UNIVERSITETET I AGDER

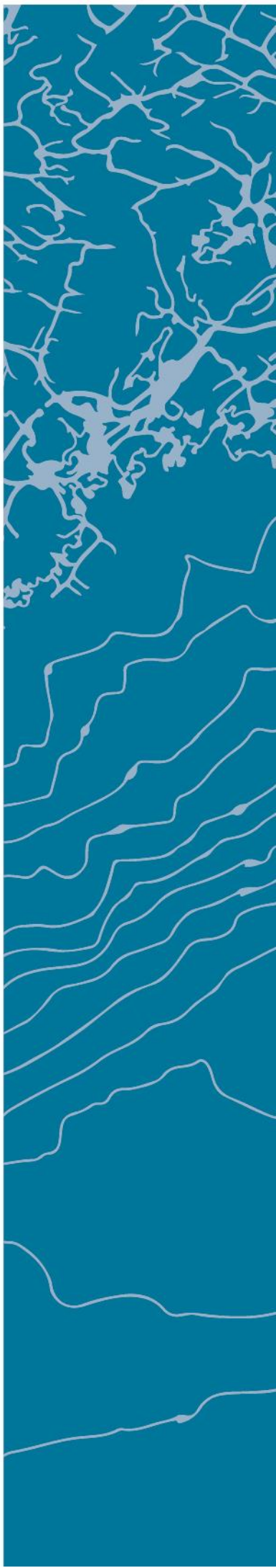
PRACTICAL APPLICATION OF ADVANCED CONTROL

An evaluation of control methods on a Quanser AERO

JAKUB MATEUSZ FRASIK
SVENN INGE LUND GABRIELSEN

SUPERVISOR
Jing Zhou

University of Agder, 2018
Faculty of Engineering and Science
Department of Engineering Sciences



Abstract

In the past years, rotorcraft have received much attention in our society. These provide great accessibility and are utilized in many areas of application. The complex dynamics of rotorcraft make them difficult to control, and the applications demand stable, safe, and agile devices. This thesis investigates the use of advanced control methods and evaluates their effectiveness using a 2DOF (pitch and yaw) 'flight control experiment-platform'; the Quanser AERO.

The main goal of the project is development and evaluation of angular-position controllers for the AERO, using conventional and advanced control theory. This includes cascade P-PI, LQR state-feedback, MPC and MRAC. The controllers are tested in simulation using linear and nonlinear models, and implemented for control of the AERO.

The performance of each controller is evaluated in a standardized test. This test is combined step response sequence for pitch and yaw axes. The performance of each controller is evaluated through a performance index, which is based on the squared tracking error between command signal and plant output.

This project has shown that model based controllers like LQR and MPC works well when tuned to a specific case (for both simulation and practical application), but the performance crumbles when the system is changed without changes to the controller parameters. The simulation of control of a linear system for the conventional controller and MRAC gave excellent results, which confirms that the controllers work well in theory. When it comes to practical application, the conventional controller continues to work relatively well. The MRAC, however, shows strange behaviour, which was not anticipated. The results give insight to the practical application of advanced controllers, and inspires further research into the subject.

Contents

1 INTRODUCTION **1**

2 EXPRESSIONS AND ABBREVIATIONS **3**

3 LITERATURE REVIEW **4**
3.1 MPC 4
3.2 MRAC 6

4 PLANT DESCRIPTION **11**

5 APPLIED THEORY **14**
5.1 Modeling of AERO 14
5.1.1 Kinematic equations for the AERO 14
5.1.2 Linearization of model 15
5.1.3 Quanser's method of linear model approximation 15
5.2 State feedback controller 17
5.2.1 State space 17
5.2.2 Pole placement method 18
5.2.3 Controllability 18
5.2.4 LQR 19
5.2.5 Integral control (Tracking/integral action) 19
5.2.6 Control with pre-filter 20
5.2.7 Observability 21
5.2.8 Luenberger observer 22
5.3 Model-Predictive Control (MPC) 24
5.3.1 Basic concepts of prediction in discrete MPC 24
5.3.2 Closed-loop control system 26
5.3.3 Observer 27
5.3.4 Characteristic equation of closed loop system with integrated observer . . 27
5.3.5 MPC with constraints 28
5.3.6 Quadratic Programming concept 30
5.3.7 Constraints as part of the optimal solution 32
5.3.8 MPC with Laguerrer Networks 32
5.4 MRAC; Model Reference Adaptive Control 34
5.4.1 MRAC of feedback system with MIT rule 34
5.4.2 MRAC of feedback system with normalized MIT rule 35
5.4.3 MRAC of feedback system with Lyapunov stability theory 35

6 PROCEDURE AND METHODS **37**
6.1 Setup of the AERO as a control plant 37
6.2 Modeling and system identification of the AERO 37
6.2.1 Estimation of functions for the nonlinear model 37

6.2.2 Linearized version of the nonlinear model 41
6.2.3 Linear model approximation by Quanser's method 43
6.3 Model comparison 44
6.4 Performance testing 46
6.5 Development of controllers 48
6.5.1 Conventional Cascade Controller 48
6.5.2 Pole placement and LQR 49
6.5.3 Model Predictive controller (MPC) 52
6.5.4 Model-Reference Adaptive Control (MRAC) 55
6.6 Solver Parametes 60

7 RESULTS **61**
7.1 Control performance; Test 1 61
7.2 Control performance with actuator damage; Test 2 61

8 CONCLUSIONS **62**

9 DISCUSSION **65**

10 REFERENCES **68**

11 APPENDICES **71**

List of Figures

1 Observer-based MPC 4
2 Block diagram of a model-reference adaptive system from [1](p. 20) 7
3 A photo of the Quanser AERO 11
4 Yaw body and -axis 12
5 Pitch body and -axis 12
6 CoM of pitch body 12
7 Pitch- and yaw motor. 13
8 Local coordinate system of IMUs 13
9 Integral control 19
10 State feedback control with pre-filter 20
11 Step respons of the system with and without pre-filter 21
12 Luenberger observer 22
13 Complex plane 23
14 MPC: Prediction horizon 26
15 Global optimal solution of function F(x1,x2) 31
16 Optimal solution of function F(x1,x2) with one constraint, Eq. 106 31
17 Optimal solution of function F(x1,x2) with two constraints, Eq. 106 and 107 . . 31
18 Curve of the normalizing function 35
19 I/O subsystem block for the AERO 37
20 Mass-moment data in blue, and the average value in red. 38
21 Voltage-thrust curve 39
22 Cross-torque data and curve 39
23 Caption 40
24 Result from test of yaw inertia as a function of pitch 41
25 Test Sequence 44
26 Model comparison: Nonlinear model, AERO 44
27 Model comparison: Linear Quanser, Linearized, AERO 45
28 Model comparison: Linear Quanser (tuned), AERO 46
29 Command signals for pitch and yaw for the performance test. 47
30 Low- and high efficiency propellers 48
31 Conventional MIMO cascade controller for the AERO 49
32 Step response of Pitch axes using different values of Q matrix 50
33 Step response of Pitch axes using different values of Q matrix with Pre-filter . . 50
34 The feedback controller with Observer and Pre-filter 51
35 Finally State Space based controller 51
36 Test of State Space based controller using Performance Test 52
37 MPC flowchart 52
38 The output response with different a values 54
39 The output response with different N values 54
40 Observer Gains 55
41 Structure of MIMO MRAC 56

42 Linear system MRAC at different adaption gains 58
43 Control performance of 'Normalized' MRAC controller 58
44 Comparison of MRAC versions for nonlinear plant control. 59
45 Comparison of MRAC versions for nonlinear plant control, with edited test. . . . 60
46 Pitch plot AERO Test 2 66
47 Yaw plot AERO Test 2 67

List of Tables

1 Table presenting Example 5.1 from [1](pp.187-190) 8
2 Table describing the functions in the equations of motion 14
3 Three adaption law alternatives for MIMO MRAC 57
4 Solver type 60
5 Performance indices from the first test. Lower is better. 61
6 Performance indices from the second test. Lower is better. 61

1 INTRODUCTION

In the past years, rotorcraft have received much attention in our society due to the popularization of quadcopters and other drones. As with regular helicopters, these provide great accessibility and are utilized in many areas of application, such as transport, search and rescue, police or military operations, photography and filming, and for hobby use. These applications require rotorcraft capable of executing complicated maneuvers in adverse flight conditions, while being stable, reliable, and safe. Increasing demands in these areas increases the necessity of advanced control systems. Therefore it is desired to continue the development and research of advanced control systems. This includes adaptive control systems, which do not only regulate systems, but also adapts the control parameters in response to varying operating conditions. These controllers should produce optimal performance of the system, and continue to do so throughout the operation even though the system is placed in a situation which is has not been exposed to before.

Project description

In an effort to:

- verify the advantages of applying advanced methods to flight control,
- highlight challenges with respect to these methods,
- and evaluate their effectiveness,

several controllers were developed for a 'flight control experiment platform' called the AERO. This product is produced by Quanser, and is a dual-rotor, 2DOF device intended as a 'plug-and-play' plant for control development and experiments. The controllers developed for the AERO in this project are:

- A conventional, cascade P-PI controller in MIMO configuration
- An observer based LQR state feedback controller
- A Model-Predictive Controller (MPC)
- A Model Reference Adaptive Controller (MRAC)

The progression of this project can be described by the following subtasks;

- Modeling of the Quanser AERO.
- System identification; estimating expressions or parameters for the model through practical testing and analysis.
- Development of controllers for the plant using conventional and advanced control theory.
- Simulation of the closed-loop system in MATLAB/Simulink using previously developed linear and nonlinear models of the AERO.

- Implementation of the control algorithms on the AERO platform.
- Testing the performance of the different systems, and their robustness to changes in the system.
- Comparing the results of the different controllers and evaluating their value as control approaches for these kinds of systems.

2 EXPRESSIONS AND ABBREVIATIONS

This section is a reference for expressions and abbreviations used in this report. It should serve as a tool for the readers, and has been used to make sure that the language is consistent throughout the report.

plant	In the context of control theory, a plant is the physical (or simulated) process which is to be controlled.
model	A model is a mathematical representation of a plant, usually an approximation of a physical plant, or a purely theoretical model.
controller	A controller is the algorithm which is used to automatically regulate a plant. They typically calculate an input for the plant based on the output(s) and user defined specifications.
command signal	The command signal is the user input to the controller, describing the desired output for a closed loop system. This signal is also often called 'reference signal' or 'reference input' in other literature.
control law	The equation that is the basis of a controller. It is used to calculate the input signal for a plant.
LQR	Linear Quadratic Regulator, modern control methodology. In this report used in term of State-feedback controller.
MPC	Model Predictive Control, an adaptive control methodology. (Also known as MBPC; Model Based Predictive Control.)
MRAC	Model Reference Adaptive Control, an adaptive control methodology. (Used in MRAS; Model Reference Adaptive System.)
Adaption law	In connection to MRAC, this is the equation which is used to update (adapt) control parameters.
Adaption gain	A constant which determines the rate of change in control parameters (rate of adaption).
SISO	Abbreviation for Single Input-, Single Output control systems.
MIMO	The alternative to SISO; Multiple Input-, Multiple Output control systems.

3 LITERATURE REVIEW

This chapter describes relevant and useful content found in literature such as published books and research papers.

3.1 MPC

The general concept of Model Predictive Control (MPC) is to find the best *Control Trajectory* u(k) based on predicted behaviour of the plant. The basic concept of MPC was developed in the 70s. According to [3] this method was mainly used in the petrochemical industry due to the computing limitation at this time, and the slow response of these types of systems. However, computing power has increased, and MPC has become widely used in more demanding systems.

The main advantages of MPC, over conventional control, are:

- It handles MIMO systems and dead-time without any modification.
- It can be used with constraints.
- It handles challenging dynamics.

Knowing this, MPC may be an appropriate controller for use with the Quanser Aero.

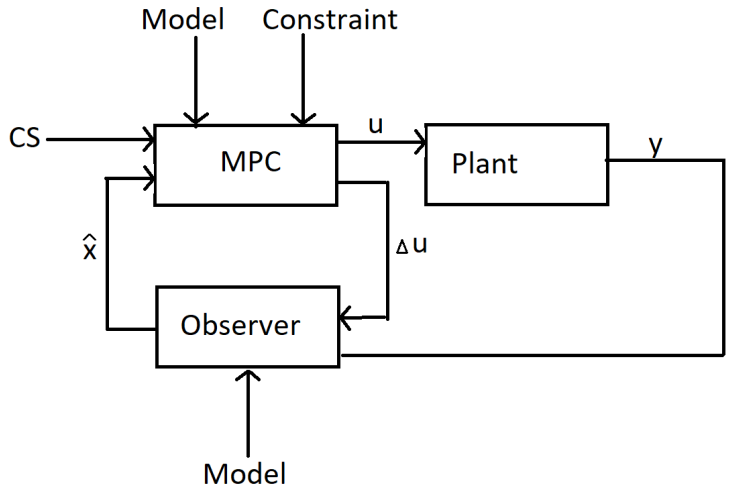


Figure 1: Observer-based MPC

MPC method is a model based controller. This model should be a best possible linear approximation of the plant. Typically, *State Space* representation is used in design of the MPC (Concept of State-Space representation is discussed in Sec. 5.2.1). Additionally, these types of models are implemented in discrete time, due to the fact that decision making and prediction cannot be instantaneous [11]. Based on the mathematical representation of the plant, future behavior of the plant can be calculated, and the best control trajectory u. Basic structure of the MPC with *Observer* is shown in fig. 1. In this case, *Command Signal* (CS) is used as input for the controller. Input for the plant is u(k), and it is the solution of the optimization problem subjected to constraints. This constraints can be applied on input signal u, rate of change Δu

and output from the plant y. In real application, it may happen that some states are missing, and Observer can be used to reconstruct missing signal $\hat{x}$.

Calculation of the control law $u(k)$ using the basic MPC algorithm can be simplified in to 3 steps

- Step 1: Capture current measurement x(k)
- Step 2: Solve optimization problem, in order to find required plant input $u(k)$
- Step 3: Apply u(k)

Step 1: If every state are measurable over time, then states values x(k) can be measured directly from the plant, [5]. However this may not always be the case. It may happen that some states are not measurable, or there is too much noise, which makes the signal un-usable. Maciejowski [3] suggests use of an Observer (or State Estimator) that can reconstruct missing signal(s) from a model of the plant. This method was used in [4]. Additionally, an observer can be used as a noise filter, according to [7]. The linear state space model in discrete is given as:

$$x(k+1) = A_m x + B_m u \tag{1}$$

$$y(k) = C_m x_m(k) \tag{2}$$

System matrices must be rewritten into following form:

$$\overbrace{\begin{bmatrix} \Delta x(k+1) \\ y(k+1) \end{bmatrix}}^{x(k+1)} = \overbrace{\begin{bmatrix} A_m & o_m^T \\ C_m A_m & 1 \end{bmatrix}}^{A} \overbrace{\begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix}}^{x(k)} + \overbrace{\begin{bmatrix} B_m \\ C_m B_m \end{bmatrix}}^{B} \Delta u(k) \tag{3}$$

and

$$y(k) = \overbrace{\begin{bmatrix} o_m & 1 \end{bmatrix}}^{C} \begin{bmatrix} x_m(k) \\ y(k) \end{bmatrix} \tag{4}$$

Now, new system matrices A,B and C are called the augmented model. This method is presented in [7](pp.4-5). New system matrix are used in calculation of prediction and optimization, as well as in calculation of the Observer states values values.

Step 2: The main task of the MPC is to find the best value of η to minimize the cost function:

$$J = \frac{1}{2}\eta^T E \eta + \eta^T H \tag{5}$$

which is subjected to linear inequality constraints:

$$M\eta \leq \gamma \tag{6}$$

Optimization algorithm is based on *Quadratic Programming*. This is the standard procedure used in MPC. Quadratic Programming and optimization is its own field of study. For complete understanding of optimization, books like [9] and [10] can be helpful. (The basic concept of

quadratic programming will be presented in Sec. 5.3.6). The problem can be solved using the following function [7](p.55):

$$\eta = \overbrace{((-E^{-1}H \times CS) - \Phi_x \hat{x})}^{\eta_0} - \overbrace{E^{-1}M^T(-(ME^{-1}M^T)^{-1}(\gamma + ME^{-1}H))}^{\eta_c} \tag{7}$$

where:

- η_0 is the global solution of the optimization problem.
- η_c is the correction term subjected to constraints
- CS is the command Signal
- E, H are the cost matrices
- M, γ are the constraints
- ϕ is the system matrix

Step 3: Applying $u(k)$: Wang, [8](p.135), proposes the use of Laguerre Networks. After solving Eq. 7, the control signal can be calculated as

$$\Delta u(k) = L0^T * \eta \tag{8}$$

$$u(k) = u + \Delta u(k-1) \tag{9}$$

where L0 is initial condition of the Laguerre function. For a given **N** (Number of Languerre networks) and **a** (Pole placement for Languerre networks) L0 is given as:

$$L0^T = \sqrt{(1-a^2)}\begin{bmatrix} 1 & -a & a^2 & -a^3 & \cdots & (-1)^{N-1}a^{N-1}) \end{bmatrix} \tag{10}$$

This method reduces computation load, and makes it possible to capture more complex control trajectories. It is important that $0 < a < 1$, and $1 \leq N$.

Practical implementation of MPC

Paper [6] shows practical application of the MPC on a system with fast dynamics. The practical part of that test was done using the previous version of the Quanser AERO. It can be seen that the controller is relatively slow. Paper [5] implements and tests Constrained Predictive Controller on 3Dof Helicopter. In this case system response is much better (in comparison with results presented in [6]), however the system oscillates around *Set-point* values. These works show that MPC method can be implemented on systems with fast dynamics. Additionally, it is shown that linear mathematical models can be used in design of the MPC used with non-linear system like miniature rotor-crafts.

3.2 MRAC

Adaptive controllers are used to make systems behave as desired, and continue to do so when subjected to external or internal disturbances. These disturbances could for example be expected changes in the plant itself due to wear or heat, or random changes in the plant's environment.

As presented in [1] and [2], there are many types of adaptive controllers, altough there is some discrepancy in the definitions of what constitutes a truly adaptive controller. Some of the presented methods are Gain Scheduling/Open-Loop Adaptive Control, MRAC/Direct Adaptive Control, Self Tuning Regulators/Indirect Adaptive Control, and Dual Control. This section will examine MRAC further.

The basic premise of MRAC is to change the control parameters of a system based on the difference in output between the controlled plant and a reference model. This reference model should be a closed loop linear approximation of the plant, tuned to possess the desired behaviour of the controlled plant. The ideal outcome is that the control parameters converge to values that result in equal dynamics and outputs from the plant and reference model, meaning that the plant is behaving as desired. An illustration of the system, from [1](p.20), is showed in Fig. 2.

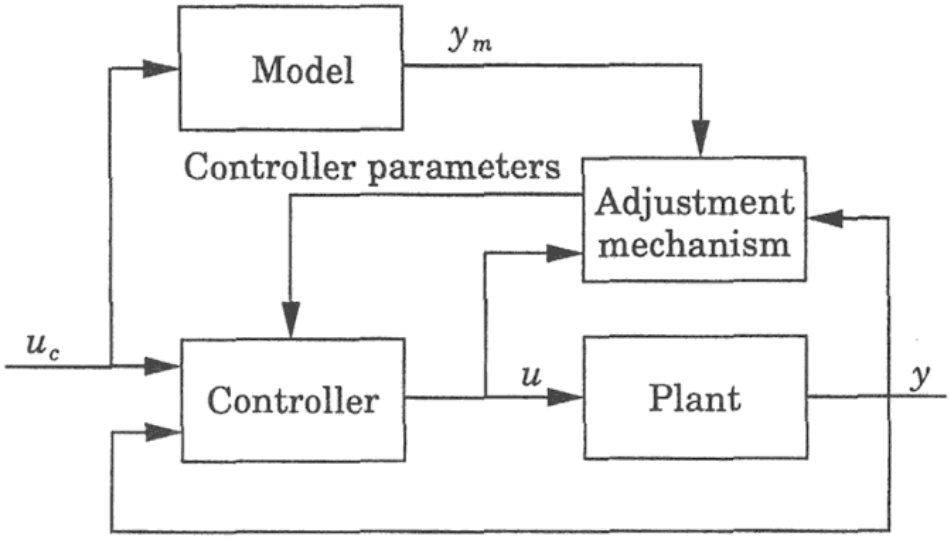


Figure 2: Block diagram of a model-reference adaptive system from [1](p.20) where u_c is the command signal, u is the input to the plant, and y and y_m are the outputs of the plant and model, respectively. The output error, e, is defined by by $e = y - ym$

MIT rule

The central part of a MRAC system is, obviously, the adjustment mechanism; the algorithm which calculates updated parameters for the controller. The seemingly most common solution for this is the MIT rule, which is a gradient method aimed at minimizing the output error. The regular version of the MIT rule is based on the cost function in Eq. (11), and is presented in Eq. (12). This states that the change in control parameters, θ, should be the product of an 'adaption gain' (γ), the output error, and the *sensitivity derivative*. The sensitivity derivative is defined as the partial derivative of the output error with respect to the control parameters. [1](pp. 186-187),[13]-[16]

$$J(\theta) = \frac{1}{2}e^2 \tag{11}$$

$$\frac{d\theta}{dt} = -\gamma e \frac{\partial e}{\partial \theta} \tag{12}$$

In examples from [1],[13],[17], MIT rule based MRAC is used for both feedforward and feedback control of first- and second order systems. The procedure for developing the control systems in these examples is simple. First, a plant and a model is presented. Then, a control law which allows perfect 'model following' (convergence between plant and reference model) is proposed. The closed loop plant output is derived from these equations, and the output error is found by subtracting the model output. This output error equation is differentiated with respect to the control parameters, yielding the sensitivity derivative, which is in turn used in the MIT rule. One of the examples is summarized below;

Plant	Model	Control law	Model following condition
$y = kG(s)u$	$y_m = k_0 G(s) u_c$	$u = \theta u_c$	$\theta = \frac{k_0}{k}$

Table 1: Table presenting Example 5.1 from [1](pp.187-190)

Here, G(s) of the plant is known, but k is not. The model is created with a desired gain k_0, and the control law allows perfect model following if the model following condition is true. From this we calculate the output error:

$$e = y - y_m = kG(s)\theta u_c - k_0 G(s) u_c \tag{13}$$

and the sensitivity derivative:

$$\frac{\partial e}{\partial \theta} = kG(s)u_c = \frac{k}{k_0} y_m \tag{14}$$

which leads to this adaption law, through the MIT rule:

$$\frac{d\theta}{dt} = -\gamma' \frac{k}{k_0} y_m e = -\gamma y_m e \tag{15}$$

where

$$\gamma = \frac{k}{k_0}\gamma'$$

[17] reaches the same adaption law (Eq. 15) as the previous example, for adaptive feedforward gain of a second order system. Similarly, example 5.2 from [1](pp.190-193), and [13] both reach the adaption law in Eq. (16) for feedback control of a first- and second order system respectively. a_m is the damping constant in the model of the second order system, and is related to the time constant of the model of the first order system.
Feedback control law for both example 5.2 in [1], and [13]:

$$u = \theta_1 u_c - \theta_2 y$$

and the following adaption law:

$$\frac{d\theta_1}{dt} = -\gamma e \left(\frac{a_m}{s + a_m} u_c \right), \frac{d\theta_2}{dt} = \gamma e \left(\frac{a_m}{s + a_m} y \right) \tag{16}$$

In another example, [18], MRAC is used to adapt the total gain of a PI controller for a 'old, worn DC motor' to make it behave as if new. The adaption law here is the same as for the two feedforward examples (Eq. 15), and the control law is:

$$u = \theta \left(P + \frac{I}{s} \right) (u_c - y)*$$

* using the same notation as in the previous MRAC examples.

All five examples mentioned in this section verify their controllers through simulation. [13] and [17] show that the systems are able to achieve a step response close to that of the reference models (desired response) with only a small time delay. Example 5.1 and 5.2 from [1], and [18] use repeating signals such as square- and sine waves as command signals, which shows more clearly how the controllers adapt to match the plant to the model over time. Within a few seconds, all three systems have gotten the plants to match the models fairly well, and it appears that the controlled plants slowly converge with the models from there. Generally, all the MRAC systems end up with quick responses and little overshoot.

[17], along with [19] and [20], also show that using the 'normalized MIT rule' (Eq. 17) can minimize undesired oscillatory effects in MRAC at large input values and adaption gains.
Normalized MIT rule:

$$\frac{d\theta}{dt} = \gamma e \frac{\phi}{\alpha + \phi^T \phi} \tag{17}$$

where $\alpha > 0$, and

$$\phi = -\frac{\partial e}{\partial \theta}$$

One interesting note from [1] on adaptive parameters, is that for feedback systems such as in Example 5.2, the parameters do not necessarily need to find their 'correct' values to get good results. (Correct meaning those that produce convergence between plant and reference model). It is, however, the relationship between the parameters that matter the most. For the particular example 5.2, it is seen that the parameters quickly reach $\theta_2 = \theta_1 - a/b$, and stays close to this until the plant and model seemingly converge. It is also stated that the parameters may not reach their 'correct' values at all, since they are highly dependent on the input of the system.

MRAC by Lyapunov stability theory

Due to the uncertain nature of its stability, an alternative to the MIT rule based MRAC was proposed in the 1960's; one based on Lyapunov stability theory. The method gave MRAC systems with guaranteed stability, allowed high adaption gains to be used, and often resulted in simplified adaption loops. [1](p.260),[21](p.221-222),[22],[23]

The procedure for Lyapunov MRAC used in [1](pp.206-215) is quite similar to the MIT rule method, in that a controller structure is defined and the error function is found, before determining an adaption law which will drive the error to zero. The adaption law is found by proposing a quadratic function which is zero when error is zero and the parameters are at their correct values. This function is verified as a Lyapunov function, and the adaption law is chosen so that the derivative of the Lyapunov function is found to be negative semidefinite. Then it

is found that the error will go to zero through the second derivative of the Lyapunov function and Theorem 5.4 (Boundedness and convergence set)[1](pp.205-206). It is concluded that the parameters will not necessarily converge to their correct values, but that they are bounded.

