trim points are found and the system is assumed to be linear around these points. Second method will use some small angle assumptions, so that the equations can be reduced to a linear form. Former method will be used in this chapter, whereas the latter will be employed by the robust controller.

Derivation of transfer functions for classical controller can be found in [14], [17] and [18], and is also given in appendix C for completeness.

Transfer functions that will be used for the classical controller are given as

$$\frac{q(s)}{\delta(s)} = \frac{M_\delta \cdot s + (M_\delta \cdot N_\alpha - M_\alpha \cdot N_\delta)}{s^2 + (N_\alpha - M_q) \cdot s - (M_q \cdot N_\alpha + M_\alpha \cdot (1 - N_q))} \tag{4.1}$$

$$\frac{a_N(s)}{\delta(s)} = V \cdot \frac{N_\delta \cdot s^2 - (M_q \cdot N_\delta - M_\delta \cdot N_q) \cdot s + (N_\alpha \cdot M_\delta - N_\delta \cdot M_\alpha)}{s^2 + (N_\alpha - M_q) \cdot s - (M_q \cdot N_\alpha + M_\alpha \cdot (1 - N_q))} \tag{4.2}$$

4.2 Obtaining Necessary Data for Design

Expressions given in 4.1 are the transfer functions that will be used in the design process. When these equations are inspected it is seen that the following parameters are necessary:

$\left.\begin{matrix} C_{n_\alpha} \\ C_{m_\alpha} \\ C_{n_\delta} \\ C_{m_\delta} \\ C_{n_q} \\ C_{m_q} \end{matrix}\right\}$ Aerodynamic Parameters $\quad \left.\begin{matrix} \mathrm{m} \\ \mathrm{I_{yy}} \\ l_{ref} \\ S_{ref} \end{matrix}\right\}$ Mass Parameters $\quad \left.\begin{matrix} \mathrm{Q} \\ \mathrm{V} \end{matrix}\right\}$ Flight Parameters

[6] i.e., the control surface deflections

Some of these parameters (namely l_{ref} and S_{ref}) are constants and are given in section 3.6. Some of them (namely m , I_{yy} , Q and V) are parameters specific for a design point and are given in section 2.2. Rest of the parameters are derivatives of the aerodynamic coefficients and will be obtained next.

The procedure of obtaining these derivatives is simple: it starts with the representation structure of the coefficients, which is explained in section 3.1.2.

$$\begin{aligned} C_i \triangleq & (b_{22}\cdot\delta^2 + b_{21}\cdot\delta + b_{20})\cdot\alpha^2 \\ & +(b_{12}\cdot\delta^2 + b_{11}\cdot\delta + b_{10})\cdot\alpha \\ & +(b_{02}\cdot\delta^2 + b_{01}\cdot\delta + b_{00}) \qquad \text{i = n,m} \end{aligned} \tag{4.3}$$

Derivatives with respect to α and δ are defined by

$$\begin{aligned} C_{i_\alpha} \triangleq & 2\cdot(b_{22}\cdot\delta^2 + b_{21}\cdot\delta + b_{20})\cdot\alpha \\ & +(b_{12}\cdot\delta^2 + b_{11}\cdot\delta + b_{10}) \qquad \text{i = n,m} \end{aligned} \tag{4.4}$$

$$\begin{aligned} C_{i_\delta} \triangleq & (2\cdot b_{22}\cdot\delta + b_{21})\cdot\alpha^2 \\ & (2\cdot b_{12}\cdot\delta + b_{11})\cdot\alpha \\ & (2\cdot b_{02}\cdot\delta + b_{01}) \qquad \text{i = n,m} \end{aligned} \tag{4.5}$$

With these definitions aerodynamic derivatives can be calculated at each design point.

4.3 Inspection of System Dynamics and Deciding on Controller Configuration

Controllers designed with classical control methods have the general form where feedback loops are closed on one another. However it's a choice of design to decide on the controller configuration. For this purpose, uncompensated system dynamics must be inspected. Open-loop response of the system at all design points are given in appendix B. Below, in Figure 4-1, is an example of DP.3, α = 3° case, acceleration response (in g's) to 1° control surface deflection. It is clear from the response that the system is stable, which is also the case for every design point.

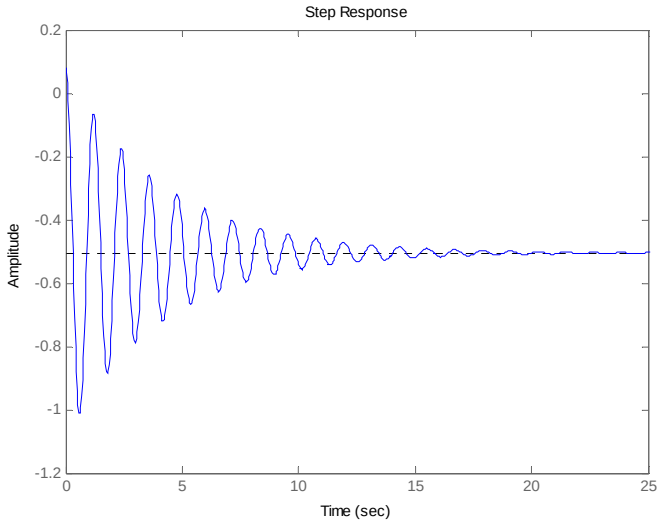


Figure 4-1 OL Response DP = 3 α = 3

As mentioned in chapter 2 the missile has to operate in a specified α range. Since the angle of attack is of crucial importance, first thing to do is to use it as a feedback variable. However, there aren't any sensors on the missile to sense the angle of attack, therefore, there will be no alpha loops in the controller structure.[7]

Decision on the best controller configuration is based on experience and sometimes trial and error in classical control. Such a decision has been made; Figure 4-2 reveals the controller configuration used.

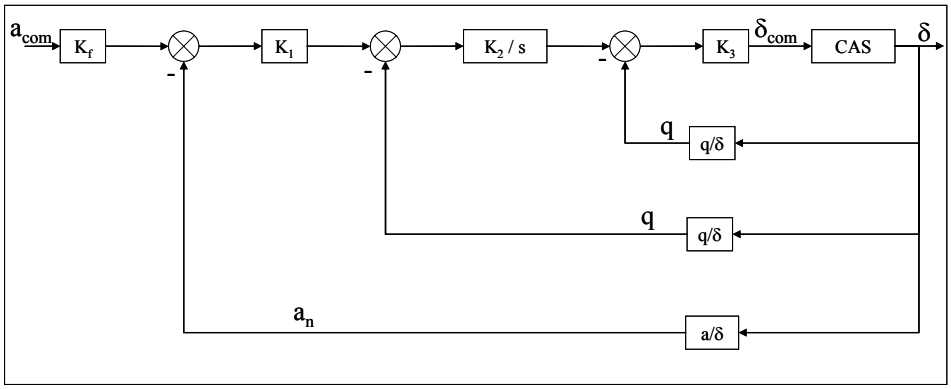


Figure 4-2 Autopilot Configuration

[7] One may also estimate the angle of attack; there are a number of ways to do this but it is not preferred in this thesis to corporate the complex estimation problem into the simple classical controller structure. For some applications of parameter and angle of attack estimation, see [19]

It can be seen that there are acceleration and pitch rate feedbacks as expected and simple proportional gains are used. The difference is that the pitch rate is integrated and fed into the system back. This helps the pitch rate to be attenuated faster.

This configuration is named as Three-Loop Autopilot in literature (see [1], [27]). The innermost loop is named rate damping loop. The loop with the integrator is named synthetic stabilization loop. Finally the outermost loop is named the acceleration loop.

4.4 Determining Autopilot Coefficients

Designing the controller consists of closing the feedback loops one by one, from inside to outside. Procedure for determining the gains, which is explained herein is generated specially for the current problem, however [1] and [26] involve different strategies for determining these four gains.

4.4.1 Closing Rate Damping Loop

Closing rate damping loop consists of choosing the gain K3.

A typical root-locus plot of rate damping loop is given in Figure 4-3.

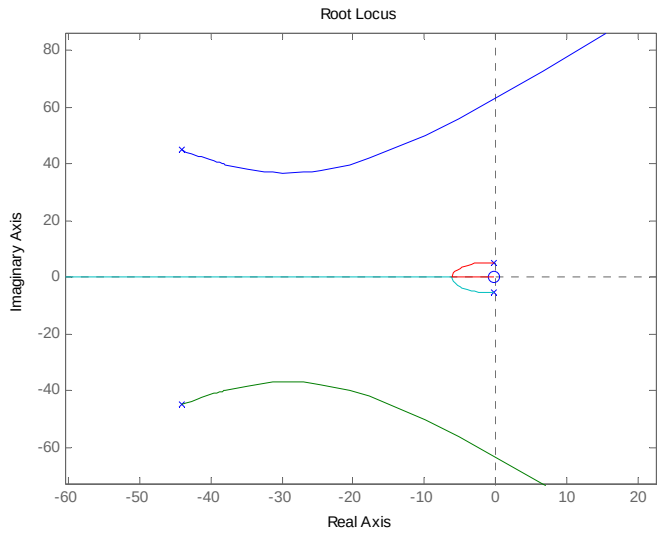


Figure 4-3 Root Locus of Rate Damping Loop

The gain K3 is selected so that the actuator dynamics are alienated from the system. Also it is important to select the system roots as far from the imaginary axis as possible.

4.4.2 Closing Synthetic Stabilization Loop

Closing synthetic stabilization loop consists of choosing the gain K2. After K3 is selected, rate damping loop is closed. With the new system in hand, root-locus can be plotted. This plot is given in Figure 4-4.

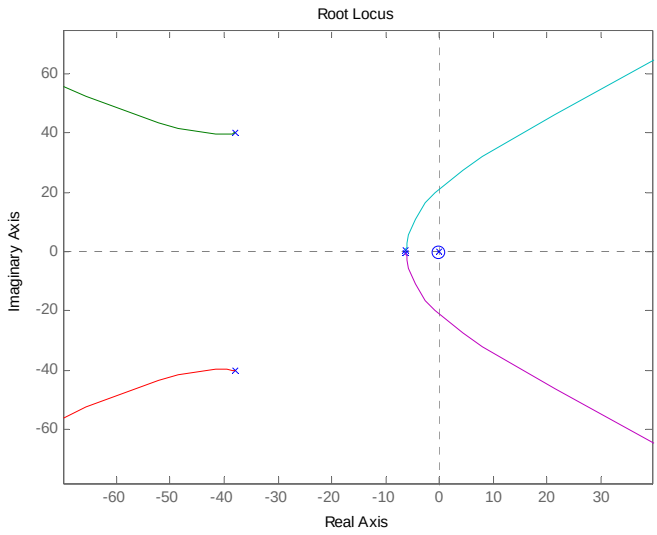


Figure 4-4 Root Locus of Synthetic Stabilization Loop

It can be seen that the actuator dynamics is completely away from the system roots. The root placed at the origin by the integrator tends to make the system slower, so the gain must be selected large to keep the root away. However, using large gains tends to make the system unstable. So a value that takes all these boundaries into account must be selected.

4.4.3 Closing Acceleration Loop

Closing acceleration loop consists of choosing the gain K1. After K2 is selected, synthetic stabilization loop is closed. With the new system in hand, root-locus can be plotted. This plot is given in Figure 4-5.

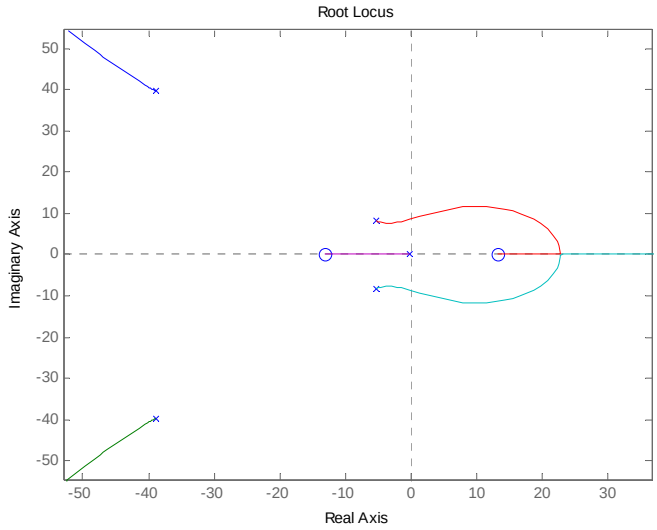


Figure 4-5 Root Locus of Acceleration Loop

It is seen that, with gains selected closer to zero, the three roots will stay in between the poles. Selection of K1 may determine whether the system will look like a first-order or second-order system. Other requirements, GM and PM also apply here.

4.4.4 Response of a System Designed Accordingly

Typical acceleration response of a system designed with the procedure above is as follows:

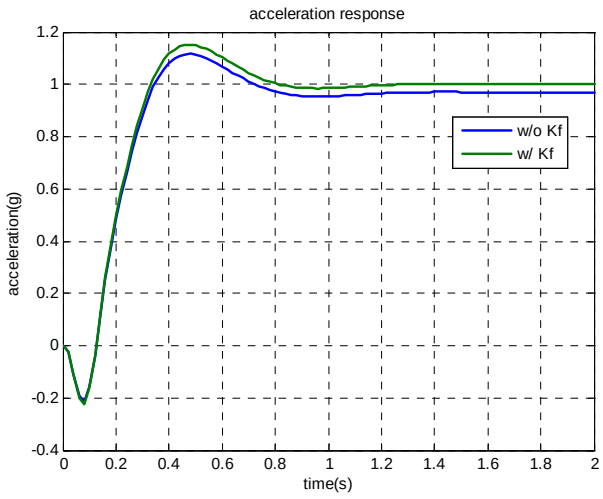


Figure 4-6 Typical Response of the System with Classical Controller

It is seen that steady state value is different than 1. K_f is a pre-filter gain that is used to scale the autopilot command so as to obtain desired steady state response[8]. Usage of K_f is as follows: Let's say normal acceleration required is 1g as computed by the guidance algorithm. Then, once it is roughly known that the autopilot will have a certain amount of steady state error, guidance algorithm can modify its command beforehand. K_f is then chosen as the reciprocal of the steady state value. For example, if desired value of the response is 1g and the system response is estimated to have a steady state value of 0.7g, the command may be multiplied by 1/0.7, in order to have the final response equal to 1. Since the aim in this thesis is the evaluate the performance of autopilot alone, K_f will be taken equal to 1.

Discussion about the α constraint is given in section 4.6.1.

4.5 Controller Gains

With the procedure explained above, following gains for the design points can be obtained.

Table 4-1Classical Controller Gains

DP	$K3$	$K2$	$K1$	K_f
1	-0.2	9	0.21	1
2	-0.5	8	0.29	1
3	-0.85	6	0.3	1
4	-0.65	5	0.25	1

System responses obtained with these gain values are given in section 4.6.1.

[8] Therefore K_f is actually a parameter more likely related to guidance than the autopilot. The reason it is involved in this discussion is to keep the consistency with the literature, [1].

4.6 Testing the Design in Linear and Non-Linear Environments

4.6.1 Testing in Linear Simulation

Once the gains are selected, one must test the design in linear and non-linear environments. This is a crucial step which complements the design. Moreover, in cases like this, it is a part of the design: As mentioned above in section 4.4, α response should be checked so that it does not go beyond the restricted region.

A SIMULINK model has been prepared as the linear test platform. Its structure is given in Figure 4-7. It can be seen that it is exactly the same as in Figure 4-2.

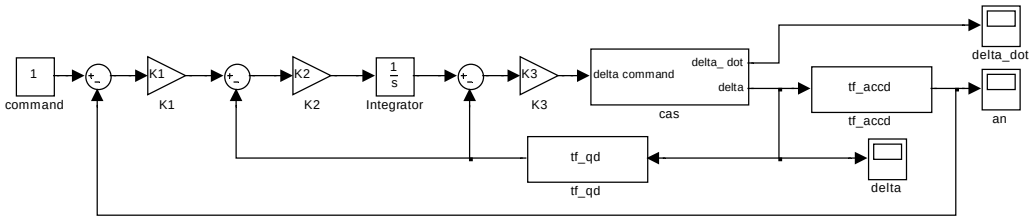


Figure 4-7 Linear Simulink Model

With the gains found in section 4.5, following closed loop acceleration responses to 1g step command are obtained:

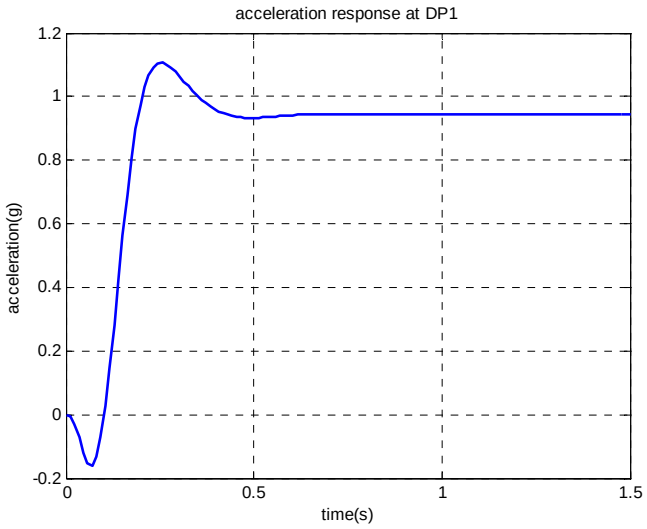


Figure 4-8 Acceleration Response in DP1

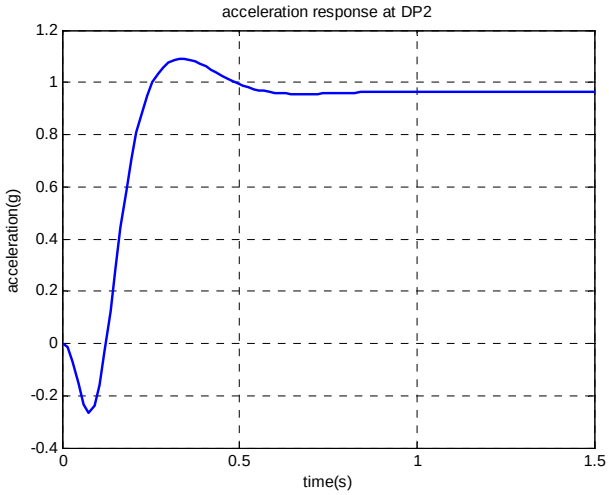


Figure 4-9 Acceleration Response in DP2

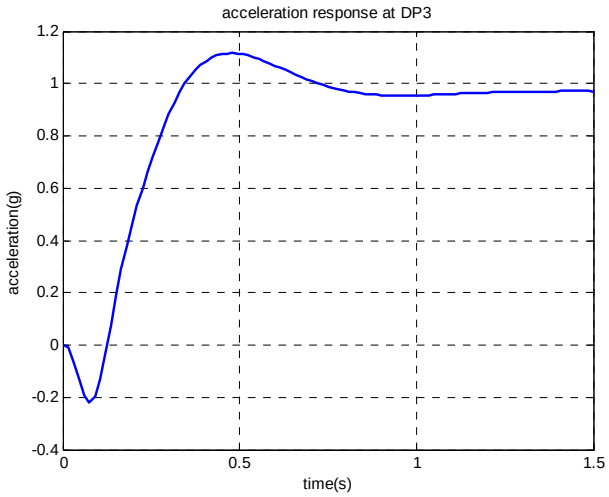


Figure 4-10 Acceleration Response in DP3

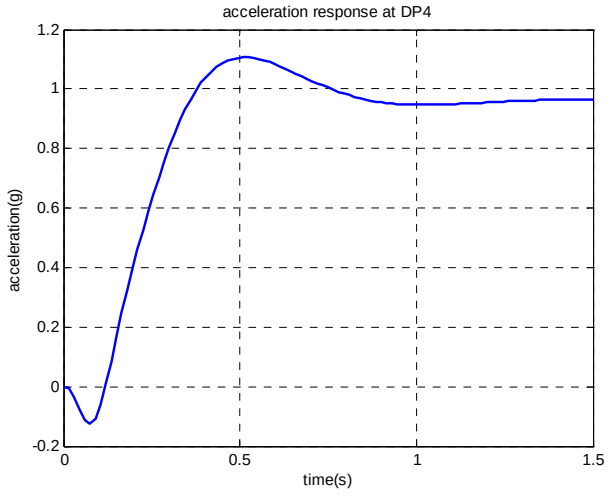


Figure 4-11 Acceleration Response in DP4

Numerical values regarding the performance of the closed system are given in 7.1.1.

Next step in designing the autopilot is to determine the allowable normal acceleration using the angle of attack responses. For one of the responses given above, DP3, α response is as given in Figure 4-12.

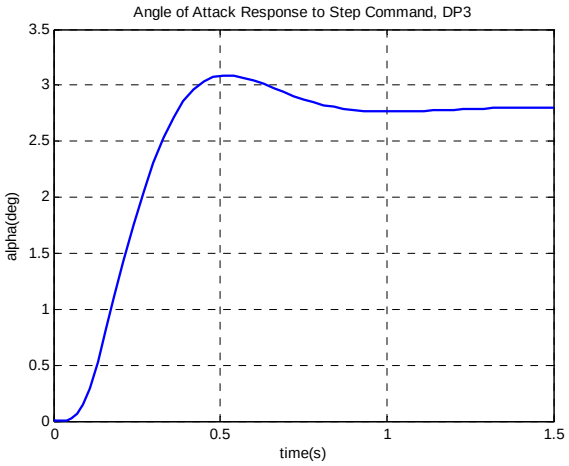


Figure 4-12 CL alpha response

It can be seen that while performing a 1g maneuver, the angle of attack becomes 3°. Considering that this is a nearly linear system, one may guess that allowable maneuver command is 2g. Indeed when command is 2g, response is just like that is expected. Angle of attack remains in the allowable region. Thus, angle of attack limitation dictates a limit on the guidance command that is ordered by the guidance section, and this is the only way the angle of attack limitation affects the design.

4.6.2 Testing in Non-Linear Simulation

Design should be tested in a non-linear simulation in order to

- See the effects of assumptions made and verify them
- Test the robustness of the design under some disturbances.

Such a tool has been introduced in section 3.9. The design has been tested with this simulation. Responses to step inputs (under no disturbances but gravity) for all design points have been given in section 7.1.

CHAPTER 5

ROBUST CONTROLLER DESIGN: THEORY

Robust Control Techniques (RCT), which is being developed since 1980s, provide powerful tools for designing controllers that guarantee the stability and the performance even in the presence of uncertainties. In this chapter and the next, techniques based on RCT will be applied to the current missile problem[9]. In this chapter, brief information about the history and theory of RCT will be given.

5.1 Historical Background

This section will provide brief information about the evolution of RCT. With this section it is also intended to explain the underlying reasons of development of such methods.

CCT, formulated in frequency domain, have been used since the 1930s. These methods are powerful tools which are easy to apply provided that the problem is not of high order. Moreover, both nominal stability and nominal performance specifications can be explicitly pronounced. Most important deficiency of CCT is the disability to handle MIMO systems in a well-defined procedure. Modern control methods followed by optimal control approach[10] have emerged in 60's, allowing comprehensive procedures for MIMO systems, yet becoming cumbersome as the dimension of the problems increase.

---

[9] Hereafter, the resulting controller will be called the "robust controller".

[10] Both are formulated in time domain

Some problematic issues about these methods are the following:

- They give a measure of "distance of instability", but this is not a quantitative one in terms of parameters involved in the system. In other words, allowed perturbations in dynamics are not quantized.
- They address the performance of the closed system only in nominal conditions where the parameters of the system are well-defined. It is always the case that dynamics of systems are different than what are used in the controller design. So, the question of how the performance will be degraded when system deviates from nominal case is not answered.

Therefore the need for a design method which guarantees both stability and some level of performance even in the presence of changes (of a pre-defined class) in plant dynamics has risen.

First formulation of H_∞ optimal control theory was given by ZAMES [2] in 1981. After earlier solution techniques which were in an input-output setting, first state-space solution has emerged in 1984. Procedure introduced by DOYLE, known as the 1984 approach, solved the general H_∞ problem, but suffered a lot from computational difficulties. After some trials to solve the problem, it was understood that H_∞ problem could be solved by Algebraic Riccati Equations. Finally, the 1988 paper by GLOVER and DOYLE [3], and the 1989 paper by DOYLE, GLOVER, KHARGONEKAR and FRANCIS [4] brought a straightforward solution to the problem. Since then state-space theory of H_∞ is being developed, expanding to time-variant and non-linear results.

Detailed information about the evolution of RCT can be found in [4], [5] and [7]. Moreover, [18], [19], [20], [21], [22] and [24] contain some applications of RCT to missile autopilot problem.

5.2 Theory

This section will provide some information on notation and mathematical tools used in RCT.

5.2.1 Problem Statement

In RCT, basic block diagram that is used to formulate the problem is as given in Figure 5-1.

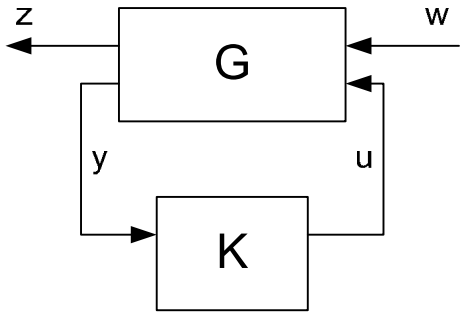


Figure 5-1 Basic Block Diagram in RCT

In Figure 5-1, G is the "generalized" plant. The formulation in this thesis requires that the plant to be a Finite Dimensional Linear Time Invariant (FDLTI) system. K is the controller. w is the signal that contains all the external inputs including disturbances, reference signals and noise. z represents the error signals, u is the controller signal. y represents the measured output signals. This type of representation is called the "linear fractional transformation" of the system (explained in section 5.2.3)

When the controller is implemented, the resulting closed system relating reference signals and disturbances to error signals is denoted by T_{zw}.

The problem is to, with the uncertainties and disturbances included, minimize some norm of the transfer function from the inputs to errors.

5.2.2 Norms of Systems

In RCT, performance and stability of systems are determined by their norms.

In general, a real-valued function $\|\cdot\|$ is said to be a norm, *iff* it satisfies the following relations:

1. $\|x\| \geq 0$ (positivity) (5.1)

2. $\|x\| = 0 \;\; iff \;\; x = 0$ (positive definiteness) (5.2)

3. $\|\alpha x\| = |\alpha| \|x\|$ for any scalar α (homogeneity) (5.3)

4. $\|x + y\| \le \|x\| + \|y\|$ (triangle inequality) (5.4)

$\forall x, y \subset X$, X being a vector space.

In order to define norm of a system (or a transfer function) definition of another quantity, Root Mean Square (RMS) is needed.

RMS of a time varying signal is defined as

$$\|u\| \triangleq \sqrt{\lim_{T\to\infty} \frac{1}{T} \int_{-\infty}^{\infty} \left(u(t)\right)^2 dt} \qquad (5.5)$$

RMS is actually a semi-norm; it can be seen that it does not satisfy the second requirement in the definition of the norm (see (5.2)).

With the system in Figure 5-2 formulated in frequency domain, RMS can also be given as

Z(s) = G(s)w(s)

Figure 5-2 Figure of System G(s) Operating on Signal w(s)

$$\|G\|_{rms} \triangleq \sqrt{\frac{1}{2\pi} \int_{-\infty}^{\infty} |G(j\omega)|^2 S_\omega(\omega) d\omega} \qquad (5.6)$$

where S_ω is the power spectral density of the input.

RMS is also named the “power norm”.

Various norms, like H_2, L_1 or H_∞, can be defined for systems. Use of these norms lead to different control paradigms. The norm to be used in this thesis is the H_∞ norm.

H_∞ norm is defined as the maximum magnitude of G on imaginary axis.

$$\|G\|_\infty \triangleq \sup_\omega |G(j\omega)| = \sup_{\substack{\omega \\ \|w\|_2 \neq 0}} \frac{\|z\|_2}{\|w\|_2} \qquad (5.7)$$

For a MIMO system with $G(j\omega)$ is a transfer function matrix, definition is:

$$\|G\|_\infty \triangleq \sup_\omega \bar{\sigma}(G(j\omega)) \qquad (5.8)$$

where $\bar{\sigma}$ is the largest singular value of $G(j\omega)$. (5.8) is the relation to be used in robustness studies.

5.2.3 Linear Fractional Transformations (LFTs)

In RCT, general framework is called the "linear fractional transformation", introduced by J.C. DOYLE in 1984. Consider the plant P in usual $P = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$ configuration and partition it in a form $P \triangleq \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}$ such that $\begin{bmatrix} z \\ v \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} w \\ u \end{bmatrix}$, forming the system in Figure 5-3.

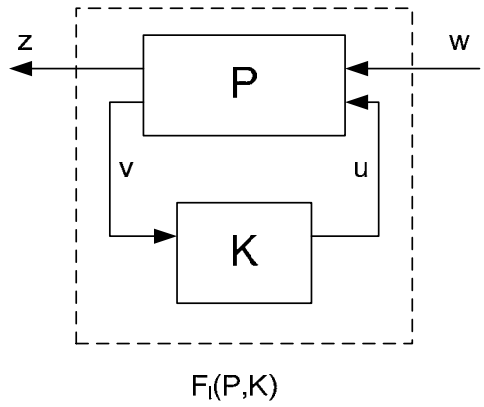


Figure 5-3 Lower Fractional Transformation

In this figure, K represents the controller.

