# Semi-automatic generation of control law parameters for generic fighter aircraft 

Markus Lindblom<br>Supervisor: Shervin Parvini Ahmadi<br>Examiner: Martin Enqvist

Linköping University

Semi-automatic generation of control law parameters for generic fighter aircraft

Markus Lindblom

July 15, 2020


#### Abstract

Control law design can be an iterative and time-consuming process. The design procedure can often include manual tuning, not uncommonly in the form of trial and error. Modern software tools may alleviate this process but are generally not developed for use within any specific industry. There is therefore an apparent need to develop field-specific tools to facilitate control law design.

The main contribution of this thesis is the investigation of a systematic and simplified approach to semi-automatic generation of control law parameters for generic fighter aircraft. The investigated method aims to reduce human workload and time spent on complex decision making in the early stages of aircraft development. The method presented is based on gain scheduled LQI-control with piece-wise linear interpolation. A solution to the automated tuning problem of the associated weighting matrices $Q$ and $R$ is investigated. The method is based on an LQ-optimal eigenstructure assignment. However, the derived method suffers from problem regarding practical implementation, such as the seemingly narrow LQ-optimal root-loci of the linearized aircraft model.

Furthermore, the inherent problem of hidden coupling is discussed in relation to gain scheduled controllers based on conventional series expansion linearization. An alternative linearization method is used in order to circumvent this problem. Moreover, the possible benefits and disadvantages of control allocation is addressed in the context of actuator redundancy. It is concluded that one may achieve a somewhat simpler handling of constraints at the expense of some model accuracy due to the inevitable exclusion of servo dynamics.


## Contents

1 Introduction ..... 1
1.1 Purpose ..... 1
1.2 Background ..... 1
1.3 Approach ..... 1
1.4 Limitations ..... 2
1.5 Thesis outline ..... 2
2 Linear quadratic control ..... 4
2.1 LQI-control ..... 5
2.2 Prescribed degree of stability ..... 6
3 Gain scheduling ..... 7
3.1 Introduction to hidden coupling ..... 7
3.2 Handling hidden coupling terms ..... 9
3.2.1 A small mathematical example ..... 10
3.3 Slow variation requirements ..... 11
4 Gain scheduling example ..... 12
4.1 Modeling and design goals ..... 12
4.2 Design procedure ..... 13
4.3 Results ..... 14
4.4 Discussion ..... 19
5 Modelling of aircraft dynamics ..... 20
5.1 Rigid body dynamics ..... 20
5.2 Current aircraft layout ..... 21
5.3 Linearized flight mechanical model ..... 22
5.4 Servo dynamics ..... 25
5.5 Validation ..... 27
6 Tuning of weighting matrices ..... 32
6.1 The inverse LQ-problem ..... 33
6.1.1 Solution to the inverse LQ-problem ..... 36
6.2 Eigenstructure assignment ..... 37
6.2.1 Solution to the eigenstructure assignment ..... 39
6.3 A small example ..... 40
7 Actuator redundancy ..... 44
7.1 Optimal control ..... 44
7.2 Control allocation ..... 45
7.2.1 Control allocation and servo dynamics ..... 46
8 Results ..... 49
8.1 Program description ..... 49
8.2 Linear simulation results ..... 55
8.3 Nonlinear simulation results ..... 56
9 Conclusions ..... 59
Appendices ..... 63
A Non-linear missile dynamics ..... 64
B Linearized missile dynamics ..... 66
C Missile guidance system script ..... 68
D Coefficients of linearized equations ..... 73
E Inverse LQ-algorithm script ..... 74
F LQ-optimal eigenstructure assignment script ..... 81

## Chapter 1

## Introduction

The work included in this thesis has been performed at the request of SAAB Aeronautics and supervised and examined by the department of electrical engineering at Linköping University.

### 1.1 Purpose

The main purpose of this thesis is to create a semi-automatic program that generates initial values of control parameters for generic fighter aircraft. The aim is to facilitate control law design and create a basis for early assessment of aircraft performance. The program is intended to reduce human workload and relieve complex decision making in the early stages of new aircraft development.