It is noted that the Lyapunov based laws for feedback and feedforward control in [1](pp.206-215) are very similar to the previous MIT rule laws, and that the small changes allow the use of arbitrarily high adaption gains. This is stated to be a product of the very strong assumptions made about the system (that the transfer function is positive real), and the author even remarks that "The result is of limited practical value because of the strong assumptions that are made." [1](p.214).

Practical implementation of MRAC

A common factor for most of the sources on MRAC in this chapter, as well as in [24], is that they are either purely theoretical, or theoretical and verified through simulation; they lack practical testing. Sometimes, even when a physical system is presented, the results of the method is only verified by controlling a linear model of the physical system. This does not guarantee that the method works in practice, since mathematical models are only approximations of reality, and in reality, a process is rarely (if ever) perfectly linear.

The exceptions are [14], [25], and [26], each of which uses MRAC on a DC motor or -servomotor. This leaves uncertainty in whether MRAC would work for more complicated systems with more disturbances and nonlinearities. Perhaps the aforementioned "very strong assumptions" made about plants in connection with Lyapunov MRAC theory is a large hinder in the practical application of the method, and maybe this applies to some degree for the MIT rule MRAC as well.

4 PLANT DESCRIPTION

The Quanser AERO, seen in Fig. 3 is a fully integrated 2DOF dual-motor laboratory test platform, designed for control experiments and -research for aerospace application. Though it can not fly, the AERO possesses some similar dynamics to rotorcraft.

Figure 3: A photo of the Quanser AERO

The three bodies of the AERO are:

1. The base, which is stationary, and holds electronics such as the I/O interface and a power amplifier.

2. The *yaw body* (Fig. 4), which rotates freely about the yaw axis in relation to the base. Zero degrees yaw is determined by its starting position. Angular velocity of this body will be referred to as *yaw velocity*.

3. The *pitch body* (Fig. 5), which is part of the yaw body, and rotates about the pitch axis in relation to the rest of the yaw body. The pitch body's center of mass is offset from the pitch axis, as illustrated in Fig. 6. This means that it behaves like a pendulum, and will return to a horizontal orientation (as seen in Fig. 3) if no other forces are applied. This orientation is defined as zero degrees pitch. Due to physical limitations, the pitch body is limited to ± 60 degrees. Angular velocity of this body will be referred to as *pitch velocity*.

The pitch and yaw joints can be independently locked by the use of an allen key.

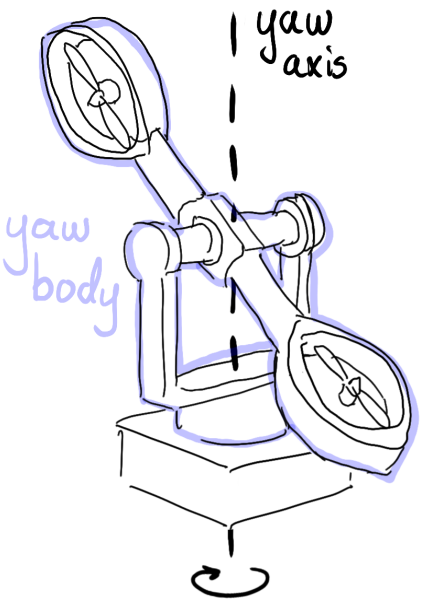


Figure 4: Yaw body and -axis

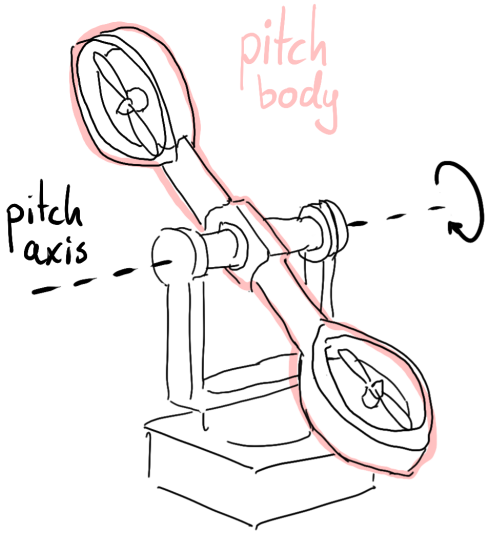


Figure 5: Pitch body and -axis

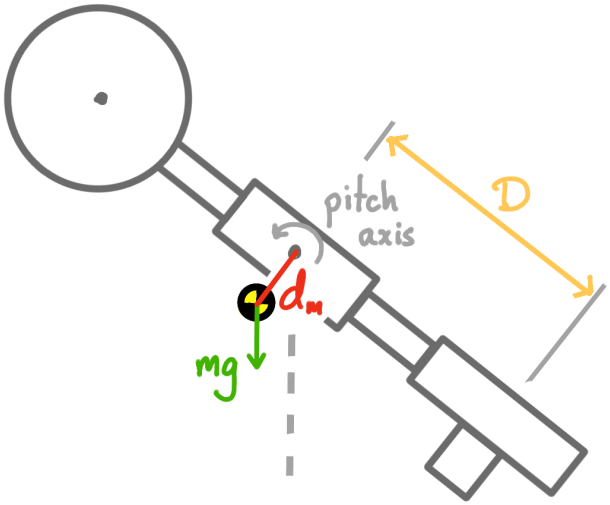


Figure 6: Illustration of center of mass of the pitch body, and force of gravity mg. Length d_m is exaggerated. Length between point of rotation and center of actuators, D, is also included.

The actuators on the AERO are two perpendicular 8-bladed propellers driven by DC motors. One is directed to produce positive rotation about the pitch axis, and is called the pitch motor. The other is directed to produce positive rotation about the yaw axis, and is called the yaw motor. Negative voltage can also be applied to the motors to run them backwards, and get the opposite effects. As a by-product of producing thrust, the propellers also produce cross-torque. This means that the pitch motor will also produce negative rotation about the yaw axis, and the yaw motor will also produce positive rotation about the pitch axis. For the 8-bladed, low efficiency propellers that are used in this project, the effect of the cross-torque is almost as large as from thrust. This means that equal voltage of the same sign to both motors will result

in movement about the pitch axis (since the moments on the yaw axis cancel out), and equal voltage of opposite signs will result in movement about the yaw axis (since the moments on the pitch axis cancel out). This is visualized in Fig. 7, where *Th* is thrust, and *Cr* is cross-torque, for positive voltages on the motors.

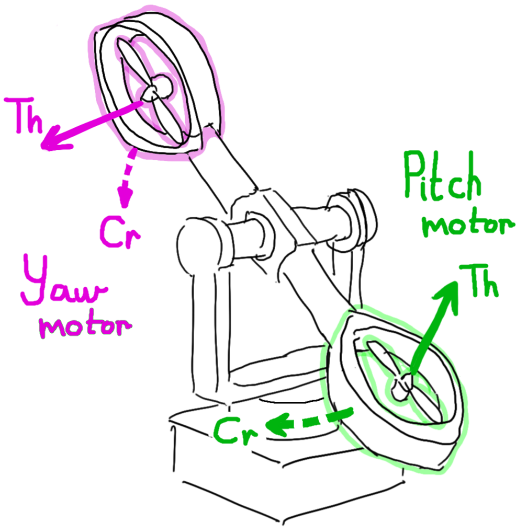


Figure 7: Pitch- and yaw motor.

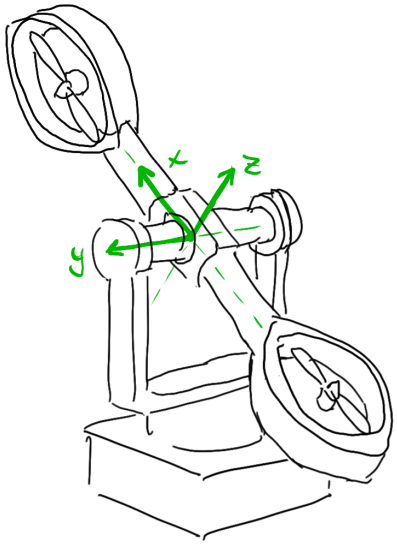


Figure 8: Local coordinate system of IMUs

The AERO is equipped with optical encoders in the joints and motors, which will output relative angular position and velocity. There is also an accelerometer and gyroscope module at the center of the pitch body. Their common local coordinate system is shown in Fig. 8. Positive rotation about the joints is the same as positive rotation about the y- and z- axes when pitch is zero.

The AERO is powered by a 12V adapter, connects to a computer via USB, and is interfaced with through Simulink ®.

5 APPLIED THEORY

This chapter will explain relevant theory applied in this project. Prior knowledge of some conventional control engineering topics including feedback, feedforward, PID, and cascade, is assumed. Some excellent literature regarding these subjects are [27] and [28].

5.1 Modeling of AERO

A model in the context of control theory is a set of mathematical equations which describe the behaviour of the plant. This model should, in a time simulation, produce the same response as the plant when both are excited by an equal input. The inputs for the AERO (and its models) are voltages V_0 and V_1, and the outputs are angles θ_p and θ_y.

5.1.1 Kinematic equations for the AERO

Through modeling, the two degrees of freedom for the AERO (pitch and yaw) gives two equations showing angular acceleration as a result of sum of moments;

Euler-Newton equation of motion for the pitch body (18), and yaw body (19):

$$J_p\ddot{\theta}_p = F_p(V_0)D + M_{py}(V_1) - M_{rp}(sign(\dot{\theta}_p), \Sigma M_p) - M_{Dp}(\dot{\theta}_p) - m_m g \sin\theta_p - M_s(|\dot{\theta}_y|, \theta_p) \quad (18)$$

$$J_y(\theta_p)\ddot{\theta}_y = F_y(V_1)\cos\theta_p D - M_{yp}(V_0)\cos\theta_p - M_{ry}(sign(\dot{\theta}_y), \Sigma M_y) - M_{Dy}(\dot{\theta}_y) \quad (19)$$

The functions in these equations are described in Table 2 below.

Pitch	Yaw	
$J_p\ddot{\theta}_p$	$J_y(\theta_p)\ddot{\theta}_y$	Inertia and acceleration about axis
$F_p(V_0)D$	$F_y(V_1)\cos\theta_p D$	Moment about axis from thrust of motors
$M_{py}(V_1)$	$M_{yp}(V_0)\cos\theta_p$	Cross-torque on body from motors
$M_{rp}(sign(\dot{\theta}_p), \Sigma M_p)$	$M_{ry}(sign(\dot{\theta}_y), \Sigma M_y)$	Moment of Coulomb friction in the joint
$M_{Dp}(\dot{\theta}_p)$	$M_{Dy}(\dot{\theta}_y)$	Damping on the body
$m_m g \sin\theta_p$		Moment of gravity on the body
$M_s(\|\dot{\theta}_y\|, \theta_p)$		Moment from centrifugal force on the body

Table 2: Table describing the functions in the equations of motion

The less straight-forward expressions are explained in more detail here:

$J_y(\theta_p)$ is the inertia of the yaw body, which decreases when the inclination of the pitch body increases, due to the fact that the mass is more concentrated about the axis of rotation.

$M_{rp}(sign(\dot{\theta}_p), \Sigma M_p)$ and $M_{ry}(sign(\dot{\theta}_y), \Sigma M_y)$ is the Coulomb friction in the joints connecting the bodies. If there is movement in the joints, the friction is constant (viscous friction is covered by damping), and if there is not, the friction matches the sum of moments between the joints.

$M_s(|\dot{\theta}_y|, \theta_p)$ the centrifugal moment created on the pitch body when rotating about the yaw axis. This moment will drive the pitch body towards a horizontal position, this this is the plane of rotation.

Physical effects that are not included in this model due to complexity of modeling and identification, and/or due to relative significance;

Conservation of energy in yaw rotation when the inertia decreases. It is quite clear that yaw velocity increases when the pitch inclination increases, and vice versa. However, this is not included in the model.

The gyroscopic effect of the pitch body. We have simplified the inertia into pitch and yaw rotation, excluding the fact that the pitch axis can have rotational inertia in planes other than that of the pitch and yaw axis.

Acceleration and inertia of propellers and motorshafts. There is a certain amount of cross-torque produced by accelerating the propellers and motorshafts, but since the rotation of the propellers is not included, this is not included either.

The decrease in thrust efficiency in 'head-wind', or increase in 'tail-wind'. During rotation, the motors will have velocity in relation to the air around them, which affects the thrust efficiency of the motors. The same effect has also been observed in cross-torque. This is not included due to low operating speeds, and the fact that general wind and pressure differences in the air around the AERO is always changing, and is seemingly more significant.

The fact that damping from air resistance on the yaw body will be lower with a larger inclination of the pitch body, since the speed of the ends of the pitch axis in relation to the air is lower.

5.1.2 Linearization of model

"The differential equations of the motion for almost all processes selected for control are nonlinear" [28](p.661)

In general, modern control methods are based on linear models of the plant. Since most processes are not linear, we can use linearization to find a linear approximation to a nonlinear plant dynamic. In this project, nonlinear parameter used in equation of motion for the pitch and yaw body was linearized through linear approximation using *Basic fitting tool* in Matlab.

5.1.3 Quanser's method of linear model approximation

In the documentation supplied with the Quanser AERO, there is a walkthrough of creating a linear model approximation of the device. They view rotation about the yaw axis as a first order system, and about the pitch axis as a second order system. These are the equations*:

$$J_p\ddot{\theta}_p + D_p\dot{\theta}_p + K_{sp}\theta = K_{pp}V_0 + KpyV_1 \tag{20}$$

$$J_y\ddot{\theta}_y + Dy\dot{\theta}_y = K_{yp}V_0 + KyyV_1 \tag{21}$$

where

- θ_p and θ_y are pitch- and yaw angles.
- J_p and J_y are inertias of pitch- and yaw bodies.
- D_p and D_y are damping constants in pitch and yaw.
- K_{sp} is the spring constant of pitch.
- K_{pp} and K_{yy} are thrust of pitch- and yaw motors.
- K_{py} and K_{yp} are cross-torque of pitch- and yaw motors.

* Some of the parameters' names have been changed to match our modeling.

Estimating parameters

Free-oscillation response about pitch is used to estimate K_{sp} and D_p through the natural frequency and decay of oscillations. Free-damping response about yaw is used to estimate D_y through the time constant. Step responses are used to estimate K_{pp}, K_{yy}, K_{py} and K_{yp} through acceleration.

5.2 State feedback controller

This section will provide a short introduction of 'modern control theory'. This includes modeling of the system using *state space* and the basic concept of a state feedback controller with a pre-filter.

5.2.1 State space

State-space representation is a method of using state variables to describe a dynamic system by a set of first-order differential equations, instead of using nth-order differential equations. A 2^{nd} order mass-spring-damper system can be used as an example to describe how the n-th order equation is described with state variables. The system is described with the following equation: [29]

$$m\ddot{y}(t) + d\dot{y}(t) + ky(t) = u(t) \rightarrow \ddot{y}(t) = \frac{1}{m}(u(t) - d\dot{y}(t) - ky(t)) \tag{22}$$

This equation can be described by dynamic state variables as:

$$y(t) = x_1(t) \tag{23}$$

$$\dot{y}(t) = x_2(t) \tag{24}$$

This gives:

$$\dot{x}_1 = x_2 \tag{25}$$

And by inserting Eq.(23) and (24) for $\ddot{y}$, $\dot{y}$ and y in the 2^{nd} order equations the following equation is obtained:

$$\begin{aligned} \dot{x}_2 &= \frac{1}{m}(u(t) - dx_2(t) - kx_1(t)) \\ \dot{x}_2 &= -\frac{k}{m}x_1(t) - \frac{d}{m}x_2(t) + \frac{1}{m}u(t) \end{aligned} \tag{26}$$

Equation (25) and (26) can now be notated as a matrix:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{d}{m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix} u \tag{27}$$

$$y = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{28}$$

This is a uniform system description with a system matrix (A), an input vector (B) and an output vector (C):

$$\dot{x} = Ax + Bu \tag{29}$$

$$y = C^T x \tag{30}$$

5.2.2 Pole placement method

The poles of a system describe its behaviour. In open-loop $\dot{x} = Ax + Bu$, the poles are the eigenvalues of matrix A, and can be easily determined. State feedback changes the behaviour of a plant, meaning it replaces the open-loop poles with closed-loop poles. In closed-loop $\dot{x} = (A - BK)x$ the poles are eigenvalues of $A - BK$, where matrix K is the feedback gain matrix. The *Pole Placement method* can be used to obtain a feedback gain matrix K by defining the desired location of the closed-loop poles.

Assuming State Space representation of the closed loop system;

$$\dot{x} = (A - BK)x + Bu \tag{31}$$

where:

$$A - BK = \begin{bmatrix} a_{11} & a_{12} \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \begin{bmatrix} k_1 & k_2 \end{bmatrix} \tag{32}$$

Then the characteristic equation is given by:

$$det[\lambda I - (A - BK)] = 0 \tag{33}$$

$$\lambda^2 - (a_{11} - k1)\lambda - (a_{12} - k_2) = 0 \tag{34}$$

and corresponding desired characteristic equation is

$$\lambda^2 + p_1\lambda + p2 \tag{35}$$

where p_1 and p_2 are desired poles placement. Required elements of **k1, k2** can be found by meching coefficients in Eq. 33 and 35. This forces system poles to be placed at the desired location.

5.2.3 Controllability

According to Nise's Control Systems Engineering [27], the definition of controllability is:

"If an input to a system can be found that takes every state variable from a desired initial state to the desired finale state, the system is said to be controllable; otherwise, the system is uncontrollable". The input signal must be able to control all the state variables, and if any of the state variables can not be controlled by the input signal, then we can not place the poles of the system in a desired location, and it will not be possible to design feed-back control gains.

It is possible to determine the controllability of the system using a controllability matrix. For a given nth-order plant:

$$\dot{x} = Ax + Bu \tag{36}$$

system is controllable if and only if the matrix:

$$Con = \begin{bmatrix} B & AB & A^2B & \cdots & A^{n-1}B \end{bmatrix} \tag{37}$$

is of rank n. In this case matrix Con is called the controllability matrix [27].

5.2.4 LQR

One of the more popular and effective approaches in design of state feedback controller is Linear Quadratic Regulator (LQR). The main objective of LQR is to minimize objective function [28]

$$J = \int_0^\infty (x^T Q x + u^T R u) dt \tag{38}$$

using state-feedback law $u = -Kx$ subjected to the following system

$$\dot{x} = Ax + Bu \tag{39}$$

In practice, if system matrix A and B are known, the designers job is to choose appropriate diagonal matrices Q and R that meets the desired system performance. According to [28] this can be done by initially choosing Q and R as:

$$Q_{ii} = 1/maximumacceptablevalueof[x^2]$$
$$R_{ii} = 1/maximumacceptablevalueof[u^2]$$

Then these weighting matrices can be adjusted in order to produce desired system performance [28].

5.2.5 Integral control (Tracking/integral action)

The pole placement and LQR methods can help to achieve desired transient response, however steady-state error can occur. In order to reduce steady-state error to zero, the system type can be increased. This can be done by adding an integrator, Fig.9

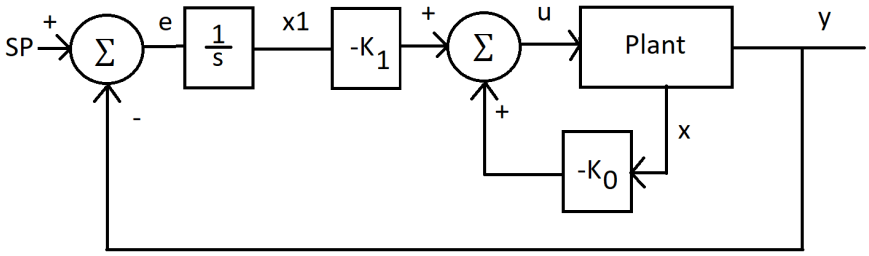


Figure 9: Integral control

In matrix form, this can be written as:

$$\begin{bmatrix} \dot{x} \\ \dot{x}_N \end{bmatrix} = \overbrace{\begin{bmatrix} A_m & 0 \\ -C_m & 0 \end{bmatrix}}^{A} \begin{bmatrix} x \\ x_N \end{bmatrix} + \overbrace{\begin{bmatrix} B_m \\ 0 \end{bmatrix}}^{B} u + \begin{bmatrix} 0 \\ 1 \end{bmatrix} SP \tag{40}$$

$$\overbrace{y = \begin{bmatrix} C_m & 0 \end{bmatrix}}^{C} \begin{bmatrix} x \\ x_N \end{bmatrix} \tag{41}$$

where matrix A_m, B_m and C_m are state space matrices of the open loop linear model. Matrix A, B and C are augmented vectors and matrices with additional state variable $\dot{x}_N = -Cx + r$. In this case, the control law becomes:

$$u = -K_0 x + K_1 x_N \tag{42}$$

[27]
Values of K_0 and K_1 can be found using the *Pole placement* method, where:

$$K = \begin{bmatrix} K1 & K0 \end{bmatrix} \tag{43}$$

5.2.6 Control with pre-filter

By [29], a state feedback control can also include a *Pre-filter*(know also as *Compensator*), which will scale the input signal $CS(t)$ to be of correct amplitude in relation to the output value $y(t)$. Fig. 10 shows, with a block diagram, how the Pre-filter is combined with the state feedback control.

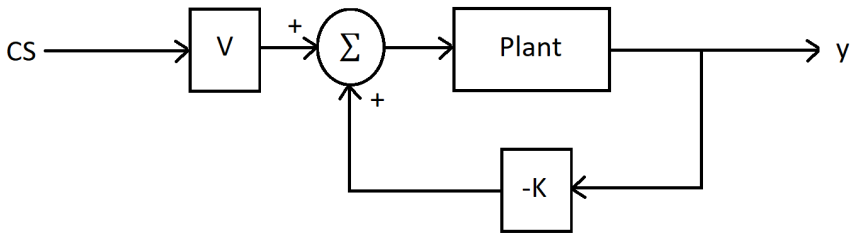


Figure 10: State feedback control with pre-filter

the equation of adding a pre-filter is given by:

$$V = (C(BK - A)^{-1}B)^{-1} \tag{44}$$

In order to show effect of Pre-filter on the state-feedback system, the step response of a simple system is shown in Fig. 11. This test shows an example of steady state error being removed by adding Eq. 44 in to the controller.

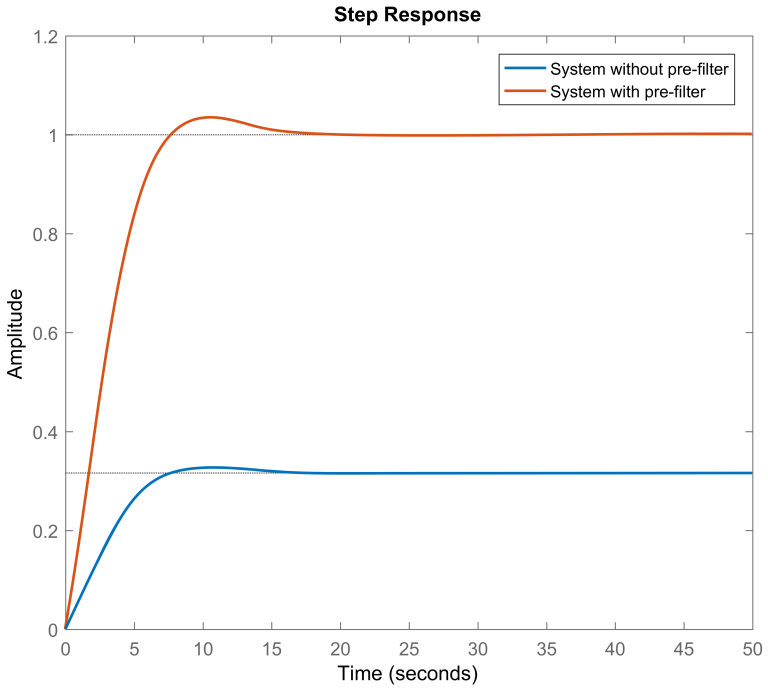


Figure 11: Step respons of the system with and without pre-filter

5.2.7 Observability

According to Nise's Control Systems Engineering [27], the definition of observability is: *If a initial-state vector, $x(t_0)$, can be found from $u(t)$ and $y(t)$ measured over a finite interval from time t_0, the system is said to be observable; otherwise the system is said to be unobservable.*

For a system to be observable all the states need to be connected to the output so that the states can be estimated from a measurement of the output. If any of the states are not connected to the output the system is not observable and it is not possible to decide the location of the poles for the observer, and also it will not be possible to design an observer gain.

In order to include observer as a part of the controller, it is necessary to control observability of the system. This can be done using a observability matrix. For given nth-order plant:

$$\dot{x} = Ax + Bu \tag{45}$$

$$y = Cx \tag{46}$$

system is observable if and only if the matrix:

$$O_M = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} \tag{47}$$

is of rank n. In this case matrix O_M is called the observability matrix [27].

5.2.8 Luenberger observer

A luenberger observer is a state observer which is useful for deterministic disturbance. By [29], the task of the observer is to estimate unavailable state variables and make the error between the actual values and the estimated values as close to zero as possible. Fig. 12 shows the structure of a luenberger observer, including the plant.