After some manipulation for eliminating u and v, one gets transfer function[11] resulting from wrapping (positive) feedback K around the lower part of P. This is called the lower-fractional transformation $F_l(P,K)$. Expression for $F_l(P,K)$ is given as

$$F_l(P,K) = P_{11} + P_{12}K(I - P_{22}K)^{-1}P_{21} \qquad (5.9)$$

[11] from w to z

Similarly, for a system in the form given in Figure 5-4, one can obtain upper fractional transformation as

$$F_u(P,\Delta) = P_{22} + P_{21}K(I - P_{11})^{-1}P_{12} \qquad (5.10)$$

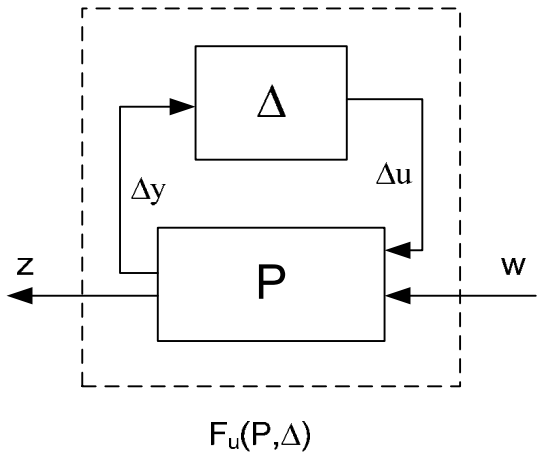


Figure 5-4 Upper Fractional Transformation

5.2.4 Transfer Functions Defined on a Feedback Structure

Consider the single degree of freedom, possibly MIMO, closed loop system structure as in Figure 5-5.

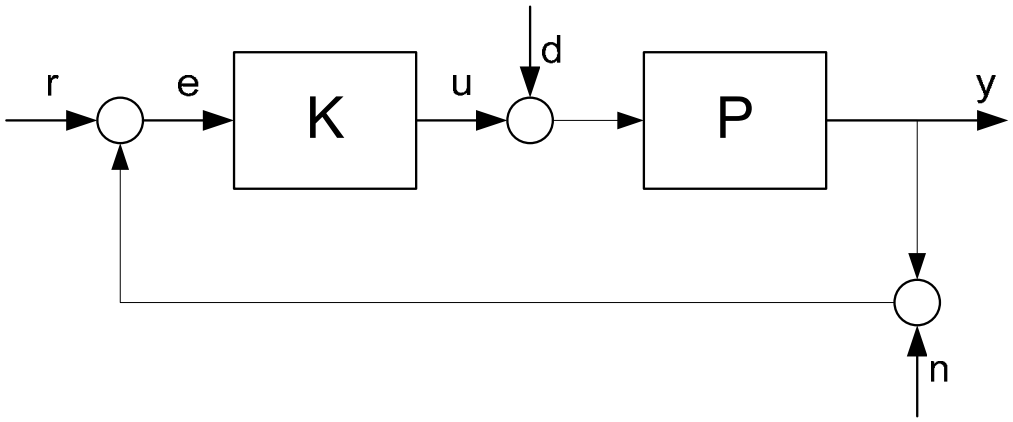


Figure 5-5 Closed Loop System Structure

In Figure 5-5, P represents the plant, e signal represents the error, r represents the reference command, d represents the disturbances and n represents the sensor noise.

Transfer function of such a system is as follows:

$$y = P(K(r - (y + n)) + d) \qquad (5.11)$$

With some manipulation, one can easily find

$$y = \underbrace{(PK + I)^{-1} PK}_{T} r - \underbrace{(I + PK)^{-1} PK}_{T} n + \underbrace{(I + PK)^{-1}}_{S} Pd \qquad (5.12)$$

It is seen that some terms appear several times. In control systems theory, these are named as S, Sensitivity Transfer Function and T, Complementary Sensitivity Transfer Function. The name "complementary" comes from the fact that $T + S = 1$. Once inspected, it can be seen that S is the transfer function from the output disturbances to the outputs and T is the transfer function from commands to outputs. These transfer functions indicate how the output is affected from elements of the system: For a good tracking, it is desired to have $y \approx r$. Then T has to be large, but this is a contradiction, because for sensor noise attenuation T must be small. The $T + S = 1$ equality explains very well why it is impossible to obtain good tracking and noise/disturbance attenuation at the same time. The solution is to make both small at different frequencies[12]:

- Make T large at low frequencies, where references are important and the noise has little effect. Then S is small, so disturbances are ignored.
- Make T small at high frequencies in order to attenuate noise.

For a very detailed discussion on T and S see [6].

Another very frequently used definition is $L = PK$, which represents the system gain. For example, if the definition for T is inspected, one may see that L has to be very large for good tracking.

5.2.5 Model Uncertainty and Stability/Performance

5.2.5.1 Definitions for Stability Functions

This section will explain the uncertainty definitions used in RCT. In a system model, "uncertainty" may represent two things:

---

[12] Note that both T and S are complex functions and it is not obligatory to have $|T| + |S| = 1$.

1. One of the parameters present in the system model may deviate from its nominal value (in the real system). This may occur because of the errors present in obtaining that parameter (testing errors, etc)
2. Some of the phenomena that are present in the system may not be involved in the model. This may be due to difficulties of modeling, or may even be ignored for simplicity. Also, higher order dynamics, that are very difficult to describe, are usually absent in the model.

In H_∞ Framework, there are two ways to describe the uncertainty possibly present in the system:

5.2.5.1.1 Multiplicative Uncertainty

Multiplicative uncertainty is defined as

$$\widetilde{P} = (1 + \Delta W_m)P \qquad (5.13)$$

where P is the nominal plant, Δ is a variable stable transfer function satisfying $\|\Delta\|_\infty \le 1$, W is a weighting function, denoting the degree of uncertainty across frequency. Typically $|W(\omega)|$ is an increasing function with frequency, in order to emphasize the increasing uncertainty in high frequency. Block diagram for representing the uncertainty in multiplicative form is given in Figure 5-6.

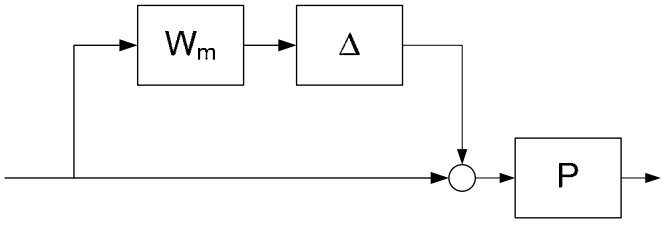


Figure 5-6 Multiplicative Uncertainty

5.2.5.1.2 Additive Uncertainty

Additive uncertainty is defined as

$$\widetilde{P} = P + \Delta W_a \qquad (5.14)$$

Block diagram for representing uncertainty in multiplicative form is given in Figure 5-7.

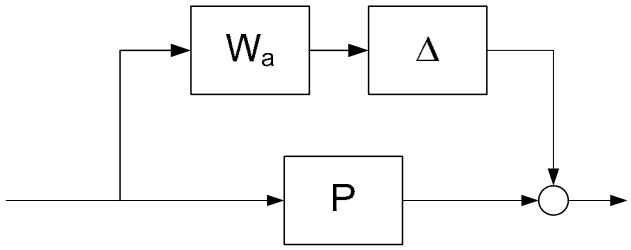


Figure 5-7 Additive Uncertainty

Note that, $W_m \neq W_a$

Both definitions are frequently used. The choice of definition to be used depends on the nature of the system and information about uncertainty. For example, if information about the absolute magnitude of the uncertainty is known, additive uncertainty definition is used. On the other side, if one just knows the plant may deviate by some fraction of the nominal, multiplicative uncertainty is preferable. Multiplicative uncertainty definition is used for all uncertainties in this study.

5.2.5.2 Stability/Performance Tests Defined for Sensitivity Functions (SISO case)

This section will define some criteria for stability and performance of systems based on the multiplicative uncertainty definition. The reference system is given in Figure 5-8. This system is essentially the combination of systems in Figure 5-5 and Figure 5-6, with the addition of a pre-filter W_1, which is used to shape the frequency content of the unitary reference signal r_{pf}.

The proofs for the following theorems given below can be found in [6].

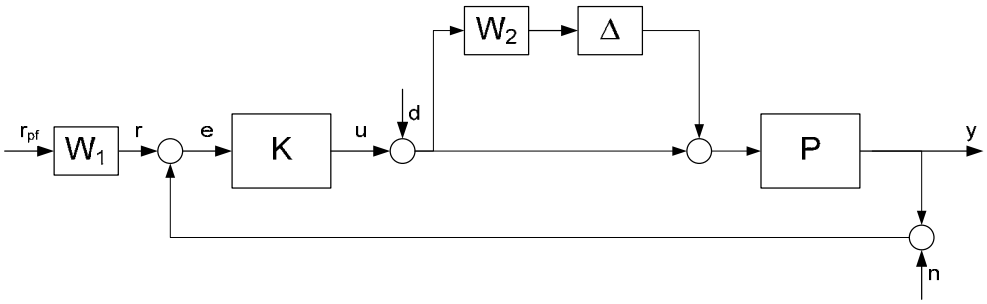


Figure 5-8 Block Diagram for Stability Tests

5.2.5.2.1 Nominal Stability (NS)

NS refers to the internal stability of the closed loop system formed by P and K, ignoring all other signals and transfer functions. This is the classical stability definition: The closed loop transfer function T has to have all its poles in LHP. This criterion is a pre-requisite for the following criteria.

5.2.5.2.2 Nominal Performance (NP)

NP means that the closed loop transfer function of the feedback connection of systems P and K, ignoring all other transfer functions, satisfies the desired performance specifications like, rise time, steady state response, etc.

In terms of norms, the "performance argument" may be interpreted as follows: For a good performance, some norm of the error signal e, or the transfer function from disturbances to errors, has to be small. The norm could be any of the ones mentioned in section 5.2.2. For example if the ∞ norm is used, the nominal performance argument is stated as:

$$\|W_1 S\|_\infty \leq 1 \qquad (5.15)$$

Recall that S is the transfer function from disturbances to errors. W_1 is a filtering/weighting function to shape the whole expression such that the error signal is required to be small especially at low frequencies.

5.2.5.2.3 Robust Stability (RS)

RS is the stability of the closed loop system in the presence of uncertainties defined by a bounded set.

In terms of ∞ norm, the system in Figure 5-8 is said to be robustly stable *iff*

$$\|W_2 T\|_\infty \leq 1 \quad \forall \|\Delta\|_\infty \leq 1 \qquad (5.16)$$

Again, W_2 is a filter/weighting function to shape the frequency content of the uncertainty.

A relevant theorem is the "small gain theorem": if a feedback consisting of stable systems and the loop-gain is less then unity, then the feedback loop is internally stable. Consider the system in Figure 5-8 and absorb everything but the Δ block into M. One then gets the block diagram given in Figure 5-9.

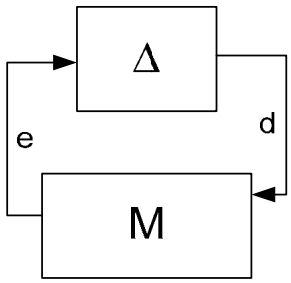


Figure 5-9 Figure for Small Gain Theorem

Then $TF_{d\to e} = \dfrac{W_2 PK}{1+PK} = W_2 T = M$ (5.17)

The $W_2 T$ expression also gives the so called "Loop Gain".

The theorem says the loop formed by M and Δ is stable *iff* $\|M\Delta\|_\infty \le 1$. If we know that $\|\Delta\|_\infty \le \beta$ for some $\beta > 0$, we can conclude $\|M\|_\infty = \|W_2 T\|_\infty \le 1/\beta$ is required for robust stability.

5.2.5.2.4 Robust Performance (RP)

RP refers to the ability of the closed loop system to perform as good as desired, in the presence of uncertainties. It is actually a combination of RS and RP criteria.

$\left\| |W_1 S| + |W_2 T| \right\|_\infty < 1$ (5.18)

5.2.5.3 Stability/Performance Tests Defined for Sensitivity Functions (MIMO case)

For MIMO systems, the idea in 5.2.5.2 is exactly the same. However, because of increased number of inputs and outputs, a new way for handling the system, namely the LFTs, is needed. Block diagram that will be used is given in Figure 5-10. In this figure, the weights in Figure 5-8 are also absorbed in . Similarly all the possible uncertainties are present in Δ. All the other signals are vectors. Since this section deals with analysis issues, K may be accepted as a part of the closed system, so P and K may be absorbed into M (so that $M = F_l(P,K)$).

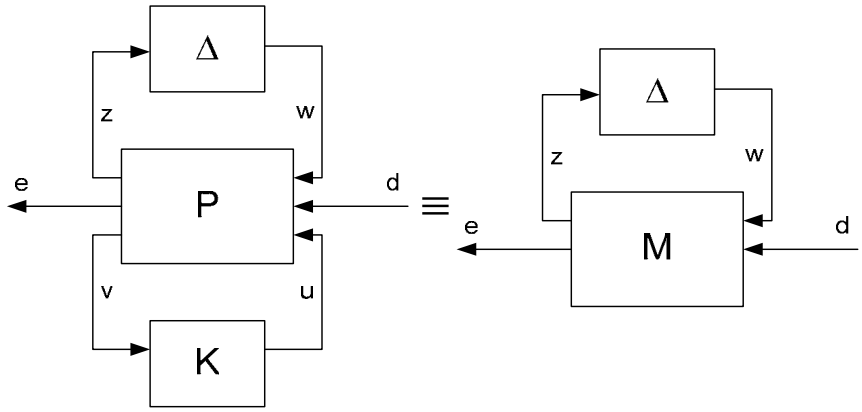


Figure 5-10 Block Diagram using LFTs

5.2.5.3.1 Nominal Stability (NS)

The criterion is that all transfer functions from all inputs to all outputs have to satisfy the criterion in SISO case. This is also a necessary condition for the existence for other remaining three tests.

5.2.5.3.2 Nominal Performance (NP)

Nominal performance deals with the transfer function from the disturbances to errors in the absence of uncertainties. Therefore, nominal performance is achieved if $\|M_{22}\|_\infty < 1$

5.2.5.3.3 Robust Stability (RS)

Small gain theorem exactly applies here: let $\|\Delta\|_\infty \le \beta$, then $\|M_{11}\|_\infty \le 1/\beta$ is required for robust stability.

5.2.5.3.4 Robust Performance (RP)

As explained for the SISO case, NP and RS conditions must be simultaneously satisfied. Once inspected it is easy to check that the transfer function from the disturbances to errors is $F_u(F_l(P,K),\Delta)$. Therefore, if a Δ with $\|\Delta\|_\infty \le 1$ is given, the problem reduces to the requirement of $\|M\|_\infty \le 1$. However, this is a conservative criterion: See section 5.2.7, which continues discussing this issue based on structured singular value, μ.

5.2.6 Solution to H_∞ problem

As a result of discussions up to now, the problem maybe re-stated as follows:

"Find all admissible controllers K, such that $\|M\|_\infty$ is minimized"

This solution K to this problem is called the "Optimal H_∞ controller". But it is not always easy to find such a controller: it may require iteration and also some MIMO systems have multiple controllers that minimizes $\|M\|_\infty$, i.e., no unique solution exists. Instead a different statement can be used:

"Given $\gamma > 0$, find all admissible controllers K, if exists, such that $\|M\|_\infty < 1/\gamma$ "

The solution to this problem, which will be discussed hereafter, finds a controller K that satisfies the above criterion, but the resulting norm $\|M\|_\infty$ is greater than the possible minimum, γ_{opt}. Such a controller is named the "Sub-Optimal H_∞ controller".

The solution to this problem is given by DOYLE and GLOVER in 1988. The complete procedure can be found in [4].

Consider the block diagram given in Figure 5-1. The equations for the system and realization of G are as follows:

$$\begin{bmatrix} z \\ y \end{bmatrix} = G(s)\begin{bmatrix} \omega \\ u \end{bmatrix} = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix}\begin{bmatrix} \omega \\ u \end{bmatrix} \qquad (5.19)$$

$$u = K(s)y \qquad (5.20)$$

$$G = \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right] \text{ so } z = F_l(G,K)\omega = T_{z\omega}\omega \qquad (5.21)$$

Now make the following assumptions:

1. (A,B_1) is stabilizable and (C_1,A) is detectable. (5.22)
2. (A,B_2) is stabilizable and (C_2,A) is detectable. (5.23)
3. $D_{11} = 0$ and $D_{22} = 0$ (5.24)
4. $D_{12}^{\,T}\begin{bmatrix} C_1 & D_{12} \end{bmatrix} = \begin{bmatrix} 0 & I \end{bmatrix}$ (5.25)

5. $\begin{bmatrix} B_1 \\ D_{21} \end{bmatrix} D_{21}{}^T = \begin{bmatrix} 0 \\ I \end{bmatrix}$ (5.26)

The H_∞ solution involves the solution of two Hamiltonian matrices:

$$H_\infty \triangleq \begin{bmatrix} A & \gamma^{-2} B_1 B_1{}^T - B_2 B_2{}^T \\ -C_1 C_1{}^T & -A^T \end{bmatrix}$$

$$J_\infty \triangleq \begin{bmatrix} A^T & \gamma^{-2} C_1 C_1{}^T - C_2 C_2{}^T \\ -B_1 B_1{}^T & -A^T \end{bmatrix}$$

Then *iff* the following three conditions hold

1. $H_\infty \in dom(Ric)$ and $X_\infty \triangleq Ric(H_\infty) \geq 0$
2. $J_\infty \in dom(Ric)$ and $Y_\infty \triangleq Ric(J_\infty) \geq 0$
3. $\rho(X_\infty Y_\infty) < \gamma^2$

There exists an admissible controller $K(s) = K_{sub}(s)$ such that $\|T_{z\omega}\|_\infty < \gamma$, and is in the form

$$K_{sub}(s) \triangleq \left[\begin{array}{c|c} \hat{A}_\infty & -Z_\infty L_\infty \\ \hline F_\infty & 0 \end{array}\right] \qquad (5.27)$$

where

$$\hat{A}_\infty \triangleq A + \gamma^{-2} B_1 B_1{}^T X_\infty + B_2 F_\infty + Z_\infty L_\infty C_2$$
$$F_\infty \triangleq -B_2{}^T X_\infty$$
$$L_\infty \triangleq -Y_\infty C_2{}^T$$
$$Z_\infty \triangleq \left(I - \gamma^{-2} Y_\infty X_\infty\right)^{-1}$$

If one wants to parameterize all controllers satisfying $\|T_{z\omega}\|_\infty < \gamma$, the controller is in the form given in Figure 5-11.

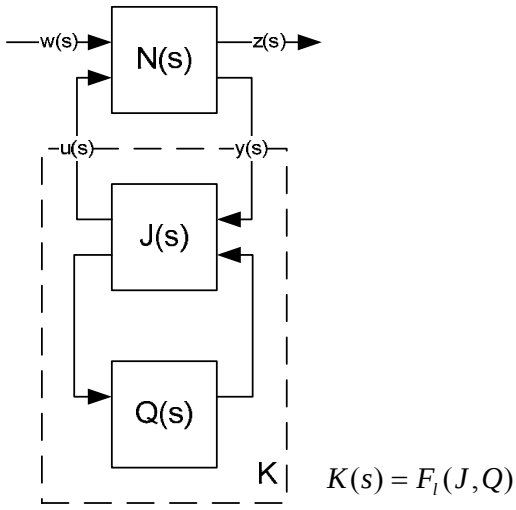


Figure 5-11 Block Diagram for the Parameterized Controller

In Figure 5-11, $Q(s)$[13] is a free parameter. Then

$$J(s) \triangleq \left[\begin{array}{c|cc} \hat{A}_\infty & -Z_\infty L_\infty & Z_\infty B_2 \\ \hline F_\infty & 0 & I \\ -C_2 & I & 0 \end{array}\right] \qquad (5.28)$$

Note that for $Q = 0$ (at the center of the set $\|Q(s)\|_\infty < \gamma$) the controller is in the form given by (5.27), so this controller is called the "central controller".

$$\left[\begin{array}{c} \dot{\hat{x}} \\ u \\ y_Q \end{array}\right] = \left[\begin{array}{c|cc} \hat{A}_\infty & -Z_\infty L_\infty & Z_\infty B_2 \\ \hline F_\infty & 0 & I \\ -C_2 & I & 0 \end{array}\right] \left[\begin{array}{c} \hat{x} \\ y \\ u_Q \end{array}\right] \qquad (5.29)$$

Block Diagram structure for the sub-optimal H_∞ is given in Figure 5-12. Definitions used in this expression are as follows:

$$\begin{aligned} B_Q &\triangleq Z_\infty B_2 \\ B_X &\triangleq \gamma^{-2} B_1 B_1^T X_\infty \end{aligned} \qquad (5.30)$$

[13] $Q(s)$ is such that $\|Q(s)\|_\infty < \gamma$

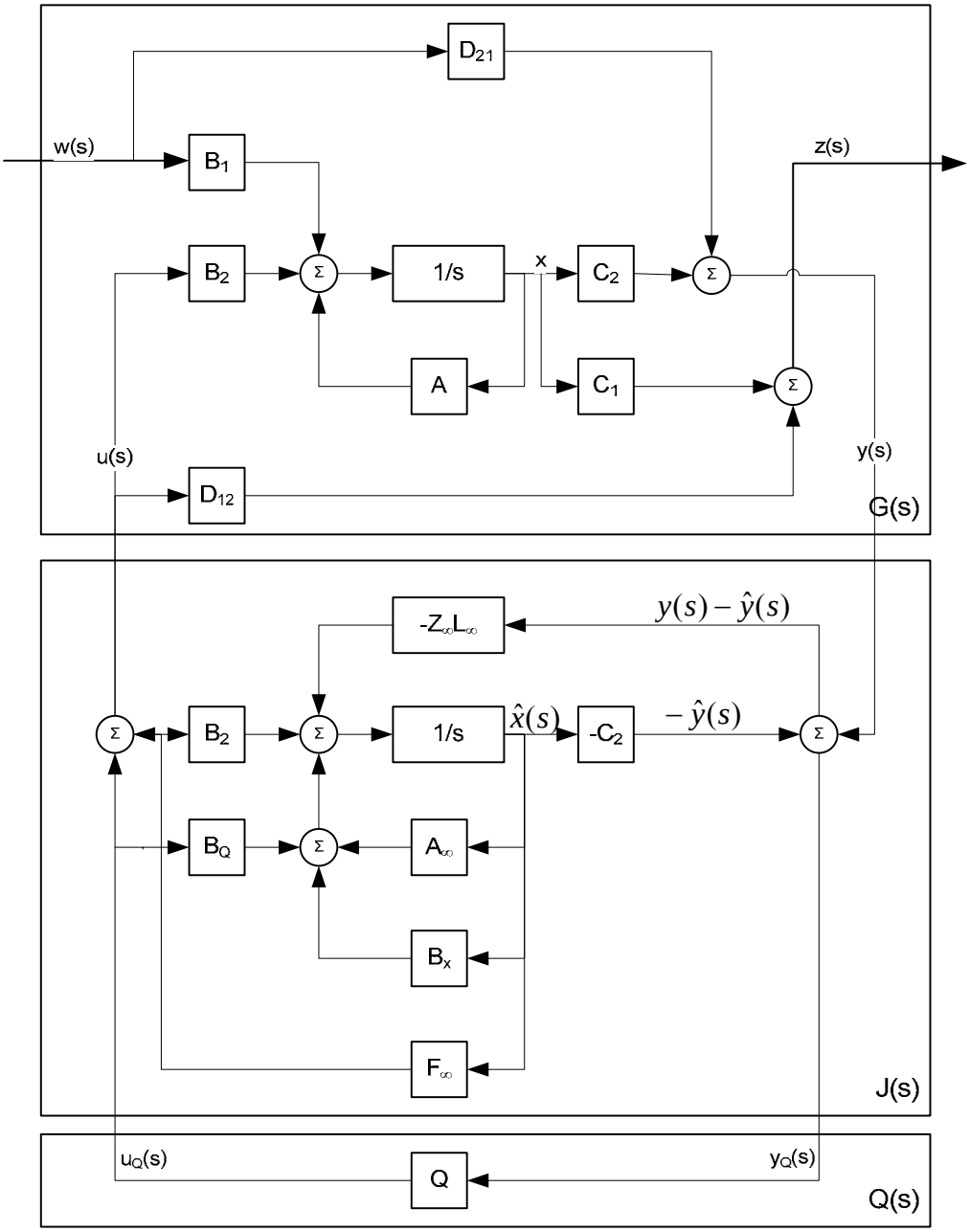


Figure 5-12 H_∞ Sub-Optimal Controller Structure

Two different approaches to solution of H_∞ problem can be found in [23] and [25].

5.2.7 Structured Singular Value (μ)

5.2.7.1 Definition

Stability and performance measures defined in 5.2.5.3 check the ∞ norm of the system. This approach does not take the structure uncertainties into account. However, systems are usually made up of elements which are themselves uncertain, so even expressed in a Δ block as given in Figure 5-10, we see a structure in the Δ block; it has a block diagonal form[14]. To account for the structure in Δ block in Figure 5-10, structured singular value has been introduced by DOYLE, [28]. Definition is as follows:

Let Δ in Figure 5-10 is a member of the set $\mathbf{\Delta} \in \mathbb{C}^{n\times n}$, which has the following diagonal structure

$$\mathbf{\Delta} = \left\{ \begin{array}{r} diag\left[\delta_1 I_{r1}, \delta_2 I_{r2}, ..., \delta_S I_{rS}, \Delta_1, \Delta_2, ..., \Delta_F\right]: \\ \delta_i \in \mathbb{C}, \Delta_j \in \mathbb{C}^{m_j \times m_j} \end{array} \right\} \quad (5.31)$$

where S and F represent the number of scalar and full block, respectively. Then, definition of μ is given as the measure of smallest structured Δ that causes instability of the matrix feedback loop:

$$\mu_{\Delta}(M) \triangleq \frac{1}{\min\left\{\bar{\sigma}(\Delta) : \Delta \in \mathbf{\Delta}, \det\left(I - M\Delta\right) = 0\right\}} \quad (5.32)$$

Structured singular value of system can not be directly computed. Instead, upper and lower bounds for μ can be given:

$$\rho(M) \le \mu_{\Delta}(M) \le \bar{\sigma}(M) \quad (5.33)$$

However, the gap between the spectral radius ρ and maximum singular value $\bar{\sigma}$ can be arbitrarily large, so a scaling that does not affect $\mu_{\Delta}(M)$ but do affect ρ and $\bar{\sigma}$ is used. First, define two sub-sets of $\mathbb{C}^{n\times n}$

$$\bar{Q} = \left\{Q \in \mathbf{\Delta} : QQ^* = I_n\right\} \quad (5.34)$$

---

[14] Therefore, ∞ norm only gives necessary conditions for robustness. In other words, $\| \|_{\infty} < 1$ yields robustness, but the converse is not true.

$$\bar{D} = \left\{ \begin{array}{c} diag\left[D_1, D_2, ..., D_S, d_1 I_{m1}, d_2 I_{m2}, ..., d_F I_{mF}\right]: \\ D_i \in \mathbb{C}^{r_i \times r_i}, D_i = D_i^* > 0, d_j \in \mathbb{R}, d_j > 0 \end{array} \right\} \qquad (5.35)$$

Then, with the following theorem, bounds given for $\mu_\Delta(M)$ in (5.33) can be tightened to

For all $D \in \bar{D}$ and $Q \in \bar{Q}$, $\max_{Q \in \bar{Q}} \rho(QM) \le \mu_\Delta(M) \le \inf_{D \in \bar{D}} \bar{\sigma}(DMD^{-1})$

(5.36)

For a detailed discussion and proof of theorems given here, see [28], [7], and [6].

Now, new measures for RS and RP depending on μ can be given.

5.2.7.2 Robust Stability (RS)

Robust stability measure is given as follows:

$\mu_\Delta(M_{11}(j\omega)) < 1 \quad \forall \omega$ (5.37)

5.2.7.3 Robust Performance (RP)

First, express the RP problem as a RS problem by adding a fictitious performance block. This is shown in Figure 5-13.

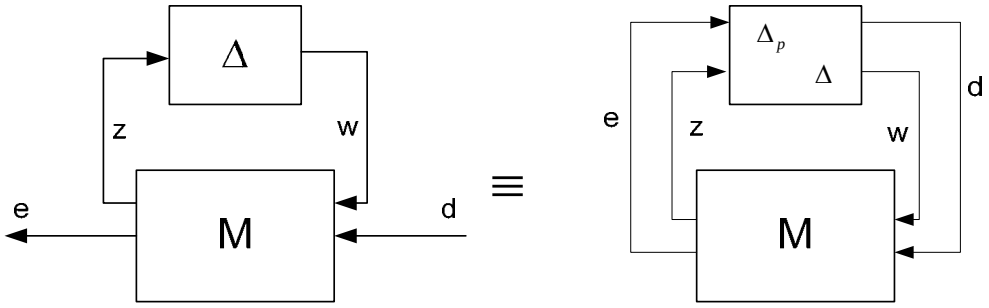


Figure 5-13 Expressing the RP Problem as a RS Problem

Then, by similar reasoning, RP criteria is given as

$\mu_{\underset{\sim}{\Delta}}(M(j\omega)) < 1 \quad \forall \omega$ (5.38)

Where $\underset{\sim}{\Delta}$ is the Δ block given in Figure 5-13, i.e.,

$$\underset{\sim}{\Delta} \triangleq \begin{bmatrix} \Delta_p & 0 \\ 0 & \Delta \end{bmatrix} \qquad (5.39)$$

CHAPTER 6

ROBUST CONTROLLER DESIGN: APPLICATION

In this chapter, Robust Control Theory presented in chapter 5 will be used to design the autopilot for the current problem.