### 1.2 Background

SAAB Aeronautics is a subset of the SAAB corporation, a military company that, among other products, designs and produces the modern multirole fighter JAS-39 Gripen. Currently the SAAB corporation owns and maintains an internally developed simulator environment known as Ares Mars. This simulator will to some degree be extended and serve as the basis for the implementation and evaluation of the developed method.

### 1.3 Approach

The controller scheme applied is gain-scheduled LQI-control with only minor variations and/or extensions which will be further introduced in the relevant sections. This choice, among other things, assures simple validation and certification of flight-worthiness. Moreover, two different linearization methods will be compared, classical series expansion (i.e. Jacobian linearization) and what is known as velocity-based linearization first introduced by Kaminer et al. [5]. The purpose of this comparison is to address the inherent problem of hidden coupling terms relating to the interpolation of controller parameters, a problem that is said to be alleviated by use of the later. Furthermore, methods for automated tuning of weighting matrices will be analyzed including solutions relating to the inverse LQ-problem.

### 1.4 Limitations

It is not within the scope of this thesis to evaluate different controller schematics, nor is it to in practice perform any studies of altering aircraft layouts. Moreover, all controller parameters created are, without exception, linearly interpolated to yield a piece-wise linear structure for their respective values. No effort will be devoted to evaluating stability preserving methods or alike. This choice ensures decoupling of the later validation process, i.e. one can simply re-tune the necessary parameters for any single sub-region of the flight envelop where performance does not confine to expectations or demands.

### 1.5 Thesis outline

In Chapter 2, a theoretical background on linear quadratic control (LQ-control) is given along with two relevant extensions. The first being linear quadratic integral control (LQI-control) which is used in order to gain setpoint tracking. The second being alpha-shifted LQ-control, sometimes also referred to as LQ-control with a prescribed degree of stability. This extension serves the purpose of reducing possible overshoot and oscillatory behaviour in the transient response.

In Chapter 3.1, the concept of gain scheduling is introduced including the associated problem of hidden coupling terms (HCT -s). Some possible methods of dealing with HCT-s are discussed including velocity based linearization first presented in [5]. An extension to this approach is given, here referred to as enhanced velocity based linearization in accordance with the title to [4]. Furthermore, a small mathematical example is given as to motivate the effect of the velocity based approach. Lastly, the slow variation requirement of gain scheduling parameters is mentioned, partially in relation to the velocity based linearization technique.

In Chapter 4, the presented theory is put to practice in a small example regarding the pitch control of a non-ballistic (fuel propelled) missile. In this example both conventional linearization, by use of a first order series expansion, and enhanced velocity based linearization is implemented and the results are subsequently compared. The possible existence of HCT-s are noted and discussed in relation to the gradient of the interpolated feedback coefficients.

In Chapter 5, the modelling of the aircraft dynamics are presented including the resulting structure of the linearized state space model along with the definition of states and inputs. The current aircraft layout in the Ares Mars simulator is described and some practicalities regarding the implementation of servo dynamics are covered. The Chapter is ended with results from a small validation process in which linear and nonlinear responses are compared.

In Chapter 6, tuning of weighting matrices are discussed. The idea of automation is introduced through some notable examples. A new method of automation through the sequential solving of an eigenstructure- and inverse LQproblem is introduced. Some practical obstacles are consequentially addressed and the Chapter is ended in a small example for a randomly created linear time invariant (LTI) system.

In Chapter 7, two possible methods of handling actuator redundancy are discussed. Focus lies on optimal control allocation along with its possible benefits and apparent drawbacks. Subsequently the concept of virtual control signals is introduced and the necessary alteration of the linear state space model is presented.

In Chapter 8 the resulting program is introduced along with some brief commentary regarding which of the addressed procedures and techniques were implemented. Examples of linear simulation results are given and the Chapter is ended by presenting results from nonlinear simulations when the aircraft is operating in rate mode.