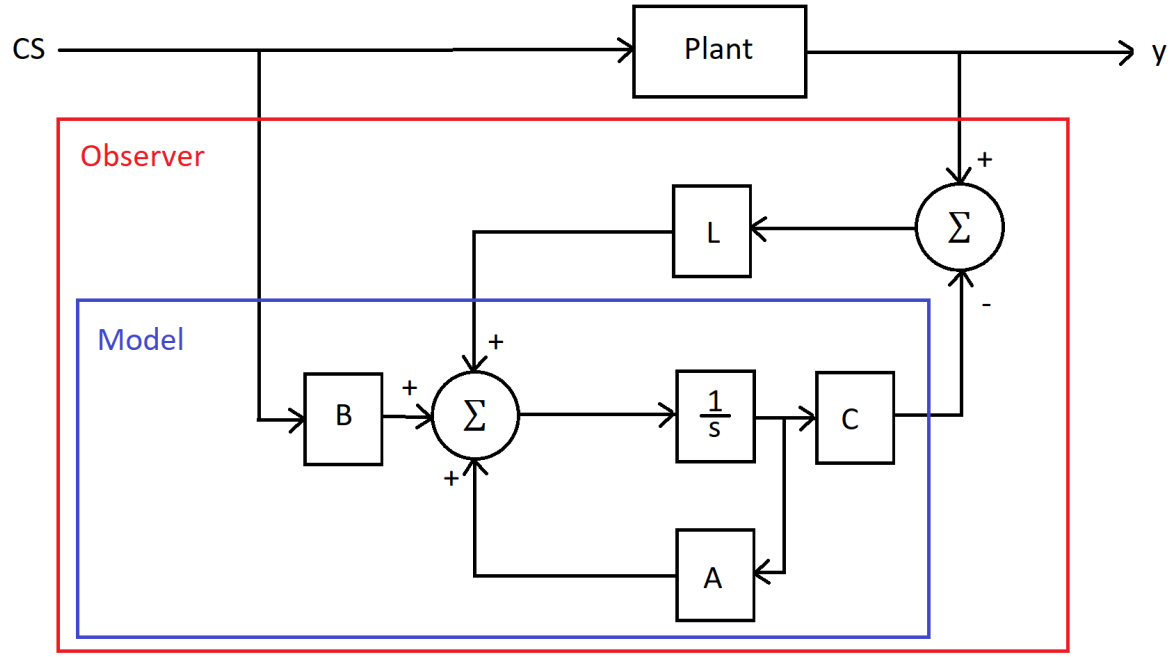


Figure 12: Luenberger observer

In this case **A,B** and **C** are State Space matrix, **L** are Luenberger gains. Consider a plant:

$$\dot{x} = Ax + Bu \tag{48}$$

$$y = Cx \tag{49}$$

and an observer:

$$\dot{\hat{x}} = A\hat{x} + Bu \tag{50}$$

$$\hat{y} = C\hat{x} \tag{51}$$

It is possible to find the error between the actual states and the estimated states and the error between the actual output and the estimated output. The error is shown in Eq. (52) and its derivative is shown in Eq. (53), while the output error is shown in Eq. (54).

$$e(t) = \dot{x} - \hat{x} \tag{52}$$

$$\dot{e}(t) = (A - LC)e(t) = (A - LC)(\dot{x} - \hat{x}) \tag{53}$$

$$\dot{y} - \hat{y} = Ce(t) = C(x - \hat{x}) \tag{54}$$

A rule of thumb is that the observer response should be 5 to 10 times faster than the plant, which means that they should be located further to the left on the *Complex plane*,Fig 13 .

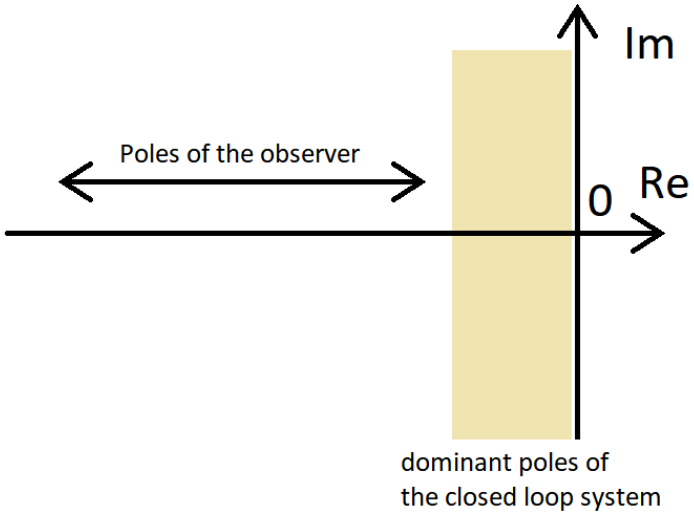


Figure 13: Complex plane

5.3 Model-Predictive Control (MPC)

This section is based on theory presented in [7]

5.3.1 Basic concepts of prediction in discrete MPC

MPC is based on a mathematical model of the plant, that is used for prediction of further behavior of the plant. A State Space representation of the plant in discrete time is most commonly used. Additionally, the mathematical model used in design of the MPC is *Augmented*. This can be done by rewriting the linear system matrix:

$$x_m(k+1) = A_m x + B_m u \tag{55}$$

$$y(k) = C_m x_m(k) \tag{56}$$

as:

$$\overbrace{\begin{bmatrix}\Delta x_m(k+1)\\ y(k+1)\end{bmatrix}}^{x(k+1)} = \overbrace{\begin{bmatrix}A_m & 0_m\\ C_m A_m & 1\end{bmatrix}}^{A}\overbrace{\begin{bmatrix}\Delta x_m(k)\\ y(k)\end{bmatrix}}^{x(k)} + \overbrace{\begin{bmatrix}B_m\\ C_m B_m\end{bmatrix}}^{B}\Delta u(k) \tag{57}$$

$$y(k) = \overbrace{\begin{bmatrix}o_m & 1\end{bmatrix}}^{C}\begin{bmatrix}\Delta x_m(k)\\ y(k)\end{bmatrix} \tag{58}$$

where A_m, B_m and C_m are linear model matrices and 0_m is zero matrix with appropriate dimensions. Matrices **A,B** and **C** are used in development of MPC. Now, a new state value x(k+1) can be calculated as:

$$x(k+1) = Ax(k) + B\Delta u(k) \tag{59}$$

Calculation of the new values of x is basically one-step prediction. This fact can be used to find an n-steps ahead prediction, and can be expressed as:

$$x(k+2) = Ax(k+1) + B\Delta u(k+1) \tag{60}$$

$$x(k+3) = Ax(k+2) + B\Delta u(k+2) \tag{61}$$

and by inserting Eq.59 in to Eq.60 and Eq.60 in to Eq.61 we obtain following equations:

$$x(k+1) = Ax(k) + B\Delta u(k) \tag{62}$$

$$x(k+2) = A[Ax(k) + B\Delta u(k)] + B\Delta u(k+1) \tag{63}$$

$$x(k+3) = A(A[Ax(k) + B\Delta u(k)] + B\Delta u(k+1)) + B\Delta u(k+2) \tag{64}$$

In general form, prediction of state variables can be rewritten as:

$$x(k+n) = A^{N_p}x(k) + A^{N_p-1}B\Delta u(k) + A^{N_p-2}B\Delta u(k+1) + ... + A^{N_p-N_c}B\Delta u(k+N_c-1) \tag{65}$$

Now Eq. 62 - 65 can be writen in matrix form:

$$\begin{bmatrix} x(k+1) \\ x(k+2) \\ \vdots \\ x(k+n) \end{bmatrix} = \begin{bmatrix} Ax(k) + B\Delta u(k) \\ A^2x(k) + AB\Delta u(k) + B\Delta u(k+1) \\ \vdots \\ A^{N_p}x(k) + A^{N_p-1}B\Delta u(k) + A^{N_p-2}B\Delta u(k+1) + ... + A^{N_p-N_c}B\Delta u(k+N_c-1) \end{bmatrix} \tag{66}$$

and by separating model data and parameters we discover the following matrix

$$\begin{bmatrix} x(k+1) \\ x(k+2) \\ \vdots \\ x(k+n) \end{bmatrix} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{N_p} \end{bmatrix} x(k) + \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \cdots & \\ \vdots & \vdots & \ddots & \vdots \\ A^{N_p-1} & A^{N_p-2}B & \cdots & A^{N_p-N_c}B \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_c-1) \end{bmatrix} \tag{67}$$

Using the same method, prediction of system output is found in general form to be:

$$y(k+n) = CA^{N_p}x(k) + C(A^{N_p-1}B\Delta u(k) + A^{N_p-2}B\Delta u(k+1) + ... + A^{N_p-N_c}B\Delta u(k+N_c-1)) \tag{68}$$

This equation can be rewritten into matrix form as:

$$\begin{bmatrix} y(k+1) \\ y(k+2) \\ \vdots \\ y(k+n) \end{bmatrix} = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^n \end{bmatrix} x(k) + \begin{bmatrix} CB & 0 & \cdots & 0 \\ CAB & CB & \cdots & \\ \vdots & \vdots & \ddots & \vdots \\ CA^{N_p-1}B & CA^{N_p-2}B & \cdots & CA^{N_p-N_c}B \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N_c-1) \end{bmatrix} \tag{69}$$

In order to simplify further calculation, following notation will be used for elements in the matrix above:

$$Y = Fx(k) + \Phi\Delta u(k) \tag{70}$$

where, the dimension of Y is determined by the length of Prediction Horizon N_p and the dimension of the Δu is determined by Control Horizon N_c Fig.14 illustrate Prediction Horizon.

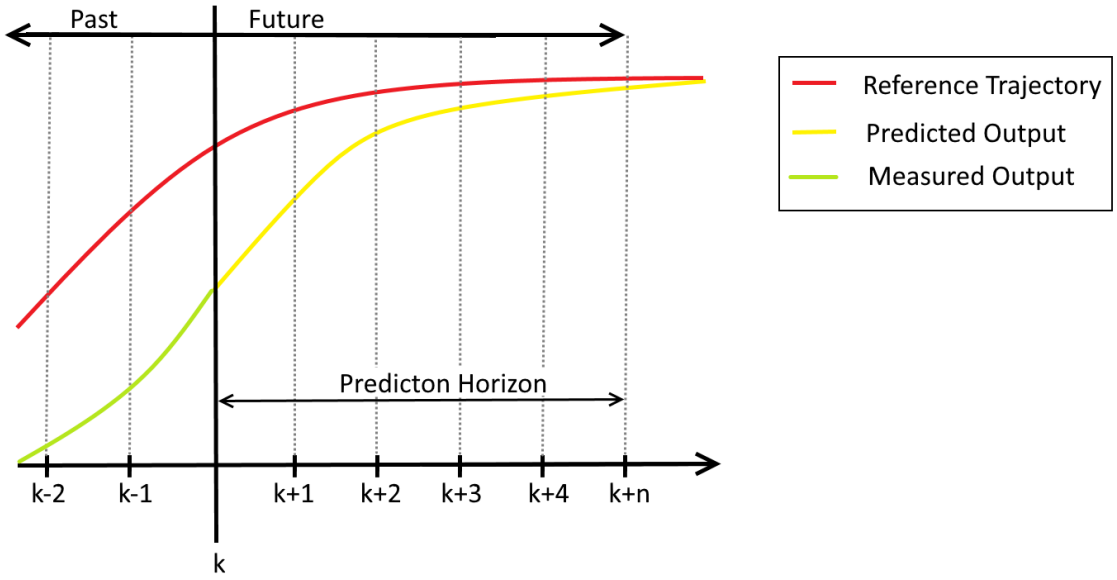


Figure 14: MPC: Prediction horizon

5.3.2 Closed-loop control system

For a given set-point signal $r(k)$ at sample time k optimal solution for control signal is as:

$$\Delta u = (\Phi^T \Phi + \bar{R})^{-1} \Phi^T (Rs - Fx(k)) \tag{71}$$

where:

- $\bar{R} = r_w I_{N_c \times N_c}$ where r_w is used as a tuning parameter closed loop system ($r_w > 0$)
- R_s=[1 1 ... 1]$r(ki)$, is vector containing set-point information, with length equal to length of prediction horizon N_c

Optimal parameter vector Δu is given as a set of information about change in control signal for each step within given control horizon Nc, and can be expressed as:

$$\Delta u = \Delta u(k) + \Delta u(k+1) + ... + \Delta u(k + Nc - 1) \tag{72}$$

However only the first sample of this sequence is used in this controller i.e.Δu(k), thus:

$$\Delta u(k) = \overbrace{\begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}}^{Nc} (\Phi^T \Phi + \bar{R})^{-1} (\Phi \bar{R}_s r(k) - \Phi^T F x(k)) \tag{73}$$

$$= K_y r(k) - K_{mpc} x(k)) \tag{74}$$

where:

- K_y is the first element of $(\Phi^T \Phi + \bar{R})^{-1} (\Phi \bar{R}_s r(k)$,
- K_{mpc} is the first row of $(\Phi^T \Phi + \bar{R})^{-1} \Phi F$

Knowing this fact, closed-loop system can be expressed as:

$$x(k+1) = Ax(k) - BK_{mpc} x(k) + BK_y r(k) \tag{75}$$

$$= (A - BK_{mpc}) x(k) + BK_y r(k)) \tag{76}$$

5.3.3 Observer

If one or more states are not measurable, Observer (known also as State Estimator) may be used to estimate unknown state values or reconstruct part of the missing signal. The observer is based on a mathematical representation of the system. If initial states and input are known, then missing signals can be calculated as:

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) \tag{77}$$

This gives an open-loop prediction. In order to improve prediction, the error between output value(s) from Observer and Plants may be added as correction term. This makes reconstruction of States $\hat{x}$ more accurate. In this case, Observer equation becomes:

$$\hat{x}(k+1) = \overbrace{A\hat{x}(k) + Bu(k))}^{model} + \overbrace{K_{ob}(y(k) - C\hat{x}(k)))}^{correctionterm} \tag{78}$$

The first term of Eq. 78 is model based prediction of the states values, and second part of this equation is correction term based on error between output value from plant $y(k)$ and Observer $C\hat{x}(k) = y(k)$, multiplied by Luenberger gains. Now the optimal solution (Eq. 79) becomes:

$$\Delta U = (\Phi^T\Phi + \bar{R})^{-1}\Phi^T(Rs - F\hat{x}(k)) \tag{79}$$

By implementing feedback gain, signal $\Delta u(k)$ (from Eq. 73) becomes:

$$\Delta u(k) = K_y r(k) - K_{mpc}\hat{x}(k) \tag{80}$$

where state values ($x(k)$) from the plant are replaced with predicted state values ($\hat{x}_p(k)$).

5.3.4 Characteristic equation of closed loop system with integrated observer

As mentioned before, the augmented model of the plant is expressed by:

$$x(k+1) = Ax(k) + B\Delta u(k) \tag{81}$$

By introducing feedback gain, and inserting Eq. 74 into Eq. 81, following equation is obtained:

$$x(k+1) = (A - BK_{mpc})x(k) + BK_y r(k) - BK_{mpc}\hat{x}(k)) \tag{82}$$

Equation for the closed loop observer is:

$$\hat{x}(k+1) = (A - K_{ob}C)\hat{x}(k)) \tag{83}$$

Combining Eq. 82 and 83, overall system matrices are obtained

$$\begin{bmatrix} x(k+1) \\ \hat{x}(k+1) \end{bmatrix} = \overbrace{\begin{bmatrix} A - BK_{mpc} & -BK_{mpc} \\ 0_m & A - K_{ob}C \end{bmatrix}}^{Asys} \begin{bmatrix} x(k) \\ \hat{x}(k)) \end{bmatrix} + \begin{bmatrix} BK_y \\ 0 \end{bmatrix} r(k) \tag{84}$$

in this case 0_m represents zero matrix with appropriate dimensions. Now, determinant of matrix Asys leads to:

$$det(A - B \cdot Kmpc)det(A - C \cdot Kob) = 0 \tag{85}$$

This gives two independent characteristic equations:

$$det(\lambda I - (A - BK_{mpc})) = 0 \tag{86}$$

$$det(\lambda I - (A - K_{ob}C)) = 0 \tag{87}$$

Knowing this fact, the feedback and observer design can be carried out separately.

5.3.5 MPC with constraints

One advantage of Model predictive controller is the possibility to include constraints directly in design of the controller. There are three main types of constraints used with Model Predictive Controller, these are:

1. **Constraints on the amplitude of control variable**
   This type of constraint is the most common type, simply we demand that control signal $u(k)$ can operate within given condition:

$$u^{min} \leq u(k) \leq u^{max} \tag{88}$$

   where:

   - u^{min} is minimal magnitude of control signal
   - u^{max} is maximal magnitude of control signal

   This type of constraint can be used to protect components i.e. if MPC is used to control DC motor, with max input voltage equal to 120V. Constraints on the amplitude of the control variable can be used to prevent voltages higher than 120V to the motor.

   In order to include constraints on the amplitude of the control variable it is necessary to translate this equation into linear inequalities.
   Remark that Δu is the vector parameter that needs to be optimized. With control horizon $Nc = 3$, the incremental control signal is:

$$\Delta U = \begin{bmatrix} \Delta u(k) & \Delta u(k+1) & \Delta u(k+2) \end{bmatrix} \tag{89}$$

   where:

$$u(k) = u(k-1) + \Delta u(k) = u(k-1) + \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \Delta U \tag{90}$$

$$u(k+1) = u(k) + \Delta u(k+1) = u(k-1) + \Delta u(k) + \Delta u(k+1) \tag{91}$$

$$= u(k-1) + \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \Delta U \tag{92}$$

   Knowing this fact, the constraints on the input signal $u(k)$ can be expressed as:

$$\begin{bmatrix} u^{min} \\ u^{min} \\ u^{min} \end{bmatrix} \leq \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} u(k-1) + \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \Delta U \leq \begin{bmatrix} u^{max} \\ u^{max} \\ u^{max} \end{bmatrix} \tag{93}$$

   One method is to use quadratic programming to find optimal solution of cost function J with constraints. The principle of quadratic programming will be discussed later in this

chapter. However, constraints need to be expressed as two expression that reflects the lower and upper limits, this can be done rewriting Eq. 88 as:

$$-u(k) \leq -u^{min}$$
$$u(k) \leq u^{max}$$

And in matrix form constraints on control signal u(k) can be expressed in matrix form as:

$$\begin{bmatrix} -1 & 0 & 0 \\ -1 & -1 & 0 \\ -1 & -1 & -1 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \Delta U \leq \begin{bmatrix} -u^{min} + u(k-1) \\ -u^{min} + u(k-1) \\ -u^{min} + u(k-1) \\ u^{max} - u(k-1) \\ u^{max} - u(k-1) \\ u^{max} - u(k-1) \end{bmatrix} \tag{94}$$

2. **Constraints on the control variable incremental variation**

$$\Delta u^{min} \leq \Delta u(k) \leq \Delta u^{max} \tag{95}$$

where:

- Δu^{min} is minimal magnitude of incremental control signal
- Δu^{max} is maximal magnitude of incremental control signal

This type of constraint can be used to define rate of change of the control variable $\Delta u(k)$. It can also be used to define directional movement of the control variable $u(k)$ i.e. if the control signal can only increase, constraints on the incremental control signal can be defined as: $0 \leq \Delta u(k) \leq \Delta u^{max}$

Using the same method as for derivation of the constraints on Control signal u(k), constraints on the incremental control signal $\Delta u(k)$ can be expressed as:

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Delta U \leq \begin{bmatrix} -\Delta u^{min} \\ -\Delta u^{min} \\ -\Delta u^{min} \\ \Delta u^{max} \\ \Delta u^{max} \\ \Delta u^{max} \end{bmatrix} \tag{96}$$

3. **Output constraints** This type of constraint can be applied on output $y(k)$ or state variable $x(k)$

$$y^{min} \leq y(k) \leq y^{max} \tag{97}$$

This type of constraint can be included in the controller in order to limit operation of the plant to the desired working area. This constraint can also be used to reduce overshoot.

As in two previous cases, constraints need to be expressed as two expressions that reflect the lower and upper limits. This can be done by inserting Eq. 70 into Eq.97 and splitting this equation into two linear inequalities:

$$-\Phi\Delta U \leq -y^{min} + Fx(k) \tag{98}$$

$$\Phi\Delta U \leq -y^{min} - Fx(k) \tag{99}$$

and in matrix form (with Nc=3, and Np=3):

$$\begin{bmatrix} -CB & 0 & 0 \\ -CAB & -CB & 0 \\ -CA^2 & -CA^2B & -CB \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \Delta u(k+2) \end{bmatrix} \leq - \begin{bmatrix} y^{min} \\ y^{min} \\ y^{min} \end{bmatrix} + \begin{bmatrix} CA & 0 & 0 \\ CA^2 & CA & 0 \\ CA^3 & CA^2 & CA \end{bmatrix} \begin{bmatrix} x(k) \\ x(k+1) \\ x(k+2) \end{bmatrix} \tag{100}$$

$$\begin{bmatrix} CB & 0 & 0 \\ CAB & CB & 0 \\ CA^2 & CA^2B & CB \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \Delta u(k+2) \end{bmatrix} \leq \begin{bmatrix} y^{max} \\ y^{max} \\ y^{max} \end{bmatrix} - \begin{bmatrix} CA & 0 & 0 \\ CA^2 & CA & 0 \\ CA^3 & CA^2 & CA \end{bmatrix} \begin{bmatrix} x(k) \\ x(k+1) \\ x(k+2) \end{bmatrix} \tag{101}$$

5.3.6 Quadratic Programming concept

Quadratic programming can be used to find the optimal control signal subjected to constraints. This can be done by finding best solution x for cost function:

$$J = \frac{1}{2}x^T Ex + x^T H \tag{102}$$

which is subject to the following constraint:

$$Mx \leq \gamma \tag{103}$$

the optimal solution x can be found using:

$$x = \overbrace{-E^{-1}H}^{x_0} - \overbrace{E^{-1}M^T\lambda}^{x_c} \tag{104}$$

where first part of the equation x_0 is used to find the global optimal solution that minimizes the original cost function J without any constraints. The second part x_c of the Eq.104 is used as a correction term subjected to constraints.
The Quadratic Programming concept can be illustrated by minimizing the simple function:

$$F(x1, x2) = 2x_1^2 - 2x_1 + 3x_2^2 - 4x_2 \tag{105}$$

with the following constraints:

$$x_2 - 1.5x_1 \geq 2 \tag{106}$$

$$x_1 + x_2 \geq 8 \tag{107}$$

Using Eq. 105, matrix E and H can be found:

$$E = 2\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \tag{108}$$

$$H = \begin{bmatrix} -2 \\ -4 \end{bmatrix} \tag{109}$$

and by rewriting constraints (Eq. 106 and 107), matrices M and γ can be found:

$$\overbrace{\begin{bmatrix} 1.5 & -1 \\ -1 & -1 \end{bmatrix}}^{M}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \overbrace{\begin{bmatrix} -2 \\ -8 \end{bmatrix}}^{\gamma} \tag{110}$$

By solving the first part (x_0) of Eq. 104, the global solution of Eq. 105 (Fig.)can be found:

$$x_0 = \begin{bmatrix} 4 & 0 \\ 0 & 6 \end{bmatrix}^{-1}\begin{bmatrix} -2 \\ -4 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.6667 \end{bmatrix} \tag{111}$$

Now, solving the second part of Eq. 104 gives the correction based on constraints from Eq. 106 and 107:

$$x_c = (E)^{-1}M^T\lambda \tag{112}$$

$$x_c = \begin{bmatrix} 4 & 0 \\ 0 & 6 \end{bmatrix}^{-1}\begin{bmatrix} 1.5 & -1 \\ -1 & -1 \end{bmatrix}\begin{bmatrix} 8.8 \\ 20.8 \end{bmatrix} = \begin{bmatrix} -1.9 \\ -4.9333 \end{bmatrix} \tag{113}$$

where λ is calculated as:

$$\lambda = -(ME^{-1}M^T)^{-1}(\gamma + ME^{-1}F) \tag{114}$$

The global solution x_0 and correction term x_c can now be used to find a solution of the optimization problem. This can be done by inserting x_0 and x_c to Eq. 104, as:

$$\begin{bmatrix} x1_{opt} \\ x2_{opt} \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.6667 \end{bmatrix} - \begin{bmatrix} -1.9 \\ -4.9333 \end{bmatrix} = \begin{bmatrix} 2.4 \\ 5.6 \end{bmatrix} \tag{115}$$

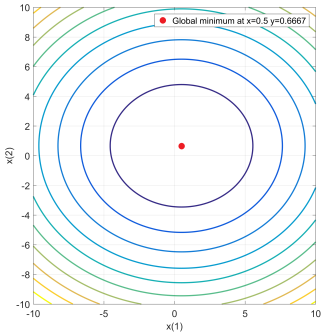


Figure 15: Global optimal solution of function F(x1,x2)

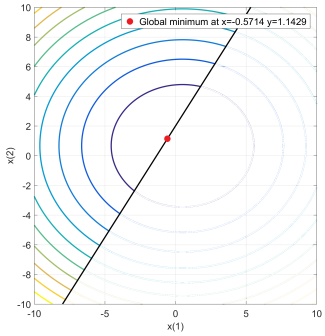


Figure 16: Optimal solution of function F(x1,x2) with one constraint, Eq. 106

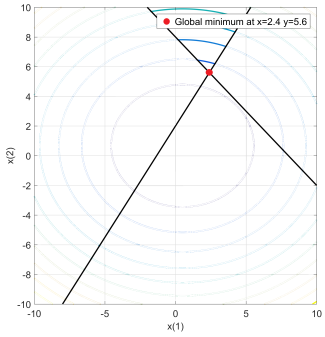


Figure 17: Optimal solution of function F(x1,x2) with two constraints, Eq. 106 and 107

5.3.7 Constraints as part of the optimal solution

In order to include constraints as part of the optimal solution in Model Predictive Controller, it is necessary to translate them into linear inequalities based on ΔU (see subsection **MPC with Constraints**) and then by using quadratic programming (see subsection **Quadratic Programming concept**) it is possible to calculate the optimal solution of function J ($u(k)$).

$$J = (R_s - Fx(k))^T(R_s - Fx(k)) - 2\Delta u^T\phi^T(R_s - F_x(k)) + \Delta u^T(\phi^T\phi + \bar{R})\Delta u \tag{116}$$

The cost function can be rewritten in to the following form:

$$J = \eta^T E\eta + 2\eta^T Hx(k) \tag{117}$$

subjected to:

$$\begin{bmatrix} -C2 \\ C2 \\ --- \\ -I \\ I \\ --- \\ -\Phi \\ \Phi \end{bmatrix} \Delta U \leq \begin{bmatrix} -U^{min} + u(k-1) \\ U^{min} - u(k-1) \\ --------- \\ -\Delta U^{min} \\ \Delta U^{max} \\ --------- \\ -Y^{min} + Fx(k) \\ Y^{max} - Fx(k) \end{bmatrix} \tag{118}$$

5.3.8 MPC with Laguerrer Networks

In real time simulation with relatively small time steps, the traditional approach of capturing future control trajectories using Control Horizon may be not sufficient and may lead to heave computation load. For fast dynamic systems, a method proposed by Wang [8] can be used. This method is based on *Laguerrer Network* that is used to capture further control trajectory. In this case Control horizon is replaced with N, that is number of Laguerrer Networks. This makes it possible to reduce the computation time. Assuming that parameter η (Optimal solution calculated using Quadratic programming) is known for each interaction, the control law becomes:

$$\Delta u(k) = \begin{bmatrix} L0_1(0)^T & 0_z & \cdots & 0_z \\ 0_z & L0_2(0)^T & \cdots & 0_z \\ \vdots & \vdots & \ddots & \vdots \\ 0_z & 0_z & \cdots & L0_{in}(0)^T \end{bmatrix} \eta = L0\eta \tag{119}$$

where 0_z are zero matrix with appropriate dimension, parameter in (length of diagonal of matrix) is number of inputs used in State Space representation of the system. Elements $L0_1, L0_2 \cdots L0_{in}$ can be calculated from:

$$L0_{in}^T = \sqrt{(1-a^2)}\begin{bmatrix} 1 & -a & a^2 & -a^3 & \cdots & (-1)^{N-1}a^{N-1}) \end{bmatrix} \tag{120}$$

where:

- Parameter N is length of Laguerrer network, and must be $1 \leq N$
- Parameter a is the pole of the discrete-time Laguerrer network, and must be $0 \leq a \leq 1$

In terms of linear state feedback, Eq.119 can be rewritten as

$$\Delta u(k) = L0(\Phi_r CS(k) - \Phi_x \hat{x}) \tag{121}$$

where:

- $\Phi_x = E^{-1}H$ where E and H is term of cost function 117
- Φ_r is last column of the Φ_x
- $CS(k)$ is Command Signal
- $\hat{x}$ are States variables estimated by Observer

5.4 MRAC; Model Reference Adaptive Control

As presented the literature review, the MRAC method is a way to automatically and continuously adjust control parameters based on an ideal reference model of the controlled system. The theory applied in the development of MRAC controllers in this project is from the literature presented in section 3.2, specifically [1](pp.185-262).