First, issues of generating system structures and expressing the specifications in the H_∞ framework will be discussed. Finally, the design will be completed and a systematic procedure for automatic design of autopilot will be presented.

6.1 Obtaining State-Space Realization

As done for the classical controller, first, transfer functions for the missile have to be obtained.

Starting point is again (3.15) and (3.16). After taking the derivative of α in (3.16) with short-period approximation and some manipulation the equations become

$$
\begin{aligned}
f_1 : \dot{\alpha} &= -\frac{\cos^2 \alpha}{m \cdot u} \cdot \left[Q \cdot S_{ref} \cdot C_n(\alpha, \delta) + q \cdot \left(C_{nq}(M) \cdot \frac{l_{ref}}{2 \cdot V} \cdot Q \cdot S_{ref} - m \cdot u \right) \right] \\
f_2 : \dot{q} &= \frac{Q \cdot S_{ref} \cdot l_{ref}}{I_y} \cdot \left[C_m(\alpha, \delta) + C_{mq}(M) \cdot \frac{l_{ref}}{2 \cdot V} q \right] \\
f_3 : a_n &= \frac{Q \cdot S_{ref}}{m} \cdot \left[C_n(\alpha, \delta) + C_{nq}(M) \cdot \frac{l_{ref}}{2 \cdot V} q \right]
\end{aligned}
\tag{6.1}
$$

Where expression for C_n and C_m are as given in (3.1)

Note that, different from chapter 4, small-angle approximation on α has not been made. Since these equations are non-linear, a linearization process is required in order to write the state-space realization. Linearization is made about some trim points, which are the δ_{trim} and q_{trim} values obtained by solving $f_i = 0$ (i=1,2) equations, i.e., equating the LHS of the state equations to zero and solving for δ_{trim} and q_{trim}, at each design point. Then the state-space realization about the trim points is given by

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} \partial f_1 / \partial \alpha \big|_{\substack{\delta_{trim} \\ q_{trim}}} & \partial f_1 / \partial q \big|_{\substack{\delta_{trim} \\ q_{trim}}} \\ \partial f_2 / \partial \alpha \big|_{\substack{\delta_{trim} \\ q_{trim}}} & \partial f_2 / \partial q \big|_{\substack{\delta_{trim} \\ q_{trim}}} \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} \partial f_1 / \partial \delta \big|_{\substack{\delta_{trim} \\ q_{trim}}} \\ \partial f_2 / \partial \delta \big|_{\substack{\delta_{trim} \\ q_{trim}}} \end{bmatrix} \delta \triangleq \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} B_{11} \\ B_{21} \end{bmatrix} \delta$$

$$\begin{bmatrix} a \\ \alpha \\ q \end{bmatrix} = \begin{bmatrix} \partial f_3 / \partial \alpha \big|_{\substack{\delta_{trim} \\ q_{trim}}} & \partial f_3 / \partial q \big|_{\substack{\delta_{trim} \\ q_{trim}}} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} \partial f_3 / \partial \delta \big|_{\substack{\delta_{trim} \\ q_{trim}}} \\ 0 \\ 0 \end{bmatrix} \delta \triangleq \begin{bmatrix} C_{11} & C_{12} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} D_{11} \\ 0 \\ 0 \end{bmatrix} \delta \tag{6.2}$$

Note that α is taken out as output because it will be used in the controller design. Since it is not measured by any direct means, it will be used to as a parameter to penalize, but not as a parameter to be fed-back into the controller. A detailed explanation about the linearization process is given in appendix D.

6.2 Formulating the Problem

As explained in section 5.2, synthesizing a H_∞ controller is a though work containing solutions of ARE's. Therefore, commercial tools are frequently used for such a study. One of these tools is MATLAB, which has a toolbox dedicated to robust controller synthesis. This tool is used in this study. Synthesizing a controller in MATLAB requires the user to express the problem as in Figure 5-1. This effort comprises transformation of the system into LFT form, deciding on uncertainty definitions to be used and selecting appropriate weights to reflect the performance parameters/uncertainties. Then the code

automatically generates the controller, by performing the calculations explained in section 5.2.6 . Below, the above mentioned efforts are explained.

6.2.1 Constructing the Structure for Use with LFT Framework

This task involves the preparation of the system with weights, system parameters and uncertainties are separated and ready to be input into MATLAB. System view is in a familiar form and is given in Figure 6-1.

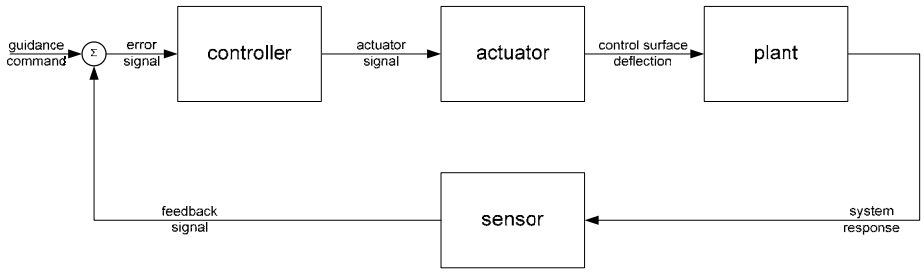


Figure 6-1 System View for Controller Structure

With a closer look, where performance and uncertainty weights are revealed:

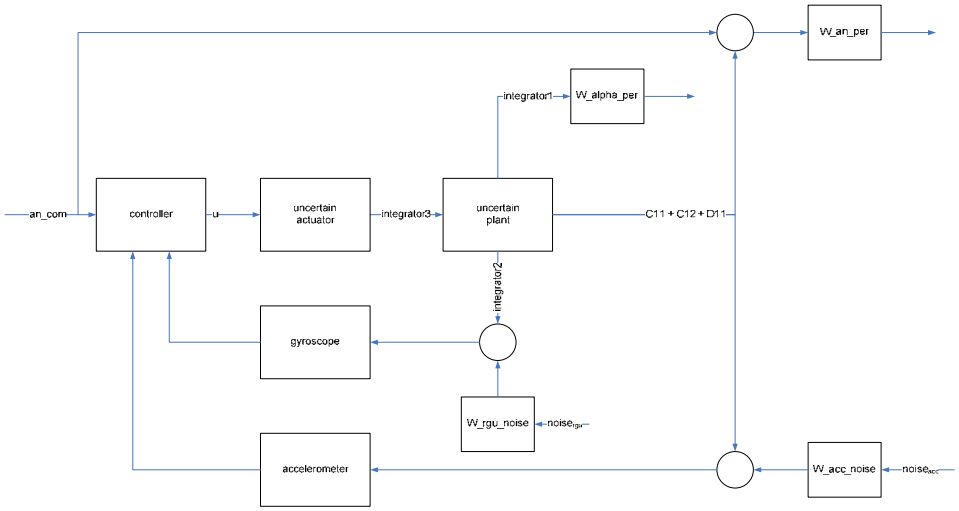


Figure 6-2 Top View System Structure with Weights Revealed

In Figure 6-2, there still are some super-blocks which will be detailed below, but the idea is completely covered: Every necessary parameter is accepted to be uncertain. Unitary disturbance and noise inputs are filtered in order to impose

the possible real-world frequency content. The parameter to be controlled is the acceleration, so the error is penalized. There are though constraints for angle of attack, so it will also be penalized. The names of the signals in Figure 6-2 is the names used in the code, which is given in appendix E.

Block diagrams for the super-blocks are given in following the figures:

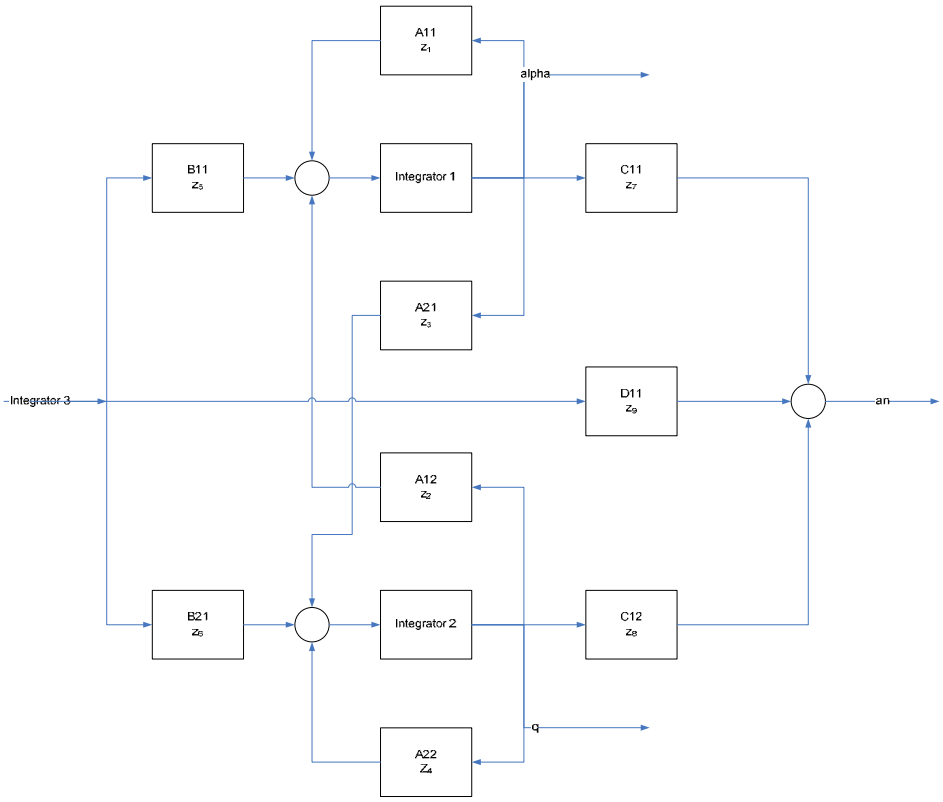


Figure 6-3 System Structure for the Plant

In this figure, all the blocks for the coefficients are again super-blocks and are in the form:

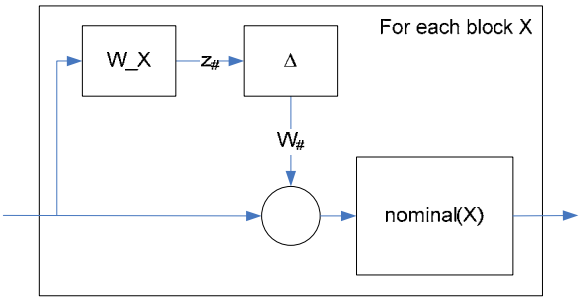


Figure 6-4 Structure for Aerodynamic Coefficients

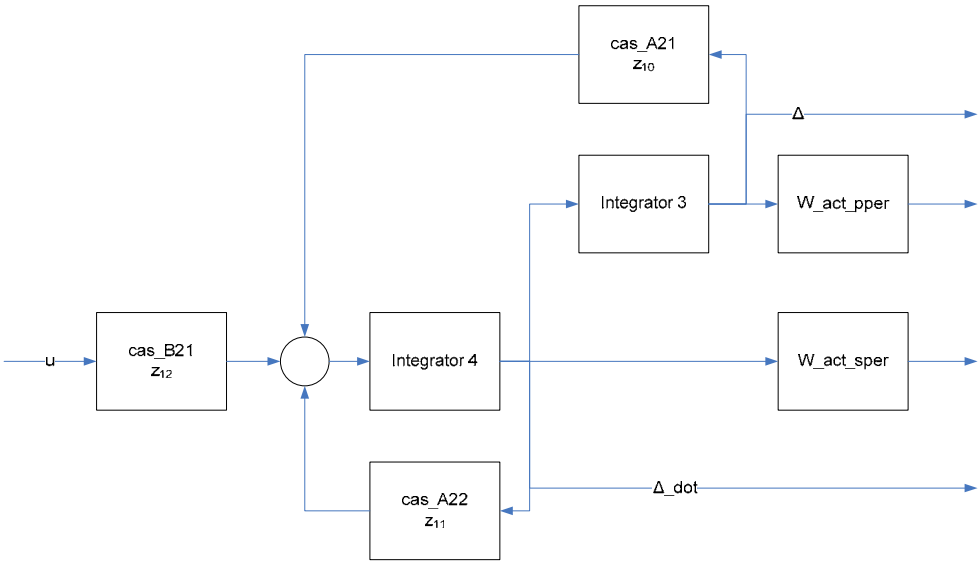


Figure 6-5 Structure for CAS

Similarly, the parameters for the CAS are super-blocks with structures given in Figure 6-4.

For most applications, the sensors are assumed to have dynamics that are fast enough compared to the missile, so are ignored. In this study, RGU and ACC transfer functions are not included in the design process but are taken into account in robustness analyses.

The system now is separated and will be ready to be inputted into MATLAB after the performance/uncertainty weights are determined.

6.2.2 From Performance Specs to Weights

In this section the work related to the determination the performance and uncertainty weights shown in Figure 6-2 to Figure 6-5 is presented.

6.2.2.1 Uncertainty Weighting Functions W1 to W9

This section will explain the structure of functions denoted as W_X in Figure 6-4, where X=1…9. These weights are reflecting the uncertainty on coefficients in state-space representation of plant, arising from mass parameters and the accuracy of MISSILE DATCOM. Each weight represents the maximum possible deviation from the nominal. Table 6-1 summarizes weights used to represent the uncertainty on plant parameters.

Table 6-1 Uncertainties for Plant Parameters

Parameter	Parameter on which uncertainty is represented
W_1	A_{11}
W_2	A_{12}
W_3	A_{21}
W_4	A_{22}
W_5	B_{11}
W_6	B_{12}
W_7	C_{11}
W_8	D_{11}
W_9	C_{12}

If section D.6.3 is inspected, it can be seen that all these parameters are functions of many parameters, for example: A_{11} is a function of mass parameters and C_n, C_{nq}, $C_{n\alpha}$ which are also functions of α and δ of second order (with parameters b_{ij}, etc…). All of these integrants are themselves uncertain. So a method for combining the uncertainties of all these parameters is needed.

First it is assumed that possible errors in parameters calculated by MISSILE DATCOM are

- 7% on C_n
- 13% on C_m
- 25% on C_{nq}
- 25% on C_{mq}

These possible errors are treated as uncertainties.

As explained in section D.6.3, in this study, aerodynamic coefficients are expressed as polynomial functions of α and δ. The structure of polynomials are given in Table 6-2.

Table 6-2 Representation of Aerodynamic Coefficients

C_i	δ^2	δ^1	δ^0
α^2	b_{22}	B_{21}	b_{20}
α^1	b_{12}	B_{11}	b_{10}
α^0	b_{02}	B_{01}	b_{00}

Examining the structure given in this table, it can be seen that the uncertainty values given for the "gross" aerodynamic coefficients must be divided among the coefficients b_{ij} of the polynomial. An easy way to do this is to evenly divide the gross uncertainty among the coefficients. However, when the relative magnitudes of polynomial coefficients b_{ij} are inspected, it is seen that some of them are dominant: b_{21}, b_{12}, b_{10} and b_{01} are always greater in magnitude than the remaining[15]. Then, the idea is to distribute the uncertainty among the polynomial coefficients so that dominant ones contain more uncertainty.

Table 6-3 Distribution of Uncertainty among Polynomial Coefficients for C_n

C_n	δ^2	δ^1	δ^0
α^2	b_{22} %0.5	b_{21} %1.0	b_{20} %0.5
α^1	b_{12} %1.0	b_{11} %0.5	b_{10} %1.0
α^0	b_{02} %0.5	b_{01} %1.0	b_{00} %0.5

Total: 6.5%

[15] Actually this is not a surprise, because aerodynamic behavior of this system change linearly with α and δ. For example, both C_n and C_m change linearly with α, so in their polynomial representation, b_{10} is of greater magnitude. Same is true for δ and b_{01}. b_{21} and b_{12} are also dominant whereas remaining coefficients expressing the effect of higher order cross-coupling between α and δ are not that recessive.

Table 6-4 Distribution of Uncertainty among Polynomial Coefficients for C_m

C_m	δ^2	δ^1	δ^0
α^2	b_{22} %1.0	b_{21} %2.0	b_{20} %1.0
α^1	b_{12} %2.0	b_{11} %1.0	b_{10} %2.0
α^0	b_{02} %1.0	b_{01} %2.0	b_{00} %1.0

Total: 13%

Procedure for distributing the gross uncertainty among the polynomial coefficients has to be validated. Graphs in Figure 6-6 and Figure 6-7 are given for this purpose. The histograms given in these graphs show distribution of 10000 probabilistic values[16] for C_n and C_m evaluated at DP=3, $\alpha = 3^\circ$, $\delta = -5^\circ$ through the use formula (3.1). Straight line in the middle is the nominal value, remaining lines show the bound of uncertainty given by DATCOM[17]. For this example, it is seen that probabilistic values obtained for both C_n and C_m do not represent the entire region defined by the uncertainty bounds.

For cases like this, the designer may need to make some corrections on the uncertainties of polynomial coefficients. It must be mentioned that, the procedure employed here is an ad-hoc one, therefore no further effort is spent to find the values that fully represent the uncertainty defined by DATCOM. However, the code enables more complex methods to be utilized and the designer may make some corrections so that the full uncertainty region is covered.

[16] Based on Gaussian distribution

[17] These lines simply show the ±7% bound for C_n, ±13% bound for C_m.

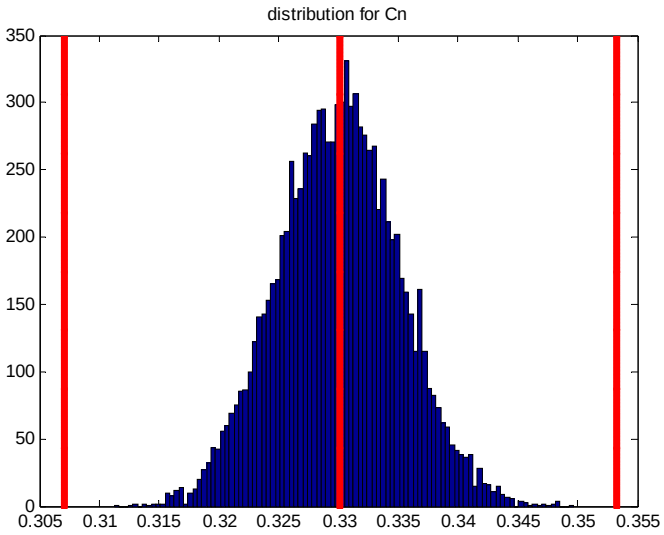


Figure 6-6 Histogram for C_n at DP3

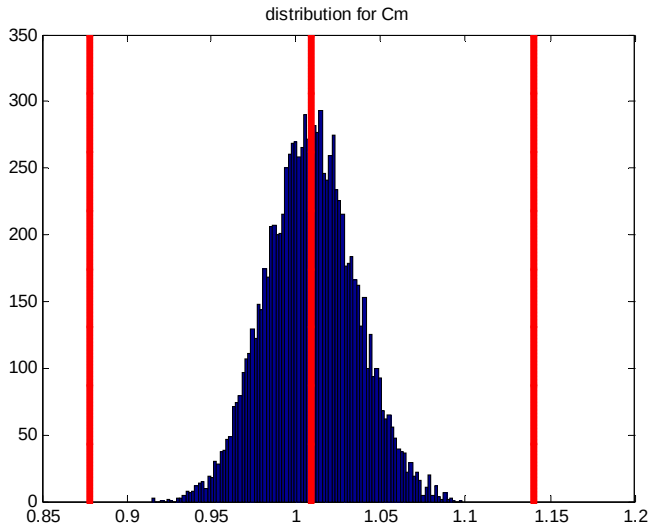


Figure 6-7 Histogram for C_m at DP4

The parameters in the system model contain both the aerodynamic coefficients themselves, as well as their derivatives. This representation also serves as a means for expressing both at the same time. For example, C_{n_α}, which is expressed as $C_{n\alpha} = 2(b_{22} \cdot \delta^2 + b_{21} \cdot \delta + b_{20}) \cdot \alpha + (b_{12} \cdot \delta^2 + b_{11} \cdot \delta + b_{10})$ contains the uncertainties of the six parameters present in its definition.

Dynamic derivatives are assumed to be constant, i.e., independent of α and δ at each design point.

Possible uncertainty on parameters related to mass and inertia are

- 1% on m
- 1% on J

Using all these definitions, uncertainty on each component can be combined into a single value for each parameter: uncertainty equals to the value of the parameter for worst combination of its integrants. For example, consider a parameter, say B_{11}, consisting of 7 uncertain components.

$$B_1 = -\cos^2(\alpha)\cdot Q\cdot S\cdot\begin{bmatrix}\left(2\cdot C_{n_{b22}}\cdot\delta_{trim}+C_{n_{b21}}\right)\cdot\alpha^2\\+\left(2\cdot C_{n_{b12}}\cdot\delta_{trim}+C_{n_{b11}}\right)\cdot\alpha\\+2\cdot C_{n_{b02}}\cdot\delta_{trim}+C_{n_{b01}}\end{bmatrix}\cdot\frac{1}{m\cdot u}$$

Let the nominal value of B_{11} be B_{11nom} at some design point. And let the value of B_{11} with its integrants uncertain be B_{11unc}. Then W_{11} is defined as

$$W_{11} = \max(\left|B_{11unc} - B_{11nom}\right|)$$

Similar definition applies for the rest of parameters.

Typical magnitude-frequency plot for these weights is given in Figure 6-8.

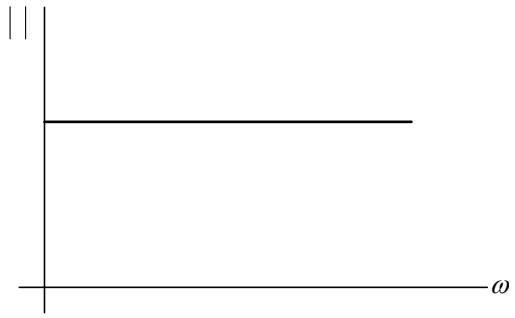


Figure 6-8 Typical Magnitude-Frequency Plot for Weights 1-9

6.2.2.2 Uncertainty Weighting Functions W10 to W12

These weights reflect the uncertainty present on the CAS. For the representation of uncertainty, the path for the missile dynamics is followed[18]: every element in the state-space representation of CAS dynamics is assumed to

[18] Since CAS is a component which has dynamics close the missile dynamics.

be uncertain, and then gross uncertainty for these coefficients is calculated. The state-space representation of the CAS is

$$\begin{bmatrix}\dot{\delta}\\ \ddot{\delta}\end{bmatrix}=\begin{bmatrix}0 & 1\\ -\omega_n{}^2 & -2\zeta\omega_n\end{bmatrix}\begin{bmatrix}\delta\\ \dot{\delta}\end{bmatrix}+\begin{bmatrix}0\\ \omega_n{}^2\end{bmatrix}u\overset{\Delta}{=}\begin{bmatrix}A_{11} & A_{12}\\ A_{21} & A_{22}\end{bmatrix}\begin{bmatrix}\delta\\ \dot{\delta}\end{bmatrix}+\begin{bmatrix}B_{11}\\ B_{21}\end{bmatrix}u$$
$$\begin{bmatrix}\delta\\ \dot{\delta}\end{bmatrix}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\delta\\ \dot{\delta}\end{bmatrix}\overset{\Delta}{=}\begin{bmatrix}C_{11} & C_{12}\\ C_{21} & C_{22}\end{bmatrix}\begin{bmatrix}\delta\\ \dot{\delta}\end{bmatrix} \qquad (6.3)$$

The table for the correspondence of weights to uncertainties on parameters is given below:

Table 6-5 Uncertainties for CAS Parameters

Parameter	Parameter on which uncertainty is represented
W_{10}	A_{21}
W_{11}	A_{22}
W_{12}	B_{21}

The uncertainties in CAS are as follows:

- 1% on ω_n
- 1% on ζ

Magnitude-frequency diagrams for these weights are the same as plant uncertainties.

As well as the uncertainty definitions, the structure for representing the uncertainty is similar to that of plant's: the structure of functions are as given in Figure 6-4, where in function W_X, X is from 10 to 12.

6.2.2.3 Performance Weighting Function "W_an_per"

This weight represents the cost function on the acceleration performance of the system. It is required that for low frequency region, where the guidance commands will be, error in reference tracking must be smallest, resulting in a big penalty on errors. The inputs in the high frequency region are usually the noise, body bending modes, etc…, so penalty on the errors for these parameters are small. Typical magnitude-frequency diagram is given in Figure 6-9

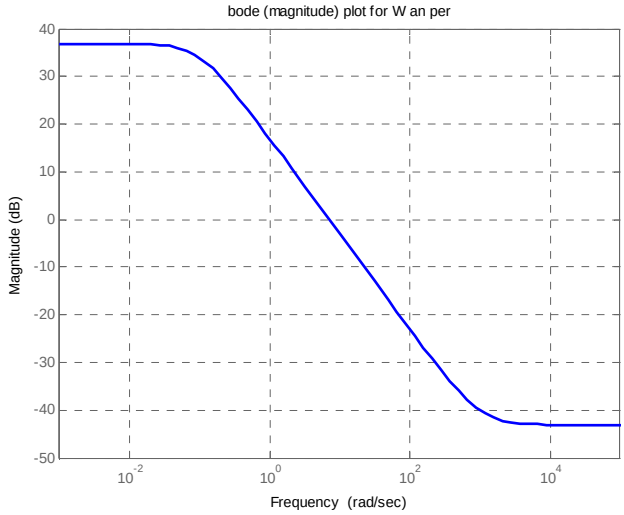


Figure 6-9 Typical Magnitude-Frequency Plot for W_an_per

If different performance levels are required/acceptable at different phases of the flight, the weighting function may be different for each design point, which is also the case in this study.

6.2.2.4 Performance Weighting Function "W_alpha_per"

This weight represents the cost function on α, resulting from the fact that the ramjet can operate in a specified region. Typical magnitude-frequency diagram is given in Figure 6-10.

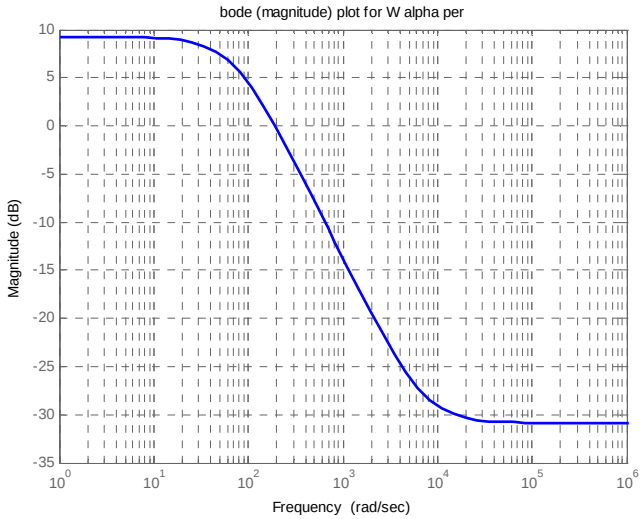


Figure 6-10 Typical Magnitude-Frequency Plot for W_alpha_per

6.2.2.5 Performance Weighting Function "W_acc_noise" and "W_gyro_noise"

"W_a_noise", "W_g_noise" represent the frequency content for the accelerometer noise and RGU noise, respectively. In chapter 3, it was mentioned that the rate gyros can sense motions up to 60°/s and the accelerometers can sense accelerations up to 10g's. In the light of this information and the fact that sensors used in a missile has to be of good quality, it is assumed that the noise in sensor measurements is small, i.e., 0.1% of the true value, at low frequency. As the frequency of the inputs increase, the effect of noise increases, reaching to values of 10 times, at the limits of the measuring ranges. Typical magnitude-frequency diagram is given in Figure 6-11.

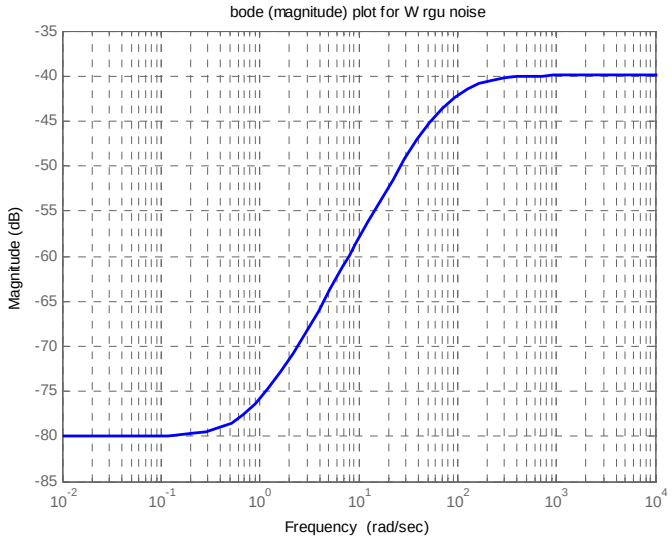


Figure 6-11 Typical Magnitude-Frequency Plot for W_a_noise, W_g_noise

6.2.2.6 Performance Weighting Functions "W_act_pper" and "W_act_sper"

W_act_pper and W_act_sper reflect the penalty on actuator states position and velocity respectively, and are limited. In chapter 3 it was mentioned that CAS is accepted to provide 15° of deflections for the control surfaces with 200°/s turn rates. Typical magnitude-frequency diagram is given in Figure 6-12.