## Chapter 2

## Linear quadratic control

Linear quadratic control is a special case of optimal control where the performance index is quadratic in both states and control effort. The notion of minimizing a quadratic cost functional (performance index) stems from work performed by Wiener [10] and Hall [11]. However the rigorous mathematical foundation needed, including the introduction of the concept of controllability, was first presented in the article Contributions to the theory of optimal control written by R.E Kalman [12], published in 1960. The LQ-problem is defined as follows, given an arbitrary initial condition $x_{0}$ of the linear system $\dot{x}=A x+B u$, find the optimal control effort $u$ that minimizes the performance index

$$
\begin{equation*}
J(x, u)=\int_{t_{0}}^{t_{1}} \frac{1}{2}\left(x^{T} Q x+u^{T} R u\right) d t+\frac{1}{2} x^{T}\left(t_{1}\right) P x\left(t_{1}\right) \tag{2.1}
\end{equation*}
$$

where $Q$ and $R$ are referred to as either cost matrices, penalty matrices or weighting matrices. Both $Q$ and $R$ are symmetric and subject to $Q \geq 0, R>0$. The last term in the above performance index is known as a terminal cost and is in principle only relevant when the final time $t_{1}$ is finite.

The solution to the LQ-problem is well known and is here only presented in a straightforward manner. If the reader wishes to have further insight into the derivation he or she is referred to, e.g. [12]. The control effort $u$ that minimizes the performance index in (2.1) is $u=-K x$ where $K=R^{-1} B^{T} P$ and $P$ is the solution to the Riccati Differential Equation (RDE)

$$
\begin{equation*}
-\dot{P}=A^{T} P+P A-P B R^{-1} B^{T} P+Q \tag{2.2}
\end{equation*}
$$

where $P$ is bounded by $P\left(t_{1}\right)=P_{1}$. Note that the solution to this ordinary differential equation is in general time-varying, however if the final time $t_{1}$ is infinite the solution $P$ kan be shown to be constant and the problem simplifies to finding the solution $P$ to

$$
\begin{equation*}
A^{T} P+P A-P B R^{-1} B^{T} P+Q=0 \tag{2.3}
\end{equation*}
$$

known as the Algebraic Riccati Equation (ARE). Note that the word simplifies does not imply that finding the solution to the above equation is trivial (neither in a theoretical nor practical sense) as can be seen by the extensive work presented in [13].

Note: It is worth mentioning the apparent similarity between the ARE and the Lyapunov equation

$$
\begin{equation*}
A P+P A^{T}=-Q . \tag{2.4}
\end{equation*}
$$

The Lyapunov equations derives from the observation that in order for a system to be stable the generalized energy (belonging to the system states) must decrease as time progress. The solution $P$, if such exist, takes the form $P=\int_{0}^{\infty} e^{A^{T} t} Q e^{A t} d t$. Noting that for an unforced LTI system the natural response, given the initial condition $x_{0}$, equals $x=e^{A t} x_{0}$ it is straightforward to show that

$$
\begin{equation*}
x_{0}^{T} P x_{0}=\int_{0}^{\infty} x_{0}^{T} e^{A^{T} t} Q e^{A t} x_{0} d t=\int_{0}^{\infty} x^{T} Q x d t \tag{2.5}
\end{equation*}
$$

The apparent conclusion of this is that finding the solution $P$ to the Lyapunov equation (and pre-/post multiplying by $x_{0}^{T} / x_{0}$ ) is equivalent to evaluating a quadratic cost functional in $x$, weighted by $Q$, over infinite time. Similarly to above, the forced system response, characterized by the optimal control $u=-K x$, may be shown to be $x=e^{(A-B K) t} x_{0}$. In accordance with previous reasoning it may be shown that in order to evaluate the quadratic cost functional over infinite time for the implied forced system one may solve the equation

$$
\begin{equation*}
(A-B K) P+P(A-B K)^{T}=-Q \tag{2.6}
\end{equation*}
$$

Substituting $K$ for $R^{-1} B^{T} P$ (and possibly transposing the result depending on convention) we find that in order to evaluate the cost functional (2.1) we must solve the ARE.

Several extensions to the LQ-problem has over time been presented and solved, including cross-coupled $L Q$-control, see for example [14]. The only extensions relevant to this thesis however are $L Q I$-control and what is known as $L Q$-control with a prescribed degree of stability, sometimes referred to as alpha-shifted LQcontrol.