5.4.1 MRAC of feedback system with MIT rule

For a first order plant

$$\dot{x} = -ax + bu \tag{122}$$

we use the ideal, closed loop reference model

$$\dot{x}_m = -a_m x_m + b_m u_c \tag{123}$$

and control law

$$u = \theta_1 u_c - \theta_2 x \tag{124}$$

which gives the closed loop plant

$$\dot{x} = -ax - b\theta_2 x + b\theta_1 u_c \tag{125}$$

and the output error

$$e = x - x_m = \frac{b\theta_1}{s + a + b\theta_2} u_c - \frac{b_m}{s + a_m} u_c \tag{126}$$

where s is the laplace operator. The terms for perfect model following (zero error) are identified: $a_m = a + b\theta_2$ and $b_m = b\theta_1$, and the sensitivity derivatives are found

$$\frac{\partial e}{\partial \theta_1} = \frac{b}{s + a + b\theta_2} u_c \tag{127}$$

$$\frac{\partial e}{\partial \theta_2} = -\frac{b^2\theta_1}{(s + a + b\theta_2)^2} u_c = -\frac{b}{s + a + b\theta_2} x \tag{128}$$

via approximation $a_m = a + b\theta_2$ and simplification $\gamma = \gamma' \dfrac{b}{a_m}$, we get the adaption law

$$\frac{d}{dt} \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = -\gamma e \left(\frac{a_m}{s + a_m} \begin{bmatrix} u_c \\ -x \end{bmatrix} \right) \tag{129}$$

through the MIT rule. Notice that the filter has unity gain.

5.4.2 MRAC of feedback system with normalized MIT rule

The normalized version of the MIT rule is essentially a constraint on the value of the sensitivity derivative. Its sign is needed for correct adaption, but the impact of its magnitude is limited to nearly nothing. This is why the normalized MIT rule allows for large signal values and adaption gains. The normalized MIT rule is presented in Eq. 17, and the curve of the normalizing function is presented in Fig. 18.

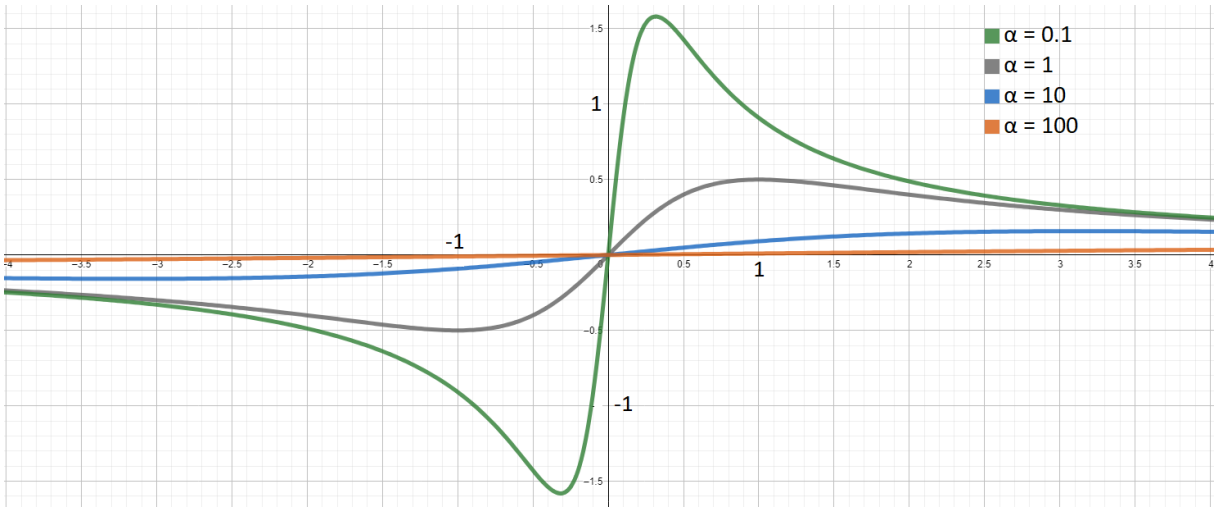


Figure 18: Curve of the normalizing function

5.4.3 MRAC of feedback system with Lyapunov stability theory

To apply Lyapunov stability theory to the control problem, we propose the differential equation

$$\frac{de}{dt} = -a_m e - (b\theta_2 + a - a_m)y + (b\theta_1 - b_m)u_c \tag{130}$$

where the error goes to zero if the parameters are equal to the model following conditions. The following quadratic function is introduced

$$V(e, \theta_1, \theta_2) = \frac{1}{2}\left(e^2 + \frac{1}{b\gamma}(b\theta_2 + a + a_m)^2 + \frac{1}{b\gamma}(b\theta_1 - b_m)^2\right) \tag{131}$$

and it is qualified as a Lyapunov function by showing that its time derivative is negative

$$\frac{dV}{dt} = e\frac{de}{dt} + \frac{1}{\gamma}(b\theta_2 + a + a_m)\frac{d\theta_2}{dt} + \frac{1}{\gamma}(b\theta_1 - b_m)\frac{d\theta_2}{dt} \tag{132}$$

$$= -a_m e^2 + \frac{1}{\gamma}(b\theta_2 + a + a_m)\left(\frac{d\theta_2}{dt} - \gamma x e\right) + \frac{1}{\gamma}(b\theta_1 - b_m)\left(\frac{d\theta_1}{dt} + \gamma u_c e\right) \tag{133}$$

which is

$$= -a_m e^2 \tag{134}$$

if we update the parameters as

$$\frac{d\theta_1}{dt} = -\gamma u_c e \tag{135}$$

$$\frac{d\theta_2}{dt} = \gamma x e \tag{136}$$

This means that the function V is negative semidefinite, and implies that e, θ_1, and θ_2 are bounded. This also implies that $x = e + x_m$ is bounded. From equation

$$\frac{d^2V}{d^2t} = -2a_m e\frac{de}{dt} = -2a_m e(-a_m e - (b\theta_2 + a - a_m)x + (b\theta_1 - b_m)u_c) \tag{137}$$

we can then see that $\ddot{V}$ is also bounded, hence $\dot{V}$ is uniformly continuous. From Theorem 5.4 in [1](pp.205-206) it now follows that the error will go to zero, and the adaption laws in Eqs. 135 and 136 are validated.

6 PROCEDURE AND METHODS

This chapter will describe the execution of the project, is sorted similarly to the chronological order of progress.

6.1 Setup of the AERO as a control plant

The AERO came with a copy of the *QUARC® Quanser Real-Time Control Software* extension for Simulink ®. This package includes Simulink function blocks for interacting with the platform. The inputs to the plant are voltages applied to the motors, in the range $\pm 24V$. The outputs used are pitch, yaw, pitch velocity and yaw velocity. Pitch and yaw are read from the encoders, and converted into degrees. Pitch- and yaw velocity are calculated from the gyroscope readings. The signals from the gyro are calibrated, converted to degrees, and filtered to reduce noise. The pitch velocity is then read from the gyro y-value, and the yaw velocity is calculated from the gyro's x- and z-value through trigonometry. All of this was combined into the subsystem block seen in Figure 19.

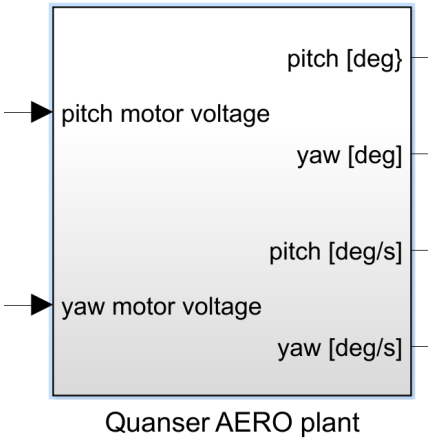


Figure 19: I/O subsystem block for the AERO

6.2 Modeling and system identification of the AERO

Four models have been made of the AERO. The first is the *nonlinear model*, and the second a linearized version of this. The third is another linear model, made through the procedure described by Quanser in their documentation for the AERO, and the last is a self-tuned version of the linear models.

6.2.1 Estimation of functions for the nonlinear model

The nonlinear model of the AERO was made by applying the Euler-Newton equation of motion as presented in Sec. 5.1. This section will explain how practical tests and calculations were used to find expressions for the functions in the model (Eq. 18 and 19). These functions are, for example, relationship between applied motor voltage and produced thrust, moment of damping in relation to angular velocity, and the effect of gravity on the pitch body.

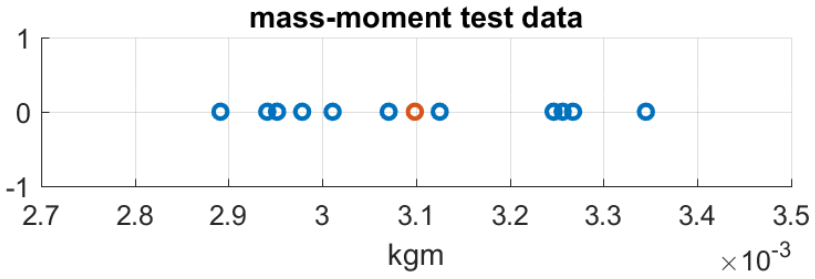


Figure 20: Mass-moment data in blue, and the average value in red.

m_m; The 'mass-moment' of the pitch body, which is used to calculate the moment from gravity. Since neither the mass of the pitch body nor the length between its center of mass and the pitch axis are known, the product of the two is approximated though a test. The test was performed by suspending tiny weights from the pitch body at a known length from the pitch axis, and the resulting steady state pitch is measured. The mass-moment is then calculated from Eq. 138, where g is the gravity constant. The average is taken of the mass-moments calculated from every data point in the test, which is shown in Fig. 20.

$$weight \cdot length \cdot cos(\theta_p) \cdot g = m_m \cdot sin(\theta_p) \cdot g \tag{138}$$

$$m_m = 0.0031[kgm] \tag{139}$$

$F_p(V_0), F_y(V_1)$; Force of thrust [N] as a function of applied motor voltage. This was found by doing three steady state tests, one using tiny counterweights on the pitch axis to oppose the thrust produced by the pitch motor, one using a force gauge to measure the thrust of both the pitch and yaw motor (separately), and the last by measuring steady state pitch and calculating thrust via the mass-moment found in the previous test. We assume that the thrust curve is identical on the two motors, and the data supports this. Fig. 21 shows the positive test data for positive and negative voltages, and the second order regression curve for the data. This curve is used in the nonlinear model if the voltage is above $2.46V$. The same curve, only mirrored about both axes, is used for negative voltages.

$$F_p(V_0) = F_y(V_1) = 0.0002661V^2 + 0.004214V - 0.01192[N] \tag{140}$$

$Mpy(V_1), Myp(V_0)$; Cross-torque [Nm] as a function of applied voltage. The same counter-weight test as for thrust was used to measure the cross-torque, and it was found that the data coincides well with the voltage-thrust curve found previously (multiplied by moment arm D), as seen in Fig. 22. Therefore, it is decided to use the same curve, which gives:

$$F_p(V_0) \cdot D = F_y(V_1) \cdot D = M_{py}(V_1) = Myp(V_0) \tag{141}$$

$M_{ry}(sign(\dot{\theta}_p)), M_{Dy}(\dot{\theta}_y)$; The Coulomb friction and damping on the yaw body was found by examining plots of the yaw velocity, as it decreases from spinning quickly to a full stop

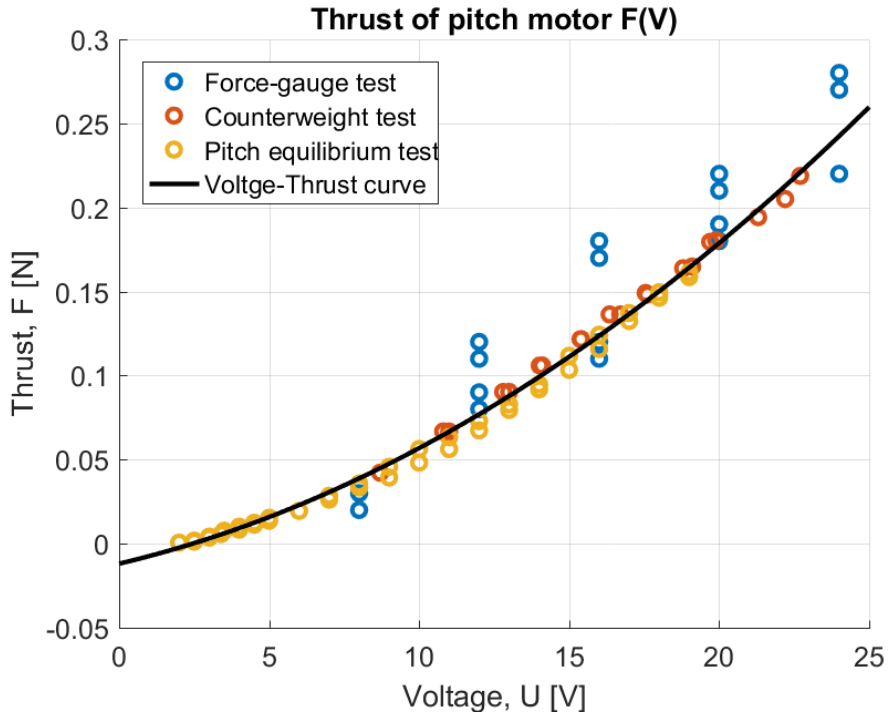


Figure 21: Voltage-thrust curve from second order regression of the test data

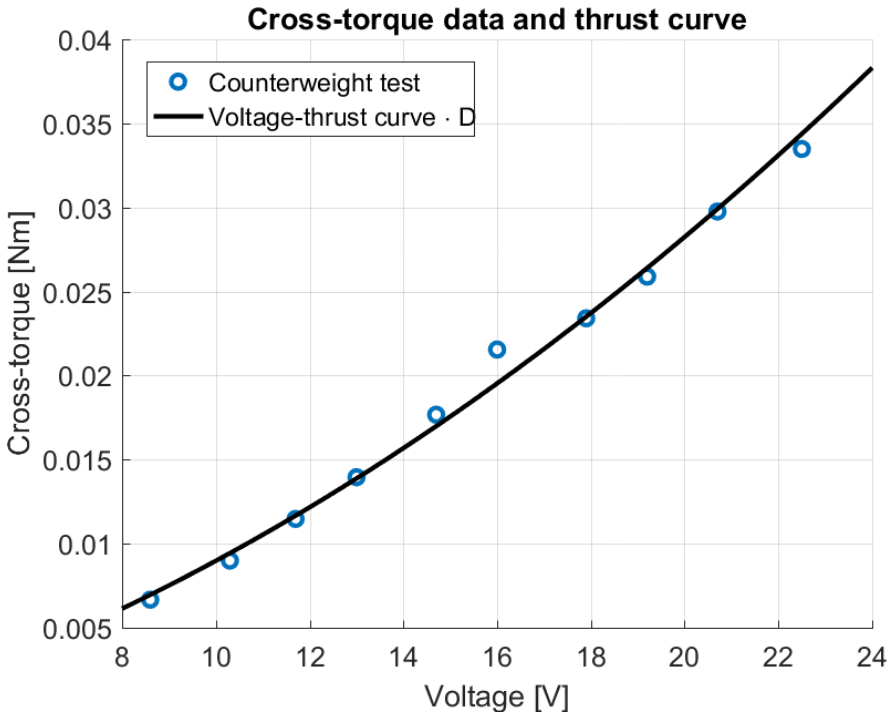


Figure 22: Cross-torque test data plotted along the voltage-thrust curve (multiplied by moment arm D).

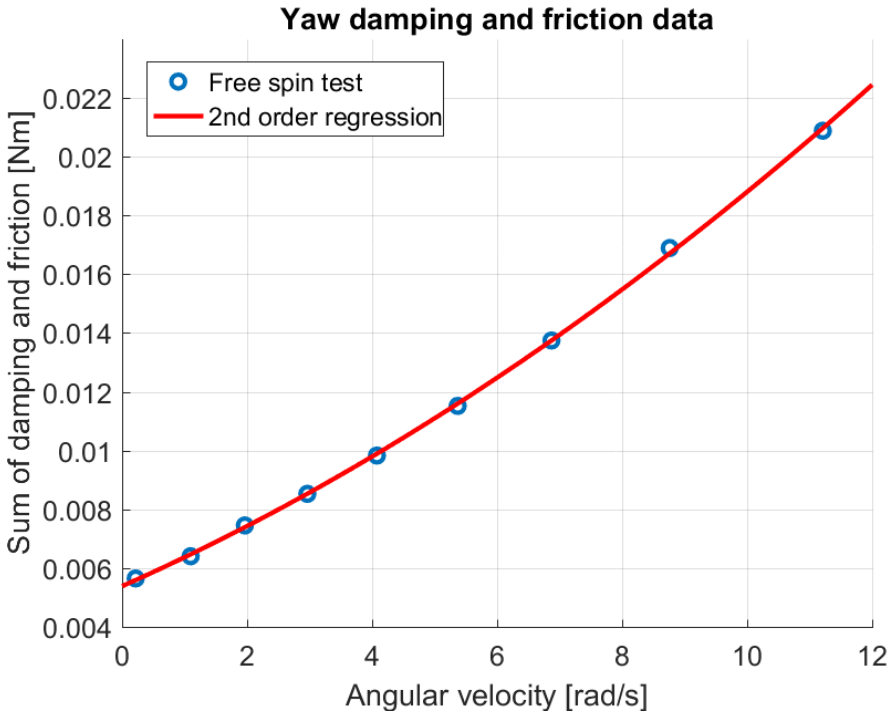


Figure 23: Caption

(with motors disengaged, and the pitch axis locked to horizontal). The angular deceleration was calculated for many points along the curve, and the moment sum of damping and friction was calculated through $J_y(\theta_p = 0) \cdot \ddot{\theta}_y = -(M_{dy} + M_{ry})$. These values were plotted against the yaw velocity. Second order regression of the plot gave the relationship between damping and angular velocity, and the value at zero velocity is the Coulomb friction of the yaw joint. The data points and resulting curve is shown in Fig. 23.

$$M_{ry}(\dot{\theta}_p > 0) = 0.00538[Nm] \tag{142}$$

$$M_{Dy}(\dot{\theta}_y) = 3.99 \cdot 10^{-5}\dot{\theta}_y^2 + 9.43 \cdot 10^{-4}\dot{\theta}_y[Nm] \tag{143}$$

$J_y(\theta_p)$; The change in yaw inertia as a function of pitch was found by measuring the initial yaw acceleration for different (positive) pitch angles and (positive) yaw motor voltages. For each of combination of pitch and voltage, the inertia was calculated from the equation $J_p\ddot{\theta}_y = F_y(V_1)cos(\theta_p)D - M_{Dy}(\dot{\theta}_y)$. This was plotted against the pitch, and regression was used to find the relationship between pitch and yaw inertia. (The data is for positive pitch, and was mirrored along the y axis to get a curve that covered both). The result of yaw inertia at zero degrees pitch is very similar to the yaw inertia given in the Quanser documentation (our: $0.02255kgm^2$, Quanser: $0.0220kgm^2$). It was also discovered that the second order regression curve fits very well with a cosine function, and therefore, Eq. 144 was used. The data, the second order regression, and the cosine function can all be seen in Fig. 24

$$J_y(\theta_p) = 0.02255cos(\theta_p)[kgm^2] \tag{144}$$

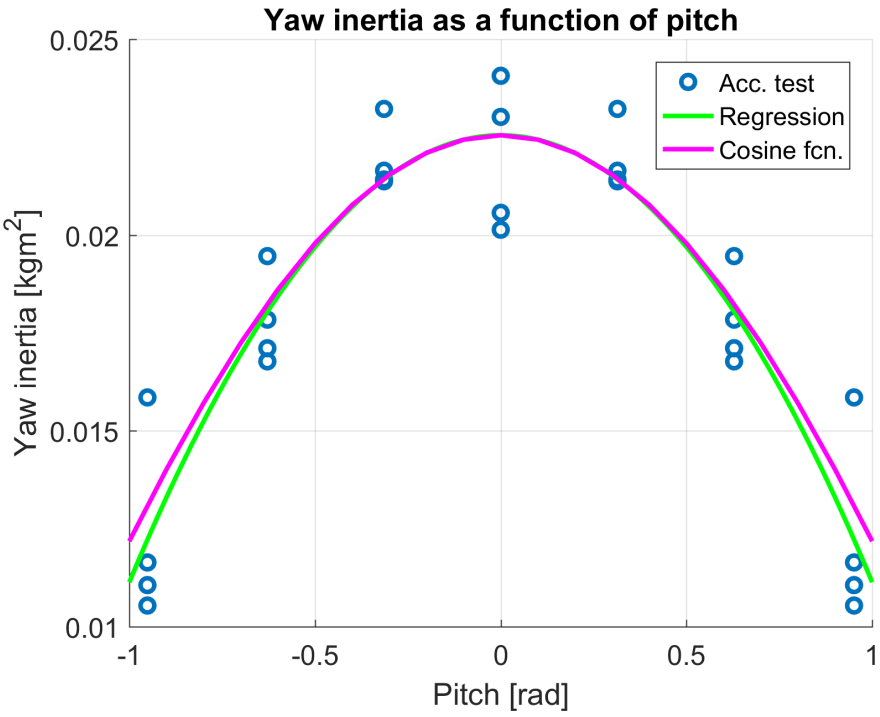


Figure 24: Result from test of yaw inertia as a function of pitch

As for the rest of the functions:

- Pitch inertia ($J_p = 0.0219[kgm^2]$) was given in the Quanser documentation.
- Pitch Coulomb friction and damping ($M_{rp}(\dot{\theta}_p) > 0) = 3.72{\cdot}10^{-4}[Nm], M_{Dp}(\dot{\theta}_p) = 0.0023\theta_p[Nm]$) were assumed to be similar to that of the yaw body, and then estimated by iterative tuning and comparison between the response of the nonlinear model and the AERO.
- Centrifugal moment ($M_s(|\dot{\theta}_y|, \theta_p) = 0.018 \cdot \dot{\theta}_y^2 sin(\theta_p)cos(\theta_p)$) was hard to isolate in a test, and a parameter for it was also estimated through iterative tuning.

The expressions presented in this section were used with Eq. 18 and 19 from Sec. 5.1 in a 'MATLAB Function' block in Simulink to create the nonlinear model. Some of the parameters have subsequently been subjected to small amounts of tuning to decrease the difference between the nonlinear model and the plant.