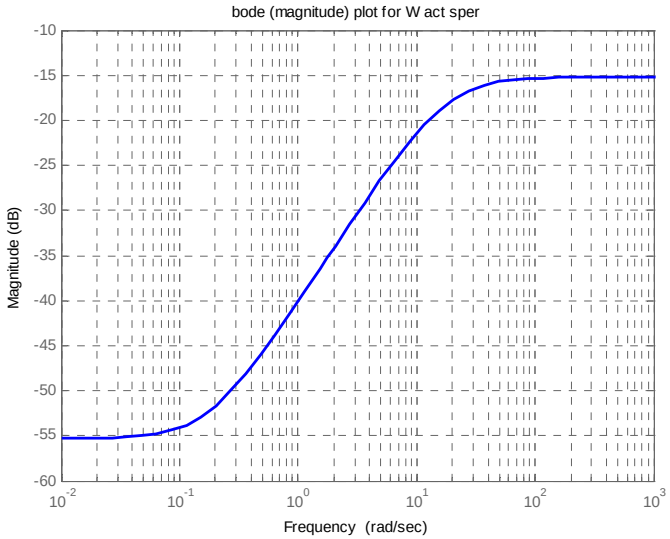


Figure 6-12 Performance weight for W_act_sper, W_act_pper

6.3 Implementation on MATLAB

By constructing the system structure and defining the character of the uncertainties and performance requirements, the system is ready for implementation in the MATLAB environment. The partition is in the form given in Figure 6-14. This is the same as the structure given in Figure 6-13, which is in H_∞ framework.

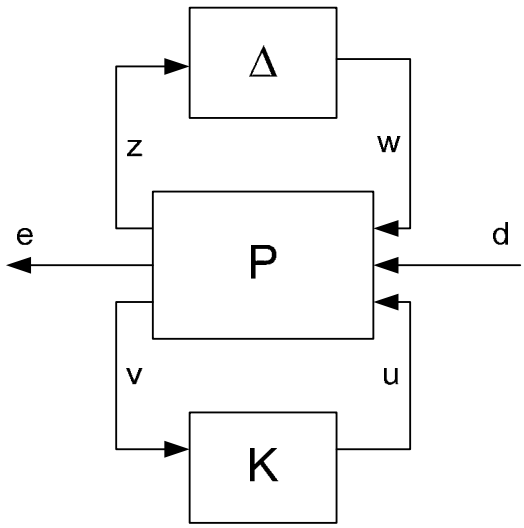


Figure 6-13 System Structure to be Input into MATLAB

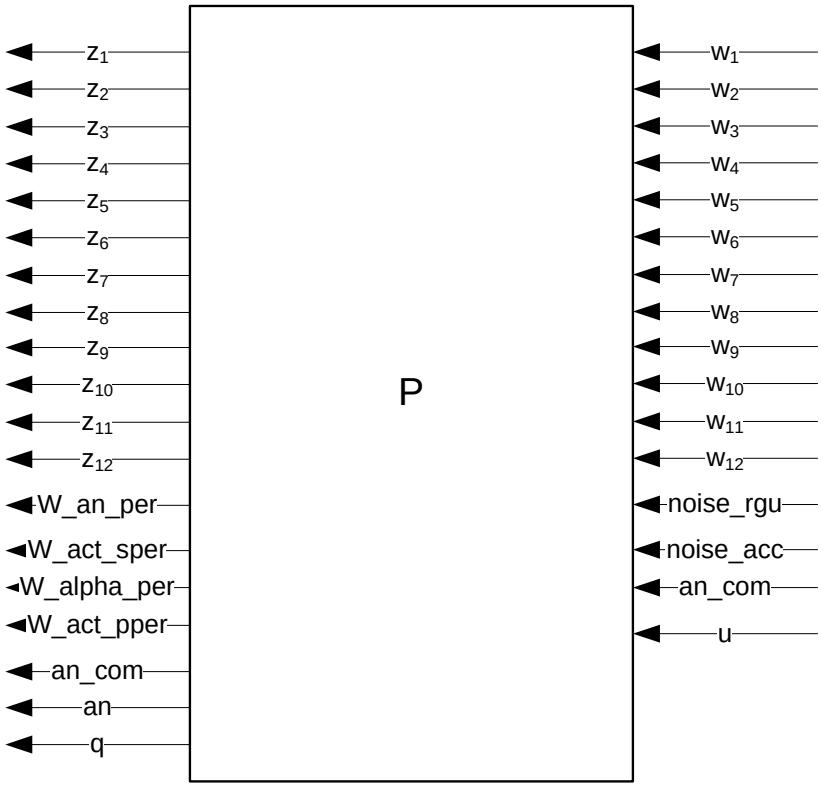


Figure 6-14 System Structure in H_∞ Framework

Therefore, all data needed is ready to be entered in the code written in MATLAB environment, in a special syntax. This syntax is available in [11]. The code that performs the entire task explained up to now is given in appendix E. This code repeats the process for each design point, with different performance weights if desired.

6.4 Obtaining Controllers

Once the structure is constructed and input into the MATLAB program with the correct syntax, the command "hinfsyn" solves the problem given in section (5.27). Acceleration response to 1g step command, which are obtained through the linear simulation are given below. Then the non-linear simulation is performed for final validation. Non-linear simulation results are given in section 7.1, in which the classical controller results are also presented for comparison purposes.

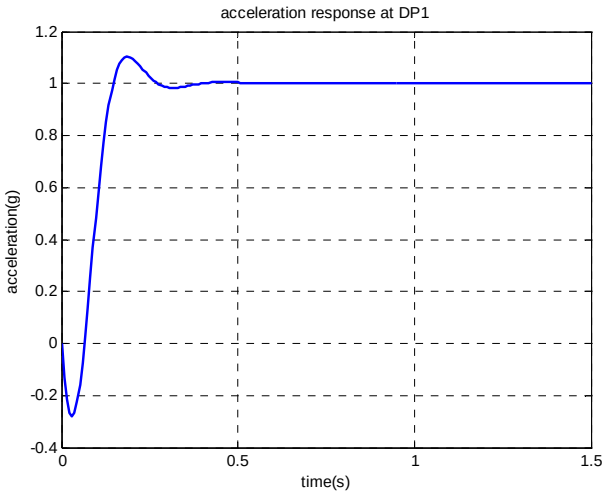


Figure 6-15 Linear Simulation Result for DP1

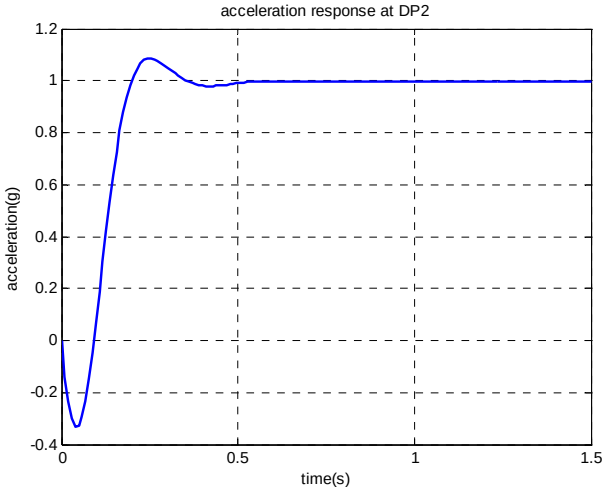


Figure 6-16 Linear Simulation Result for DP2

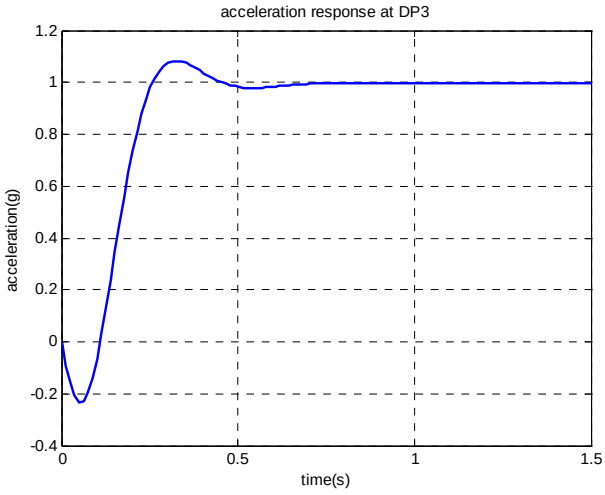


Figure 6-17 Linear Simulation Result for DP3

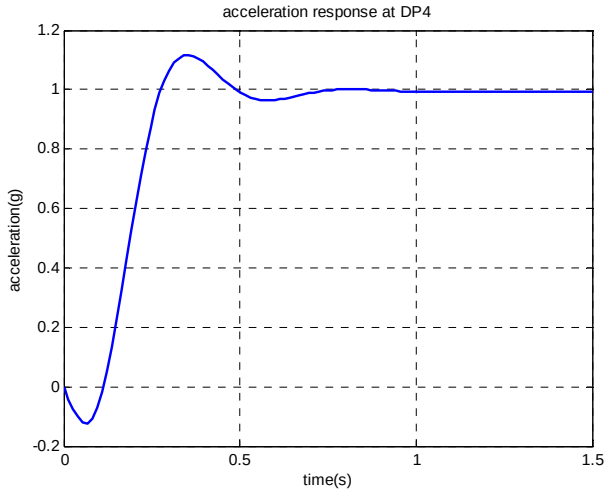


Figure 6-18 Linear Simulation Result for DP4

Numerical values regarding the performance of the closed system are given in 7.1.1. Moreover, for all the design points, the angle of attack does not go beyond 6°, and fin deflection rates are admissible.

For the closed loop systems whose closed loop responses are given above, γ values[19] are as follows:

Table 6-6 γ values for H_∞ Controllers

DP	γ value achieved
1	72.2211
2	26.123
3	22.9092
4	36.486

All of the γ values are greater than 1, however, as explained in 5.2.7, $\gamma < 1$ is a conservative criterion for evaluating RP. μ plots of the system, which are presented in section 7.1.2 reveal that RP criterion is also satisfied.

[19] See section 5.2.6 for definition of γ

6.5 Controller Order Reduction

As given in 5.2.6, the H_∞ controller has as many states as the system. This results in complex controllers for large scale systems. However, complex controllers are hard to implement on the missile computer. They even are burdens for simulations for that are conducted for validating them. It is also a known fact that order of H_∞ controllers are likely to be considerably greater than truly needed. One may, without degrading the performance much, can reduce the order of the controller. Controller order reduction is a very widely used technique in control system design.

In control theory, model reduction is achieved through keeping "energetic" states and omitting the rest. This enables the dominant states of the system to preserve most of the characteristics, while the absence of omitted ones, since they are not "energetic", does not influence the overall picture.

In mathematical representation, "dominant" or "energetic" states are expressed as their measures in terms of Hankel Singular Values. Hankel singular values of a system are defined as

$$\sigma_H = \sqrt{\lambda_i(PQ)} \qquad (6.4)$$

In (6.4), P and Q are the Controllability and Observability Grammians satisfying

$$\begin{aligned} AP + PA^T &= -BB^T \\ A^TQ + QA &= -C^TC \end{aligned} \qquad (6.5)$$

For a system $G = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ model reduction routines offer reduced models G_{red} of the real system G satisfying

$$\left\| G - G_{red} \right\|_\infty \le 2\sum_{i=k+1}^{n} \sigma_i \qquad (6.6)$$

In (6.6), k is the order of G_{red} and n is the order of G.

In MATLAB, the command "reduce" is used for model reduction. For obtaining a reduced order controller, this command is used. Resultant controllers are used in the rest of the processes.

CHAPTER 7

RESULTS and CONCLUSION

7.1 Results

Responses of the robust controller to acceleration commands is given in the figures below. All the curves are obtained by non-linear simulation. The commands are extracted from expected g-envelopes given in chapter 2. Results for the classical controller are added for comparison. However, it must be kept in mind that, in this thesis, it is not intended to claim that robust controllers give "a lot better" results than classical controllers. The aim is to show that the controllers obtained with the code give comparable/acceptable results with guaranteed stability/performance characteristics.

7.1.1 Performance Issues

Numerical values about the performance requirements related to acceleration, which are stated in section 2.3, are given below:

Table 7-1 Performance Comparison of Controller, DP1

	Requirement	**Result with CCT**	**Results with RCT**
Steady State Error	<10%	5.5%	0
% OverShoot	<15%	15%	10.1%
Rise Time (s)	<1	0.0689	0.06
Settling Time (s)	<1.5	0.387	0.249

Table 7-2 Performance Comparison of Controller, DP2

	Requirement	Result with CCT	Results with RCT
Steady State Error	<10%	3.7%	0.2%
% OverShoot	<15%	13.2%	8.88%
Rise Time (s)	<1	0.088	0.078
Settling Time (s)	<1.5	0.512	0.325

Table 7-3 Performance Comparison of Controller, DP3

	Requirement	Result with CCT	Results with RCT
Steady State Error	<10%	3.1%	0.5%
% OverShoot	<15%	15.1%	8.81%
Rise Time (s)	<1	0.158	0.108
Settling Time (s)	<1.5	0.741	0.325

Table 7-4 Performance Comparison of Controller, DP4

	Requirement	Result with CCT	Results with RCT
Steady State Error	<10%	%3.7	0.5%
% OverShoot	<15%	14.9%	9.98%
Rise Time (s)	<1	0.183	0.347
Settling Time (s)	<1.5	0.749	0.867

7.1.1.1 Design Point 1

As explained in section 2.2.1, this design point represents the flight condition at the end of the boost phase where altitude is 5000m, and the velocity is M=2.5. The missile is expected to perform maneuvers in a [-1 4]g range.

In Figure 7-1, response to commanded acceleration is given. It can be seen that all related performance requirements given in section 2.3 are satisfied, however steady-state characteristic of the robust controller is better.

For both controllers, required deflections are quite small as shown in Figure 7-2. However, classical controller requires a higher deflection rate. It is also

clear from Figure 7-4 that such an acceleration envelope can be fulfilled within the possible angle of attack envelope.

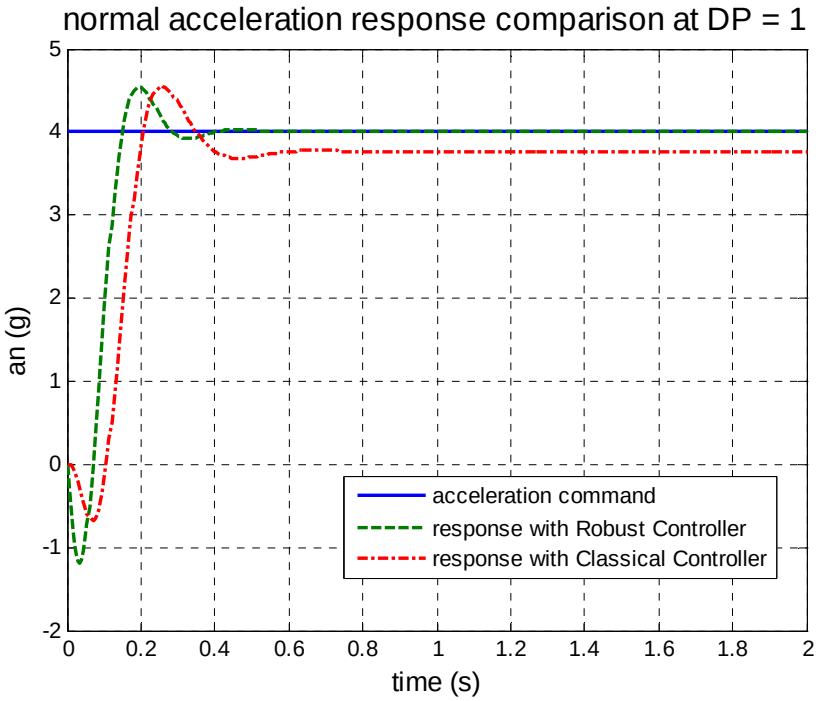


Figure 7-1 a_n response @ DP.1

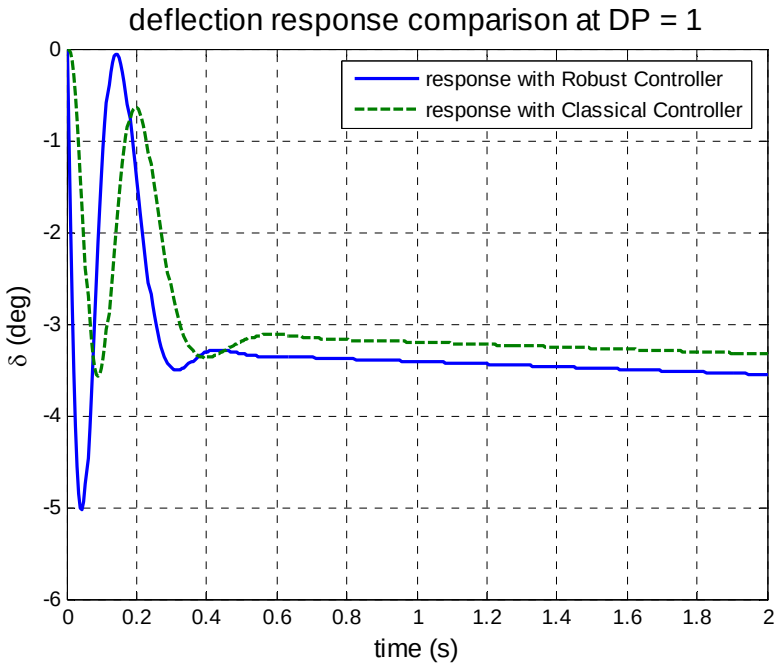


Figure 7-2 δ response @ DP.1

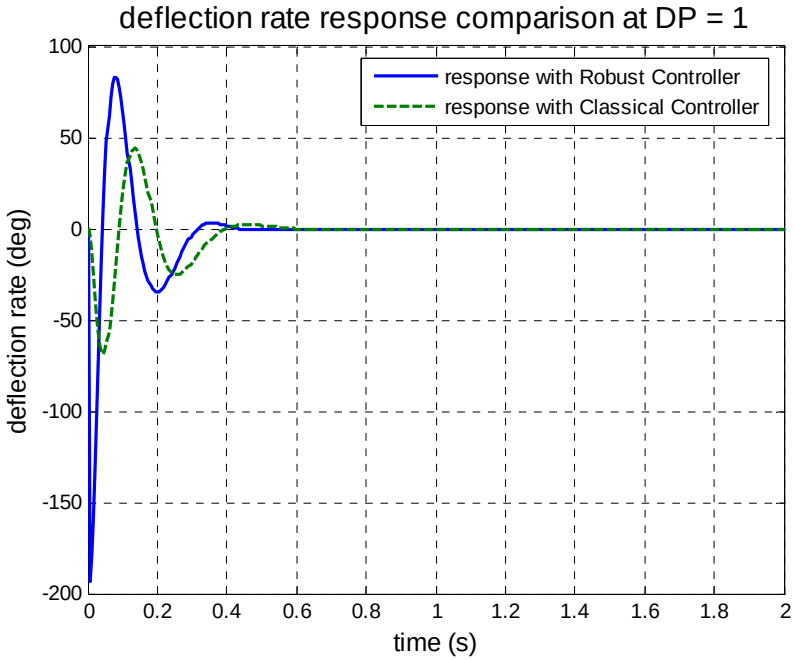


Figure 7-3 $\dot{\delta}$ response @ DP.1

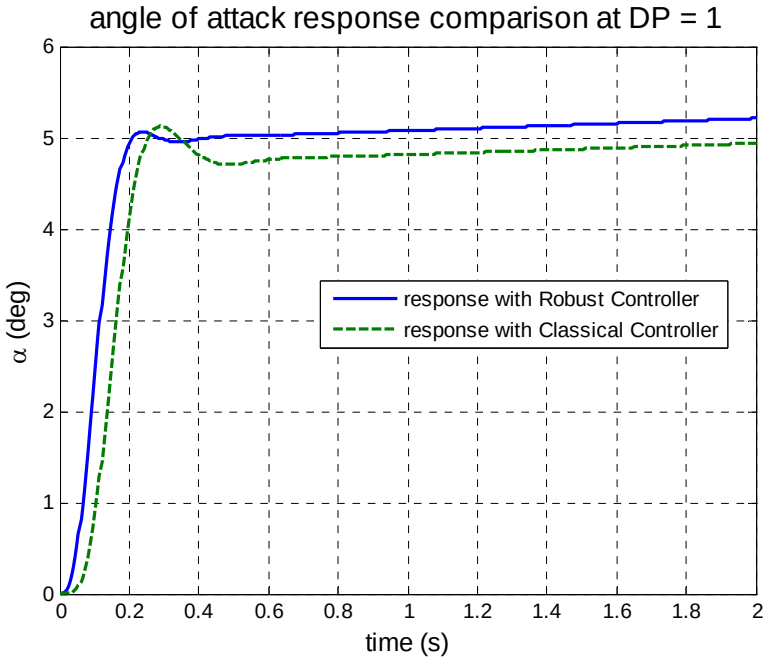


Figure 7-4 α response @ DP.1

7.1.1.2 Design Point 2

As explained in section 2.2.2, this design point represents the flight condition at the end of the first part of climbing phase, where the missile starts the pitch-

down maneuver. The altitude is 12000m, and the velocity is M=3. The missile is expected to perform maneuvers in a [-3 0]g range.

In Figure 7-5, response to commanded acceleration is given. It can be seen that all related performance requirements given in section 2.3 are satisfied, however, steady-state characteristic of the robust controller is better.

This design point is the crucial part for the missile as explained in chapter 2. It can be seen in Figure 7-6 that, deserving a safety area for disturbance rejection[20], a -1g command can hardly be accomplished without violating the angle of attack limitations.

Since the acceleration requirement is small, deflection and deflection rates are both small, as seen from Figure 7-7 and Figure 7-8.

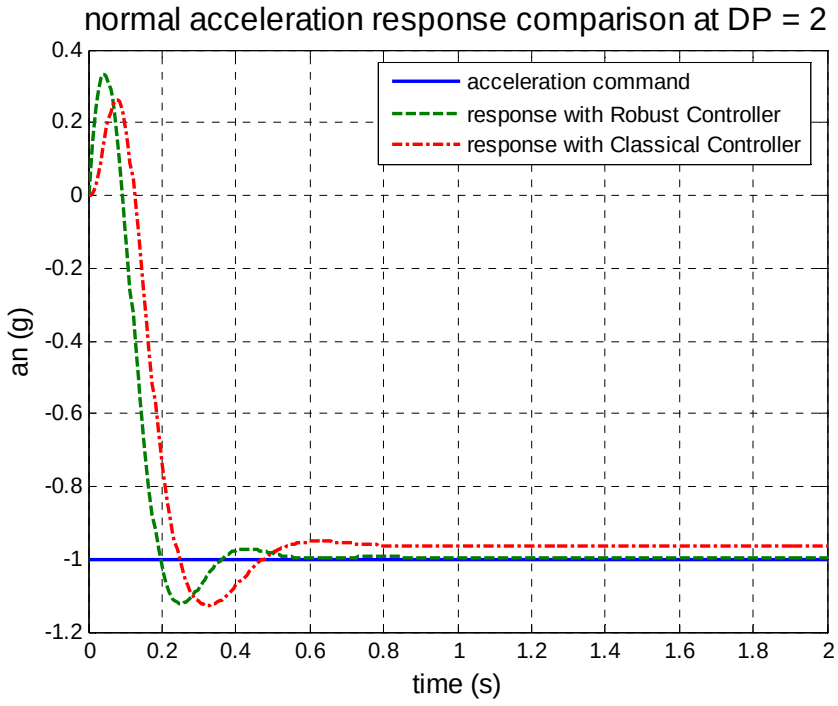


Figure 7-5 a_n response @ DP.2

[20] for possible gusts, wind etc.

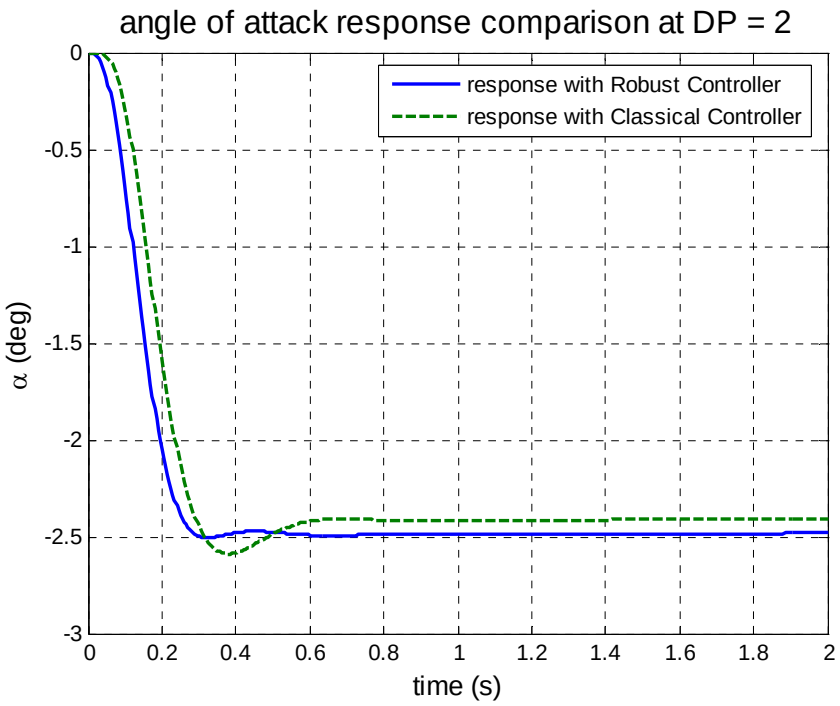


Figure 7-6 α response @ DP.2

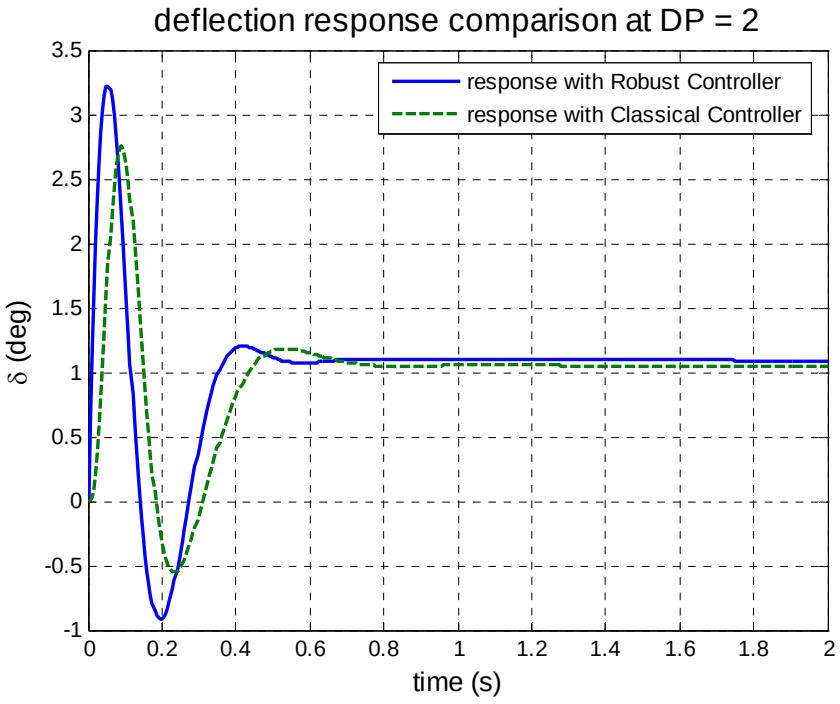


Figure 7-7 δ response @ DP.2

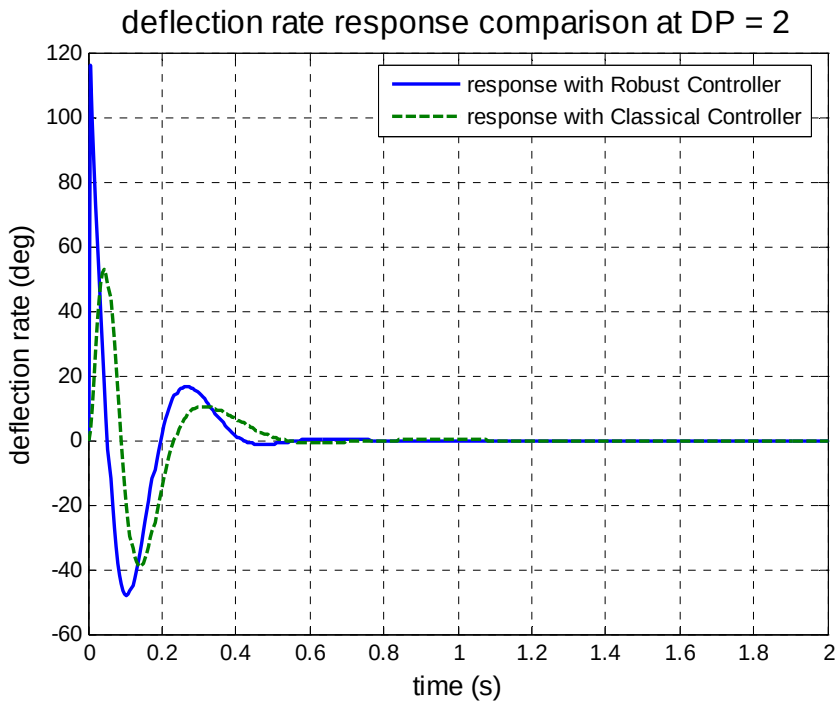


Figure 7-8 $\dot{\delta}$ response @ DP.2

7.1.1.3 Design Point 3

As explained in section 2.2.3, this design point represents the flight condition in cruise phase, where the altitude is 16000m, and the velocity is M=3.5. The missile is expected to perform maneuvers in a [-1 2]g range.