### 2.1 LQI-control

LQI-control abbreviates Linear Quadratic Integral control and is one of several ways of solving the servo problem (note that ordinary LQ-control only provides stability). In addition to the conventional state feedback, inherent to LQ-control, a second feedback loop containing simple integral control is added. In order to find appropriate values for the integral controller coefficients the system is augmented to include the controller states (i.e. the integral of the tracking error $e)$. The augmented system takes the form

$$
\begin{array}{r}
\binom{\dot{x}}{\dot{e}}=\left[\begin{array}{cc}
A & 0 \\
-C_{y i} & 0
\end{array}\right]\binom{x}{e}+\left[\begin{array}{c}
B \\
-D_{y i}
\end{array}\right] u+\left[\begin{array}{l}
0 \\
I
\end{array}\right] r  \tag{2.7}\\
y=\left[\begin{array}{ll}
C & 0
\end{array}\right]\binom{x}{e}+D u \\
x \in \mathcal{R}^{n}, u \in \mathcal{R}^{m}, e \in \mathcal{R}^{l}
\end{array}
$$



Figure 2.1: Block diagram of a general LQI controlled system.
where the matrices $C_{y i}$ and $D_{y i}$ are the output- and feed-trough matrices for the outputs that are to be tracked (of sizes $l \times n$ and $l \times m$ respectively). The subsequent feedback-matrix $K$ may be divided and written in block matrix form as $K=\left[K_{x}, K_{i}\right]$, where $K_{i}$ is of size $m \times l$. The matrix $K_{i}$ contains the LQoptimal integral controller coefficients and the system takes the general form shown in Figure 2.1.

In contrast to LQI-control one could use static- or dynamic feed forward control to gain proper tracking. However, as is well known, this approach yields high model dependency. Both LQI-control and feed forward control could in practice be combined in order gain tracking performance and introduce further freedom in shaping of the transient response and sensitivity functions. However, the main focus of this thesis is, as stated, not the control structure itself but rather the investigation of possible methods of automation. As such no feed forward control is implemented.

### 2.2 Prescribed degree of stability

The optimal control, deriving from the solution to the LQ-problem, may at times result in a step response that exhibits either overshoot and/or other unwanted transient behaviour. In order to lessen the extent of this issue the performance index may be altered accordingly

$$
\begin{equation*}
J(x, u)=\int_{t_{0}}^{t_{1}} \frac{1}{2}\left(e^{\alpha t}\right)^{T}\left(x^{T} Q x+u^{T} R u\right) e^{\alpha t} d t \tag{2.8}
\end{equation*}
$$

Note that in order for a stabilizing solution to exist, in the general case, the states $x$ must converge towards the origin (zero). For the altered performance index in (2.8) this effectively means that the states must converge to the origin faster than, or at least equally fast as, $e^{-\alpha t}$, otherwise the performance index would per definition diverge. To implement this extensions in practice one need only alter the state transfer matrix accordingly $\dot{x}=A_{\alpha} x+B u, A_{\alpha}=\alpha+A$, where $\alpha$ is a diagonal positive semi-definite matrix, for further details regarding this approach the reader is referred to [14].

## Chapter 3

## Gain scheduling

Gain scheduling is the practice of dividing an intended operating region into a set of sub-regions, conventionally about a set of operating points. The plant is then linearized about each point and a family of linear controllers is synthesized based on given performance specifications. These controllers could then either be implemented as is, with an associated switching logic, or controller parameters and/or outputs could be interpolated to estimate an appropriate controller structure at intermediate values. Regardless, the current choice of controller parameters is scheduled based on the parameters that define the operating domain. These parameters, called scheduling parameters, can be either exogenous (such as local atmospheric properties) and/or endogenous (such as angle of attack, angular rates etc.). However if one chooses to use endogenous state variables as scheduling parameters one will, if not taken particular caution, invoke what is known as hidden coupling terms. For further insight and additional references regarding gain scheduling the reader is referred to [1].

### 3.1 Introduction to hidden coupling

In order to create a basis for the introduction, and subsequent discussion, of hidden coupling terms we will here refer to the example given in [3].