6.2.2 Linearized version of the nonlinear model

To create a linearized version of the nonlinear model, the nonlinear functions were discarded, and linear regressions was used on the data in stead of second order regressions.

The linearized equation of motion for the pitch body is as follows:

$$J_p\ddot{\theta}_p = F_p(V_0)D + M_{py}(V_1) - M_{Dp}(\dot{\theta}_p) - m_m \cdot g \cdot k_{sin} \cdot \theta_p \tag{145}$$

where:

- $F_p(V_0) = 0.007899 \cdot V_0$
- $M_{py}(V_1) = -1.259 \cdot V_1$
- $M_{Dp}(\dot{\theta}_p) = 0.0016982 \cdot \dot{\theta}_p$
- $m_m g k_{sin} \theta_p = 0.0031 \cdot 9.81 \cdot 0.89 \cdot \theta_p$ (k_{sin} is an approximation of sine)
- $J_p = 2.15 \cdot 10^{-2}$

The equation of motion for the yaw body is as follows:

$$J_y \ddot{\theta}_y = F_y(V_1)D - M_{yp}(V_0) - M_{Dy}(\dot{\theta}_y) \tag{146}$$

where:

- $F_y(V_1) = 0.007653 \cdot V_1$
- $M_{yp}(V_0) = 0.0007634 \cdot V_0$
- $M_{Dy}(\dot{\theta}_y) = 0.002028 \cdot \dot{\theta}_y$
- $J_y = 2.35 \cdot 10^{-2}$

State space model

In order to create state space model, Eq. 145 and 146 was rewritten in to following form:

$$\ddot{\theta}_p = \frac{F_p(V_0) * D}{J_p} + \frac{M_{py}(V_1)}{J_p} - \frac{M_{Dp}(\dot{\theta}_p)}{J_p} - \frac{m_m * g * k_{sin} * \theta_p}{J_p} \tag{147}$$

and

$$\ddot{\theta}_y = \frac{F_y(V_1) * D}{J_y} - \frac{M_{yp}(V_0)}{J_y} - \frac{M_{Dy}(\dot{\theta}_y)}{J_y} \tag{148}$$

Now, Eq. 147 and 148 can be easily rewritten in to State Space for:

$$\dot{x} = A \begin{bmatrix} \dot{\theta}_p \\ \theta_p \\ \dot{\theta}_y \\ \theta_y \end{bmatrix} + B \begin{bmatrix} V_0 \\ V_1 \end{bmatrix} \tag{149}$$

$$y = C \begin{bmatrix} \dot{\theta}_p \\ \theta_p \\ \dot{\theta}_y \\ \theta_y \end{bmatrix} + Du \tag{150}$$

where:

$$A = \begin{bmatrix} \frac{-M_{Dp}(\dot{\theta}_p)}{J_p} & \frac{-m_m * g * 0.89}{J_p} & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{-M_{Py}}{J_y} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{151}$$

$$= \begin{bmatrix} -0.07899 & -1.259 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -0.0863 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{152}$$

$$B = \begin{bmatrix} \frac{F_p * D}{J_p} & \frac{M_{py}}{J_p} \\ 0 & 0 \\ \frac{-M_{yp}}{J_y} & \frac{F_y * D}{J_y} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0.0562 & 0.0355 \\ 0 & 0 \\ -0.0325 & 0.0515 \\ 0 & 0 \end{bmatrix} \tag{153}$$

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{154}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{155}$$

6.2.3 Linear model approximation by Quanser's method

The procedure described in Sec. 5.1.3 was performed by simply connecting the AERO to the computer, and executing the Simulink scripts provided with the documentation from Quanser. This produced the parameters, which were directly inserted into the state space model from the documentation (shown in Eq. 156 and 157 below).

$$\begin{bmatrix} \theta_p \\ \theta_y \\ \dot{\theta}_p \\ \dot{\theta}_y \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -K_{sp}/J_p & 0 & -D_p/J_p & 0 \\ 0 & 0 & 0 & -D_y/J_y \end{bmatrix} \begin{bmatrix} \dot{\theta}_p \\ \dot{\theta}_y \\ \ddot{\theta}_p \\ \ddot{\theta}_y \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ K_{pp}/J_p & K_{py}/J_p \\ K_{yp}/J_y & K_{yy}/J_y \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \end{bmatrix} \tag{156}$$

$$\begin{bmatrix} \theta_p \\ \theta_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \theta_p \\ \theta_y \\ \dot{\theta}_p \\ \dot{\theta}_y \end{bmatrix} \tag{157}$$

where:

- $K_{sp} = 0.0377[Nm/V]$
- $D_p = 0.0072[Nms/rad]$
- $D_y = 0.0085[Nms/rad]$
- $K_{pp} = 0.0011[Nm/V]$
- $K_{yy} = 0.0044[Nm/V]$
- $K_{py} = 0.0019[Nm/V]$
- $K_{yp} = 0.0026[Nm/V]$
- $J_p = 0.0219[kgm^2]$
- $J_y = 0.0220[kgm^2]$

6.3 Model comparison

At this point, project group decided to compare the response of created models with a response of AERO. This was done by implementing voltage sequence for pitch and yaw motor. The test sequence is shown in Fig 25.

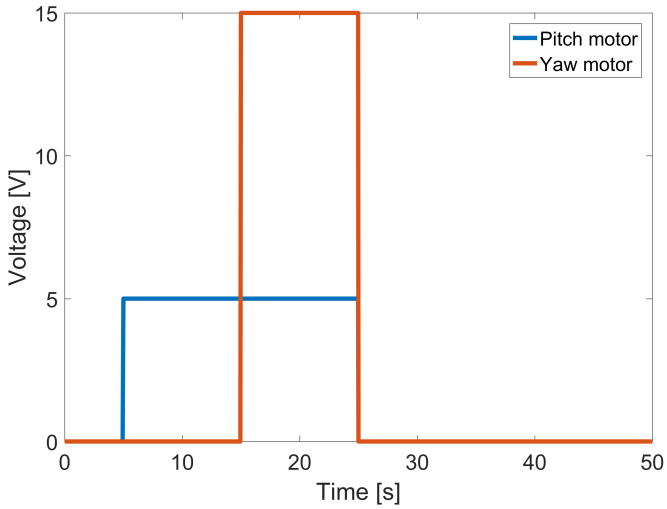


Figure 25: Test Sequence

This sequence was applied to an open-loop nonlinear model and AERO, shown in Fig 26. This figure shows that main dynamics of the AERO are captured in Nonlinear model.

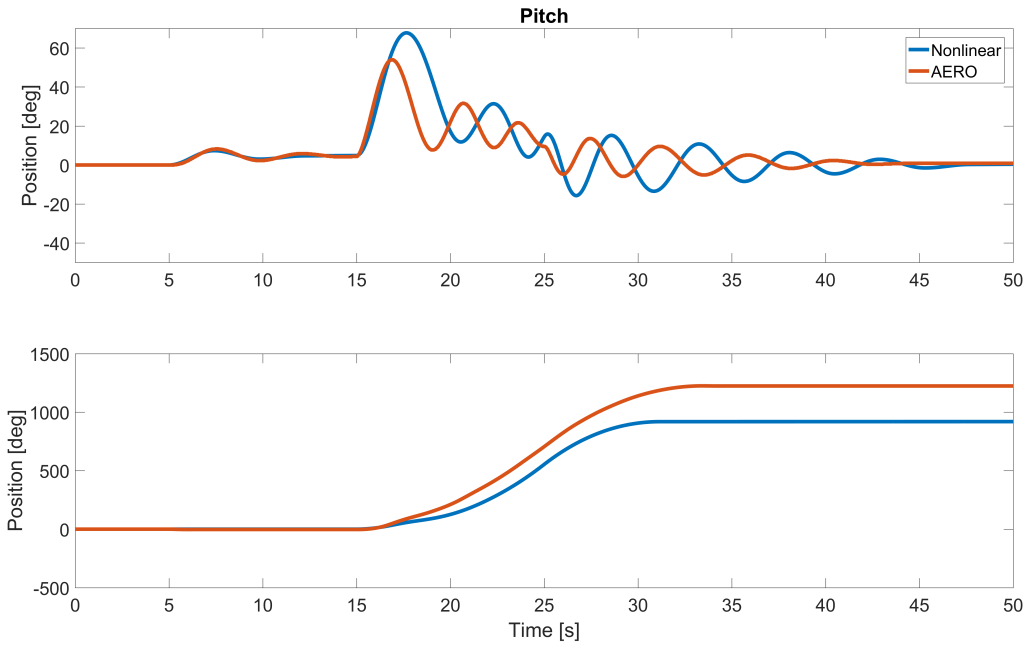


Figure 26: Model comparison: Nonlinear model, AERO

Next, a Nonlinear model was linearized and compared with dynamics of the AERO using same test sequence. Additionally, a linear model made through the procedure described by

Quanser was added to the comparison. The response of the open-loop systems can be seen in Fig 27. By visual inspection, it is easy to see that Linearized model have different dynamics then AERO. Reason for this is during linearization of the nonlinear model, friction and other nonlinearities like a centrifugal force was excluded from the model, resulting in long settling time and not accurate coupling between dynamics of the pitch and yaw axes.

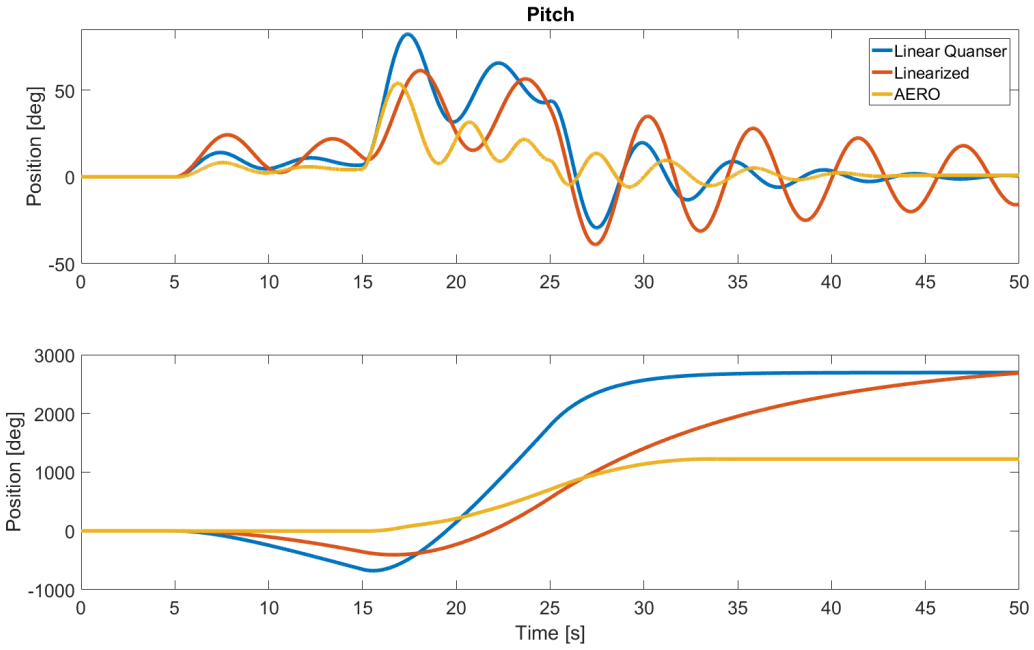


Figure 27: Model comparison: Linear Quanser, Linearized, AERO

Linear model made using the procedure described by Quanser is better then Linearized model, but still can't be compared with dynamics of the AERO. Knowing this fact, parameters of the system matrices was slightly tuned to archive better approximation of the AERO. A self-tuned version of the linear model is shown in Fig 28. In this case, state-space model becomes:

$$\begin{bmatrix} \theta_p \\ \theta_y \\ \dot{\theta}_p \\ \dot{\theta}_y \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1.45 & 0 & -0.18 & 0 \\ 0 & 0 & 0 & -0.2 \end{bmatrix} \begin{bmatrix} \dot{\theta}_p \\ \dot{\theta}_y \\ \ddot{\theta}_p \\ \ddot{\theta}_y \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.05 & 0.05 \\ -0.05 & 0.05 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \end{bmatrix} \tag{158}$$

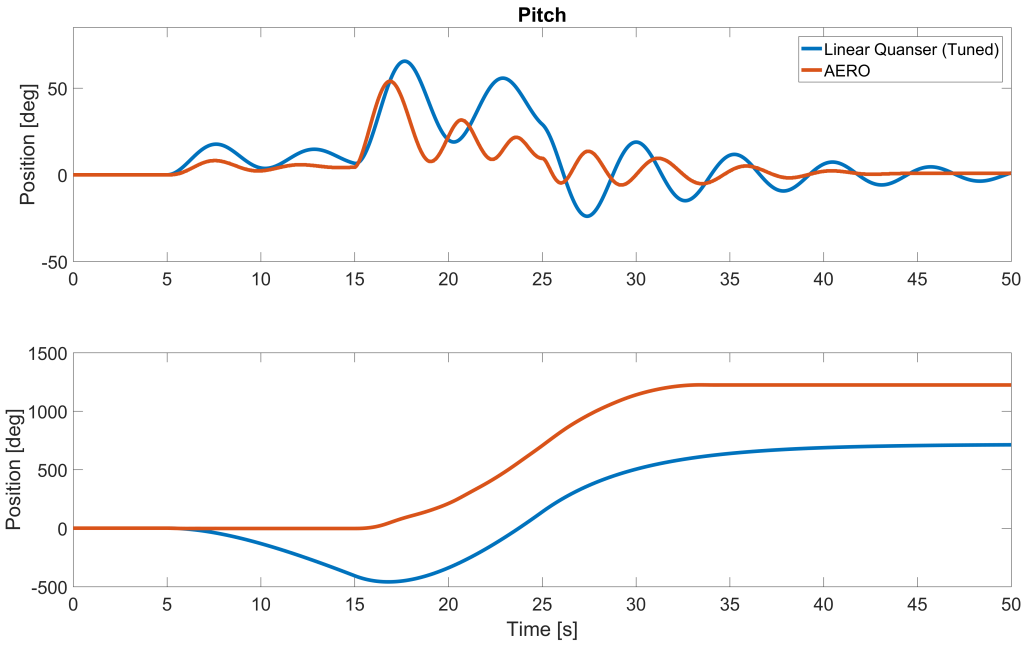


Figure 28: Model comparison: Linear Quanser (tuned), AERO

This linear model is the best approximation of AERO, that works well for input between ±24V for pitch and yaw motor. This model is called Linear model and is used in the development of the controllers further in this project.

6.4 Performance testing

Test 1

The performance of each controller is evaluated in a standardized test. This test is a combined step response sequence for pitch and yaw, as illustrated in Fig. 29. The performance of each controller is evaluated through a performance index. This index is based on the squared tracking error (a difference between the command signal and plant output) and is calculated by the formula seen in Eq. 159. Conclusions can also be drawn from the response curves of the test since the index does not capture all aspects of a response.

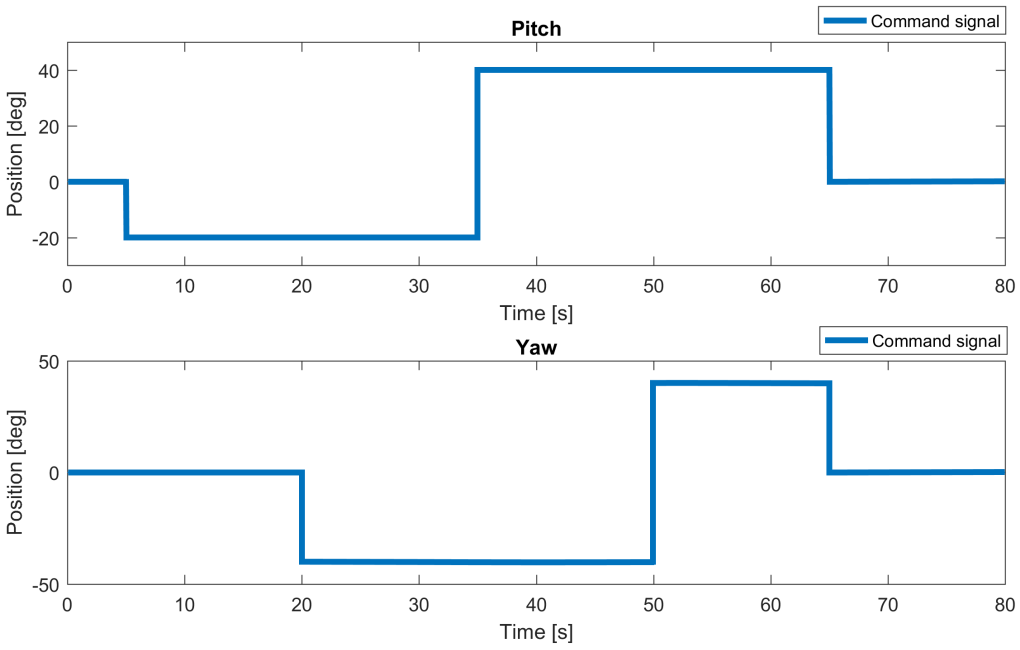


Figure 29: Command signals for pitch and yaw for the performance test.

$$performance\_index = \int_0^{80} (error_{pitch}^2 + error_{yaw}^2)dt \tag{159}$$

Each controller is tested for control of:

- the linear model (Simulation),
- the nonlinear model (Simulation),
- and the plant (Practical control),

and is tuned to minimize the performance index in each case.

Test 2

The second test is equal to the first test, except that actuator damage is emulated in the plants. The controllers have **not** been re-tuned for this test, as this would be the case if actuator damage should occur during operation. The test should give insight into how well the controllers deal with changes to the system.

The actuator damage is emulated by changing the dynamics of the actuator. For the AERO, this is achieved by exchanging one of the eight-bladed, low efficiency yaw propeller for a two-bladed high efficiency propeller. The high efficiency propeller seems to develop more thrust in one direction, far less thrust in the other direction, and less cross torque in both directions. The two propellers are seen in Fig. 30. The propeller is switched on the yaw motor.

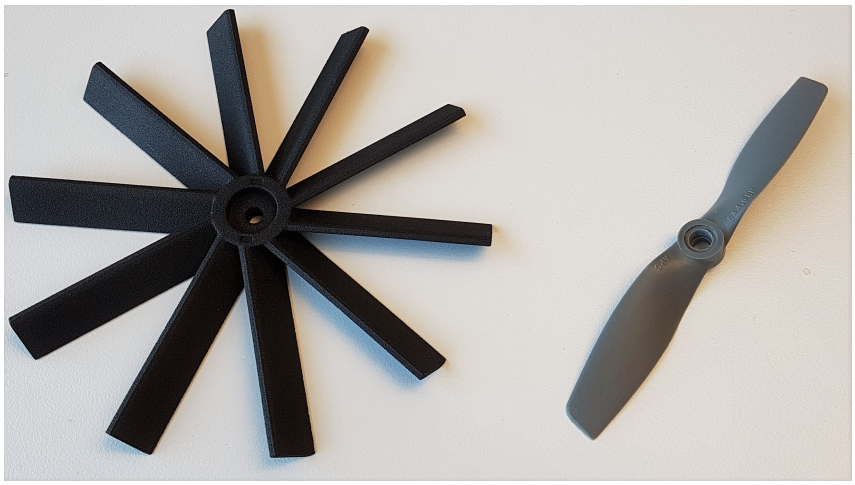


Figure 30: The original low efficiency propeller (left), and the replacement high efficiency propeller (right).

For emulated actuator damage in the linear model, the B-matrix has been edited. The 'thrust' is increased, and the 'cross-torque' is decreased for the yaw motor input. The edited B matrix is shown beside the original one in Eq. 160

$$\text{original} = \begin{bmatrix} 0.05 & 0.05 \\ 0 & 0 \\ -0.04 & 0.04 \\ 0 & 0 \end{bmatrix}, \text{replacement} = \begin{bmatrix} 0.05 & 0.06 \\ 0 & 0 \\ -0.04 & 0.01 \\ 0 & 0 \end{bmatrix} \tag{160}$$

The nonlinear model control simulation has been excluded from test 2.

6.5 Development of controllers

This section will describe the procedure of developing controllers for the AERO and its models. Input to the controllers is *Command Signal* and it is a desire position of Pitch and Yaw axis in rad. Output from the controllers are Voltage in range $\pm 24V$ used as input to the plant $u(k)$. Output from the plant is a actual position of the Pitch and Yaw axes.

6.5.1 Conventional Cascade Controller

The conventional controller is the simplest to apply to the system, as there is no need for a plant model or precise knowledge of the plants dynamics, and the setup is fairly straight-forward. The two cascade controllers (for yaw and pitch) are tuned separately, and simply added together to make the MIMO control system. The procedure is described in the steps below, and the system is visualized in Fig. 31.

1. Inner loops - pitch and yaw; Velocity feedback PI controllers were set up and tuned separately. Both outputs are limited to $\pm 24V$, and integrator clamping is enabled.*

2. Pitch feedforward; A feedforward controller was set up to cancel out the effect of gravity on the pitch body. The steady state pitch was measured for applied voltage to the motors,

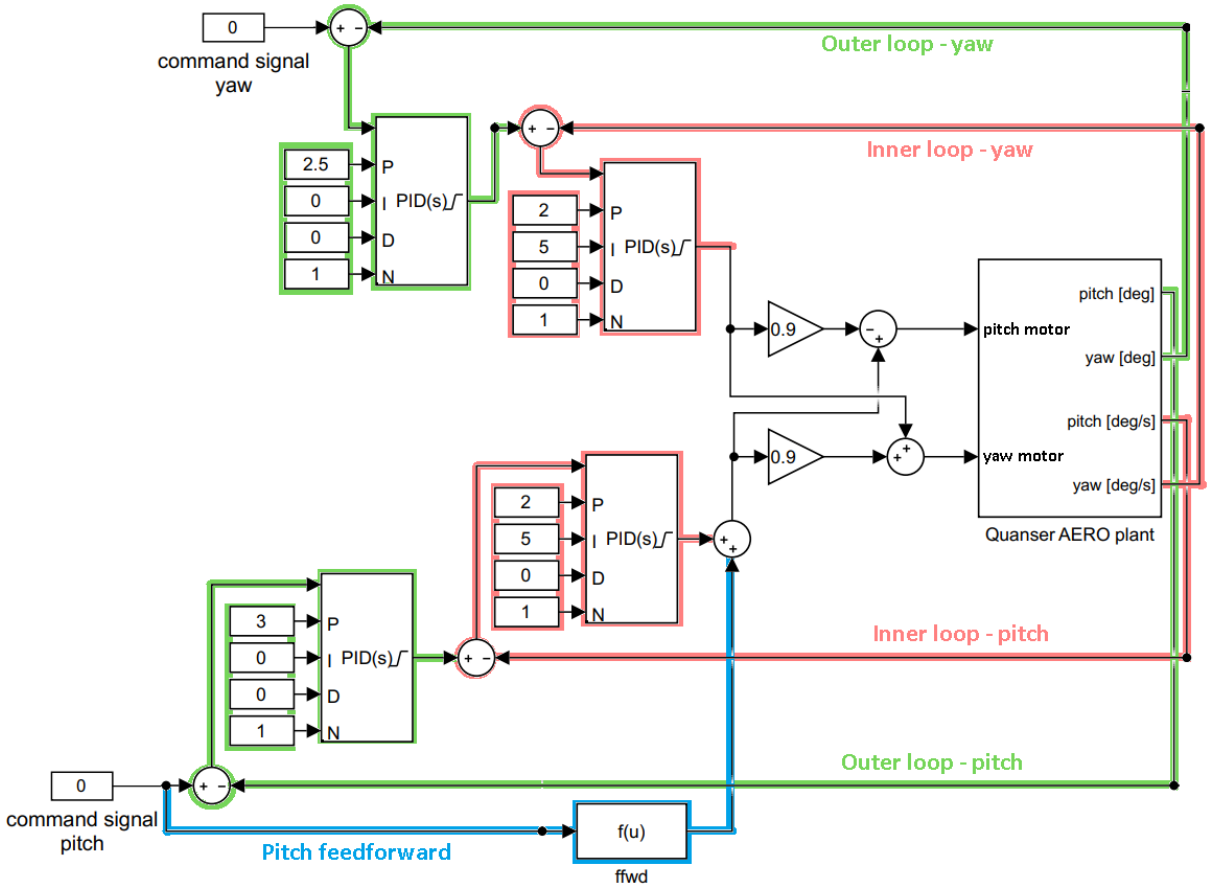


Figure 31: Conventional MIMO cascade controller for the AERO

and the inverse of this relationship is used as a feedforward from 'pitch setpoint' to applied motor voltage.

3. Outer loops - pitch and yaw; Position feedback P controllers were applied as an outer loop for the velocity controllers. Their outputs are limited to appropriate values in order to reduce maximal speed, and thereby minimize position overshoot.

* Even though the sum of the controller outputs could exceed the 24V maximum while the integrator clamping is not active, integrator windup does not seem to be a problem.

Tuning was done by adjusting the parameters of the controllers to values that gave quick responses with little overshoot for square- and sine waves of different magnitudes. Low pass filters are applied to the command signals to reduce jerk. Response to external disturbances was also considered. The output of the pitch controller is applied to both motors with the same sign due to the way that thrust and cross-torque cancel each other out, as discussed in Section 4. The same relationship applies to the yaw controller and opposite motor voltages. The 0.9 multiplier is to fine tune the relationship between thrust and cross-torque.