In Figure 7-9, response to commanded acceleration is given. It can be seen that all related performance requirements given in section 2.3 are satisfied, however, steady-state characteristic of the robust controller is better.

This design point represents the cruise phase, and the g-command is accomplished by both controllers. However, as seen from Figure 7-10, angle of attack limit is reached. Therefore, allowing a safety margin for disturbances, commanded-g margin has to be modified by the trajectory planning algorithm.

Deflection and deflection rate response characteristics are very similar to those of other DP's, as seen in Figure 7-11 and Figure 7-12.

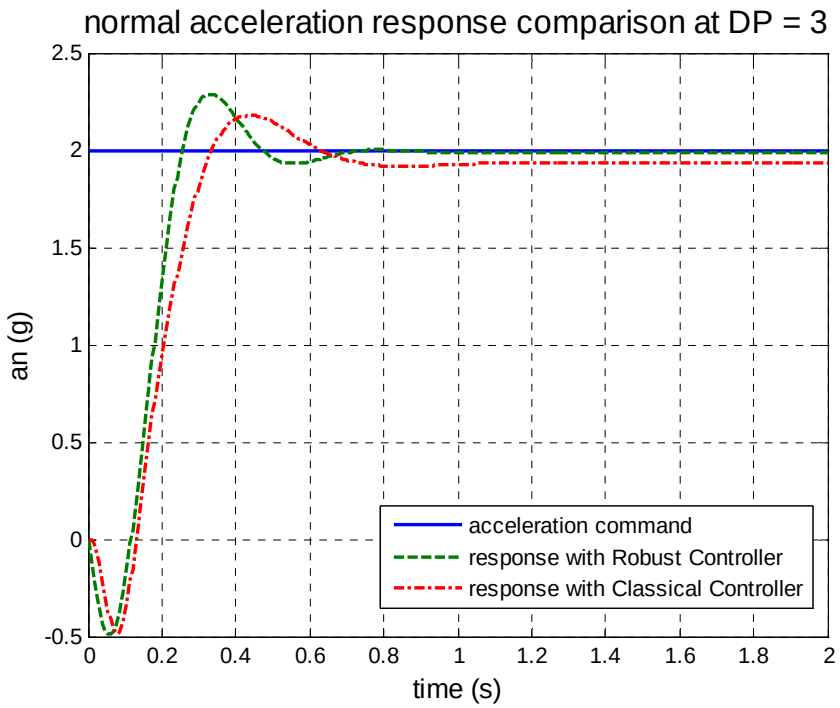


Figure 7-9 a_n response @ DP.3

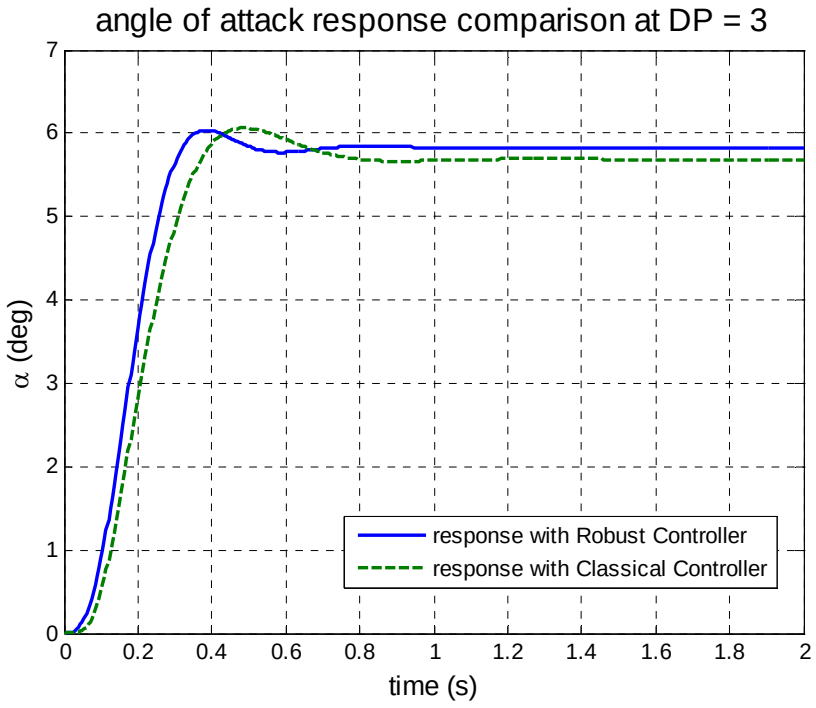


Figure 7-10 α response @ DP.3

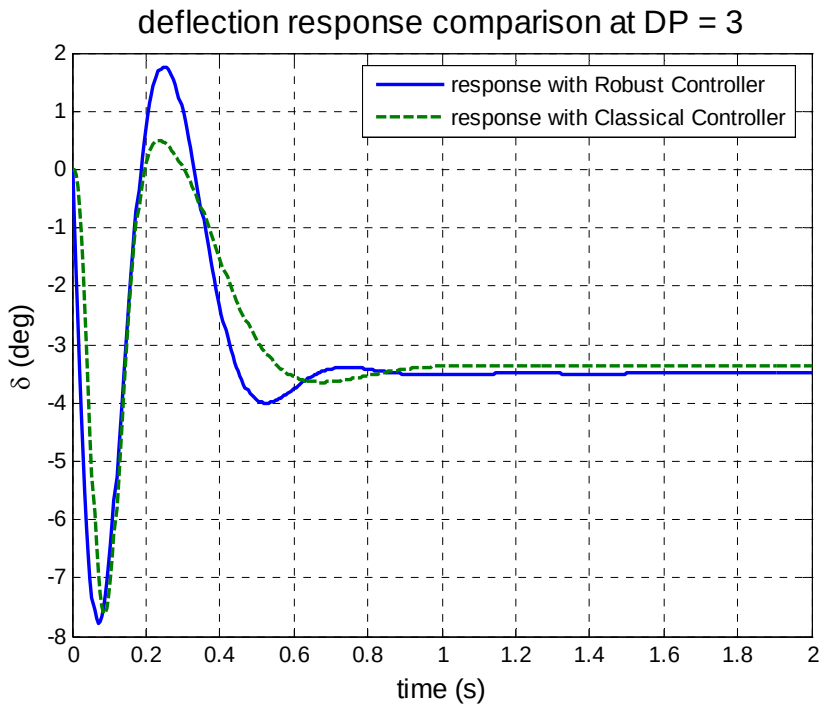


Figure 7-11 δ response @ DP.3

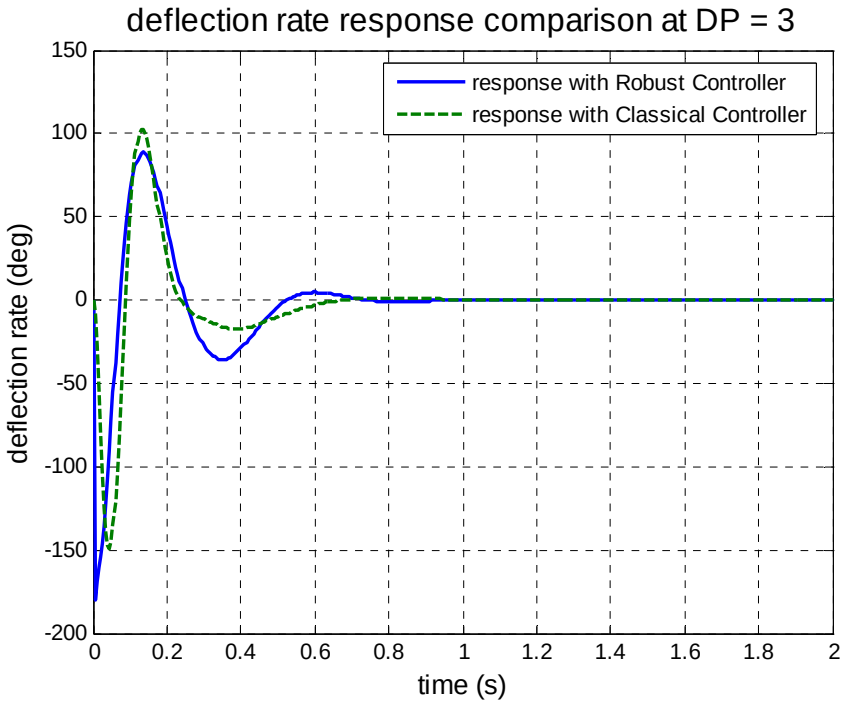


Figure 7-12 $\dot{\delta}$ response @ DP.3

7.1.1.4 Design Point 4

As explained in section 2.2.4, this design point represents the flight condition at the end of cruise phase, where the diving maneuver starts. No angle of attack limitation is present in the phase. The altitude is 16000m, and the velocity is M=3.5. The missile is expected to perform maneuvers in a [-4 2]g range.

In Figure 7-13, response to commanded acceleration is given. It can be seen that all related performance requirements given in section 2.3 are satisfied, however, steady-state characteristic of the robust controller is better.

It can be seen that more g-maneuvers can be performed, since α limitation does not exist in this DP. Moreover, there is still margin to enforce the control actuator.

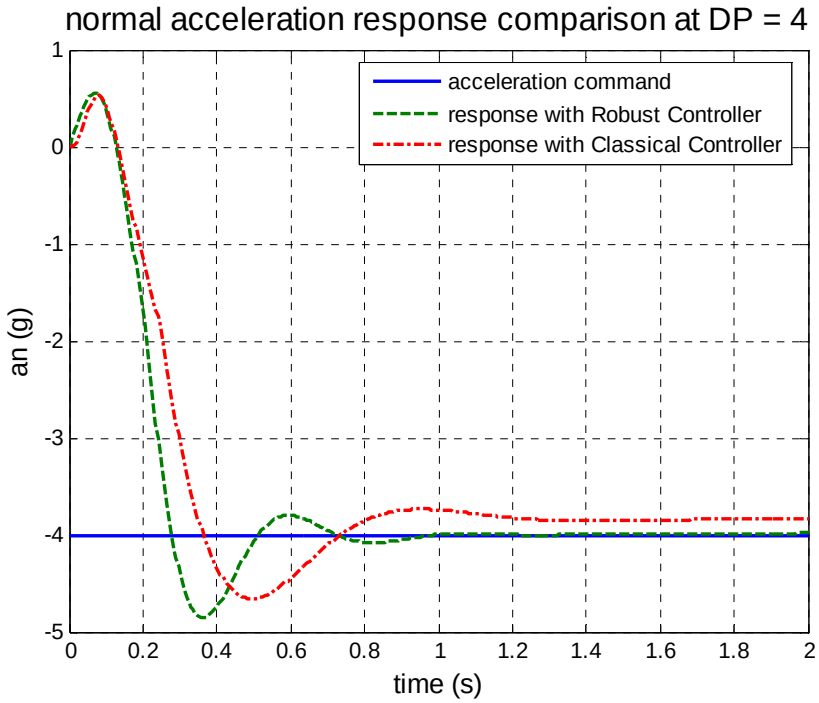


Figure 7-13 a_n response at DP.4

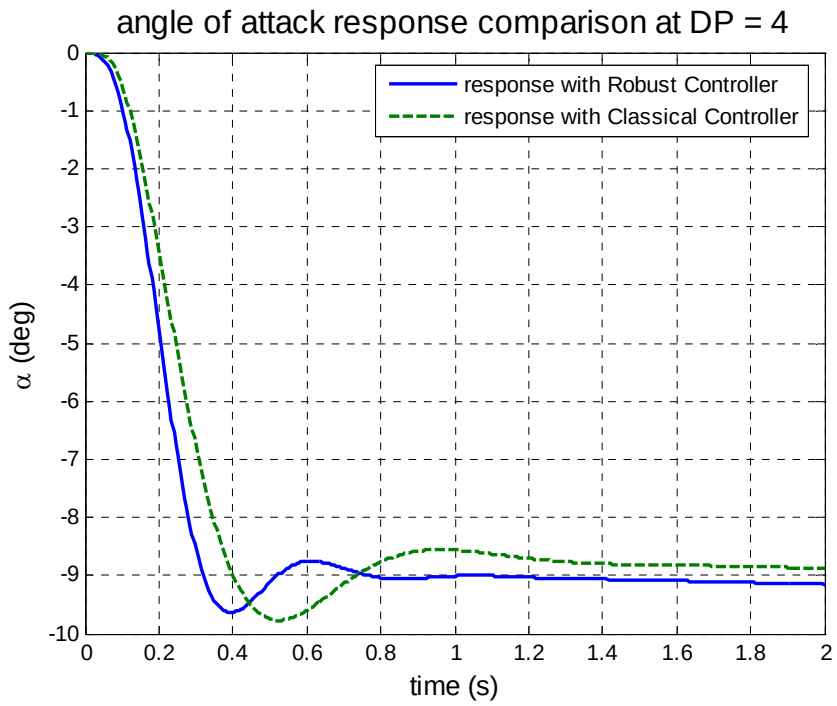


Figure 7-14 alpha response at DP.4

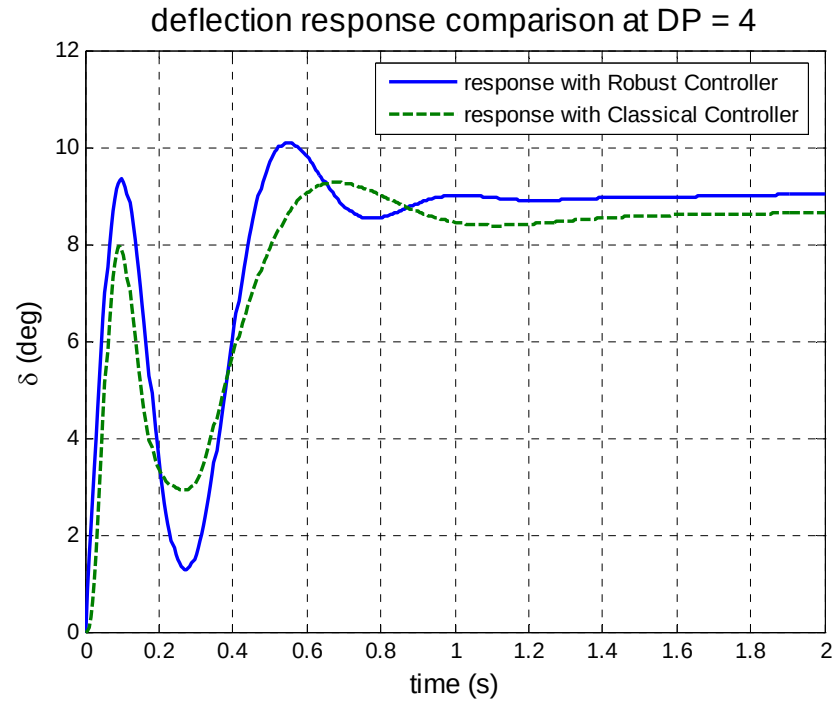


Figure 7-15 δ response @ DP.4

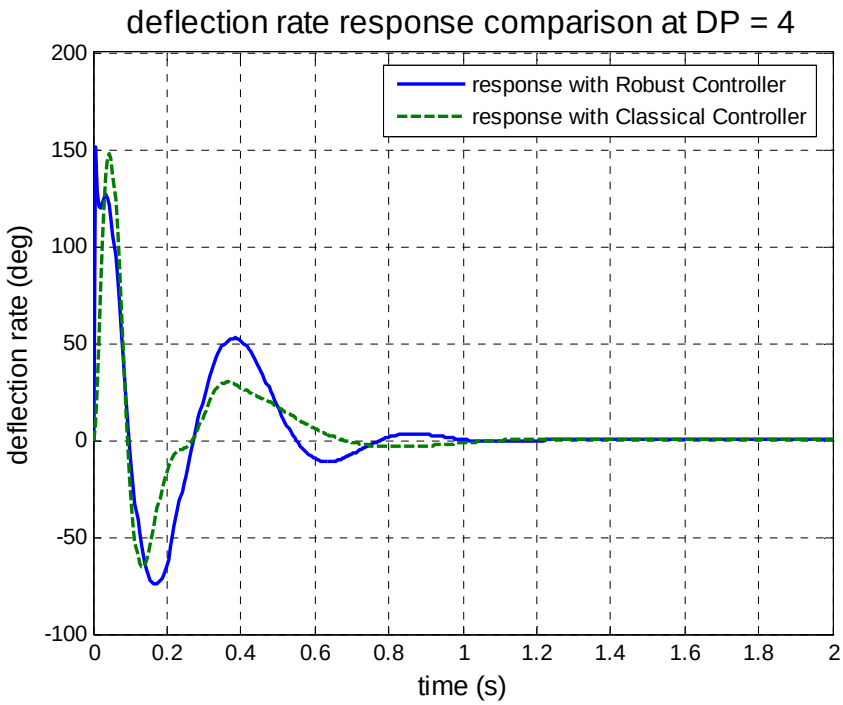


Figure 7-16 $\dot{\delta}$ response @ DP.4

7.1.2 Stability Issues

For all controllers results of which are given above, γ value was greater than 1. as explained in section 5.2.7, μ plots of systems are drawn to see whether the robust performance criteria is satisfied. These graphs are given in Figure 7-17, Figure 7-18, Figure 7-19 and Figure 7-20.

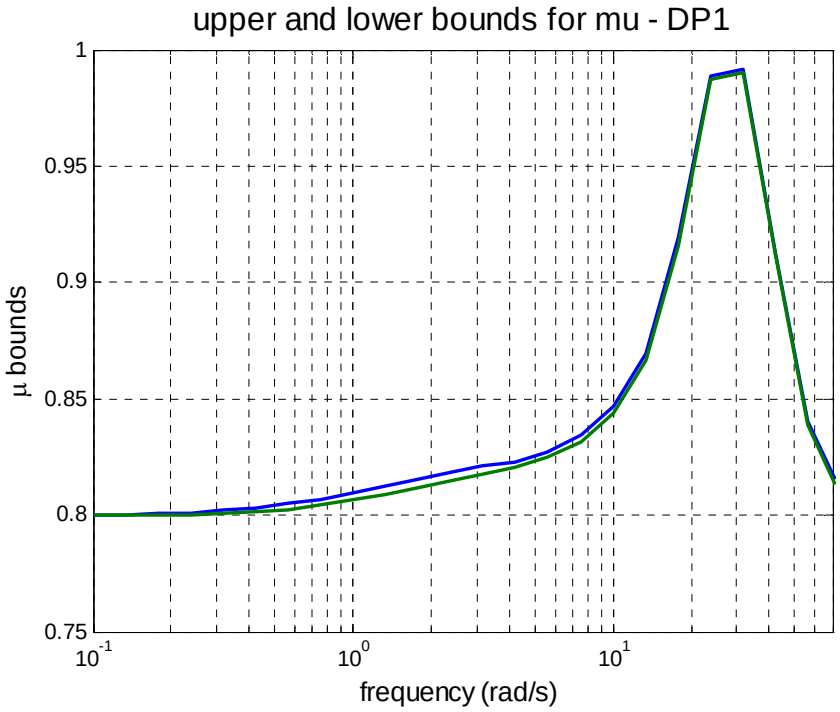


Figure 7-17 μ plot for DP1

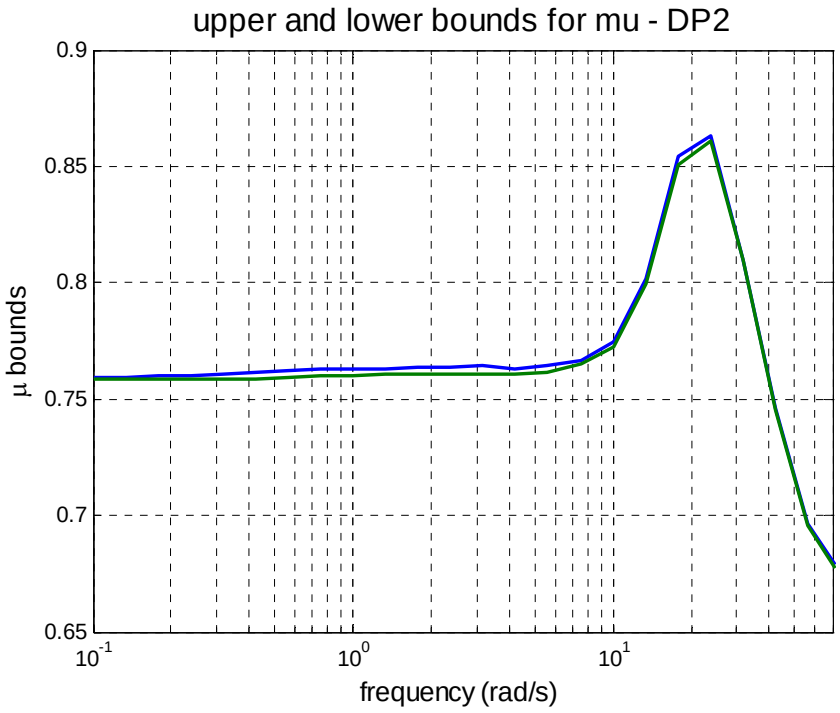


Figure 7-18 μ plot for DP2

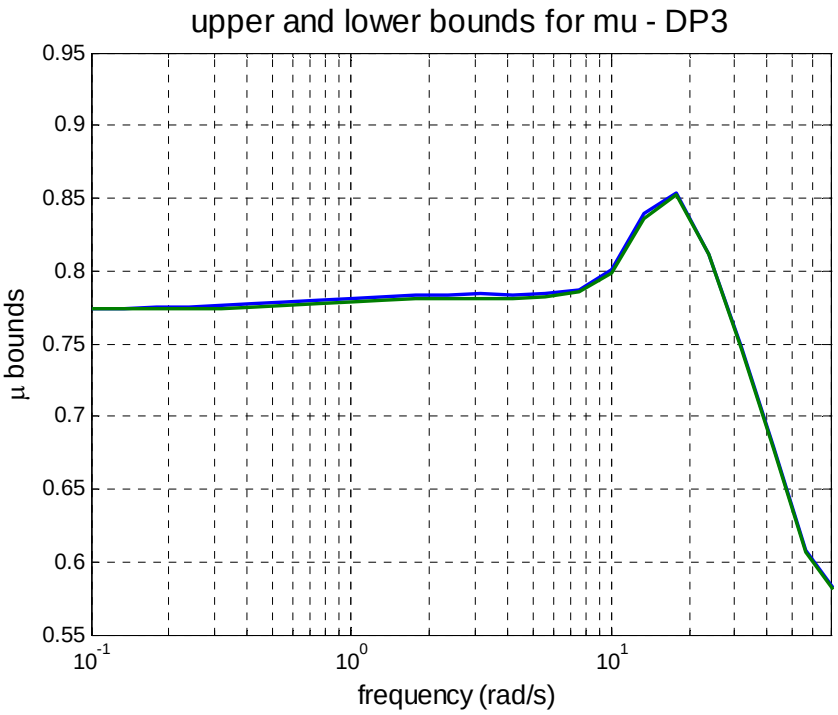


Figure 7-19 μ plot for DP3

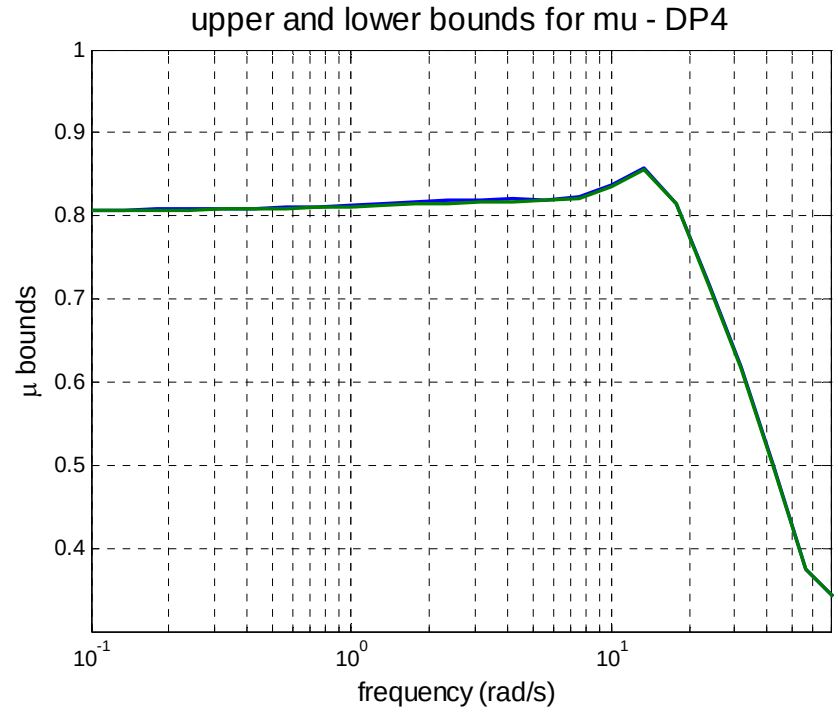


Figure 7-20 μ plot for DP4

7.2 Conclusion

In this thesis, a design study for the longitudinal autopilot of a ramjet-powered air-to-surface missile has been performed. This study fulfills the "autopilot" part of the overall design process. The aim of the study was designing an autopilot with the following achievements:

- Generating an automatic algorithm for design
- Reducing the time required for autopilot design between successive iterations
- Reducing the effort needed for the autopilot design

Realizing such aims requires a methodology that is suitable for automatization, therefore Robust Control Theory was used. Robust Control Theory not only serves as a perfect tool for automatization, but it also makes the handling of problems easier. Further advantages of the Robust Control Theory can be listed as follows:

- Involving limitations and parametric uncertainties and/or specifying "working frequency regions" are easier.
- It is possible to explicitly pronounce the performance specifications. For example, if the designer is not happy about the response of the actuator, the only thing to do is to modify the actuator performance weight, which is an explicitly defined function.
- Working with MIMO systems is not much different than working with SISO systems.

Consequently, the studies resulted in a program that can design the autopilot as soon as the preliminary data about the missile is provided. This data comprises the aerodynamic data provided by Missile DATCOM program, mass property expectations and some trajectory parameters. The program then processes this raw data, converts them into useful format for controller design, constructs the structure for the controller, and finally designs it.

The problem in hand is a ramjet missile which is launched from an aircraft to a non-moving surface target. Properties of the trajectory and flight parameters are provided in chapter 2. Ramjet missile flies much longer at high speeds than

conventional missiles, so time-to-target is shorter. This advantage of ramjet propulsion systems is overshadowed by its operational limitation in terms of angle of attack range. This difficulty is stated in chapter 2 which results in further challenges in inlet design, trajectory planning and autopilot design.

The results are provided in previous section. It can be seen that all current performance specifications are satisfied. These results are also comparable with the ones of classical controller. Yet, it must be mentioned that Robust Controller results are generated in seconds, while classical controller takes quarter of an hour to design, for each DP.

Robust Control Theory enables more analysis to be performed: With advanced uncertainty analysis, it is possible to predict uncertainties that can cause instability and/or loss of required performance. On the other hand, it is understood that while the performance specifications are held constant, the program can be used as a means of evaluating the aerodynamic performance of alternative aerodynamic configurations.

When the program is utilized for design, since all processes are done automatically, no additional effort other than changing the inputs is needed for the next iteration. If there still exists need for modification, it provides a user-friendly interface, since the limitations and uncertainties are explicitly pronounced.

The program uses the formulae and methods that are generic for STT missiles, so modification needed for implementation on a different STT missile system is minor. Though, if autopilots for BTT missiles are demanded, code must be partly modified.

For automatization purposes, aerodynamic data has been expressed in polynomials. This enabled the symbolic representation of aerodynamic coefficients in terms of polynomials, so the aerodynamic derivatives can easily be computed. Polynomial fitting algorithm implemented in the code can be improved. It is seen that second order polynomial fitting of aerodynamic coefficients might be insufficient for some flight conditions. For the current missile configuration aerodynamic coefficients vary linearly with α and δ, therefore second order representation is considered to be satisfactory, and a

higher order representation is not used. However, for a different aerodynamic configuration a higher order fitting may be necessary. In this case code needs to be partly modified and the necessary (generalized) algorithm to do this modification is also given in the appendix.

It can be concluded that, the code prepared in this thesis study, can be used as a tool for designing autopilots in an efficient way, which is sought and highly demanded by the industry.