Consider the problem of designing a missile guidance system. The problem is, for simplicity, reduced to that of only controlling the pitching dynamics of a ballistic (non propelled) missile. The goal is to design a controller in order to enforce reference tracking of commanded normal force to ensure that the missile follows a predefined trajectory. In [3] the authors choose to use angle of attack, $\alpha$, and mach number, $M$, as scheduling parameters and derive an analytic expression for a general PI-controller acting on commanded and measured normal force along with pitch rate. The controller dynamics are given as

$$
\begin{array}{r}
\dot{z}=K_{i}(\alpha, M)\left(n_{c}-n\right) \\
\delta_{e}=z+K_{n}(\alpha, M) n+K_{q}(\alpha, M) q \tag{3.2}
\end{array}
$$

where $n_{c}$ denotes the commanded normal force and $\delta_{e}$ denotes the elevator deflection.

If both $\alpha$ and $M$ were independent parameters that did not affect nor became affected by either $n$ or $q$, then the designed controller would per definition be linear, however this is not the case. There is an inherent coupling between the states that are to be controlled ( $n$ and $q$ ) and the parameters used to define the controller gains ( $\alpha$ and $M$ ), hence, the above controller is non-linear in both $n$ and $q$. To clarify (3.1) and (3.2) should perhaps be rewritten as

$$
\begin{array}{r}
\dot{z}=K_{i}(\alpha(n, q), M(n, q))\left(n_{c}-n\right) \\
\delta_{e}=z+K_{n}(\alpha(n, q), M(n, q)) n+K_{q}(\alpha(n, q), M(n, q)) q \tag{3.4}
\end{array}
$$

In order to evaluate the local behaviour of this non-linear controller it is necessary to differentiate with respect to both $n$ and $q$ and evaluate arising terms at the given operating conditions. Since the gains $K_{x}(\alpha(n, q), M(n, q))$, by above reasoning, must be regarded as composite functions one must inevitably turn to the composite function rule for derivatives, perhaps more well known as the chain rule. When applied to the general PI-controller in (3.3)-(3.4), it is straightforward to show that $\Delta \dot{z}=K_{i}\left(\Delta n_{c}-\Delta n\right)+\left.\frac{\partial K_{i}}{\partial \Theta} \frac{\partial \Theta}{\partial n}\right|_{o c} \Delta n$, as well as $\Delta \delta_{e}=\Delta z+K_{n} \Delta n+K_{q} \Delta q+\left.\frac{\partial K_{n}}{\partial \Theta} \frac{\partial \Theta}{\partial n}\right|_{o c} \Delta n+\left.\frac{\partial K_{q}}{\partial \Theta} \frac{\partial \Theta}{\partial q}\right|_{o c} \Delta q$. Note that the subscript oc denotes operating conditions. The last terms included in the above expressions derive from the aforementioned coupling and is as such properly named hidden coupling terms, this dependency may be visualized as in Figure 3.1.


Figure 3.1: When the scheduling parameters are internal states of the system the passing of their instantaneous values to the controller will cause additional feedback. In order to preserve designed performance and margins this added dynamics must be accounted for in some manner.

### 3.2 Handling hidden coupling terms

There are a number of ways to deal with hidden coupling terms (including the method from [3]), however only two methods will be mentioned here. The first of which is by simply stating that:

$$
\begin{align*}
\left.\frac{\partial K_{i}}{\partial \Theta} \frac{\partial \Theta}{\partial n}\right|_{o c} \Delta n & =0  \tag{3.5}\\
\left.\frac{\partial K_{n}}{\partial \Theta} \frac{\partial \Theta}{\partial n}\right|_{o c} \Delta n+\left.\frac{\partial K_{q}}{\partial \Theta} \frac{\partial \Theta}{\partial q}\right|_{o c} \Delta q & =0 \tag{3.6}
\end{align*}
$$

In other words, one states that the hidden couplings, deriving from the different scheduling parameters, must cancel each other out. However this imposes restrictions on any forthcoming interpolation, restrictions that may not be compatible with the original design specifications.