6.5.2 Pole placement and LQR

The controller developed in this section is a state-feedback based MIMO controller. Design of the controller can be divided into four steps. (Note that design procedure is carried out using

the Linear State Space representation of the Plant)

1. **Design of the state-feedback controller:** Before preceding with a design of feedback gains, it is necessary to check a controllability of the system. This was done using the controllability matrix. In this case, State Space model is fully controllable. Knowing this, the state-feedback controller was designed using **A, B** and weighting matrices **Q, R** as an input for the Linear-Quadratic Regulator (LQR)

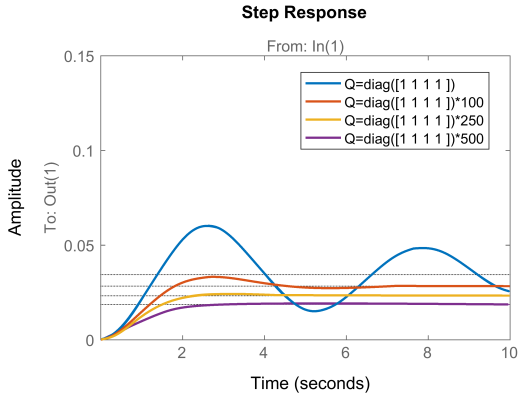


Figure 32: Step response of Pitch axes using different values of Q matrix

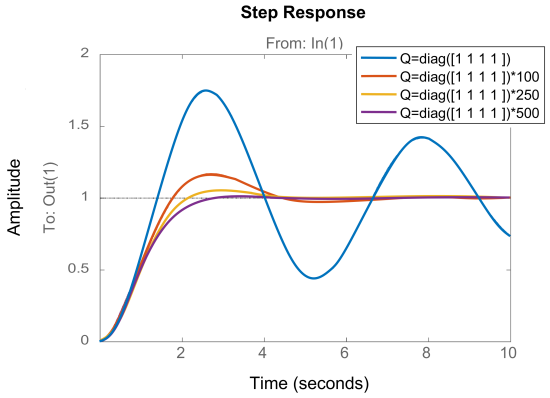


Figure 33: Step response of Pitch axes using different values of Q matrix with Pre-filter

Initially, elements of Q and R matrix was chosen to be equal 1, and by increasing values of Q matrix, desired dynamic of the system was archived. Impact of Q-matrix on the controller is shown in Fig. 32. Increasing values of Q-matrix, increased stability of the system. However, steady state error increased as well. The response of the closed loop controller on a linear model is shown in Fig. 36 (Red signal).

2. **Pre-filter:** From Fig. 36, it can be seen that system have a high steady state-error. This error can be removed by adding *Pre-Filter* (For more information look Sec. 5.2.6). This is done by multiplying *Command Signal* by Eq. 44. A response of the state-feedback with the Pre-Filter system can be seen in Fig. 36 (Orange signal). Steady state error was removed width great success.

3. **Observer:** Due to poor resolution and general limitation of the tachometers used in AERO, some part of the signal is missing round equilibrium point. There are two possible way to reconstruct missing part of the signal, by use of *Gyro* or *Observer* Due to the fact that mathematical model of the plant is available, it was decided that use of the observer will be sufficient. Before preceding with the design, it was necessary to check if model/plant is observable. In this case, the system is fully observable. The general setup of the state-feedback controller with Observer and Pre-filter is shown in Fig. 34.

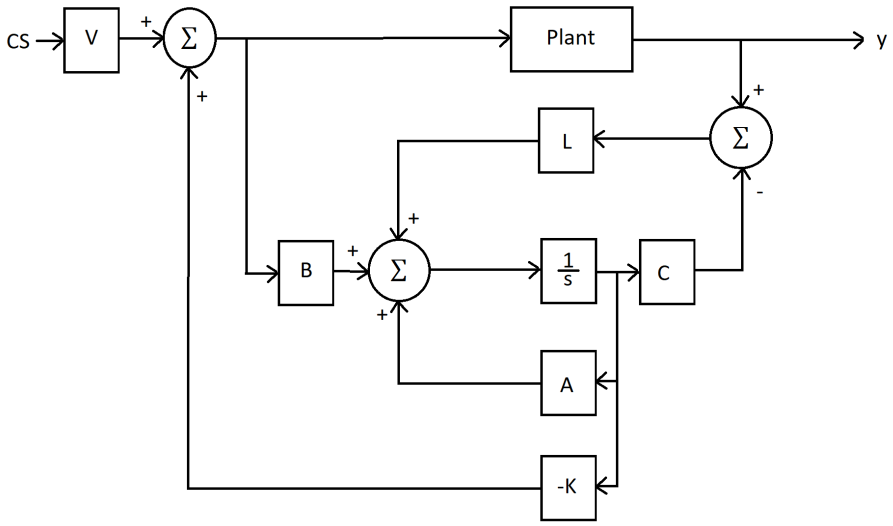


Figure 34: The feedback controller with Observer and Pre-filter

The luenberger gains **L** was found using Poles placement method. The Observer poles were seven times further to the left (in the Complex plane) then poles of the closed-loop system. The response of the state-feedback with Observer and Pre-Filter can be seen in Fig. 36 (Purple signal). Use of the observer makes it possible to reconstruct missing signals, however steady-state error was obtained.

4. **Integral Action:** In order to reduce steady state error from Observer, an integral action was added to the controller. The new controller is shown in Fig. 35

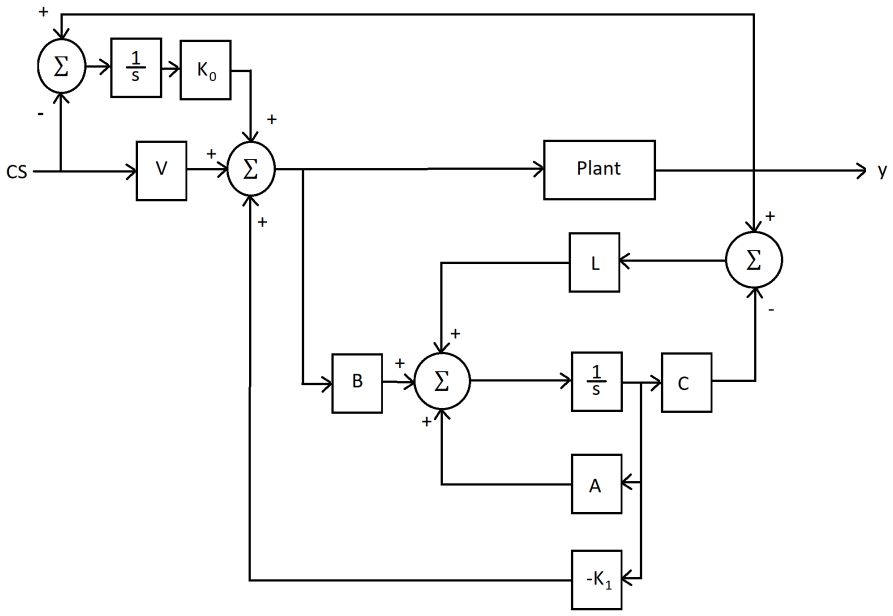


Figure 35: Finally State Space based controller

The feedback gains were recalculated including two new poles, one for pitch and one for yaw. A response of the final controller can be seen in Fig. 36 (Green signal). The steady

state error is removed and all states are reconstructed. Resulting controller (shown in Fig 35) is donated as *LQR controller* further in this rapport.

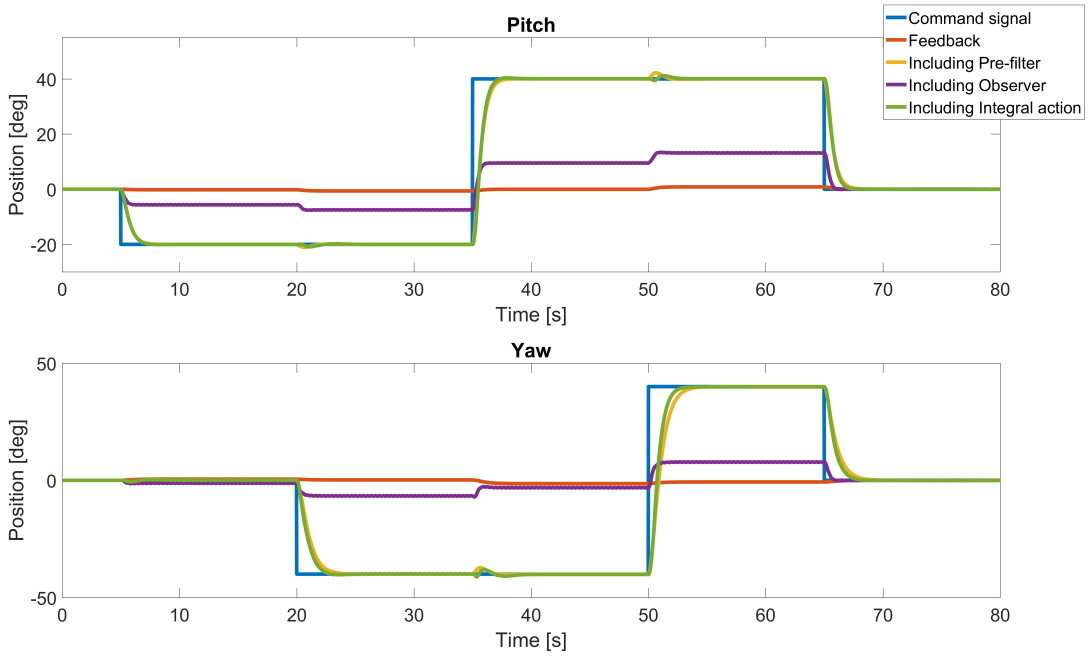


Figure 36: Test of State Space based controller using Performance Test

6.5.3 Model Predictive controller (MPC)

For a better understanding of MPC, the controller can be divided into three different steps as fig. 37 shows. First two steps are the calculation made directly in Matlab. These calculations are made only once, and are shown with grey lines. The last step is Hardware in the loop simulation. In this case values that change over time are shown with blue lines.

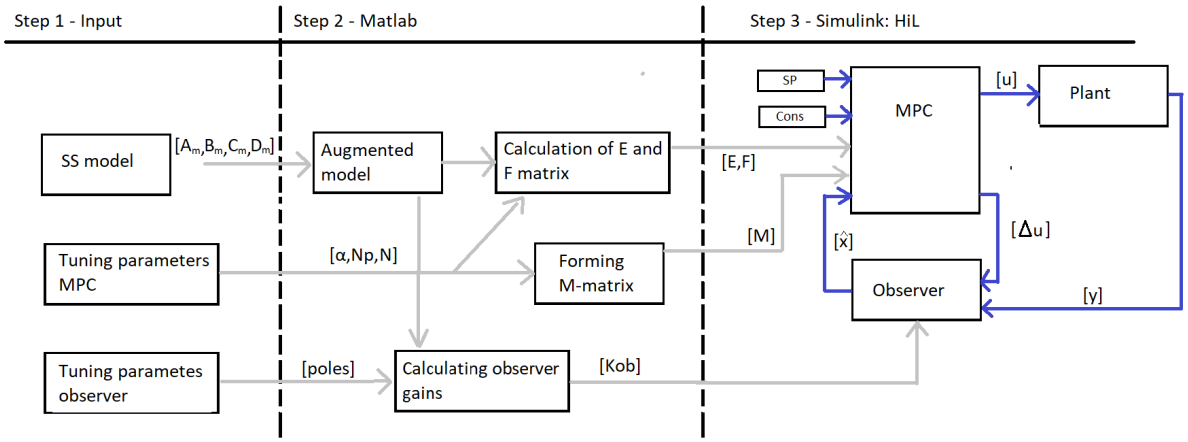


Figure 37: MPC flowchart

where:

- **Step 1:** This is initialization step, where three different sets of parameters need to be defined before proceeding with Step 2. This is:
  - **SS model:** As mentioned before, MPC is based on a mathematical model of the plant which is augmented State Space representation (A,B,C,D) of linearized model described in section 5.1.2
  - **Tuning parameters MPC:** These parameters are used to obtain desired performance of the MPC and were found by trial and error.
  - **Tuning parameters observer:** Contains vector with desired poles placement that is necessary to calculate observer gains [Kob]. Placement of the poles was found by trial and error.
- **Step 2:** Using input values from Step 1, matrices E,F and M can be calculated based on the theory presented in section 5.3.5 and 5.3.6. These matrices are used in Quadratic programming. Those calculations must be done before proceeding with Step 3.
- **Step 3:** All calculations presented in this step are made in the Simulink. However, constant values (shown as grey lines in Fig. 37) are transferred directly from the Matlab as parameters. Project group decided to use the Observer in order to reconstruct missing part of output signals from the plant due to a practical limitation of tachometers used in the AERO.

  Two *Matlab function* within Simulink was used, one for the *Observer* and one for the *MPC*. In order to calculate the observer states Eq. 78 is used.

  Calculation of control signal using function *MPC* is more complex and can be divided into three steps for each time step:

  1. Create/update matrix γ used in Quadratic programming. This matrix must be included in real time simulation due to fact that values used to find optimal value of cost function J including constraints on input ($u(k)$) must be updated with most recent value of $u(k-1)$ and $x(k)$ for each step. For more detail see section 5.3 **MPC with Constraints**
  2. Quadratic Programming is used to find the best solution of cost function ($J = \frac{1}{2}\eta^T E\eta + \eta^T H$) based on constraints defined as $M\eta \leq \gamma$
  3. Calculation of Δu and u.

Tuning parameters in MPC

- **Prediction horizon (P_h):** According to [12], Prediction horizon concept is not one of the tuning parameters. However, it is necessary to choose this value right. The thumb rule says that Prediction horizon must be greater than settling time of the plant. The approximated settling time of the plant is 2s. Sufficient sample time is 0.02s. That gives minimal prediction horizon equal to 100 steps. Prediction horizon was chosen to be 150

- **Number of Languerrier networks:** N=[N1 N2] and a=[a1 a2]. Parameter N is a number of the Languerrier network used to capture control trajectories for each input. By increasing number of N terms, increase the ability to capturing unstable and complex signals.

  Parameter a is Poles placement of the discrete-time Laguerre network for each input. This values should be $0 < a < 1$ in order to ensure the stability of a system.

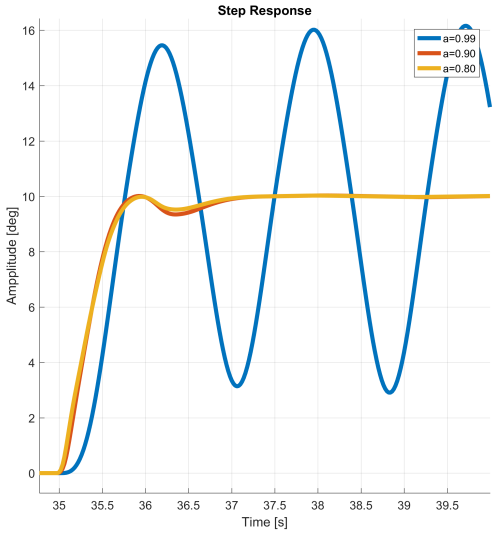


Figure 38: The output response with different a values

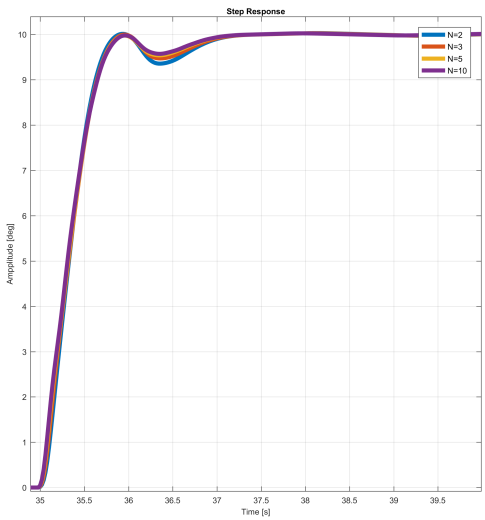


Figure 39: The output response with different N values

Parameters **N** and **a** was determined by testing different combination. Fig. 38 shows different values of **a** when **N** is sett to be equal 3 for both inputs. The values of a=0.85 give the best response of the system. Higher values make the system unstable. In the case when $a < 0.85$ controllers become too aggressive and it starts to oscillate close to Command Signal. Next, different N values were tested. Initially, N was sett to be 2 for both channels, this value was gradually increased. Fig. 39 shows different values of **N** (when **a** is constant for both inputs). By increasing N from 2 to 5 error between output trajectory and Command signal is slightly reduced. However, a further increment of N gives none visible reduction of the error. Knowing this fact N was chosen to be 5 for both inputs.

- **Observer Gains:** Observer poles was placed at

$$Poles_{obs} = \begin{bmatrix} 0.5985 & 0.5990 & 0.6000 & 0.6010 & 0.6020 & 0.6030 \end{bmatrix} \tag{161}$$

This parameter was found by trial and error, Fig 40 shows different combination of poles placement. Slow observer introduces oscillation into the system. In could be reasonable to make the observer poles placed around zero values (in *Uniny cycle*) however, very fast observer dynamics increase noise sensitivity and makes plant unstable.

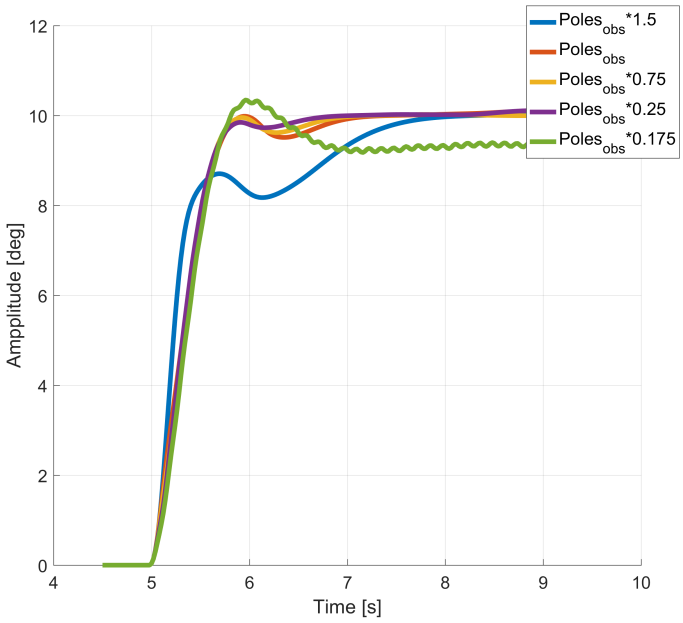


Figure 40: Observer Gains

6.5.4 Model-Reference Adaptive Control (MRAC)

The MIMO MRAC system is based on the conventional cascade controller from section 6.5.1. The reasoning is that the P-PI cascade controllers (for pitch and yaw) work well, and might be improved by having adaptive inner loops. The outer loops would work as usual, and the inner loops would provide the desired velocities regardless of changes to the plant. The MIMO MRAC system uses the same pitch feedforward as the conventional cascade controller as well.

By viewing the inner loops as first order systems, we apply the MIT rule based MRAC from Sec. 5.4. This means that we exchange the PI controller for a 'KV' controller, which has separate gains for feedforward and feedback, and is also capable of eliminating steady state error. The structure of the system is presented in Figure 41. Two alternatives to the MIT rule approach are presented in the literature review (section 3.2), which are the Normalized MIT rule, and the Lyapunov based MRAC. The former should reduce oscillatory effects at large input values, the latter should guarantee stability, and both are claimed to allow arbitrarily large adaption gains. The resulting three adaption laws are presented in Table 3.

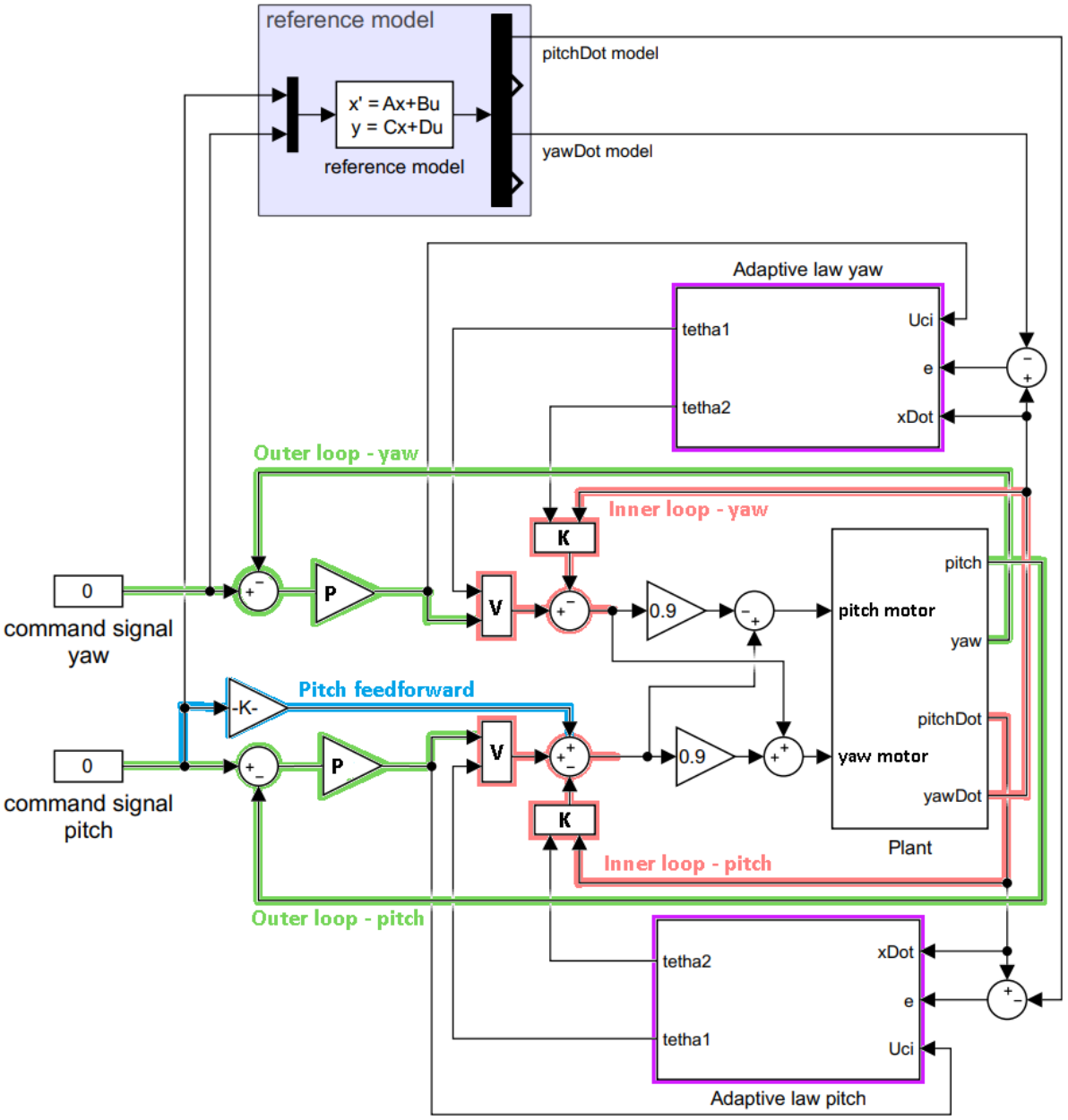


Figure 41: Structure of MIMO MRAC system. The outer loop (green) and pitch feedforward (blue) is the same as for the conventional cascade controller. The inner loop (orange) uses separate gains for feedforward and feedback, and these gains (control parameters) are calculated by the adaptive law in the (purple) subsystems.

MIT rule	Normalized MIT rule	Lyapunov based
$\frac{d}{dt}\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix} = -\gamma e \frac{\partial e}{\partial \theta}$	$\frac{d}{dt}\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix} = \gamma e \frac{\phi}{\alpha + \phi^T \phi}$	$\frac{d}{dt}\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix} = -\gamma e \begin{bmatrix}u_{ci}\\ -\dot{x}\end{bmatrix}$
$\frac{\partial e}{\partial \theta} = \frac{a_m}{s + a_m}\begin{bmatrix}u_{ci}\\ -\dot{x}\end{bmatrix}$	$\phi = -\frac{a_m}{s + a_m}\begin{bmatrix}u_{ci}\\ -\dot{x}\end{bmatrix}$	

Table 3: Table showing the three alternatives for MRAC adaption laws in the MIMO MRAC system. These are applied in the purple subsystems of Figure 41. The theory behind these is explained in Sec. 5.4

These three versions of MIMO MRAC have been tested, and are compared in Fig. 42, where the controlled plant is the linear model of the AERO, and the reference model is a closed loop version of this model. The plotted response is movement about the yaw axis, and pitch is at zero degrees. In each plot, the adaptive controllers are using the same adaption gain; γ. The 'no adaption' response is $\gamma = 0$, and is equal to a (poorly tuned) P-cascade controller. The goal of the adaptive control is to make the output of the controlled plant equal to that of the reference model, which again follows the command signal. From this test, it is apparent that the 'MIT rule' and 'Lyapunov' controllers are working as intended, and at $\gamma = 10$, the outputs of these two are almost identical to that of the 'reference model'. The 'Normalized' controller does not show the same level of performance for $\gamma = 10$, but further testing reveals that much(!) larger values of γ gives almost identical output between the reference model and the 'Normalized' controller. These responses can be seen in Figure 43. Increasing gamma further (beyond 10 for 'MIT rule' and 'Lyapunov, and beyond 100k for 'Normalized') causes the simulations to crash due to insufficiently small time-steps. The adaptive control parameters for the $\gamma = 10$ response are included in the second plot of the aforementioned figure, to give a visual representation of the adaption (change in control parameters over time).