REFERENCES

1. P. ZARCHAN, Tactical and Strategic Missile Guidance, 3rd Ed., AIAA Progress in Astronautics and Aeronautics vol. 176, 1997
2. G. ZAMES, "Feedback and Optimal Sensitivity: Model Reference Transformation, Multiplicative Semi-norms and Approximate Inverses", IEEE Transactions on Automatic Control, vol. AC-26, pp.301-320, 1981
3. K. GLOVER, J.C. DOYLE, "State-space Formulae for All Stabilizing Controllers that Satisfy an H_∞ Norm Bound and Relations to Risk Sensitivity", Systems and Control Letters, vol. 11, pp. 167-172, 1988
4. J.C. DOYLE, K. GLOVER, P.P.KHARGONEKAR, B.A.FRANCIS, "State-space Solutions to Standard H_2 and H_∞ Problems", IEEE Transactions on Automatic Control, vol.AC-34, No. 8, pp 831-847, 1989
5. K. ZHOU, J.C. DOYLE, K. GLOVER, Robust and Optimal Control, Prentice Hall, 1996
6. S. SKOGESTAD, I. POSTLETHWAITE, Multivariable Feedback Control, John Wiley & Sons, 1996
7. K. ZHOU, J.C. DOYLE, Essentials of Robust Control, Prentice Hall, 1998
8. A. DAMEN, S. WEILAND, Robust Control, Unpublished (Draft version), 2002
9. S. TOFFNER-CLAUSEN, P. ANDERSEN, J. STOUSTRUP, Robust Control, 4th Ed., Lecture Notes for Course on Robust and Optimal Control in Department of Control Engineering of Aalborg University, 2001

10. B.A. FRANCIS, G. ZAMES, "On H_∞-Optimal Sensitivity Theory for SISO Feedback Systems", IEEE Transactions on Automatic Control, vol. AC-29, No. 1, 1984
11. The MATHWORKS, MATLAB Robust Control Toolbox (ver. 3), User's Guide, 2005
12. G. ZAMES, "Input-Output Feedback Stability and Robustness, 1959-85", IEEE Control Systems Magazine, 1996
13. M. ALEMDAROĞLU, "Conceptual Internal Design and Computational Fluid Dynamics Analysis of a Supersonic Inlet", Master Thesis, Middle East Technical University, 2005
14. D. McLEAN, Automatic Flight Control Systems, Prentice Hall, 1990
15. K. OGATA, Modern Control Engineering, 3rd Ed., Prentice Hall, 1997
16. J.H. BLAKELOCK, Automatic Control of Aircraft and Missiles, 2nd Ed., John Wiley & Sons, 1997
17. Ö. ATEŞOĞLU, "Different Autopilot Designs and Their Performance Comparison for Guided Missiles", Master Thesis, Middle East Technical University, 1996
18. V.NALBANTOĞLU, "Autopilot Design for Missiles", Master Thesis, University of Minnesota, 1994
19. P.BENDOTTI, M.M'SAAD, "Advanced Control for the Design of STT Missile Autopilots", Conference Proceeding, IEEE, 1992
20. J.C.JUANG, C.F.LIN, J.R.CLOUTIER, J.H.EVERS, "Robust Full-Envelope Missile Autopilot Design", Conference Proceeding, IEEE, 1992
21. G.J. BALAS, A.K. PACKARD, "Design of Robust Time-Varying Controllers for Missile Autopilots", Conference Proceeding, 1st IEEE Conference on Control Applications, 1992
22. J.C.JUANG, C.F.LIN, J.R.CLOUTIER, J.H.EVERS, "Generalized Singular Robust Control Design", Conference Proceeding, 30th Conference on Decision and Control, 1991

23. K.Z. LIU, R. HE, “A Simple Derivation of ARE Solutions to the Standard H_∞ Control Problem Based on LMI Solution”, Conference Proceeding, 42nd IEEE Conference on Decision and Control, 2003
24. B.A. WHITE, Y.PATEL, D.H. VORLEY, M.HORTON, “Loop Shaping Design of a Robust Missile Autopilot”, Conference Proceeding, American Control Conference, 1995
25. S.ARIKI, “A New System Invariant and an Algebraic Proof of the Standard H_∞ Problem”, Conference Proceeding, 35th Conference on Decision and Control, 1996
26. The MATHWORKS, MATLAB Aerospace Blockset Help, Modeling a Classical Three Loop Autopilot
27. E. DEVAUD, J.P. HARCAUT, H. SIGUERDIDJANE, “Three-axes Autopilot Design: From Linear to Nonlinear Control Strategies”, Journal of Guidance, Control and Dynamics, vol. 24, No.1, pp.65-71, 2001
28. J. C. DOYLE, “Analysis of Feedback and Systems with Structured Uncertainties”, IEEE Proceedings, Part D, 129, 1982

APPENDIX A

GRAPHS FOR THE AERODYNAMIC COEFFICIENTS

This section involves the graphical presentation of aerodynamic data on which controllers will be designed. The data is obtained by Missile DATCOM.
Two important coefficients C_n and C_m are presented in detailed 2D graphs for α and δ derivatives. Other coefficients are given in 3D plots.

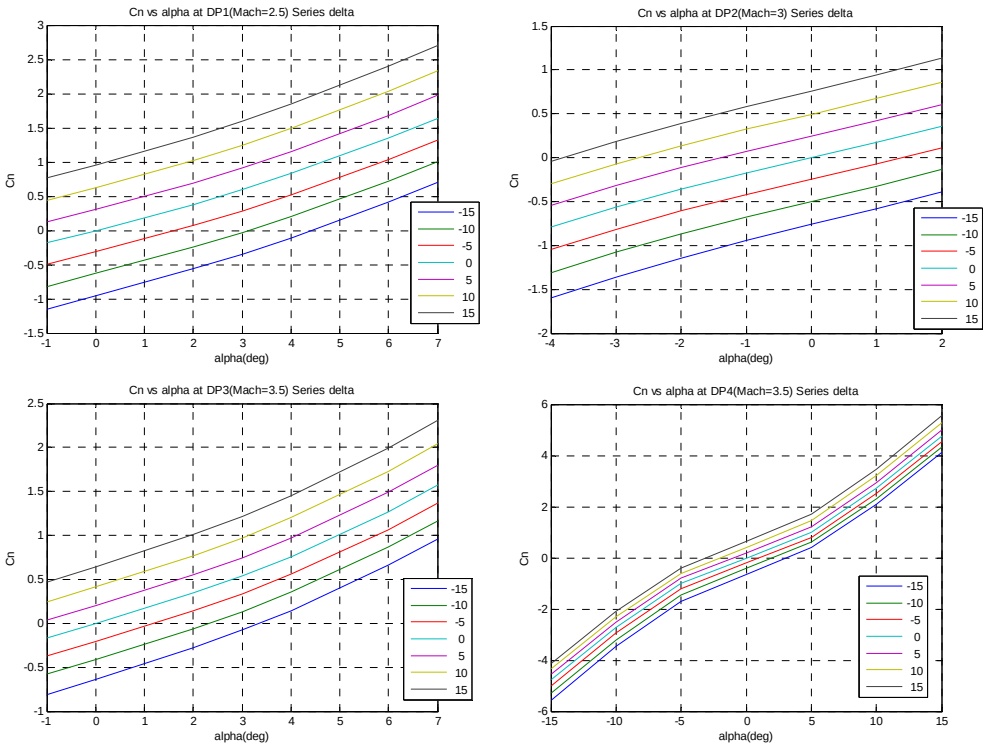


Figure A-1 C_n vs. α graphs at each DP (Series δ)

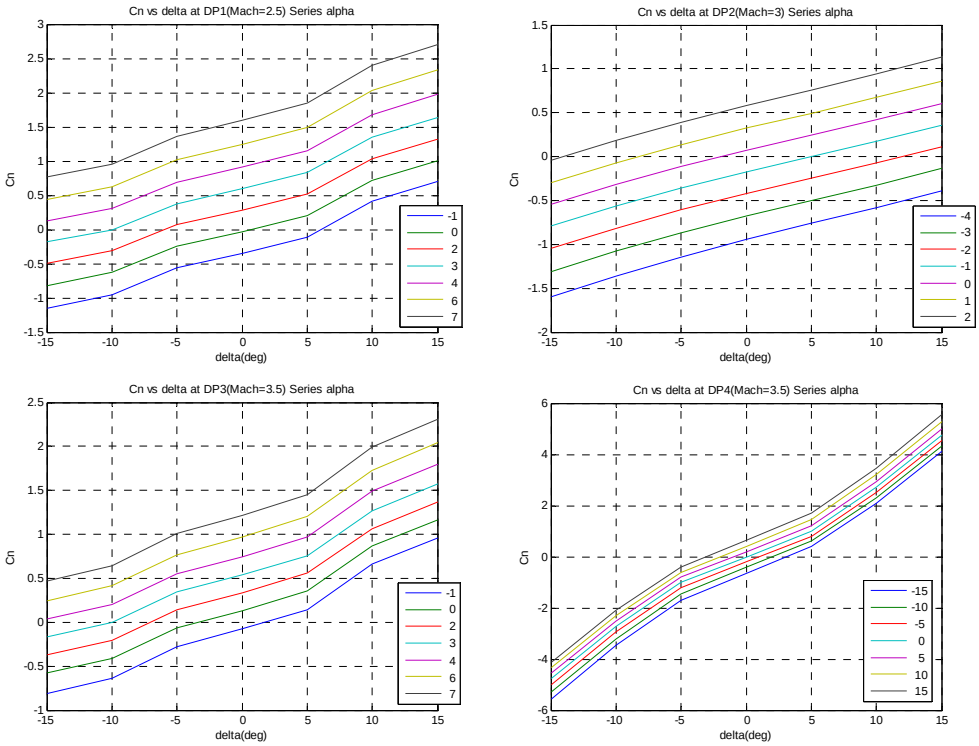


Figure A-2 C_n vs. δ graphs at each DP (Series α)

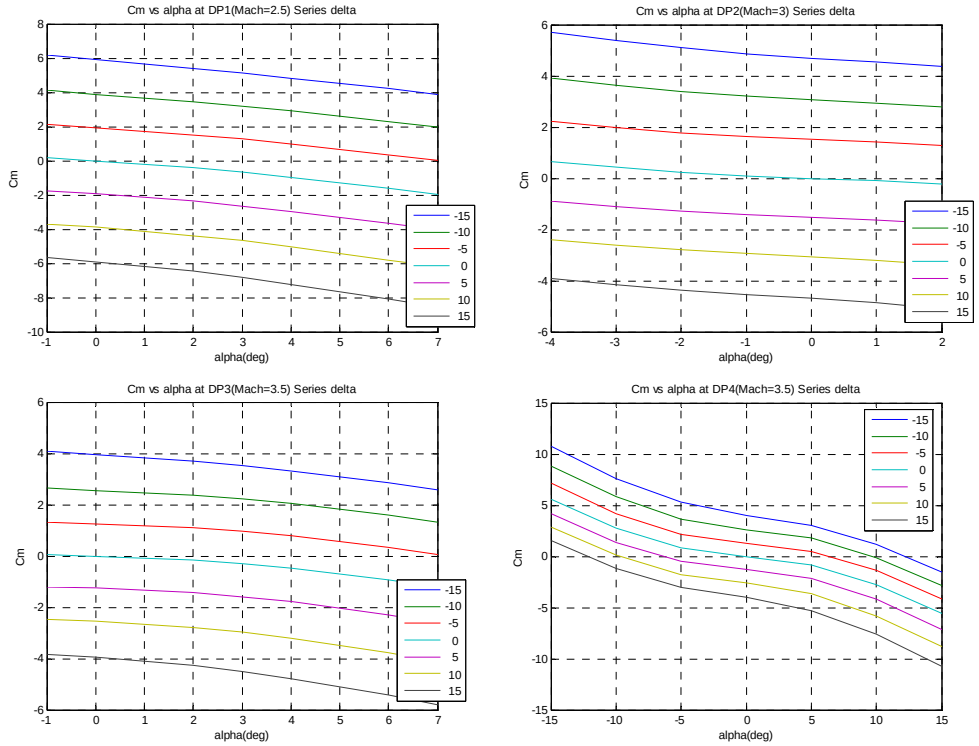


Figure A-3 C_m vs. α graphs at each DP (Series δ)

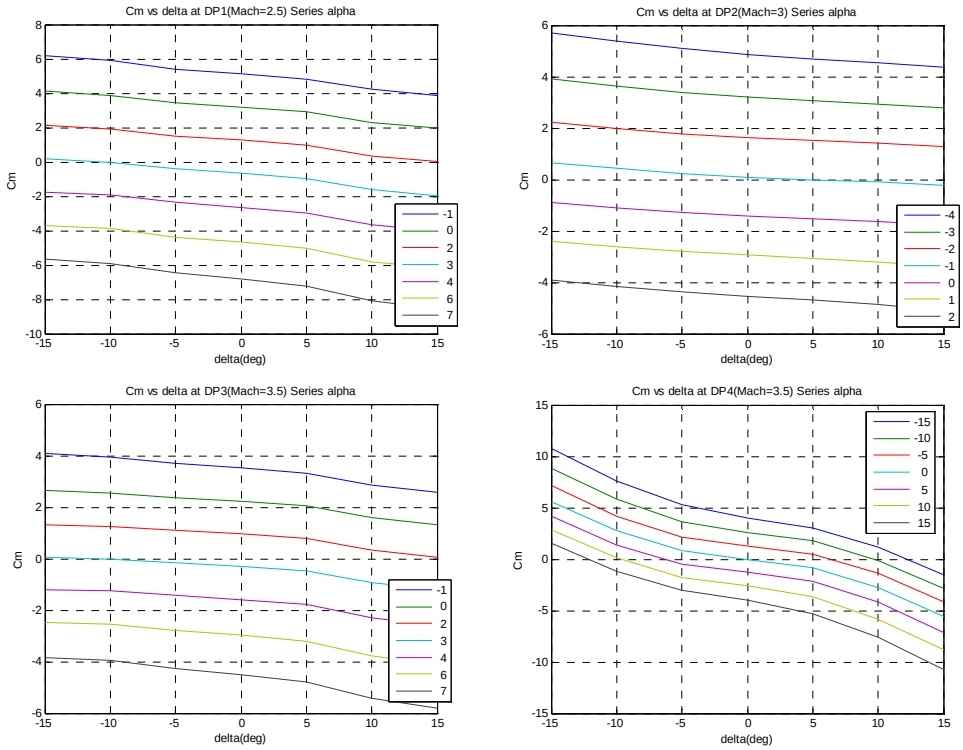


Figure A-4 C_m vs. δ graphs at each DP (Series α)

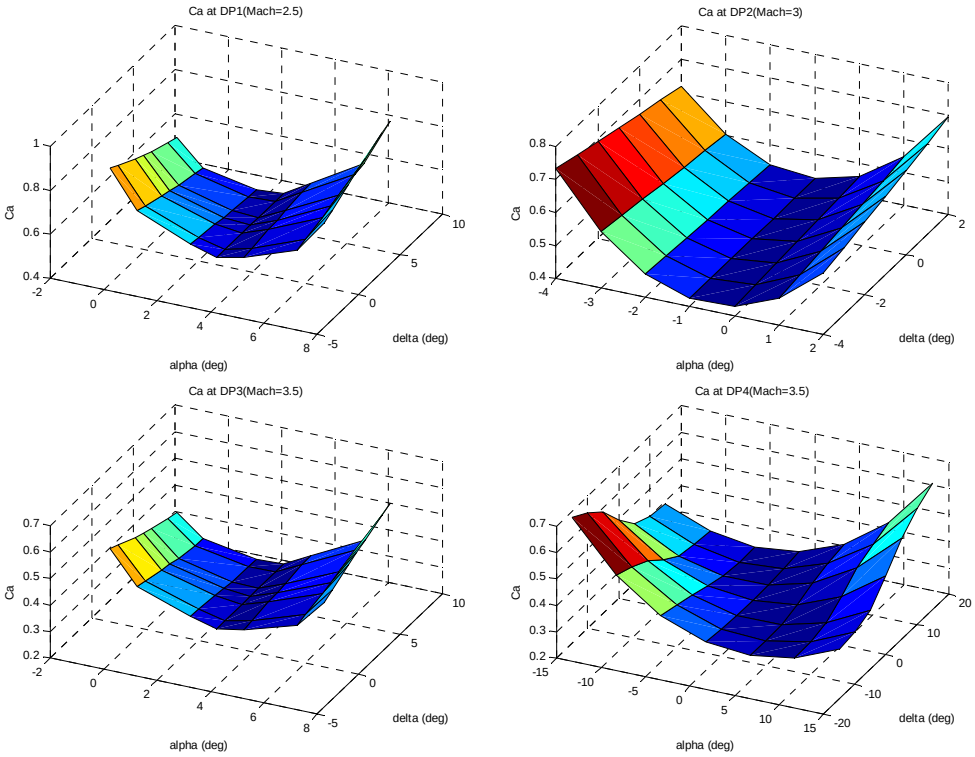


Figure A-5 C_a at each DP

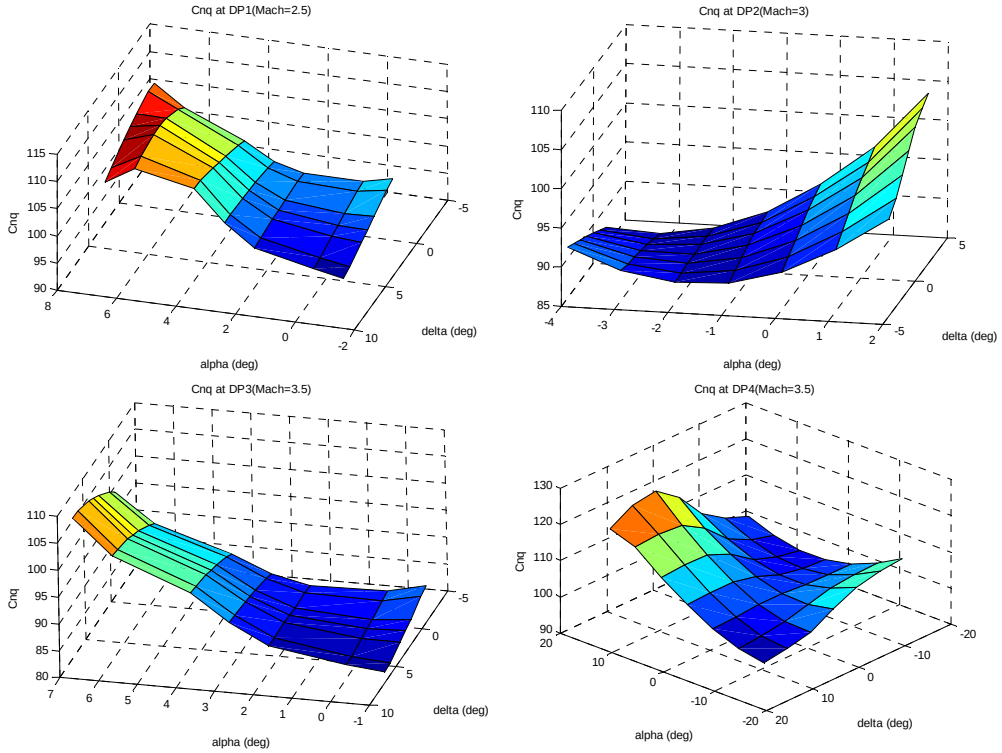


Figure A-6 C_{nq} at each DP

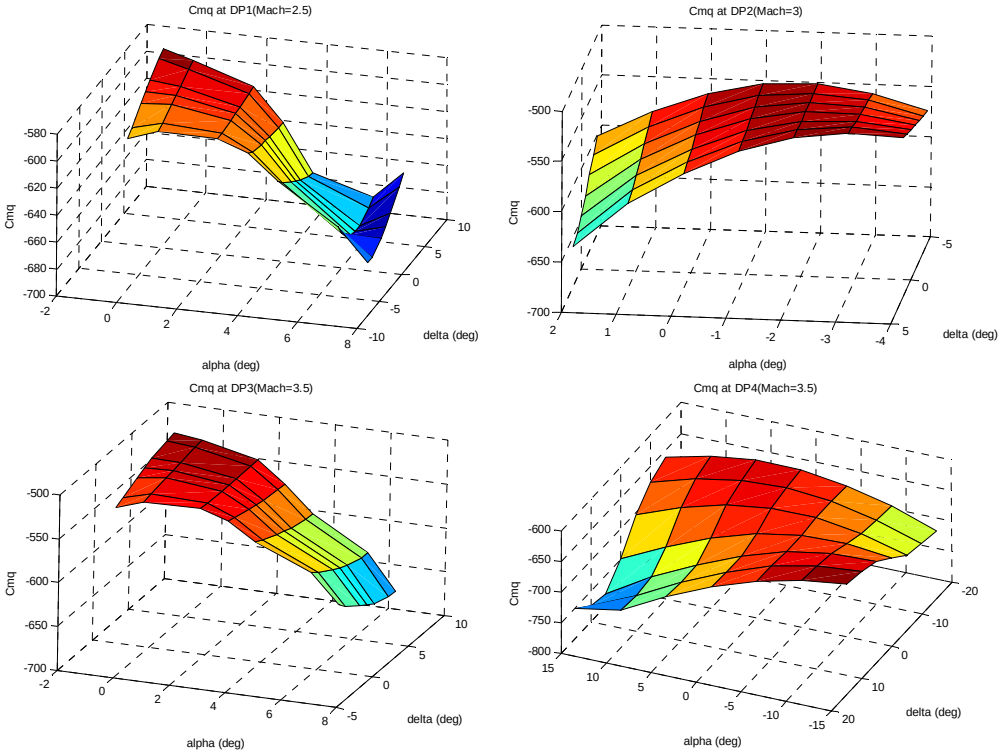


Figure A-7 C_{mq} at each DP

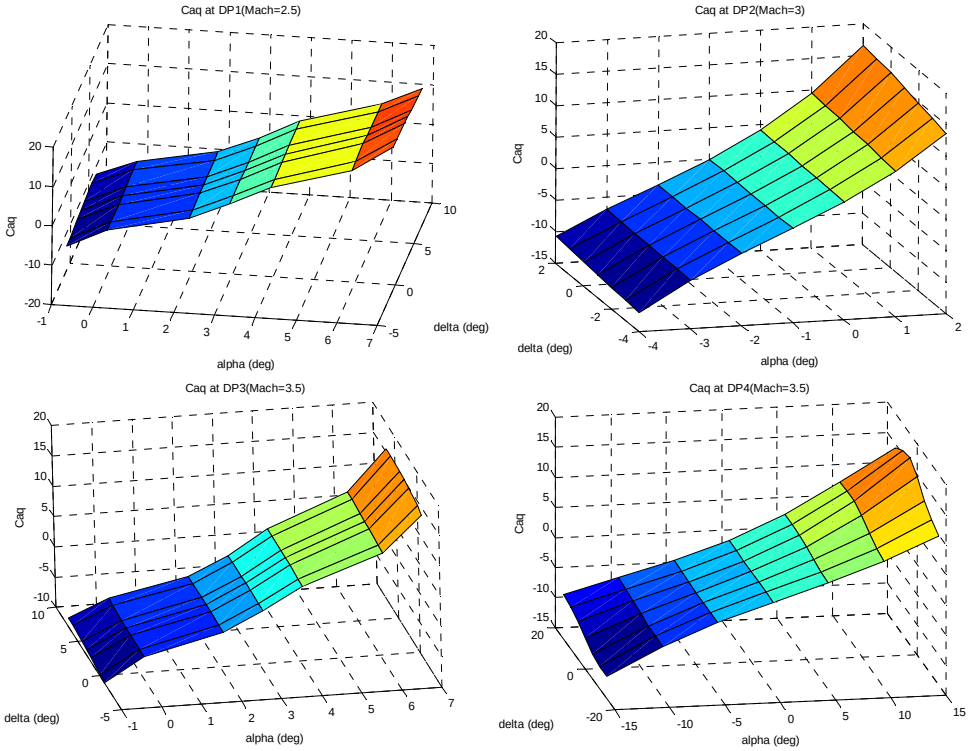


Figure A-8 C_{aq} at each DP

Note: The “q” derivatives are normalized and are in units of (1/rad)

APPENDIX B

OPEN-LOOP RESPONSES

This section involves the responses of missile transfer functions at each DP. Each graph shows the acceleration response of the missile when the control surface regarding the pitch axis is deflected by +1°.

Each graph involves responses of three systems.

1. The robust controller, which is obtained through linearization.
2. The classical controller.
3. Nonlinear simulation result

The graphs show that all three agree to a certain degree. The differences come from the modeling, especially from the polynomialization process in linearization. Therefore, the controllers will be designed for plants which are a bit different than assumed "the designer". Since the controllers are tested in a non-linear environment, it will be a robustness test for the controller.

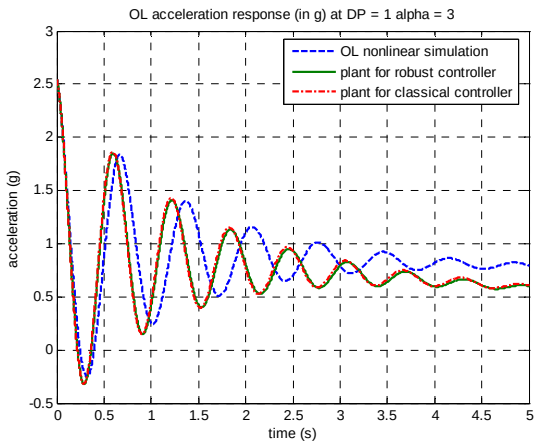


Figure B-1 OL Response for DP1 α = 3°

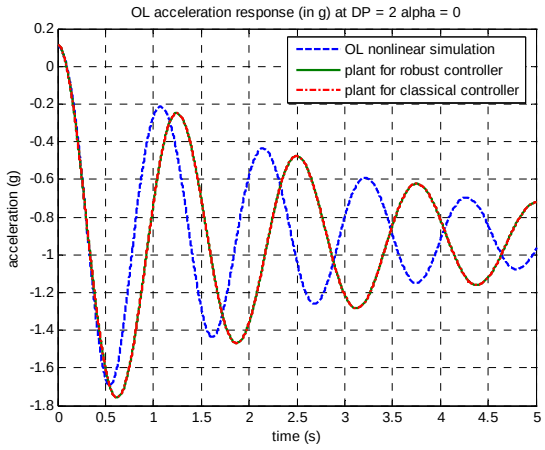


Figure B-2 Response for DP2 α = 0°

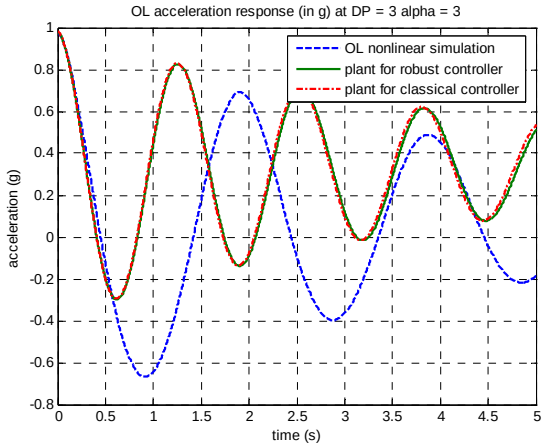


Figure B-3 Response for DP3 α = 3°

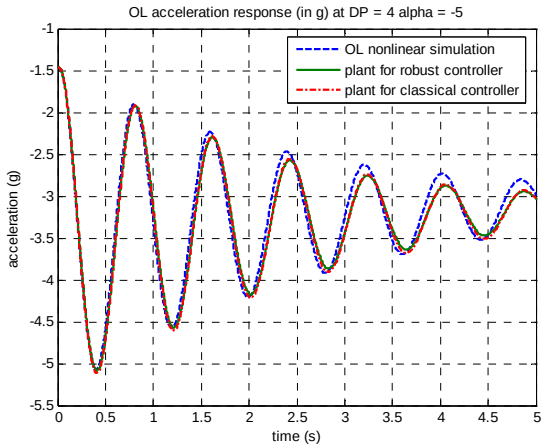


Figure B-4 Response for DP4 α = -5°

APPENDIX C

OBTAINING TRANSFER FUNCTIONS FOR THE CLASSICAL CONTROLLER

Obtaining transfer functions starts from (3.15) and (3.16). where the equations of motion were derived for the pitch plane. First, small angle for assumption for α has to be made

$$\alpha = \frac{w}{u} \qquad \text{(C-1)}$$

Time derivative of this expression gives

$$\dot{\alpha} = \frac{\dot{w}}{u} \qquad \text{(C-2)}$$

Note that short-period approximation is used. This is a valid assumption since controller dynamics are much faster than system dynamics in digital controllers.

Combining pitch axes equations from (3.15) and (C-2) gives

$$\dot{\alpha} = \frac{F_z}{m \cdot u} + q \qquad \text{(C-3)}$$

In these equations, F_z and M terms must be extracted. They can be written as linear combinations of their integrants as

$$F_z = Q \cdot S_{ref} \cdot (C_{z\alpha} \cdot \alpha + C_{z\delta} \cdot \delta_e + C_{zq} \cdot \frac{l_{ref}}{2 \cdot V} \cdot q) \qquad \text{(C-4)}$$

$$M = Q \cdot S_{ref} \cdot (C_{m\alpha} \cdot \alpha + C_{m\delta} \cdot \delta_e + C_{mq} \cdot \frac{l_{ref}}{2 \cdot V} \cdot q) \qquad \text{(C-5)}$$

Note that all coefficients in the equation above are functions of Mach number and local α only. This is the assumption of classical controller, which eliminates the linearization process.