The second approach includes applying a velocity-based plant model first presented in [5]. In contrast to the conventional series expansion approach this method relies on the use of state and output derivatives rather than the introduction of the familiar deviation variables. In [5] the authors prove that using a velocity-based approach will preserve the local input-output relation of the designed linear controller, i.e. one avoids the presence of hidden coupling terms. Furthermore, the authors present a method of realizing the resulting controller structure when necessary measurements of derivatives are not available. This method simply involves using pseudo derivatives in the form of low-pass filtered ordinary derivatives. In [4] the authors extend upon this idea and conclude that adapting this approach on pseudo derivatives leads to the inclusion of an unwanted pole located at $-\frac{1}{\tau}$, where $\tau$ represents the low-pass filter time constant. This in turns leads to an additional trade off between performance and system noise injection. The authors then suggest a post-filtering structure that introduces a zero at the unwanted pole in order to compensate for the pre-filtering dynamics. They conclude that the new structure is mostly insensitive to the choice of time constant $\tau$ and that one therefore can use fairly high values of the same without severely degrading performance. Furthermore, the authors prove that among the possible choices of pre- and post-filtering structures that could be made, a standard first order low-pass filter is the only one that will preserve the local input-output relations and avoid hidden coupling. The resulting generic structure is illustrated in Figure 3.2. For further reference, this approach will (in accordance with the title of the original article) be referred to as enhanced velocity-based linearization.


Figure 3.2: Enhanced velocity-based gain scheduled controller operating on pseudo derivatives. Note that the only practical difference between this approach and those based on the conventional series expansion is the introduction of the low pass filtered derivative and its subsequent inverse.

### 3.2.1 A small mathematical example

Consider the SISO-system

$$
\begin{align*}
\dot{x} & =a x+b u  \tag{3.7}\\
y & =x \tag{3.8}
\end{align*}
$$

Assume that a gain scheduled feedback law of the type $u=k(x) x$ has been derived (by any preferable method). Further assume that $k(x)$ takes the simple form $k(x)=c x$, where $c$ is some constant. Linearization of the controller structure at the operation point $x=x_{0}$ yields

$$
\begin{equation*}
\Delta u=k\left(x_{0}\right) \Delta x+\left.\frac{\partial k(x)}{\partial x}\right|_{x_{0}} x_{0} \Delta x=2 c x_{0} \Delta x \tag{3.9}
\end{equation*}
$$

The deviation in input, $\Delta u$, is obviously twice that what was anticipated (due to hidden coupling). Suppose instead that the feedback law takes the structure

$$
\begin{equation*}
u=\int k(x) \dot{x} d t \tag{3.10}
\end{equation*}
$$

as suggested by the velocity based approach. The integral term may be evaluated by partial integration accordingly

$$
\begin{equation*}
u=k(x) x-\int \frac{\partial k(x)}{\partial t} x d t=k(x) x-\int \frac{\partial k(x)}{\partial x} \frac{\partial x}{\partial t} x d t \tag{3.11}
\end{equation*}
$$

If the structure for $k(x)$ is inserted and and the integral evaluated one ends up at

$$
\begin{equation*}
u=k(x) x-c \frac{x^{2}}{2} \tag{3.12}
\end{equation*}
$$

Linearizing at $x=x_{0}$ now yields

$$
\begin{equation*}
\Delta u=\left.\frac{\partial}{\partial x}\left(k(x) x-c \frac{x^{2}}{2}\right)\right|_{x_{0}} \Delta x=c x_{0} \Delta x \tag{3.13}
\end{equation*}
$$

The deviation in input $\Delta u$ now matches exactly what was desired and expected.

### 3.3 Slow variation requirements

Any non-linear system with a controller structure that has been based on a finite number of linear approximations will inevitably, for nominal stability, be subject to a slow variation requirement on the inherent scheduling parameters. This may in practice limit the effectiveness about some of the system extremes where such an assumption could be far from valid. One approach to alleviate this problem may be to design a sufficiently high bandwidth controller, but for obvious reasons this may not be realizable for the intended application. Such restrictions may be relaxed by the use of the former velocity-based linearization method since it allows for linearization at off-equilibrium points. However, since one instead operates on the derivatives, or pseudo-derivatives, of the system states and/or outputs one may, if not taking proper precaution, create a static offset through the integration of existing modelling errors [6].

## Chapter 4

## Gain scheduling example

In order to show the principles, a small example will here be given in the form of the LQ gain-scheduling of a non-ballistic missile.

### 4.1 Modeling and design goals

Consider the pitch dynamics of a ballistic missile given in [7], all necessary equations can be found in Appendix A. Note that this system offers only one degree of freedom when it comes to classical gain scheduling, i.e. the equilibrium domain over which to schedule is a single trajectory.