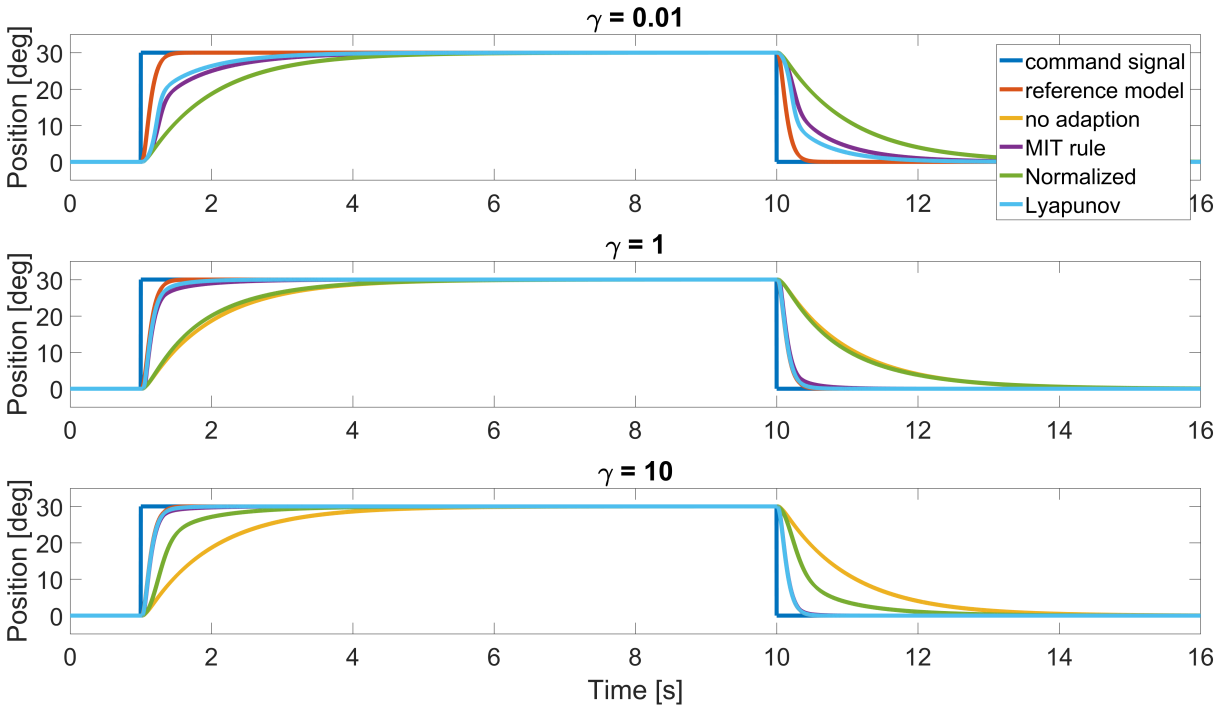


Figure 42: Linear system control performance of MRAC at different adaption gains.

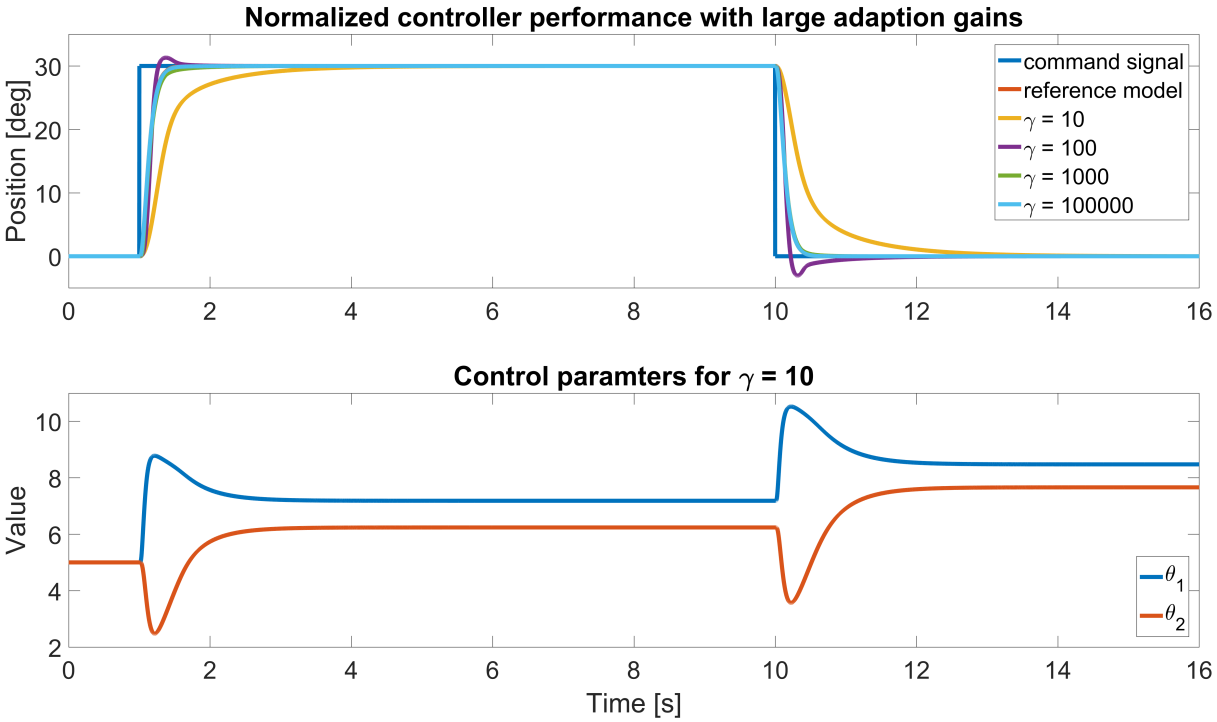


Figure 43: Control performance of the 'Normalized' controller at large adaption gains. The second plot is of the control parameters for the $\gamma = 10$ response.

As all three MRAC versions work well in control of linear systems, we have also tested how well they can perform control of the nonlinear AERO model. In Fig. 44, the performance test presented in section 6.4 is used to evaluate the MRAC versions in control of a nonlinear plant. All three controllers are tuned (by pole placement of reference model, adaption gain γ, and

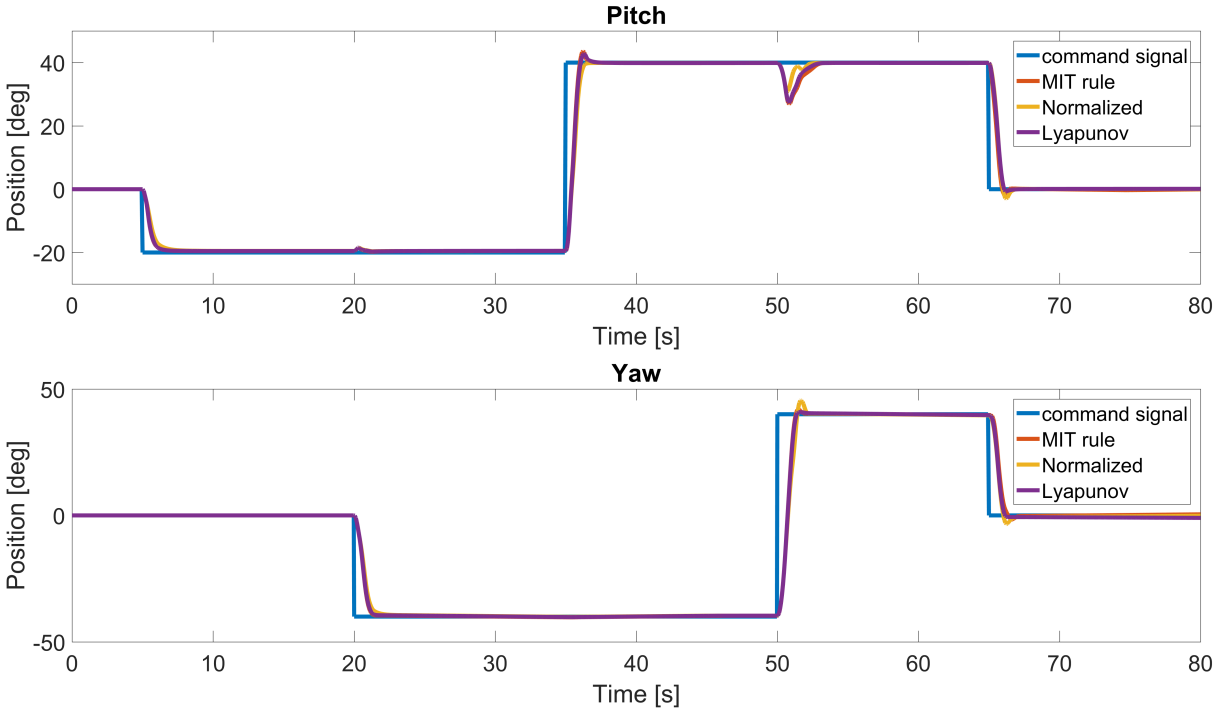


Figure 44: Comparison of MRAC versions for nonlinear plant control.

in the case of the 'Normalized' controller; α) to minimize the value of the performance index. Due to actuator saturation of the controlled nonlinear model, the adaption rates are set to zero if either input to the nonlinear model is outside of $\pm 24V$. A constraint of ± 24 is placed on the control effort of the linear reference model, to emulate actuator saturation in the reference model, which increases performance of the overall system, since the system will not be trying to track a signal that it can not reach.

This comparison gave almost the same performance for all three versions. Therefore, another comparison is made. Here, the same controllers (with unchanged tuning) is used for an edited version of the test (with larger steps, and steps in pitch and yaw simultaneously). This means that the controllers are operating outside of the work area they were tuned for, and since this is not a linear system, that impacts the performance. The result is seen in Fig. 45.

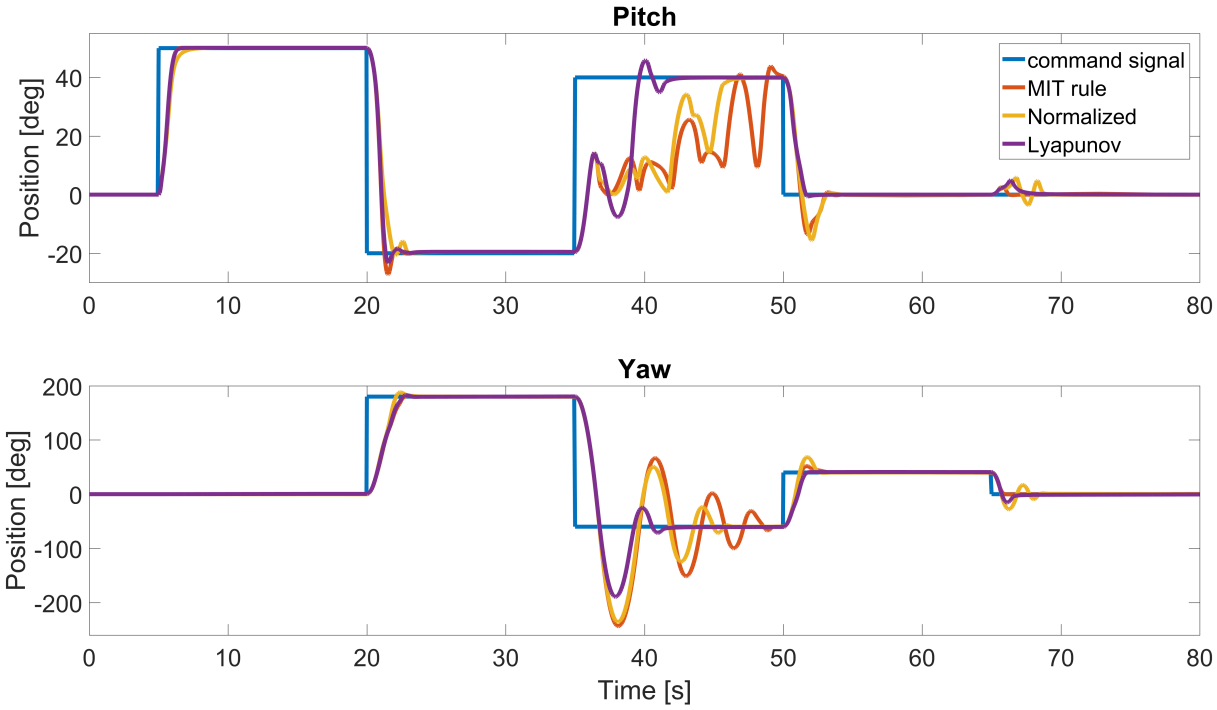


Figure 45: Comparison of MRAC versions for nonlinear plant control, with edited test.

Since the performance of the Lyapunov based MRAC controller is the least terrible, it is chosen as the MRAC type which is used in comparison with conventional, LQR, and MPC controllers.

6.6 Solver Parametes

In this project, Matlab Simulink was used to make RT (Real-time) simulation. Simulink can operate with different Solver types, like *Continuous Time* (ode1, ode2, ode3 et al.) and *Discrete*. Additional, solver can be used with fixed- or variable-steps time. For simulation of conventional, LQR and MRAC continuous solver was used with fixed time steps. In this cases, Simulink chose appropriate solver that is necessary to solve all equations in Real Time. For simulation of MPC, discrete time solver was used with step-size 0.02s. Additionally, the *System Timebase* function block was used in all simulation. The main task of this block is to checks if chosen Step-size is appropriate, if not simulation is terminated.

	Solver	Step-size [s]
Conventional	Continuous	0.002
LQR	Continuous	0.002
MPC	Discrete	0.02
MRAC	Continuous	0.002

Table 4: Solver type

7 RESULTS

7.1 Control performance; Test 1

This is the first test of controller performance, as described in section 6.4. Plots of the tests are supplied in Appendix A-C.

	Linear Model	Nonlinear Model	AERO
Conventional	6	79	70
LQR	74	78	75
MPC	27	71	69
MRAC	16	75	73

Table 5: Performance indices from the first test. Lower is better.

7.2 Control performance with actuator damage; Test 2

This is the test of controller performance wi emulated actuator damage, as described in section 6.4. Plots of the tests are supplied in Appendix D-E.

	Linear Model	AERO
Conventional	6	121
LQR	167	127
MPC	108	98
MRAC	11	191

Table 6: Performance indices from the second test. Lower is better.

8 CONCLUSIONS

The results of this project do not produce straight-forward conclusions, but we will try to highlight the important aspects of the results here.

The simulations of control of the linear model in Test 1 show that the conventional controller, the MPC, and the MRAC could be tuned follow the command signal nearly perfectly (especially the conventional controller), which we would claim confirms the theory behind the controllers. The LQR also achieves good results, but seems to be limited by the relative speed of the controller and the observer.

The linear model simulations for Test 2 show equally good performance as Test 1 for the conventional controller and the MRAC. This is logical, since the conventional controller and the plant is linear, and will give the same response regardless of amplitude on the signals. Similarly, the MRAC is built on the structure of the model, but not tuned to its specific values, it rather finds parameters that fit the values of the plant. This is why the controller works similarly, and the small difference in the performance index is simply due to the initial values of the adaptive parameters. (The initial values are closer to their 'correct' values in Test 2 than in Test 1)

The other two controllers in the linear model simulations in Test 2, the LQR and MPC, show a drastic decline in performance. Both systems experience large oscillations at several instances. This must be due to the fact that these two controllers are based on, and tuned for, the model of the system. Both use 'knowledge' about the system, and since this knowledge is no longer accurate, the performance declines.

The nonlinear model control simulations in Test 1 show that the nonlinearities in the plant limits the performance of the system. The most significant factor in this is certainly the actuator saturation present, and the nonlinear functions may play a role as well. The actuator saturation will produce a theoretical limit to the performance of the system, and we believe our controllers are not too far from this limit, considering the good responses seen in the plot. It is also observed that the controllers have more or less the same performance.

Test 1 for control of the AERO shows very similar (good) results to the nonlinear model simulation. The plots are perhaps a little less smooth, which is a result of the unmodelled dynamics and disturbances, but these effects are insignificant. The performance index values are slightly lower than in the nonlinear model simulation, which indicates that the nonlinear model is somehow a little harder to control, however, this is also insignificant. We should specify that the performance index values for the AERO are the values for the specific plots seen in the appendices, and not averages. Variations in performance index of ± 2 have been observed form test to test of the same controllers, but we do not consider these small changes significant.

The change in actuator dynamics from Test 1 to Test 2 had virtually no effect in the linear simulation of the conventional controller and the MRAC. This was, however, **not** the case for the practical application of the control methods on the AERO. We can assume that the change in actuator efficiency limits the performance of the controllers here as well, but this does not account for the decline in performance, which can be asserted from the fact that the MPC achieves a much better performance index, despite some oscillation. The plots show that the performance of the conventional is not terrible, it is rather just ill-tuned. The MRAC, on the other hand, shows massive steady state errors, the cause of this is discussed in Sec. 9. Otherwise,

the response of the MRAC is not terrible either, meaning that it reaches its steady state error fairly quickly, and has virtually no overshoot.

Notice that the feedforward used in both the conventional controller and the MRAC is no longer correct for Test 2 (since we have changed the actuator output). Nevertheless, both controller work excellently in the linear test, and the conventional controller still manages to find zero steady state error in the AERO control.

Test 2 on the AERO for the LQR and the MPC show similar, but not as large, oscillations as linear Test 2. The reason for the decrease in performance from AERO test 1 must be the same as for the linear tests, and the smaller oscillations in the AERO test 2 compared to the linear Test 2 indicate that the change of propellers is actually less significant than the change of the B-matrix. This is reasonable, since the two actuator changes were not tuned to have the exact same effect. It should also be noted that the LQR controlled plant hit the physical barrier of 60 degrees pitch at around 53 seconds, so this response might have looked even worse.

Overall, this project has shown that when it comes to development of controllers, it is important to consider the practical application of them. Linear models are not necessarily good, especially when it comes to propeller-driven flight and similar processes. The LQR and MPC rely on linear models, and they work well when tuned to a specific case, but the performance crumbles when the system is changed without changes to the controller. The linear test 1 and 2 for the conventional controller and MRAC give excellent results, which confirms that the controllers work well in theory, and when it comes to practical application, the conventional controller continues to work relatively well. The MRAC, however, shows strange behaviour, which was not anticipated.

An important observation for these tests is also that none of the systems became unstable. It is not guaranteed that they are stable for all scenarios, and it is rather likely that they will not be, but in these cases all remained use-able.

One final note on the conclusion is that the conventional controller is by far the easiest controller to implement. A lot of time was spent during this project to research and try to understand the MPC and MRAC, and how to use them. Even when comparing only the controller scipts themselves, the conventional controller is the simplest one. And still, the conventional controller subjectively outperforms the other controllers (even though the MPC has a better score in AERO Test 2, it is still more oscillatory, which is undesirable), which makes it easy to deem this the best choice for a controller in this application. The other methods, especially the MRAC, work very well in theory, and seems to have a lot of potential, but further work is needed to fully understand the method, and to apply it in a manner that achieves the results we are looking for; a controller that effortlessly tunes itself to the best performance it can have regardless of changes to the system. We feel that this project did a fair job of evaluating the controller methods, and inspires further research into advanced control topics, especially adaptive controller like the MRAC.

A quote from [1] that fits quite well with these results;

"In previous chapters we showed that adaptive control can be very useful and can give good closed-loop performance. However, that does not mean that adaptive control is the universal tool that should always be used. A control engineer should be equipped with a variety of tools

and the knowledge of how to use them. A good guideline is to use the simplest control algorithm that satisfies the specifications."

9 DISCUSSION

This section will address possible sources of error in the execution of the project, suggestions to what could have been done differently, and thoughts on further work. This section will mainly discuss the MRAC, since the other controllers are largely working well, and as expected.

MRAC

The practical application of MRAC in Test 2 showed a large steady state error, which was unexpected. The reason for this must be that the relationship between the adaptive parameters θ_1 and θ_2 somehow counteracts the 'effort' of the outer loop P controller. Looking at the first and second plot of Fig. 47 and 47, we see that the steady state errors actually make sense. During steady state errors, the velocity of the plant matches the velocity of the model, which it is supposed to. During the changes in position though, the velocity of the plant can not follow the reference model due to actuator saturation. So the steady state error in position is actually the integrated velocity error between the plant and the reference model. It is then assumed that slowing down the response of the reference model until the plant could follow it, without reaching actuator saturation, would completely remove the steady state error.

This gives the conclusion that the problem is the combination of controller setup and actuator saturation. Initially, actuator saturation was not considered as the problem since the MRAC worked so well for AERO control in Test 1. However, it seems that the problem itself is present here as well, only not noticeable, since the actuators are able to keep up with the reference model. The same must be true for the nonlinear control of Test 1.

Since we can not do anything about the reality of actuator saturation, focus should be put on fixing the setup of the controller. During the course of the project, several setups of MRAC were tested. Among those was one that used the same adaptive KV setup for both the inner and outer loops. Attempts were also made at developing our own adaption laws, for example by combining the position and velocity loops into one adaptive setup, or by adding integrators to the adaptive law. Unfortunately, the development of MRAC adaption laws for more complicated systems proved difficult, as none of these controllers worked as they should in simulations. Typically, the resulting controllers were unpredictable and unstable.

Another issue that should probably be mentioned is that the adaptive parameters of MRAC would sometimes go beyond zero in the nonlinear and AERO applications, causing positive feedback and instability. They have since been limited to a minimal value of zero, but we can see that the parameters occasionally hit zero even in our final test (Fig. 46 and 47). It seems that there is actually nothing stopping them from doing so in the original MRAC method, it is just assumed that they will not.

Further work should be put into the research of these adaptive controllers, specifically to understand their complicated behaviour, and how we can apply them to more complicated models such as high order, highly nonlinear MIMO systems.

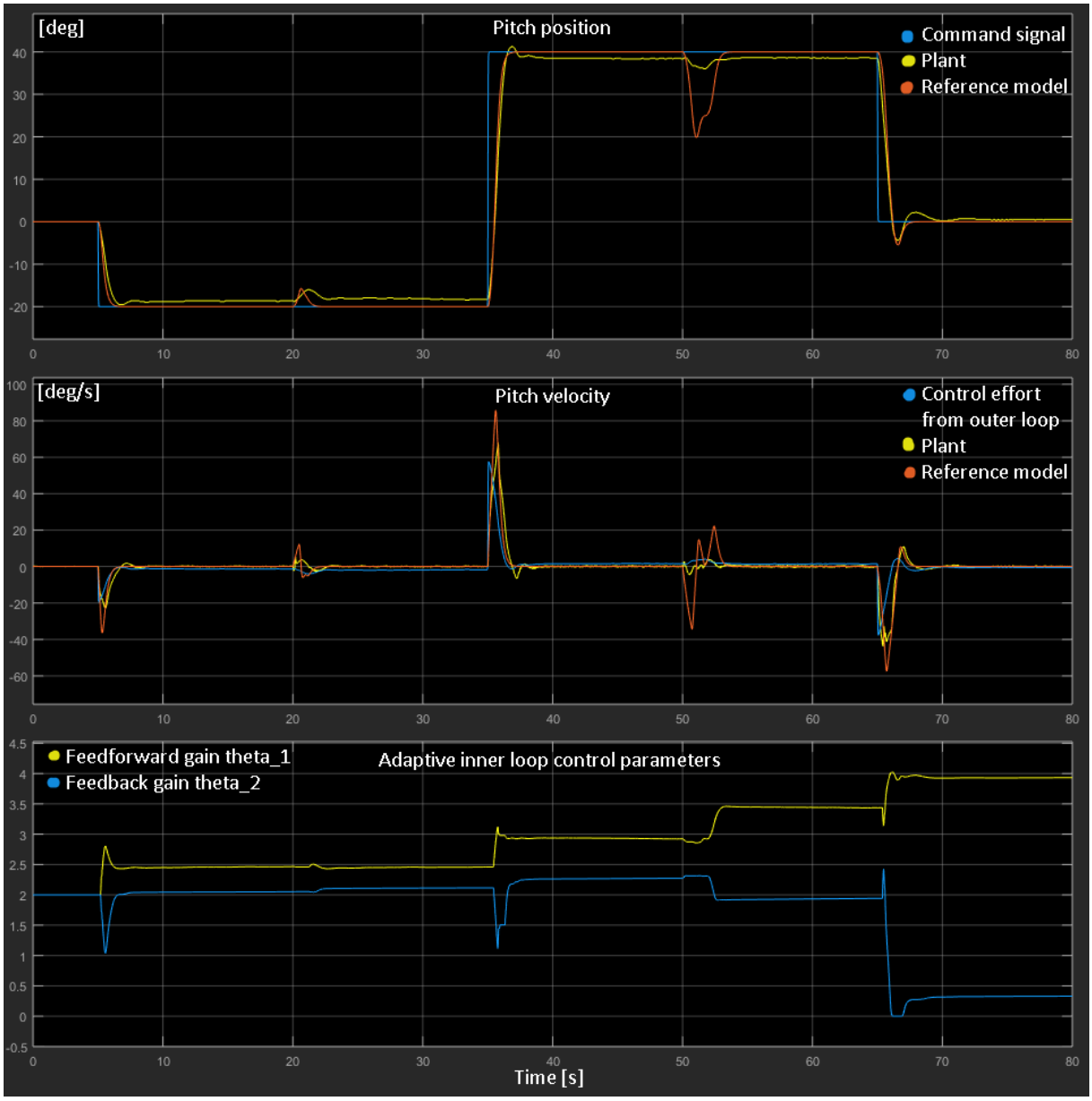


Figure 46: Plot of position, velocity and inner loop control parameters for pitch in Test 2: AERO control

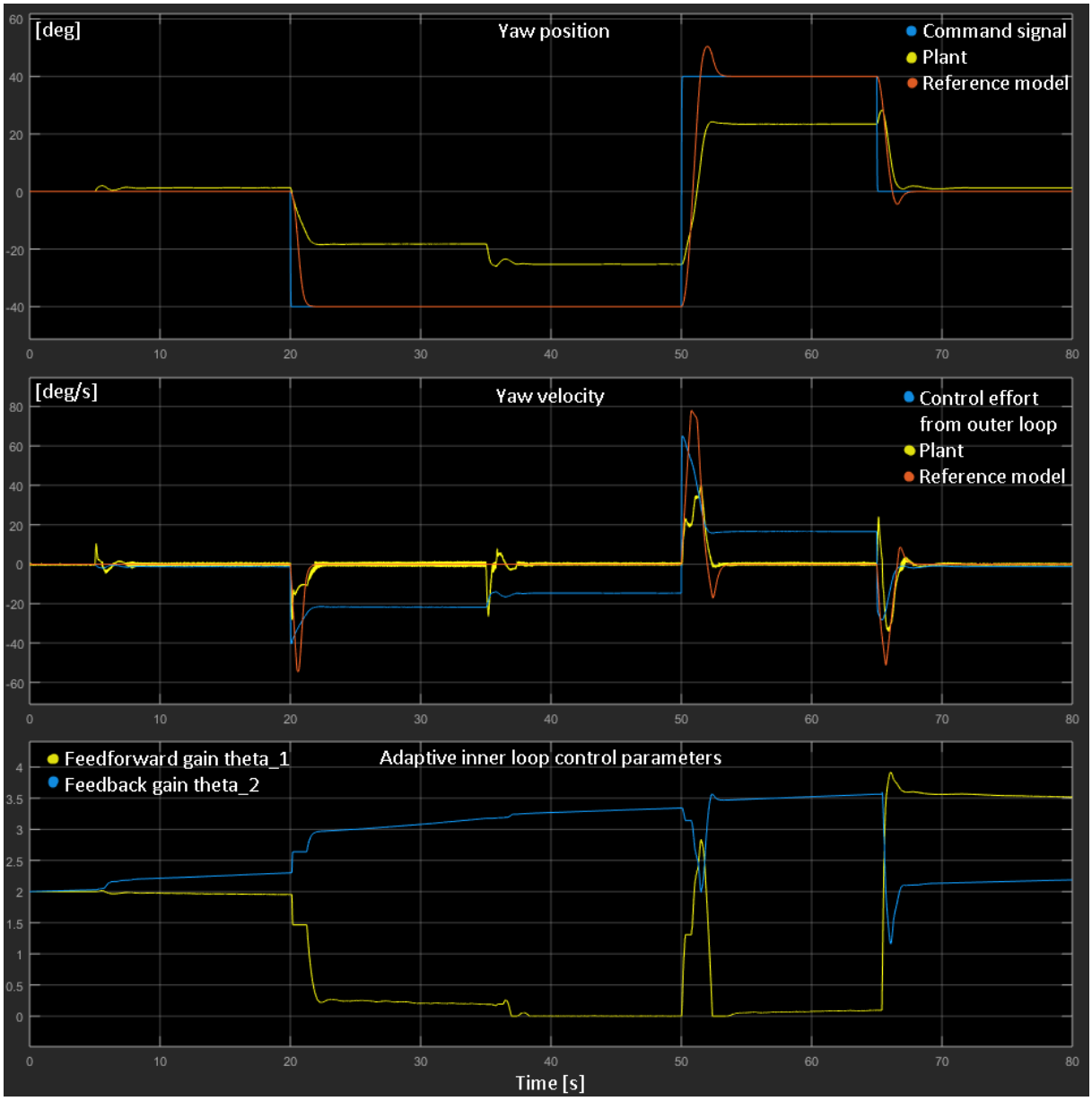


Figure 47: Plot of position, velocity and inner loop control parameters for yaw in Test 2: AERO control

10 REFERENCES

[1] Åström, K.J., and Wittenmark, B. (2008) *Adaptive Control*. 2nd ed. New York: Dover Publications, pp. 1-33, 185-255.