With the following definitions,

$$Z_\alpha \overset{\Delta}{=} C_{z_\alpha} \frac{Q \cdot S_{ref}}{m \cdot V} \qquad M_\alpha \overset{\Delta}{=} C_{m_\alpha} \frac{Q \cdot S_{ref}}{I_{yy}} l_{ref}$$

$$Z_\delta \overset{\Delta}{=} C_{z_\delta} \frac{Q \cdot S_{ref}}{m \cdot V} \qquad M_\delta \overset{\Delta}{=} C_{m_\delta} \frac{Q \cdot S_{ref}}{I_{yy}} l_{ref} \qquad \text{(C-6)}$$

$$Z_q \overset{\Delta}{=} C_{z_q} \frac{l_{ref}}{2 \cdot V} \frac{Q \cdot S_{ref}}{m \cdot V} \qquad M_q \overset{\Delta}{=} C_{m_q} \frac{l_{ref}}{2 \cdot V} \frac{Q \cdot S_{ref}}{m \cdot V} l_{ref}$$

(C-3) and pitch moment equation $M = I_{yy} \cdot \dot{q}$ in (3.16) become

$$\dot{\alpha} = q + Z_\alpha \cdot \alpha + Z_\delta \cdot \delta + Z_q \cdot q \qquad \text{(C-7)}$$

$$\dot{q} = M_\alpha \cdot \alpha + M_\delta \cdot \delta + M_q \cdot q \qquad \text{(C-8)}$$

Now, the Laplace transform of these equations can be found as

$$\frac{\alpha(s)}{\delta(s)} = \frac{Z_\delta \cdot s + (M_\delta \cdot (Z_q + 1) - M_q \cdot Z_\delta)}{\Delta(s)} \qquad \text{(C-9)}$$

$$\frac{q(s)}{\delta(s)} = \frac{M_\delta \cdot s + (M_\alpha \cdot Z_\delta - M_\delta \cdot Z_\alpha)}{\Delta(s)} \qquad \text{(C-10)}$$

$$\Delta(s) = s^2 - (M_q + Z_\alpha) \cdot s + (M_q \cdot Z_\alpha - (M_\alpha \cdot (Z_q + 1))$$

Δ(s) is called the characteristic equation of the system

For the acceleration equation, (C-3) can be used; minding (F_z/m) term stands for acceleration, one can write

$$a_z = u \cdot (\dot{\alpha} - q) \qquad \text{(C-11)}$$

With the Laplace transform,

$$a_z(s) = u \cdot (s \cdot \alpha(s) - q(s)) \qquad \text{(C-12)}$$

Substituting (C-9) and (C-10) and also assuming $u \cong V$

$$\frac{a_z(s)}{\delta(s)} = V \cdot \frac{Z_\delta \cdot s^2 + (M_\delta \cdot Z_q - M_q \cdot Z_\delta) \cdot s + (Z_\alpha \cdot M_\delta - Z_\delta \cdot M_\alpha)}{\Delta(s)} \qquad \text{(C-13)}$$

(C-9), (C-10) and (C-13) are the equations that define the behavior of the system.

Note that autopilot coordinate system differs from the body axes. In order to make the transformation, following definitions are used:

$$\begin{aligned} Z_\alpha &\stackrel{\Delta}{=} -N_\alpha \\ Z_\delta &\stackrel{\Delta}{=} -N_\delta \\ Z_q &\stackrel{\Delta}{=} -N_q \\ a_z &\stackrel{\Delta}{=} -a_N \end{aligned} \qquad \text{(C-14)}$$

With these, equations that will be used for the autopilot design can be obtained:

$$\frac{q(s)}{\delta(s)} = \frac{M_\delta \cdot s + (M_\delta \cdot N_\alpha - M_\alpha \cdot N_\delta)}{s^2 + (N_\alpha - M_q)\cdot s - (M_q \cdot N_\alpha + M_\alpha \cdot (1 - N_q))} \qquad \text{(C-15)}$$

$$\frac{a_N(s)}{\delta(s)} = V \cdot \frac{N_\delta \cdot s^2 - (M_q \cdot N_\delta - M_\delta \cdot N_q)\cdot s + (N_\alpha \cdot M_\delta - N_\delta \cdot M_\alpha)}{s^2 + (N_\alpha - M_q)\cdot s - (M_q \cdot N_\alpha + M_\alpha \cdot (1 - N_q))} \qquad \text{(C-16)}$$

Note that the state-space representation can be given as

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -N_\alpha & 1 - N_q \\ M_\alpha & M_q \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} -N_\delta \\ M_\delta \end{bmatrix} \delta \qquad \text{(C-17)}$$

$$\begin{bmatrix} a_n \\ \alpha \\ q \end{bmatrix} = \begin{bmatrix} u \cdot N_\alpha & u \cdot N_q \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} u \cdot N_\delta \\ 0 \\ 0 \end{bmatrix} \delta$$

APPENDIX D

OBTAINING TRANSFER FUNCTIONS FOR THE ROBUST CONTROLLER

D.1 Theory

Given a non-linear function $\dot{x} = f(x,u)$, and the trim conditions $x = x_0$ and $u = u_0$, in the vicinity of trim conditions, the function can be expressed in linear form as

$$\dot{x} = f(x,u) = f(x_o,u_o) + \left.\frac{\partial f}{\partial x}\right|_{x_o,u_o}(x - x_o) + \left.\frac{\partial f}{\partial u}\right|_{x_o,u_o}(u - u_o) \qquad \text{(D-1)}$$

Writing $f(x_o,u_o) = \dot{x}_o$, the equation becomes

$$\dot{x} - \dot{x}_o = \left.\frac{\partial f}{\partial x}\right|_{x_o,u_o}(x - x_o) + \left.\frac{\partial f}{\partial u}\right|_{x_o,u_o}(u - u_o) \qquad \text{(D-2)}$$

This expression tells the change in $\dot{x} = f(x,u)$ is proportional to changes in x and u from the trim values.

Extending to a multivariable case and using the variables in missile control problem, one can write

$$\begin{bmatrix}\dot{\alpha}\\ \dot{q}\end{bmatrix} = \begin{bmatrix}\partial f_1/\partial\alpha & \partial f_1/\partial q\\ \partial f_2/\partial\alpha & \partial f_2/\partial q\end{bmatrix}\bullet\begin{bmatrix}\alpha\\ q\end{bmatrix} + \begin{bmatrix}\partial f_1/\partial\delta\\ \partial f_2/\partial\delta\end{bmatrix}\bullet\delta \qquad \text{(D-3)}$$

$$\begin{bmatrix}a_n\\ q\end{bmatrix} = \begin{bmatrix}\partial f_3/\partial\alpha & \partial f_3/\partial q\\ 0 & 1\end{bmatrix}\bullet\begin{bmatrix}\alpha\\ q\end{bmatrix} + \begin{bmatrix}\partial f_3/\partial\delta\\ 0\end{bmatrix}\bullet\delta \qquad \text{(D-4)}$$

with derivatives evaluated at trim values, δ_{trim}, q_{trim}.

Trim values for δ_{trim}, q_{trim} are to be found from the $f_i = 0$ equations which are the equations for $\dot{\alpha}, \dot{q}, a_n$ respectively.

D.2 Obtaining f_i

The coordinate system to be used is given in Figure D-1.

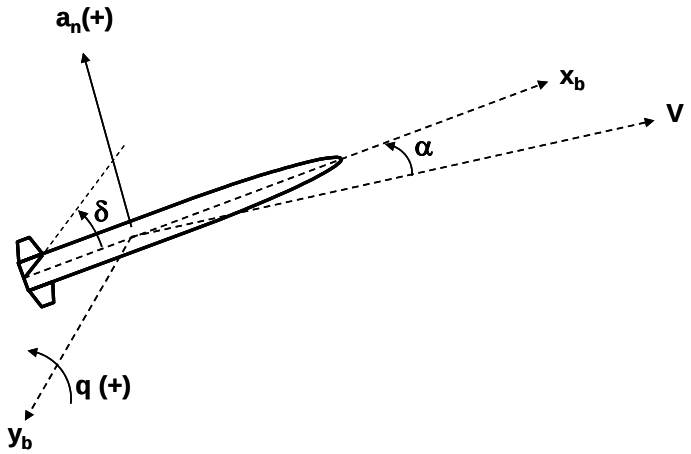


Figure D-1 Coordinate System for Transfer Function derivation

Equation of motion for total longitudinal acceleration in body axis

$$\dot{w} = \frac{F_z}{m} - q \cdot u \qquad \text{(D-5)}$$

In (D-5), F_z/m represents the acceleration felt by the accelerometers of the missile. Thus, F_z consists of aerodynamic forces[21].

Equation of motion for angular acceleration in body axis

$$f_2 : \dot{q} = \frac{M}{I_y} \qquad \text{(D-6)}$$

In (D-6), M represents the net moment felt by the gyros of the missile.

Equation for the angle of attack is as follows:

$$\tan(\alpha) = -\frac{w}{u} \qquad \text{(D-7)}$$

[21] It is assumed that no thrust misalignment causing a vertical acceleration exists. Also gravity is not taken into account as it can not be sensed by accelerometers.

Taking the time derivative of last equation w.r.t. time

$$\frac{\partial}{\partial t}\tan(\alpha)=\frac{\partial}{\partial t}(-w/u)\Rightarrow\dot{\alpha}=-\frac{\dot{w}}{u}\cdot\cos^2\alpha \quad [22] \qquad \text{(D-8)}$$

State equation for $\dot{\alpha}$(i.e., f_1) can be obtained by substituting the $\dot{w}$ term.

$$f_1:\dot{\alpha}=-\cos^2\alpha\cdot\left(\frac{F_z}{m\cdot u}-q\right)=\cos^2\alpha\cdot\left(q-\frac{F_z}{m\cdot u}\right) \qquad \text{(D-9)}$$

For the output equation, no additional calculations are needed since $a_n \triangleq F_z/m$ by definition, provided that the accelerometers are placed close to center of gravity of the missile.

$$f_3:a_n=F_z/m \qquad \text{(D-10)}$$

In these equations

$$\begin{aligned}F_z&=Q\cdot S_{ref}\cdot\left[C_n(M,\alpha,\delta)+C_{nq}(M,\alpha,\delta)\cdot\frac{l_{ref}}{2\cdot V}q\right]\\ M&=Q\cdot S_{ref}\cdot l_{ref}\cdot\left[C_m(M,\alpha,\delta)+C_{mq}(M,\alpha,\delta)\cdot\frac{l_{ref}}{2\cdot V}q\right]\end{aligned} \qquad \text{(D-11)}$$

It can be seen that aerodynamic coefficients have to be expressed in terms of their parameters in a non-linear fashion, which requires implementation of a surface fitting algorithm. The form usually being used is the polynomials. Such an algorithm, based on least-squares method has been developed; it expresses the non-linear aerodynamic data as explained in next section.

D.3 Expressing Aerodynamic Data as Polynomials

For an N[th] order fitting of a 3D matrix A its dimensions are based on reference vector α, δ and M

$$\begin{aligned}A(\alpha,M,\delta)&=a_N\alpha^N+a_{N-1}\alpha^{N-1}+\ldots+a_0\\ a_i&=b_N\delta^N+b_{N-1}\delta^{N-1}+\ldots+b_0\\ b_i&=c_NM^N+c_{N-1}M^{N-1}+\ldots+c_0\end{aligned} \qquad \text{(D-12)}$$

[22] Note that short period approximation is used. However, different from appendix C, small α assumption is not used.

This expression results in a (N+1)× (N+1)× (N+1) coefficient matrix

This method is extendable to hyper-surfaces of any dimension.

So, when such an approximation is used coefficient derivatives can easily be found as follows: (example is for α derivative)

$$\begin{aligned} A(\alpha, M, \delta) &= N a_N \alpha^{N-1} + a_{N-1}(N-1)\alpha^{N-2} + \ldots + a_1 \\ a_i &= b_N \delta^N + b_{N-1}\delta^{N-1} + \ldots + b_0 \\ b_i &= c_N M^N + c_{N-1} M^{N-1} + \ldots + c_0 \end{aligned} \qquad \text{(D-13)}$$

Code for implementing this algorithm is given in section D.6.1

This general form of surface fit will not be used during calculations. The coefficients will be formulized as polynomials that are functions of α and δ only. The reason underlying is that the accuracy of the method is dependent on the accuracy of Least Squares Method. As the number of dimensions increase (with not according increase in data points) results get worse. If the study above is inspected, it can be seen that no derivative w.r.t. Mach # exists. In all parameters it will be present as a coefficient, i.e. no derivative will be affected by the variance in Mach #. So in order to obtain better results, the parameters will be polynomialized as functions of parameters, which w.r.t the derivatives will be taken.

Code for obtaining parameters as functions of α and δ are given in D.6.2.

Similarly, C_{mq} and C_{nq} will be assumed to be functions of Mach # only, since no significant change with α and δ in these coefficients are expected.

Finally, f_i are obtained as follows

$$\begin{aligned} f_1 : \dot{\alpha} &= -\frac{\cos^2 \alpha}{m \cdot u} \cdot \left[Q \cdot S_{ref} \cdot C_n(\alpha, \delta) + q \cdot \left(C_{nq}(M) \cdot \frac{l_{ref}}{2 \cdot V} \cdot Q \cdot S_{ref} - m \cdot u \right) \right] \\ f_2 : \dot{q} &= \frac{Q \cdot S_{ref} \cdot l_{ref}}{I_y} \cdot \left[C_m(\alpha, \delta) + C_{mq}(M) \cdot \frac{l_{ref}}{2 \cdot V} q \right] \\ f_3 : a_n &= \frac{Q \cdot S_{ref}}{m} \cdot \left[C_n(\alpha, \delta) + C_{nq}(M) \cdot \frac{l_{ref}}{2 \cdot V} q \right] \end{aligned} \qquad \text{(D-14)}$$

where, for example for a 3rd order fitting,

$$\begin{aligned} C_i = {} & (b_{33} \cdot \delta^3 + b_{32} \cdot \delta^2 + b_{31} \cdot \delta + b_{30}) \cdot \alpha^3 + (b_{23} \cdot \delta^3 + b_{22} \cdot \delta^2 + b_{21} \cdot \delta + b_{20}) \cdot \alpha^2 \\ & + (b_{13} \cdot \delta^3 + b_{12} \cdot \delta^2 + b_{11} \cdot \delta + b_{10}) \cdot \alpha + (b_{03} \cdot \delta^3 + b_{02} \cdot \delta^2 + b_{01} \cdot \delta + b_{00}) \qquad \text{i} = \text{n, m} \end{aligned}$$

In this study parameters are fitted as second order polynomials. The reason is that; the higher the order, the harder the solutions of the non-linear equations in trim calculations get.

Linearization should be performed for each Mach #, which represents the design point that the controller design shall be performed for.

D.4 Calculating Trim Conditions

Trim conditions are ones in which $f_{1,2} = 0$ at a given Mach # and α. So at each design point the following set of equations are to be solved for δ_{trim} and q_{trim}.

$$0 = C_n(\alpha,\delta) + q \cdot \left(C_{nq}(M) \cdot \frac{l_{ref}}{2 \cdot V} - \frac{m \cdot u}{Q \cdot S_{ref}} \right)$$
$$0 = \left[C_m(\alpha,\delta) + q \cdot C_{mq}(M) \cdot \frac{l_{ref}}{2 \cdot V} \right] \qquad \text{(D-15)}$$

With these data trim points about the design points can be found as follows:

DP	δ_{trim} **(rad)**	q_{trim} **(rad/ s)**
1	-0.0295	0.0275
2	0.0011	-1.58e-4
3	-0.0194	0.0086
4	0.1074	-0.0237

D.5 Obtaining Linearized Equations

Only remaining step for a linearized system is to obtain $\partial f_i / \partial x_j$ (i = 1,2,3 j = α, q, δ) parameters, evaluated at trim values. With the definitions given in D.1, $\partial f_i / \partial x_j$ terms can be found. Analytical expressions for these parameters can be found in D.6.3. Numerical values are given below.

DP	$\partial f_1/\partial\alpha$	$\partial f_1/\partial q$	$\partial f_1/\partial\delta$	$\partial f_2/\partial\alpha$	$\partial f_2/\partial q$	$\partial f_2/\partial\delta$	$\partial f_3/\partial\alpha$	$\partial f_3/\partial q$	$\partial f_3/\partial\delta$
1	-0.728	0.996	-0.204	-102.7	-1.003	-152.8	585	1.07	164.7
2	-0.277	0.999	-0.078	-25.02	-0.428	-64.29	245.6	0.46	69.53
3	-0.213	0.997	-0.042	-24.07	-0.269	-40.3	221	0.29	43.67
4	-0.327	0.992	-0.046	-55.38	-0.361	-44.57	340.8	0.38	48.14

D.6 Supplementary Codes

D.6.1 Code for "Implementing Nth Order Fitting for an 3D Database" Algorithm

```
% program for 3D Nth order fitting

N = 3 ;              % this parameter can be changed for different accuracy

for i = 1:1:length(mach_vector)
    for j=1:1:length(delta_vector)
      fit(j,:) = polyfit(alpha_vector',Cn_matrix(:,i,j),N) ;
   end
   for k =1:1:N+1
      fit2(k,:) = polyfit(delta_vector',fit(:,k),N) ;
   end
   fit3(:,:,i) = fit2(:,:) ;
end

for i = 1:1:length(fit2)
   for j = 1:1:length(fit2)
      for k = 1:1:length(mach_vector)
      fitx(k,1) = fit3(i,j,k) ;
      end
      fit4(i,j,:) = polyfit(mach_vector',fitx,N) ;
   end
end

for i = 1:1:length(mach_vector)
   mach = mach_vector(i) ;
   for l = 1:1:N+1
      m_v(l,1) = mach^(N+1-l) ;
   end
   for m = 1:1:length(fit4)
      for n = 1:1:length(fit4)
         for o = 1:1:N+1
            fit6(1,o) = fit4(m,n,o) ;
         end
```

```
        fit5(m,n) = fit6*m_v ;
      end
   end

   for j = 1:1:length(alpha_vector)
      for k = 1:1:length(delta_vector)
         alpha = alpha_vector(j) ;
         delta = delta_vector(k) ;
         for p = 1:1:N+1
            a_v(p,1) = alpha^(N+1-p) ;
            d_v(p,1) = delta^(N+1-p) ;
         end
         results (j,i,k) = (fit5*d_v)'*a_v ;
      end
   end
end
```

The output matrix "results" is the matrix of coefficients of the polynomial.

D.6.2 Code for Simplified Algorithm

```
% program for 2D Nth order fitting

for i=1:1:length(delta_vector)
   fit(i,:) = polyfit(alpha_vector',data_matrix(:,i),N) ;
end

for j =1:1:N+1
   fit2(j,:) = polyfit(delta_vector',fit(:,j),N) ;
end
```

D.6.3 Analytical Derivatives for State Equations

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} \delta$$

$$\begin{bmatrix} a_n \\ \alpha \\ q \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} D \\ 0 \\ 0 \end{bmatrix} \delta \qquad \text{(D-16)}$$

Parameters in this expression are defined as follows:

$$A_{11}= 2\cdot\cos(\alpha)\cdot\left\{\begin{array}{l}Q\cdot S\cdot\left[\begin{array}{l}\left(C_{n_{b22}}\cdot{\delta_{trim}}^{2}+C_{n_{b21}}\cdot\delta_{trim}+C_{n_{b20}}\right)\cdot\alpha^{2}\\+\left(C_{n_{b12}}\cdot{\delta_{trim}}^{2}+C_{n_{b11}}\cdot\delta_{trim}+C_{n_{b10}}\right)\cdot\alpha\\+\left(C_{n_{b02}}\cdot{\delta_{trim}}^{2}+C_{n_{b01}}\cdot\delta_{trim}+C_{n_{b00}}\right)\end{array}\right]\\+q_{trim}\cdot\left(\frac{1}{2}\cdot\frac{C_{n_q}\cdot l_{ref}\cdot Q\cdot S}{u}-m\cdot u\right)\end{array}\right\}\cdot\frac{\sin(\alpha)}{m\cdot u}$$

$$-\cos^{2}(\alpha)\cdot Q\cdot S\cdot\left(\begin{array}{l}2\cdot\left(C_{n_{b22}}\cdot{\delta_{trim}}^{2}+C_{n_{b21}}\cdot\delta_{trim}+C_{n_{b20}}\right)\cdot\alpha\\+\left(C_{n_{b12}}\cdot{\delta_{trim}}^{2}+C_{n_{b11}}\cdot\delta_{trim}+C_{n_{b10}}\right)\end{array}\right)\cdot\frac{1}{m\cdot u}$$

$$A_{12}=-\cos^{2}(\alpha)\cdot\left(\frac{1}{2}\cdot\frac{C_{n_q}\cdot l_{ref}\cdot Q\cdot S}{u}-m\cdot u\right)\cdot\frac{1}{m\cdot u}$$

$$B_{1}=-\cos^{2}(\alpha)\cdot Q\cdot S\cdot\left[\begin{array}{l}\left(2\cdot C_{n_{b22}}\cdot\delta_{trim}+C_{n_{b21}}\right)\cdot\alpha^{2}\\+\left(2\cdot C_{n_{b12}}\cdot\delta_{trim}+C_{n_{b11}}\right)\cdot\alpha\\+2\cdot C_{n_{b02}}\cdot\delta_{trim}+C_{n_{b01}}\end{array}\right]\cdot\frac{1}{m\cdot u}$$

$$A_{21}=Q\cdot S\cdot l_{ref}\cdot\left(\begin{array}{l}2\cdot\left(C_{m_{b22}}\cdot{\delta_{trim}}^{2}+C_{m_{b21}}\cdot\delta_{trim}+C_{m_{b20}}\right)\cdot\alpha\\+\left(C_{m_{b12}}\cdot{\delta_{trim}}^{2}+C_{m_{b11}}\cdot\delta_{trim}+C_{m_{b10}}\right)\end{array}\right)\cdot\frac{1}{J}$$

$$A_{22}=\frac{1}{2}\cdot\frac{C_{m_q}\cdot{l_{ref}}^{2}\cdot Q\cdot S}{J\cdot u}$$

$$B_{2}=Q\cdot S\cdot l_{ref}\cdot\left[\begin{array}{l}\left(2\cdot C_{m_{b22}}\cdot\delta_{trim}+C_{m_{b21}}\right)\cdot\alpha^{2}\\+\left(2\cdot C_{m_{b12}}\cdot\delta_{trim}+C_{m_{b11}}\right)\cdot\alpha\\+2\cdot C_{m_{b02}}\cdot\delta_{trim}+C_{m_{b01}}\end{array}\right]\cdot\frac{1}{J}$$

$$C_{11}=\left[\begin{array}{l}2\cdot\left(C_{n_{b22}}\cdot{\delta_{trim}}^{2}+C_{n_{b21}}\cdot\delta_{trim}+C_{n_{b20}}\right)\cdot\alpha\\+\left(C_{n_{b12}}\cdot{\delta_{trim}}^{2}+C_{n_{b11}}\cdot\delta_{trim}+C_{n_{b10}}\right)\end{array}\right]\cdot\frac{1}{m}$$

$$C_{12}=\frac{1}{2}\cdot\frac{C_{n_q}\cdot l_{ref}\cdot Q\cdot S}{m\cdot u}$$

$$D=Q\cdot S\cdot\left[\begin{array}{l}\left(2\cdot C_{n_{b22}}\cdot\delta_{trim}+C_{n_{b21}}\right)\cdot\alpha^{2}\\+\left(2\cdot C_{n_{b12}}\cdot\delta_{trim}+C_{n_{b11}}\right)\cdot\alpha\\+2\cdot C_{n_{b02}}\cdot\delta_{trim}+C_{n_{b01}}\end{array}\right]\cdot\frac{1}{m}$$

Finally, including the actuator dynamics to (D-16), state space representation of the plant consisting of the missile and actuator dynamic can be given as follows:

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \\ \dot{\delta} \\ \ddot{\delta} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & B_{11} & 0 \\ A_{21} & A_{22} & B_{21} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & A_{21_{CAS}} & A_{22_{CAS}} \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \delta \\ \dot{\delta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ B_{21_{CAS}} \end{bmatrix} u$$

$$\begin{bmatrix} a_n \\ \alpha \\ q \\ \delta \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & D_{11} & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ q \\ \delta \\ \dot{\delta} \end{bmatrix} \qquad \text{(D-17)}$$

where u is the deflection command of the autopilot to the actuator, and

$$A_{21_{CAS}} = -\omega_{n_{CAS}}{}^2$$

$$A_{22_{CAS}} = 2\zeta_{CAS}\omega_{n_{CAS}}$$

$$B_{21_{CAS}} = \omega_{n_{CAS}}{}^2$$

APPENDIX E

CODE FOR ROBUST CONTROLLER DESIGN

The code for robust controller design is given below. The execution file is "main.m" which is given in section E.1 and it calls some subroutines which are given in section E.2 to E.8. The code is compatible with MATLAB (v7.01) software.