> Note: Simply put, without propulsion the only motion in which the missile is experiencing equilibrium is gliding. The gliding motion has a slope explicitly determined by the angle of the tail fin (all else held constant) and each slope is associated with a terminal velocity, i.e. a velocity at which the missile will no longer accelerate due to drag. Intuitively one might replace the missile with a paper airplane, the principles are much the same. This effectively means that the value of any single parameter, meaning input or state, defines an arbitrary equilibrium point and, in effect, the values of all other associated parameters. Stated differently, we can only choose one single scheduling parameter.

However, similar to [3] and [4], both angle of attack and missile velocity are here chosen as scheduling parameters. In both [3] and [4] the authors choose to exclude the dynamics of the missile velocity in the scheduling procedure (as it is deemed to vary slow enough in relation to other parameters). As such they effectively regain one degree of freedom for scheduling purposes. Here an alternative approach is employed by introducing an additional control signal in the form of an imagined engine. Equation (A.4c), from appendix A, therefore becomes

$$
\begin{equation*}
\dot{V}_{m}=\frac{F_{x}+T}{m} \cos (\alpha)-\left|\frac{F_{l}}{m} \sin (\alpha)\right| \tag{A.4c}
\end{equation*}
$$

where $T$ denotes the thrust produced by the engine. Note that this is a simple summation of forces in the direction of the missile velocity.

The problem is thus converted from that of controlling a ballistic missile to that of controlling a non-ballistic missile. Note that in reality this thrust is realistically a function of both height, speed and perhaps several other system and/or environmental parameters, but this approach will however be realistic enough for this example. Further note that the engine is assumed to be perfectly aligned with the central-axis of the missile body such that it creates no pitching or yawing moment. The overall design goal is to control the missile normal acceleration as to force the missile to follow a desired flight path. The performance requirements are

- the closed loop effective time constant is to be no greater than 0.35 seconds
- the overshoot may be no more than $5 \%$

This is, as someone may point out, not a complete performance description but will however suffice. The missile operating domain is defined by $\alpha \in\left[-20^{\circ}, 20^{\circ}\right]$ and $V_{m} \in[2,4] \times$ sos, where sos denote speed of sound

### 4.2 Design procedure

The system is linearized using a first order series expansion resulting in a family of linear systems of the form:

$$
\begin{align*}
\Delta \dot{x} & =A\left(x_{e}, u_{e}\right) \Delta x+B\left(x_{e}, u_{e}\right) \Delta u  \tag{4.1}\\
\Delta y & =C\left(x_{e}, u_{e}\right) \Delta x
\end{align*}
$$

where the subscript $e$ denotes equilibrium. The feed-through matrix $D$ has been excluded since all elements equal zero (note that this is only valid if the servo model in question is included in the state-space matrix $A$, otherwise the tail-fin deflection, $\delta$, will show up as a feed-through term to the normal acceleration $\eta$ ). The full list of linearized equations can be found in Appendix B. Once linearized a grid of $20 \times 20$ equilibrium points are calculated and each point is inserted into (4.1). This results in a family of in total four hundred linear systems for each of which an LQI-controller is to be designed where the integral term operates on the error in the output $\eta$. This may seem excessive but note that as $\alpha$ spans a full forty degrees the resulting grid-resolution is $2^{\circ}$, which, from an aerodynamic point of view, is fairly large. Since LQ-design is inherently based on full state feedback the matrix $C\left(x_{e}, u_{e}\right)$ takes the form of $\left[\begin{array}{ll}I_{4 \times 4} & C_{y}\end{array}\right]$ (where $C_{y}$ derives from the linearization of (A.4e)) assuming that all states are measurable. In order to include the integral of the tracking error the system is augmented in accordance with Section 2.1. Note that the approach for the enhanced velocity based technique is initially the same, the only difference is the later addition of the aforementioned pre and post filtering structure, see Section 3.2. The above procedure is hence common to both linearization techniques.

The practical implementation and evaluation is performed by use of Matlab and Simulink, the Matlab script can be found in Appendix C. In order to interpolate the static feedback coefficients a look-up table with automated linear interpolation was used in Simulink. Note that, by default, Simulink performs a bilinear interpolation that in reality is quadratic in the grid-position.