[2] Landau, I.D., Lozano, R., M'Saad, M., Karimi, A. (2011) *Adaptive Control: Algorithms, Analysis and Applications*. 2nd ed. Springer-Verlag London, pp. 1-33.

[3] Maciejowski, J.M. (2002) *Predictive Control with Constraints*. New Jersey: Pearson Education (US)

[4] Ruf M. (2014) *Model Predictive Control of a Quanser 3DOF Helicopter*. Stuttgart: University of Stuttgart.

[5] Lopes, R.V., Galvao, R.K.H., Milhan, A.P., Becerra, V.M., Yoneyama, T. (2006) *Modelling and Constrained Predictive Control of a 3DOF helicopter*. Available at: https://www.researchgate.net/publication/228783626_Modelling_and_Constrained_Predictive_Control_of_a_3DOF_Helicopter?enrichId=rgreq-757f06fb-8a2d-4780-b2a6-c11d341a9f45&enrichSource=Y292ZXJQYWdlOzIyODc4MzYyNjtBUzozMTcwMTY2ODA5MjcyMzJAMTQ1MjU5NDA3NTQ5Nw%3D%3D&el=1_x_3.(Accessed: 18.03.2018)

[6] Ramalakshmi, A.P.S. Manoharan P.S, Harshath K, Varatharajan, M. (2016) Model predictive control of 2DOF helicopter. *International Journal of Innovation and Scientific Research*. 2016 2 Jun. ISSR Journals, pp. 337-346. ISSN 2351-8014 Vol. 24

[7] Wang, L. (2009) *Model Predictive Control System Design and implementation Using MATLAB*. Springer-Verlag London.

[8] Wang, L. (2003) Discrete model predictive controller design using Laguerrer functions,*Journal of Process Control*. Volume 14, pp. 131–142 doi: https://doi.org/10.1016/S0959-1524(03)00028-3

[9] Luenberger, D.G. Ye,Y (2008) *Linear and Nonlinear Programming*. New York: Springer

[10] Boyd, S. Vandenberhe, L (2004) *Convex Optimization*. Cambride: Cambridge University Press

[11] Rossiter, A. (2014) *Model predictive control (Anthony Rossiter)* Available at: https://www.youtube.com/watch?v=-ahA3kkIqD0&index=3&list=PLs7mcKy_nInFEpygo_VrqDFCsQVnGaoy (accessed: 2018, May 07)

[12] Rossiter, A. (2014) *Model predictive control (Anthony Rossiter)* Available at: https://www.youtube.com/watch?v=w_GA_7sNM3g&index=1&list=PLs7mcKy_nInFEpygo_VrqDFCsQVnGaoy- (accessed: 2018, May 07)

[13] Swarnkar, P., Jain, S., and Nema, R.K. (2011) Effect of Adaption Gain in Model Reference Adaptive Controlled Second Order System, *ETASR*. Vol. 1, No.3, pp. 70-75. Available at: http://www.etasr.com/index.php/ETASR/article/view/11/98 (accessed: 2018, April 19)

[14] Ehsani, M.S. (2007) Adaptive Control of Servo Motor by MRAC Method, *IEEE VPPC*. Arlington, TX, USA: 2007 September 9-12. doi: 10.1109/VPPC.2007.4544102

[15] Jain, P., and Nigam, M.J. (2013) Real Time Control of Ball and Beam System with Model Reference Adaptive Control Strategy using MIT Rule, *ICCIC*. Enathi, India: 2013 December 26-28. doi: 10.1109/ICCIC.2013.6724180

[16] Kochummen, S.A., Jaffar, N.E., and Nasar, A. (2015) Model Reference Adaptive Controller Designs of Steam Turbine Speed Based on MIT Rule, *ICCC*. Trivandrum, India: 2015 November 19-21. doi: 10.1109/ICCC.2015.7432861

[17] Jain, P., and Nigam, M.J. (2013) Design of a Model Reference Adaptive Controller Using Modified MIT Rule for a Second Order System, *AEEE*. Vol.3, No.4, pp.477-484. Available at: https://www.ripublication.com/aeee/62_pp%20%20%20477-484.pdf (accessed: 2018, April 19)

[18] Chirag (2016) *Simple Adaptive Control Example* Available at: https://se.mathworks.com/matlabcentral/fileexchange/44416-simple-adaptive-control-example (accessed: 2018, April 20)

[19] Swathi, M., and Ramesh, P. (2017) Modeling and Analysis of Model Reference Adaptive Control by Using MIT and Modified MIT Rule for Speed Control of DC Motor, *7th IACC*. Hyderabad, India: 2017 January 5-7. doi: 10.1109/IACC.2017.0105

[20] Kumar, N., and Khanduja, N. (2012) Mathematical Modelling and Simulation of CSTR Using MIT Rule, *5th IEEE IICPE*. Delhi, India: 2012 December 6-8. doi: 10.1109/IICPE.2012.6450458

[21] Lavretsky, E., Wise, K.A. (2013) *Robust and Adaptive Control: with Aerospace Applications*. 1st ed. London: Springer-Verlag. doi: 10.1007/978-1-4471-4396-3

[22] Parks, P.C. (1966) Liapunov Redesign of Model Reference Adaptive Control Systems, *IEEE TAC*. Vol.11, No.3, pp.362-367. doi: 10.1109/TAC.1966.1098361

[23] Swarnkar, P., Jain, S., and Nema, R.K. (2011) Comparative Analysis of MIT Rule and Lyapunov Rule in Model Reference Adaptive Control Scheme, *ISDE*. Vol.2, No.4, pp.154-162. Available at: http://www.iiste.org/Journals/index.php/ISDE/article/view/414/295 (accessed: 2018, April 19)

[24] Hung, V.M., Stamatescu, T., Dragana, C., and Paraschiv, N. (1993) Comparison of model reference adaptive control and cascade PID control for ASTank2, *9th IEEE IDAACS*. Bucharest, Romania: 2017 September 21-23. doi: 10.1109/IDAACS.2017.8095263

[25] Butler, H., Honderd, G., and van Amerongen, J. (1989) Model reference adaptive control of a direct-drive DC motor, *IEEE Control Systems Magazine*. Vol.9, No.1, pp.80-84. doi: 10.1109/37.16756

[26] Benjelloun, K., Mechlih, H., and Boukas, E.K. (1993) A modified model reference adaptive control algorithm for DC servomotor, *2nd IEEE CCA*. Vancouver, BC, Canada, Canada: 1993 September 13-16. doi: 10.1109/CCA.1993.348210

[27] Nise, N.S. (2015) *Control systems engineering*. 7th ed. Singapore: Markon Print Media Pte Ltd.

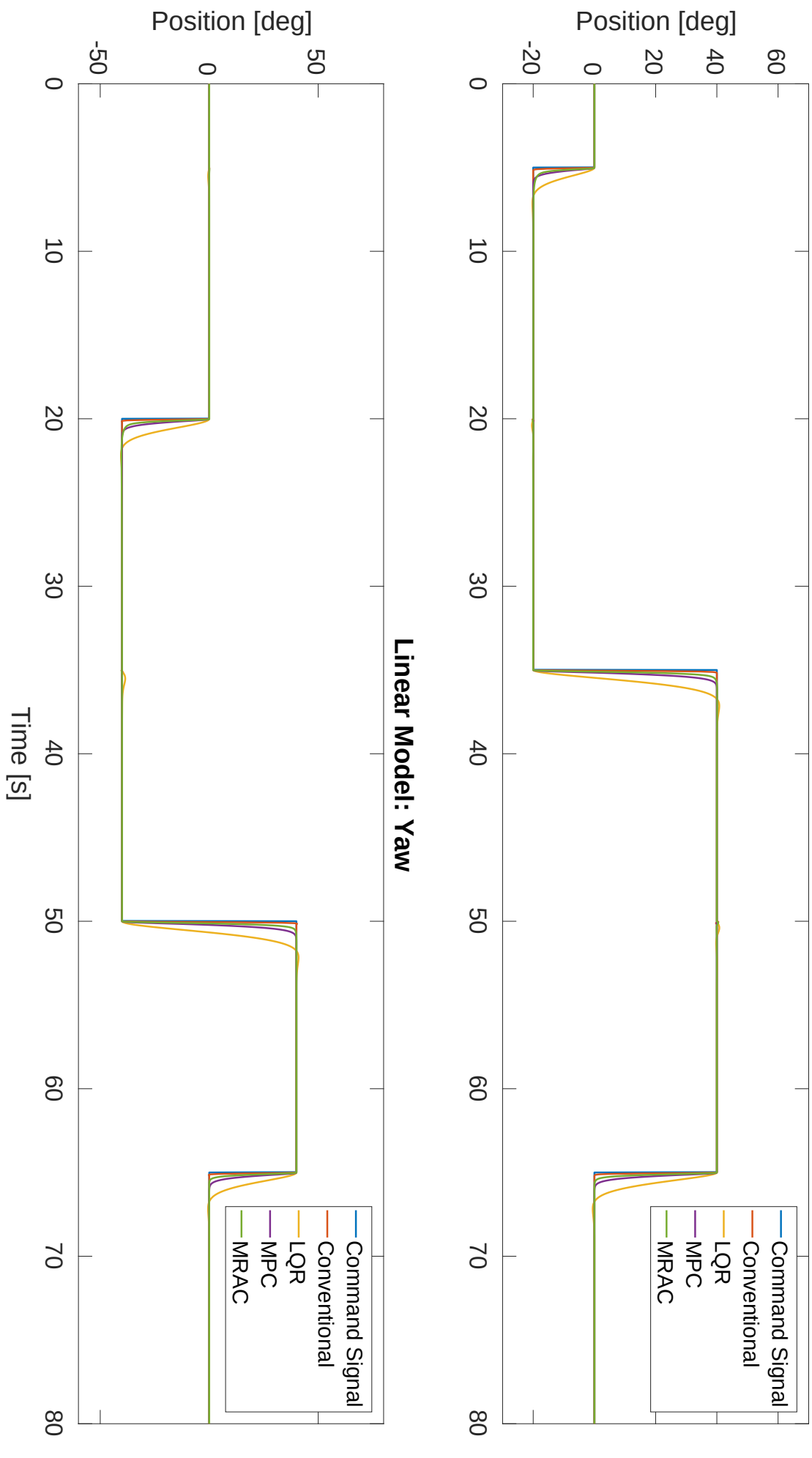
[28] Franklin, G.F., Powell, J.D., and Emani-Naeini, A. (2015) *Feedback Control of Dynamic Systems*. 7th ed. London: Pearson.

[29] Ruderman, M.2017. State-space methods Available at: https://fronter.com/uia/ (Accessed: 11. April. 2018)

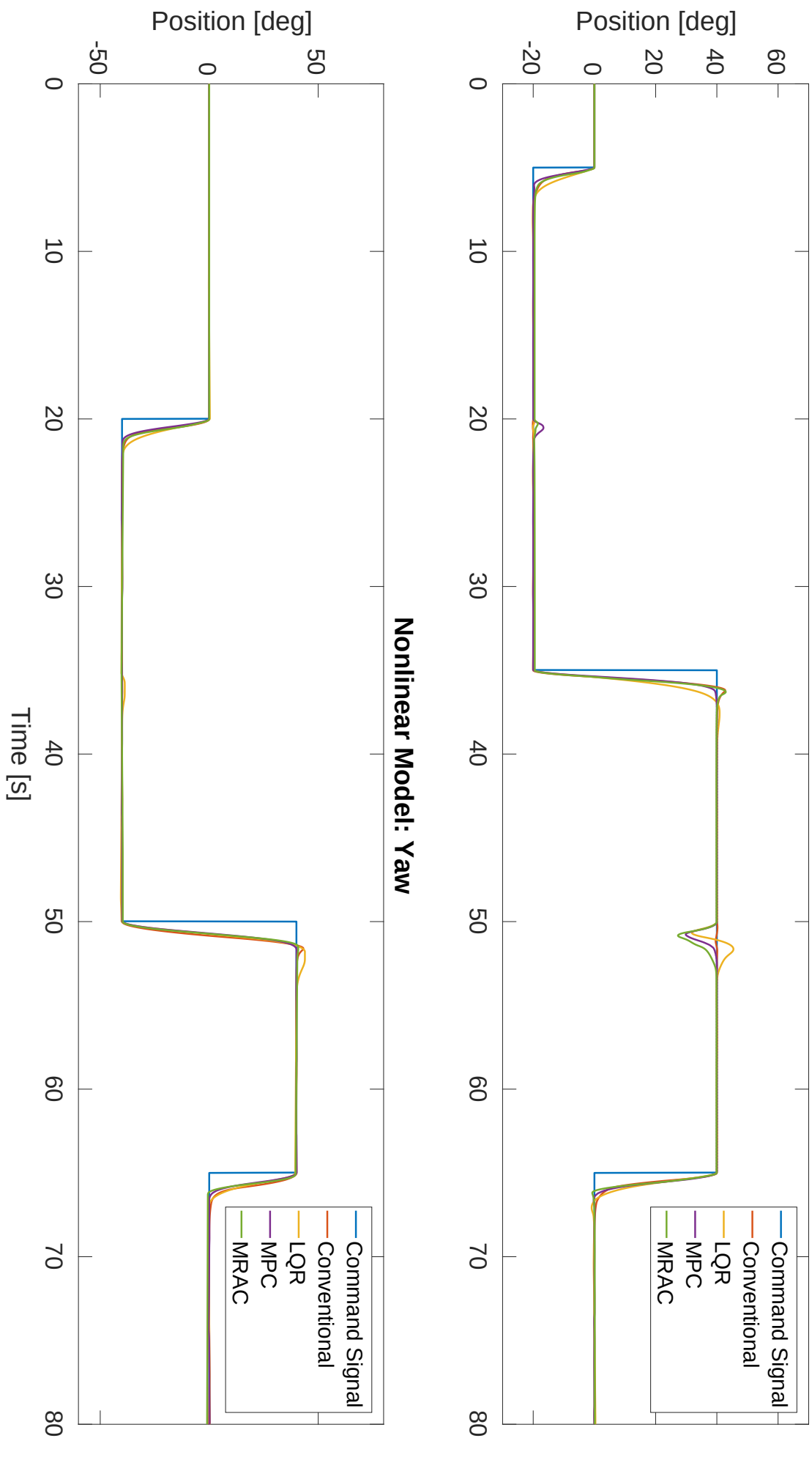
11 APPENDICES

APPENDIX A: Test 1 Linear Model
APPENDIX B: Test 1 Nonlinear Model
APPENDIX C: Test 1 AERO
APPENDIX D: Test 2 Linear Model
APPENDIX E: Test 2 AERO

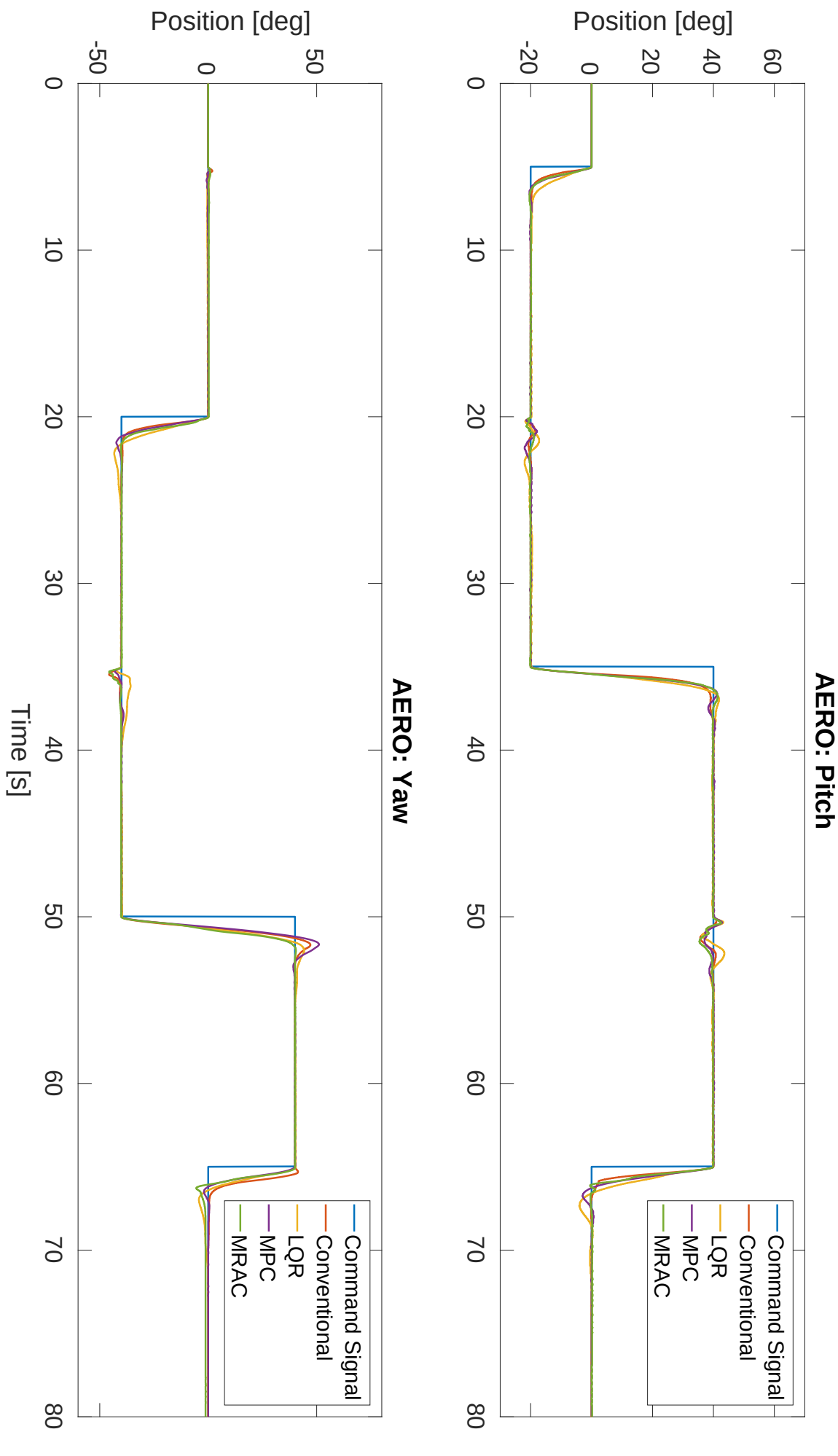
APPENDIX A: Test 1 Linear Model



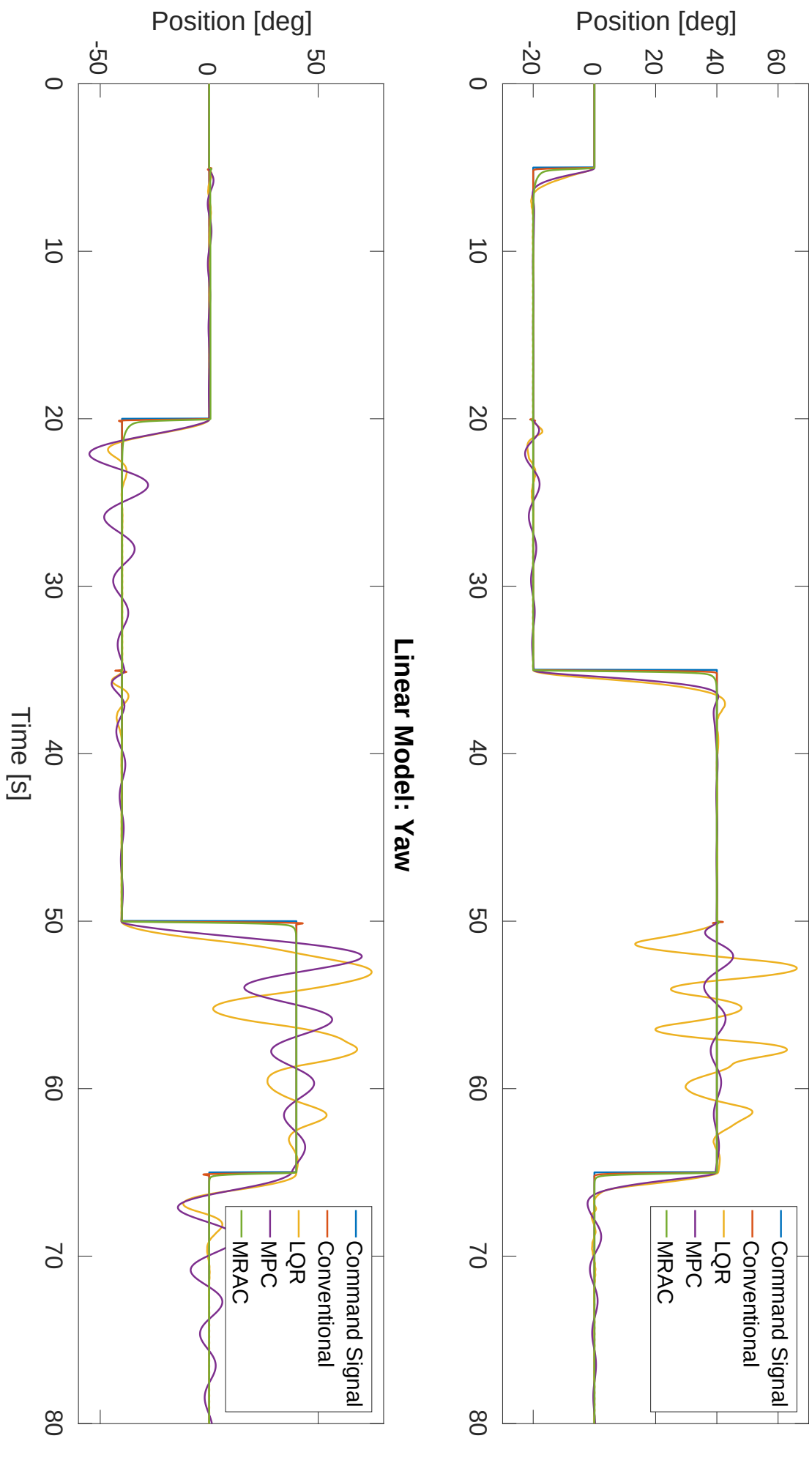
APPENDIX B: Test 1 Nonlinear Model



APPENDIX C: Test 1 AERO



APPENDIX D: Test 2 Linear Model



APPENDIX E: Test 2 AERO