E.1 Main Function "main.m"

```
% this is the top level function for RCT tasks. it
% - calls "\robust_initialize.m" for initialization of parameters
% - calls "\robust_synthesize.m" for controller synthesis
% - calls "\linear_simulation.m" for generating linear closed loop responses
% - calls "\rp_analysis.m" for mu anaysis

%%
clear all; close all; clc; commandwindow ;
tic

%% design condition (USER EDITABLE FIELD)

DP = 4 ;
alpha = -5*pi/180 ;      %(rad)
disp(' ') ; disp(['task started for DP = ' num2str(DP) ' alpha = ' num2str(alpha*180/pi) ' deg'])

%% initialize DP data (for flight conditions) and system components

freq_of_interest = 20 ;     % the frequency modelling is accurate up to (rad/s)
[plant,cas,rgu,acc,delta_trim,q_trim,W,cas_rate_limit,cas_pos_limit,wn_cas,dr_cas,rgu_meas_l
imit,acc_meas_limit] = robust_initialize(DP,alpha,freq_of_interest) ;

%% additional assignments

dacc = 1 ;                      % design acc
drgu = 1 ;                      % design rgu
nrgu = rgu.nominal ;                % nominal rgu
```

```
nacc = acc.nominal ;                % nominal acc
nplant = plant.nominal ;            % nominal plant
ncas = cas.nominal ;                % nominal cas

%% synthesize the controller

[k,k_red,per_weight] =
robust_synthesize(freq_of_interest,plant,cas,drgu,dacc,W,cas_rate_limit,cas_pos_limit,wn_cas,
dr_cas,rgu_meas_limit,acc_meas_limit) ;

%% create test inputs and simulate time response of nominal CL system

[nom_CL_system,nom_red_CL_system,lsim_reg,lsim_red,lsim_time,lsim_an_com] =
linear_simulation(ncas,nplant,nrgu,nacc,k,k_red,DP) ;

%% test for robust stability
disp(' ') ; disp('...checking for robust stability')

%% test for robust performance
disp(' ') ; disp('...checking for robust performance')

[perfmarg,perfmargunc,perf_report,perf_info,sysrp] =
rp_analysis(plant,rgu,acc,cas,k_red,per_weight,DP) ;

%%
disp(' ') ; disp(['total process time is ' num2str(toc) ' seconds'])

save (['current_design_data_DP' num2str(DP)])
```

E.2 Sub-function "robust_synthesize.m"

```
function [k,k_red,per_weight] =
robust_synthesize(freq_of_interest,plant,cas,rgu,acc,W,cas_rate_limit,cas_pos_limit,wn_cas,dr
_cas,rgu_meas_limit,acc_meas_limit)
% this is the code for robust controller synthesis.
% it synthesis the regular and reduced order controllers

disp(' ') ; disp(['controller synthesis started (elapsed time: ' num2str(toc) ' seconds)' ])

%% creating sysic components
disp(' ') ; disp('...defining necessary functions')

A11 = plant.a(1,1).nominal ;
A12 = plant.a(1,2).nominal ;
A21 = plant.a(2,1).nominal ;
A22 = plant.a(2,2).nominal ;
B11 = plant.b(1,1).nominal ;
B21 = plant.b(2,1).nominal ;
C11 = plant.c(1,1).nominal ;
C12 = plant.c(1,2).nominal ;
D11 = plant.d(1,1).nominal ;

cas_A21 = cas.a(2,1).nominal ;
cas_A22 = cas.a(2,2).nominal ;
```

```
cas_B21 = cas.b(2,1).nominal ;

W_A11 = W.W1 ;
W_A12 = W.W2 ;
W_A21 = W.W3 ;
W_A22 = W.W4 ;
W_B11 = W.W5 ;
W_B21 = W.W6 ;
W_C11 = W.W7 ;
W_C12 = W.W8 ;
W_D11 = W.W9 ;
W_cas_A21 = W.W10 ;
W_cas_A22 = W.W11 ;
W_cas_B21 = W.W12 ;

integ1 = tf([1],[1 0]) ;
integ2 = tf([1],[1 0]) ;
integ3 = tf([1],[1 0]) ;
integ4 = tf([1],[1 0]) ;

%% define performance weights

df = freq_of_interest/2 ;                    % frequency up to where performance is needed

W_cmd = 1*makeweight(1.01,df/2,0.01) ;
W_an_per = 0.5*makeweight(100,df,0.01) ;                    % acceleration performance weight
W_alpha_per = 0.1*makeweight(1.01,df,0.01)/(15*pi/180) ;% angle of attack performance
weight
W_act_sper = 2*makeweight(0.01,2*wn_cas,1.01)/cas_rate_limit ; % deflection rate penalty
weight
W_act_pper = 1*makeweight(0.01,2*wn_cas,1.01)/cas_pos_limit ; % deflection penalty
weight
W_acc_noise = 0.01*makeweight(0.01,10*2*pi*2,1.01) ;          % accelerometer noise (g)
W_rgu_noise = 0.01*makeweight(0.01,45*2*pi*2,1.01) ;          % rgu noise (rad/s)

% define performance weight function for outputting purposes

per_weight.W_cmd = W_cmd ;
per_weight.W_an_per = W_an_per ;
per_weight.W_alpha_per = W_alpha_per ;
per_weight.W_act_sper = W_act_sper ;
per_weight.W_act_pper = W_act_pper ;
per_weight.W_acc_noise = W_acc_noise ;
per_weight.W_rgu_noise = W_rgu_noise ;

%% construct system structure
disp(' ') ; disp('...constructing system structure for controller design')

systemnames = ' A11 A12 A21 A22 B11 B21 C11 C12 D11 cas_A21 cas_A22 cas_B21 integ1
integ2 integ3 integ4 rgu acc ' ;      % system components
systemnames = [systemnames, ' W_cmd W_acc_noise W_rgu_noise W_act_pper W_act_sper
W_alpha_per W_an_per ' ] ;                % performance weights
systemnames = [systemnames, ' W_A11 W_A12 W_A21 W_A22 W_B11 W_B21 W_C11
W_C12 W_D11 W_cas_A21 W_cas_A22 W_cas_B21 ' ] ;  % Uncertainty Weights

inputvar = ' [ noise_rgu ; noise_acc ; '  ;   % noise inputs
```

```
inputvar = [ inputvar, ' w1 ; w2 ; w3 ; w4 ; w5 ; w6 ; w7 ; w8 ; w9 ; w10 ; w11 ; w12 ; ' ] ; % uncertainty inputs
inputvar = [ inputvar, ' an_com ; u ] ' ] ; % reference controller commands

outputvar = ' [ W_act_pper ; W_act_sper ; W_alpha_per ; W_an_per ; ' ; % performance weights
outputvar = [ outputvar, ' W_A11 ; W_A12 ; W_A21 ; W_A22 ; W_B11 ; W_B21 ; W_C11 ; W_C12 ; W_D11 ; W_cas_A21 ; W_cas_A22 ; W_cas_B21 ; ' ] ;  % uncertainties
outputvar = [ outputvar, ' an_com ; acc + W_acc_noise ; rgu + W_rgu_noise ] ' ] ; % controller inputs

input_to_A11 = ' [ w1 + integ1 ] ' ;
input_to_A12 = ' [ w2 + integ2 ] ' ;
input_to_A21 = ' [ w3 + integ1 ] ' ;
input_to_A22 = ' [ w4 + integ2 ] ' ;
input_to_B11 = ' [ w5 + integ3 ] ' ;
input_to_B21 = ' [ w6 + integ3 ] ' ;
input_to_C11 = ' [ w7 + integ1 ] ' ;
input_to_C12 = ' [ w8 + integ2 ] ' ;
input_to_D11 = ' [ w9 + integ3 ] ' ;
input_to_cas_A21 = ' [ w10 + integ3 ] ' ;
input_to_cas_A22 = ' [ w11 + integ4 ] ' ;
input_to_cas_B21  = ' [ w12 + u ] ' ;
input_to_integ1 = ' [ A11 + A12 + B11 ] ' ;
input_to_integ2 = ' [ A21 + A22 + B21 ] ' ;
input_to_integ3 = ' [ integ4 ] ' ;
input_to_integ4 = ' [ cas_A21 + cas_A22 + cas_B21 ] ' ;
input_to_rgu  = ' [ integ2 ] ' ;
input_to_acc = ' [ C11 + C12 + D11 ] ' ;
input_to_W_cmd = ' [ an_com ] ' ;
input_to_W_acc_noise = ' [ noise_acc ] ' ;
input_to_W_rgu_noise = ' [ noise_rgu ] ' ;
input_to_W_act_pper = ' [ integ3 ] ' ;
input_to_W_act_sper = ' [ integ4 ] ' ;
input_to_W_alpha_per = ' [ integ1 ] ' ;
input_to_W_an_per = ' [ W_cmd - acc ] ' ;
input_to_W_A11 = ' [ integ1 ] ' ;
input_to_W_A12 = ' [ integ2 ] ' ;
input_to_W_A21 = ' [ integ1 ] ' ;
input_to_W_A22 = ' [ integ2 ] ' ;
input_to_W_B11 = ' [ integ3 ] ' ;
input_to_W_B21 = ' [ integ3 ] ' ;
input_to_W_C11 = ' [ integ1 ] ' ;
input_to_W_C12 = ' [ integ2 ] ' ;
input_to_W_D11 = ' [ integ3 ] ' ;
input_to_W_cas_A21 = ' [ integ3 ] ' ;
input_to_W_cas_A22 = ' [ integ4 ] ' ;
input_to_W_cas_B21 = ' [ u ] ' ;

cleanupsysic = 'yes' ;

P = sysic ;

%% Hinf controller synthesis
disp(' ') ; disp('...synthesizing the controller')

nmeas = 3 ;            % # of measurement for 2DOF controller
```

```
ncon = 1 ;              % # of control inputs

[k,cl,gam,info] = hinfsyn(P,nmeas,ncon) ;          % synthesize the controller

disp(' ')
disp(['......synthesis completed at ' num2str(toc) ' seconds'])
disp(['......gama value achieved = ' num2str(gam) ])
disp(['......controller is of order ' num2str(order(k))]) ; clear info gam

%% obtain reduced controller
disp(' ') ; disp('...obtaining reduced order controller')

for cont_index = 1:1:order(k)
[k_red,info] = reduce(k,cont_index) ;                          % reduce the controller
if info.ErrorBound < 1e-3
   break
end
end

disp(' ') ; disp(['......reduced controller is of order ' num2str(order(k_red))])
disp(['......error in approximation is ' num2str(info.ErrorBound)]) ; clear info

%%
disp(' ') ; disp(['controller synthesis finished (elapsed time: ' num2str(toc) ' seconds)'])
```

E.3 Sub-function “robust_initialize.m”

```
function [plant,cas,rgu,acc,dt,qt,W,...
        cas_rate_limit,cas_pos_limit,wn_cas,dr_cas,...
        rgu_meas_limit,acc_meas_limit] = robust_initialize(DP,alpha,freq_of_interest)

% this code initializes the parameters for RCT tasks and creates the system components for
robust controller design
% it calls the pre_initialization codes
% - "\data\initialize_DP_data.m"
% - "\data\initializa_cas_data.m"
% - "\data\initializa_imu_data.m"
% and intermediate processing codes
% - "\robust\polynomialize.m"
% - "\robust\linearize.m"
% - "\robust\construct_components.m"

disp(' ') ; disp(['robust_initialize.m is running for initializing data (elapsed time: ' num2str(toc) '
seconds)'])

%% reach to the data to be processed
disp(' ') ; disp('...getting data')
cd ..
cd data

%% get data
%cas data
[cas_rate_limit,cas_pos_limit,wn_cas,dr_cas] = initialize_cas_data ;
%imu data
```

```
[wn_rgu,dr_rgu,rgu_meas_limit,wn_acc,dr_acc,acc_meas_limit] = initialize_imu_data ;
% DP data
[alpha_vector,mach_vector,delta_vector,design_alt,sos,design_mach,u0,rho,T,m,J,Q,...
        Cn_matrix,Cm_matrix,Ca_matrix,Cnq_matrix,Cmq_matrix,Caq_matrix,...
        l_ref,S] = initialize_DP_data(DP) ;
% delete unnecessary data
clear Ca_matrix Caq_matrix

%% return back to base path
cd ..
cd robust

%% process aerodynamic data (transform aerodynamic data from 3d to 2d and 0d)
disp(' ') ; disp('...pre-processing aerodynamic data')

% find necessary indices
mi = find(mach_vector == design_mach) ;      % find the indice of DP mach number in
reference mach vector
di = find(delta_vector == 0) ;         % find the indice of zero delta in reference delta vector
ai = find(alpha_vector == 0) ;        % find the indice of zero alphak in reference alpha vector

% error checking
if size(mi)>1 | size(di)>1 | size(ai)>1
   error('invalid reference vectors')
end

% transformation
Cn_matrix_m(:,:) = Cn_matrix(:,mi,:) ;       % obtain Cn(alpha,delta) matrix at DP
Cm_matrix_m(:,:) = Cm_matrix(:,mi,:) ;       % obtain Cm(alpha,delta) matrix at DP
Cnq = Cnq_matrix(ai,mi,di) ;                % obtain Cnq at DP, at zero alpha and delta
Cmq = Cmq_matrix(ai,mi,di) ;                % obtain Cmq at DP, at zero alpha and delta
clear mi di ai Cn_matrix Cm_matrix Cnq_matrix Cmq_matrix

%% fit polynomials to aerodynamic data
disp(' ') ; disp('...polynomializing aerodynamic coefficients')

% "N" being the order of the fitting, code polynomializeN.m must be called for correct
solutions
% (default) order used in this study is 2.
% note that inputs are altered for unit conversion since data matrices "Cn_matrix_m" and
"Cm_matrix_m"
% are in "radians" in terms reference vectors "alpha_vector" and "delta_vector"

[Cn,Cm] =
polynomialize(alpha_vector*pi/180,delta_vector*pi/180,Cn_matrix_m,Cm_matrix_m) ;
clear Cn_matrix_m Cm_matrix_m

%% get trim conditions
disp(' ') ; disp('...obtaining trim conditions')
[df1,df2,df3,dt,qt] = linearize(alpha,Cn,Cm,Cnq,Cmq,m,u0,Q,S,l_ref) ;

%% create system components with uncertainties
disp(' ') ; disp(['...creating system components with uncertainties (elapsed time: ' num2str(toc) '
seconds)'])
[plant,cas,rgu,acc,W] = construct_components(freq_of_interest,alpha,Cn,Cm,Cnq,Cmq,dt,qt,...
                                    df1,df2,df3,m,J,u0,Q,S,l_ref,...
                                    wn_cas,dr_cas,...
```

```
                                wn_rgu,dr_rgu,wn_acc,dr_acc) ;

%%
disp(' ') ; disp(['job finished for robust_initialize.m (elapsed time: ' num2str(toc) ' seconds)'])
```

E.4 Sub-function "construct_components.m"

```
function [plant,cas,rgu,acc,W] =
construct_components3(freq_of_interest,alpha,Cn,Cm,Cnq,Cmq,dt,qt,...
                                        df1,df2,df3,m,J,u0,Q,S,l_ref,...
                                        wn_cas,dr_cas,...
                                        wn_rgu,dr_rgu,wn_acc,dr_acc)

% this code defines the uncertainty in system components

s_s = 1000 ;                            % sample size for unceratainty creation

%% define uncertainty on plant
disp(' ') ; disp('.....defining plant')

% define parametric uncertainties on smallest elements

m = ureal('m',m,'percentage',.1) ;
J = ureal('J',J,'percentage',.1) ;

Cnq = ureal('Cnq',Cnq,'percentage',25) ;
Cmq = ureal('Cmq',Cmq,'percentage',25) ;

Cn_b22 = Cn.Cn_b22 ; if abs(Cn_b22) <= 0.00001 ; Cn_b22 = 0 ; else Cn_b22 =
ureal('Cn_b22',Cn_b22,'percentage',0.5) ; end ;
Cn_b21 = Cn.Cn_b21 ; if abs(Cn_b21) <= 0.00001 ; Cn_b21 = 0 ; else Cn_b21 =
ureal('Cn_b21',Cn_b21,'percentage',1.0) ; end ;
Cn_b20 = Cn.Cn_b20 ; if abs(Cn_b20) <= 0.00001 ; Cn_b20 = 0 ; else Cn_b20 =
ureal('Cn_b20',Cn_b20,'percentage',0.5) ; end ;
Cn_b12 = Cn.Cn_b12 ; if abs(Cn_b12) <= 0.00001 ; Cn_b12 = 0 ; else Cn_b12 =
ureal('Cn_b12',Cn_b12,'percentage',1.0) ; end ;
Cn_b11 = Cn.Cn_b11 ; if abs(Cn_b11) <= 0.00001 ; Cn_b11 = 0 ; else Cn_b11 =
ureal('Cn_b11',Cn_b11,'percentage',0.5) ; end ;
Cn_b10 = Cn.Cn_b10 ; if abs(Cn_b10) <= 0.00001 ; Cn_b10 = 0 ; else Cn_b10 =
ureal('Cn_b10',Cn_b10,'percentage',1.0) ; end ;
Cn_b02 = Cn.Cn_b02 ; if abs(Cn_b02) <= 0.00001 ; Cn_b02 = 0 ; else Cn_b02 =
ureal('Cn_b02',Cn_b02,'percentage',0.5) ; end ;
Cn_b01 = Cn.Cn_b01 ; if abs(Cn_b01) <= 0.00001 ; Cn_b01 = 0 ; else Cn_b01 =
ureal('Cn_b01',Cn_b01,'percentage',1.0) ; end ;
Cn_b00 = Cn.Cn_b00 ; if abs(Cn_b00) <= 0.00001 ; Cn_b00 = 0 ; else Cn_b00 =
ureal('Cn_b00',Cn_b00,'percentage',0.5) ; end ;

Cm_b22 = Cm.Cm_b22 ; if abs(Cm_b22) <= 0.00001 ; Cm_b22 = 0 ; else Cm_b22 =
ureal('Cm_b22',Cm_b22,'percentage',0.5) ; end ;
Cm_b21 = Cm.Cm_b21 ; if abs(Cm_b21) <= 0.00001 ; Cm_b21 = 0 ; else Cm_b21 =
ureal('Cm_b21',Cm_b21,'percentage',1.0) ; end ;
Cm_b20 = Cm.Cm_b20 ; if abs(Cm_b20) <= 0.00001 ; Cm_b20 = 0 ; else Cm_b20 =
ureal('Cm_b20',Cm_b20,'percentage',0.5) ; end ;
```

```
Cm_b12 = Cm.Cm_b12 ; if abs(Cm_b12) <= 0.00001 ; Cm_b12 = 0 ; else Cm_b12 =
ureal('Cm_b12',Cm_b12,'percentage',1.0) ; end ;
Cm_b11 = Cm.Cm_b11 ; if abs(Cm_b11) <= 0.00001 ; Cm_b11 = 0 ; else Cm_b11 =
ureal('Cm_b11',Cm_b11,'percentage',0.5) ; end ;
Cm_b10 = Cm.Cm_b10 ; if abs(Cm_b10) <= 0.00001 ; Cm_b10 = 0 ; else Cm_b10 =
ureal('Cm_b10',Cm_b10,'percentage',1.0) ; end ;
Cm_b02 = Cm.Cm_b02 ; if abs(Cm_b02) <= 0.00001 ; Cm_b02 = 0 ; else Cm_b02 =
ureal('Cm_b02',Cm_b02,'percentage',0.5) ; end ;
Cm_b01 = Cm.Cm_b01 ; if abs(Cm_b01) <= 0.00001 ; Cm_b01 = 0 ; else Cm_b01 =
ureal('Cm_b01',Cm_b01,'percentage',1.0) ; end ;
Cm_b00 = Cm.Cm_b00 ; if abs(Cm_b00) <= 0.00001 ; Cm_b00 = 0 ; else Cm_b00 =
ureal('Cm_b00',Cm_b00,'percentage',0.5) ; end ;

% find magnitudes of combined uncertainties on elements of Jacobian matrix

uA11 = eval(df1.df1da) ; xA11 = usample(uA11,s_s) ;
uA12 = eval(df1.df1dq) ; xA12 = usample(uA12,s_s) ;
uA21 = eval(df2.df2da) ; xA21 = usample(uA21,s_s) ;
uA22 = eval(df2.df2dq) ; xA22 = usample(uA22,s_s) ;
uB11 = eval(df1.df1dd) ; xB11 = usample(uB11,s_s) ;
uB21 = eval(df2.df2dd) ; xB21 = usample(uB21,s_s) ;
uC11 = eval(df3.df3da) ; xC11 = usample(uC11,s_s) ;
uC12 = eval(df3.df3dq) ; xC12 = usample(uC12,s_s) ;
uD11 = eval(df3.df3dd) ; xD11 = usample(uD11,s_s) ;

% construct the system matrix

A = [uA11 uA12 ; uA21 uA22] ;
B = [uB11 ; uB21] ;
C = [uC11/9.81 uC12/9.81 ; 1 0 ; 0 1] ;          % scale so that acceleration output is "g"
D = [uD11/9.81 ; 0 ; 0] ;                        % scale so that acceleration output is "g"

plant = ss(A,B,C,D,'InputName','delta','OutputName',{'acceleration','alpha','q'}) ;

clear Cn* Cm* m J A B C D

% define uncertainty weighting functions on a statistical basis

W.W1 = 3*std(abs(xA11))/abs(uA11.nominal) ;
W.W2 = 3*std(abs(xA12))/abs(uA12.nominal) ;
W.W3 = 3*std(abs(xA21))/abs(uA21.nominal) ;
W.W4 = 3*std(abs(xA22))/abs(uA22.nominal) ;
W.W5 = 3*std(abs(xB11))/abs(uB11.nominal) ;
W.W6 = 3*std(abs(xB21))/abs(uB21.nominal) ;
W.W7 = 3*std(abs(xC11))/abs(uC11.nominal) ;
W.W8 = 3*std(abs(xC12))/abs(uC12.nominal) ;
W.W9 = 3*std(abs(xD11))/abs(uD11.nominal) ;

clear x*
clear u*

%% define uncertainty on cas
disp(' ') ; disp('.....defining cas')

% define parametric uncertainties on smallest elements

wn_cas = ureal('wn_cas',wn_cas,'percentage',1) ;
```

```
dr_cas = ureal('dr_cas',dr_cas,'percentage',1) ;

% only some of the elements in the state-space representation of cas can be uncertain
% find magnitudes of combined uncertainties only for those

ucas_A21 = -(wn_cas^2) ;        xcas_A21 = usample(ucas_A21,s_s) ;
ucas_A22 = -2*dr_cas*wn_cas ;   xcas_A22 = usample(ucas_A22,s_s) ;
ucas_B21 = wn_cas^2 ;           xcas_B21 = usample(ucas_B21,s_s) ;

% construct the system matrix

cas_A = [0 1 ; ucas_A21 ucas_A22] ;
cas_B = [0 ; ucas_B21] ;
cas_C = [1 0 ; 0 1] ;
cas_D = [0 ; 0] ;

cas = ss(cas_A,cas_B,cas_C,cas_D) ;

clear cas_* wn_cas dr_cas

% define uncertainty weighting functions on a statistical basis

W.W10 = 3*std(abs(xcas_A21))/abs(ucas_A21.nominal) ;
W.W11 = 3*std(abs(xcas_A22))/abs(ucas_A22.nominal) ;
W.W12 = 3*std(abs(xcas_B21))/abs(ucas_B21.nominal) ;

%% define uncertainty on imu
disp(' ') ; disp('.....defining imu')

% rgu

% define parametric uncertainties on smallest elements

wn_rgu = ureal('wn_rgu',wn_rgu,'percentage',.1) ;
dr_rgu = ureal('dr_rgu',dr_rgu,'percentage',.1) ;

% construct the tf

rgu = tf([wn_rgu^2],[1 2*wn_rgu*dr_rgu wn_rgu^2]) ;

clear wn_rgu dr_rgu

% acc

% define parametric uncertainties on smallest elements

wn_acc = ureal('wn_acc',wn_acc,'percentage',.1) ;
dr_acc = ureal('dr_acc',dr_acc,'percentage',.1) ;

% construct the tf

acc = tf([wn_acc^2],[1 2*wn_acc*dr_acc wn_acc^2]) ;

clear wn_acc dr_acc
```

E.5 Sub-function “linearize.m”

```
function [df1,df2,df3,dt,qt] = linearize(alpha,Cn,Cm,Cnq,Cmq,m,u0,Q,S,l_ref)
% this code finds the trim points and constructs the Jacobian

%% obtain the analytical expressions for Jacobian matrix and trim conditions

[df1,df2,df3,dt,qt] = robust_analytic_soln ;

%% obtain numeric value for the trim conditions

Cn_b22 = Cn.Cn_b22 ; Cn_b21 = Cn.Cn_b21 ; Cn_b20 = Cn.Cn_b20 ;
Cn_b12 = Cn.Cn_b12 ; Cn_b11 = Cn.Cn_b11 ; Cn_b10 = Cn.Cn_b10 ;
Cn_b02 = Cn.Cn_b02 ; Cn_b01 = Cn.Cn_b01 ; Cn_b00 = Cn.Cn_b00 ;

Cm_b22 = Cm.Cm_b22 ; Cm_b21 = Cm.Cm_b21 ; Cm_b20 = Cm.Cm_b20 ;
Cm_b12 = Cm.Cm_b12 ; Cm_b11 = Cm.Cm_b11 ; Cm_b10 = Cm.Cm_b10 ;
Cm_b02 = Cm.Cm_b02 ; Cm_b01 = Cm.Cm_b01 ; Cm_b00 = Cm.Cm_b00 ;

K1 = l_ref / (2 * u0) ;
K2 = m*u0 / (Q * S) ;

dt = eval(dt) ;
qt = eval(qt) ;

%% choose a solution
% since the equations are quadratic, there exists two solution pairs for q_trim and delta_trim
% choose the one with greater delta_trim

if abs(dt(1)) < abs(dt(2))
   dt = dt(1) ;
   qt = qt(1) ;
else
   dt = dt(2) ;
   qt = qt(2) ;
end
```

E.6 Sub-function “rp_analysis.m”

```
function [perfmarg,perfmargunc,perf_report,perf_info,sysrp] =
rp_analysis(plant,rgu,acc,cas,k,per_weight,DP)

disp(' ') ; disp(['performing mu analysis for robust performance (elapsed time: ' num2str(toc) '
seconds)'])

%% initialize performance weights (defined in robust_synthesize.m)

W_cmd = per_weight.W_cmd ;
W_an_per = per_weight.W_an_per ;                    % acceleration performance weight
W_alpha_per = per_weight.W_alpha_per ;              % angle of attack performance
weight
W_act_sper = per_weight.W_act_sper ;                % deflection rate penalty weight
```

```
W_act_pper = per_weight.W_act_pper ;                    % deflection penalty weight
W_acc_noise = per_weight.W_acc_noise ;                  % accelerometer noise (g)
W_rgu_noise = per_weight.W_rgu_noise ;                  % rgu noise (rad/s)

%% generate closed-loop structure

systemnames = ' plant rgu acc cas k ' ;                 % system components
systemnames = [systemnames, ' W_acc_noise W_cmd W_rgu_noise W_act_pper W_act_sper
W_alpha_per W_an_per ' ] ;      % performance weights

inputvar = ' [ noise_rgu ; noise_acc ; '  ;             % noise inputs
inputvar = [ inputvar, ' an_com ] ' ] ;                 % reference controller commands

outputvar = ' [ W_act_pper ; W_act_sper ; W_alpha_per ; W_an_per ] ' ;% performance
weights

input_to_k = ' [ an_com ; acc + W_acc_noise ; rgu + W_rgu_noise ] ' ;
input_to_plant = ' [ cas(1) ] ' ;
input_to_rgu  = ' [ plant(3) ] ' ;
input_to_acc = ' [ plant(1) ] ' ;
input_to_cas = ' [ k ] ' ;
input_to_W_cmd = ' [ an_com ] ' ;
input_to_W_acc_noise = ' [ noise_acc ] ' ;
input_to_W_rgu_noise = ' [ noise_rgu ] ' ;
input_to_W_act_pper = ' [ cas(1) ] ' ;
input_to_W_act_sper = ' [ cas(2) ] ' ;
input_to_W_alpha_per = ' [ plant(2) ] ' ;
input_to_W_an_per = ' [ W_cmd - acc ] ' ;

cleanupsysic = 'yes' ;

sysrp = sysic ;

%% generate frequency grid and perform robust performance analysis

omega = logspace(-1,2,25);
tty = frd(sysrp,omega);
[perfmarg,perfmargunc,perf_report,perf_info] = robustperf(tty)
figure ; semilogx(perf_info.MussvBnds,'LineWidth',2) ; title(['upper and lower bounds for mu -
DP' num2str(DP-1) ],'FontSize',14)
xlabel('frequency (rad/s) ','FontSize',12) ; ylabel('\mu bounds','FontSize',12) ;

disp(['mu analysis completed (elapsed time: ' num2str(toc) ' seconds)'])
```

E.7 Sub-function “polynomialize.m”

```
function [Cn,Cm] = polynomialize(...
        alpha_vector,delta_vector,Cn_matrix_m,Cm_matrix_m)
% this code polynomializes aerodynamic data of 2nd order

%% obtain coefficients by calling function "fitting"

Cn_fit = fitting(2,alpha_vector,delta_vector,Cn_matrix_m) ;
```

```
Cn.Cn_b22 = Cn_fit(1,1) ; Cn.Cn_b21 = Cn_fit(1,2) ; Cn.Cn_b20 = Cn_fit(1,3) ;
Cn.Cn_b12 = Cn_fit(2,1) ; Cn.Cn_b11 = Cn_fit(2,2) ; Cn.Cn_b10 = Cn_fit(2,3) ;
Cn.Cn_b02 = Cn_fit(3,1) ; Cn.Cn_b01 = Cn_fit(3,2) ; Cn.Cn_b00 = Cn_fit(3,3) ;

Cm_fit = fitting(2,alpha_vector,delta_vector,Cm_matrix_m) ;

Cm.Cm_b22 = Cm_fit(1,1) ; Cm.Cm_b21 = Cm_fit(1,2) ; Cm.Cm_b20 = Cm_fit(1,3) ;
Cm.Cm_b12 = Cm_fit(2,1) ; Cm.Cm_b11 = Cm_fit(2,2) ; Cm.Cm_b10 = Cm_fit(2,3) ;
Cm.Cm_b02 = Cm_fit(3,1) ; Cm.Cm_b01 = Cm_fit(3,2) ; Cm.Cm_b00 = Cm_fit(3,3) ;

%% subfunction "fitting"
function fit2 = fitting(N,alpha_vector,delta_vector,data_matrix)
% this code performs 2D Nth order fitting

for i=1:1:length(delta_vector)
   fit(i,:) = polyfit(alpha_vector',data_matrix(:,i),N) ;
end

for j =1:1:N+1
   fit2(j,:) = polyfit(delta_vector',fit(:,j),N) ;
end
```

E.8 Sub-function “analytic_soln.m”

```
function [df1,df2,df3,dt,qt] = analytic_soln
% this function finds the analytical expressions for Jacobian matrix and trim conditions
% this code is only called from "\robust\linearize.m"
% original copy of this code can be found in "\gecmis\asama
5\linearizasyon\realization\trans.m"

%% state and output equations

f1 = '-((cos(alpha))^2)/(m*u0)*(Q*S*((Cn_b22*dt^2+Cn_b21*dt+Cn_b20)*alpha^2 +
(Cn_b12*dt^2+Cn_b11*dt+Cn_b10)*alpha + (Cn_b02*dt^2+Cn_b01*dt^1+Cn_b00)) + qt *
(Cnq  * l_ref/(2*u0) * Q*S - m*u0))' ;
f2 = 'Q*S/J*l_ref * (((Cm_b22*dt^2+Cm_b21*dt+Cm_b20)*alpha^2 +
(Cm_b12*dt^2+Cm_b11*dt+Cm_b10)*alpha + (Cm_b02*dt^2+Cm_b01*dt^1+Cm_b00)) +
Cmq * l_ref/(2*u0) * qt)' ;
f3 = 'Q*S/m * (((Cn_b22*dt^2+Cn_b21*dt+Cn_b20)*alpha^2 +
(Cn_b12*dt^2+Cn_b11*dt+Cn_b10)*alpha + (Cn_b02*dt^2+Cn_b01*dt^1+Cn_b00)) + Cnq
* l_ref/(2*u0) * qt)' ;

%% Jacobian matrix

df1.df1da = diff(f1,'alpha') ;
df1.df1dq = diff(f1,'qt') ;
df1.df1dd = diff(f1,'dt') ;

df2.df2da = diff(f2,'alpha') ;
df2.df2dq = diff(f2,'qt') ;
df2.df2dd = diff(f2,'dt') ;
```

```
df3.df3da = diff(f3,'alpha') ;
df3.df3dq = diff(f3,'qt') ;
df3.df3dd = diff(f3,'dt') ;

%% find the trim cnditions

[dt qt] = solve('(Cn_b22*dt^2+Cn_b21*dt+Cn_b20)*alpha^2 +
(Cn_b12*dt^2+Cn_b11*dt+Cn_b10)*alpha + (Cn_b02*dt^2+Cn_b01*dt^1+Cn_b00) + qt *
(Cnq * K1 - K2) = 0 ','(Cm_b22*dt^2+Cm_b21*dt+Cm_b20)*alpha^2 +
(Cm_b12*dt^2+Cm_b11*dt+Cm_b10)*alpha + (Cm_b02*dt^2+Cm_b01*dt^1+Cm_b00) +
Cmq * K1 * qt = 0') ;
```