The apparent implication is that the feedback coefficients at times may be over- or underestimated compared to a true piece-wise linear interpolation. However, true piece-wise linear interpolation requires triangularization of the grid which in effect, if to avoid all ambiguity, would imply the need of an additional $19 \times 19$ grid values, as illustrated in Figure 4.1. This bilinear interpolation is therefore deemed adequate for this introductory example.


Figure 4.1: Four three dimensional data points (black dots), representing arbitrary values, are shown in the leftmost figure. For ease of visualization they are connected by solid blue lines. The grey solid lines beneath them indicate their height (or equally functional value). Subfigures a and b both represent valid piece-wise linear interpolations. As can be seen they may yield different interpolated values at identical position. In order to avoid this ambiguity one additional data point is needed in the space between them, as illustrated in figure $\mathbf{c}$.

### 4.3 Results

Below is presented a series of figures showing the result from the non-linear simulations. Note that in these simulations a static feed-forward term has been added, which includes the thrust and tail-fin deflection needed to keep the missile at equilibrium. The responses shown in Figures 4.2-4.4 correspond to the weighting matrices

$$
\begin{align*}
& Q=\left[\begin{array}{ccccc}
1 & 0 & \ldots & & 0 \\
0 & 1000 & & & \\
\vdots & & 1 & & \vdots \\
0 & & & 1 & 0 \\
0 & & \ldots & 0 & 4000
\end{array}\right]  \tag{4.2}\\
& R=\left[\begin{array}{cc}
0.001 & 0 \\
0 & 0.01
\end{array}\right] \tag{4.3}
\end{align*}
$$

In Figure 4.3 the annotation provided indicate that the missile, for both linearization techniques, has reached just below 12.7 g , or approx. $63.5 \%$ of the total initial step size, after roughly 0.15 seconds. This proves that the requirement regarding effective time constant has been fulfilled (with some margin). Likewise, in Figure 4.4 the annotation provided to the right shows that the initial (and equally largest) overshoot is around $3.75 \%$ and $1.15 \%$ respectively. As such both techniques further fulfill the requirement regarding overshoot.


Figure 4.2: Response to a series of steps for the closed loop nonlinear system. The red curve shows the commanded normal acceleration. The black curve corresponds to the conventional series expansion approach (First order SE). The blue curve corresponds to the enhanced velocity based approach (Enhanced $V B)$.


Figure 4.3: Response to the initial step. The enhanced velocity based approach reaches approx. $63.40 \%$ at around 0.1487 seconds. Likewise the conventional series expansion approach reaches approx. $63.45 \%$ at around 0.1503 seconds.


Figure 4.4: Response to the initial step. The enhanced velocity based approach reaches its maximum value, corresponding to roughly $3.75 \%$ overshoot, at about 0.3340 seconds. Likewise the conventional series expansion approach reaches its maximum value, corresponding to roughly $1.15 \%$ overshoot, at 0.3671 seconds.

In Figures 4.5a and 4.5b one can see how two elements from the feedback matrix $K$ changes with the element $R_{22}$ (lower right element of weighting matrix $R$ ) according to $R_{22}=0.01(10 i-9), \quad i=1,2,3 \ldots 21$. In this instance, element $R_{11}$ (upper left element of matrix $R$ ) equals 0.001 . All other elements are set to zero, i.e. R is diagonal. Note that $Q$ is chosen as in (4.2).


Figure 4.5

Figure 4.6 shows the response when the system has been subjected to the same series of steps as in Figure 4.2 with $R_{22}$ chosen as $1.01(\mathrm{i}=21)$. In Figure 4.7 one can see how the values of the angle-of-attack, $\alpha$, vary over the scope of the simulation. Note that the minimum value lies just above -0.24 radians (or roughly -13.75 degrees).


Figure 4.6: Response to a series of steps for the closed loop nonlinear system. The red curve shows the commanded normal acceleration. The black curve corresponds to the conventional series expansion approach (First order SE). The blue curve corresponds to the enhanced velocity based approach (Enhanced $V B)$.


Figure 4.7: Angle-of-attack, $\alpha$, belonging to the response shown in Figure 4.6. The minimum value, for both linearization techniques, lies at roughly -0.24 radians (with only a slight initial difference).

Figure 4.8 shows the response with the same commanded normal acceleration and weighting matrices as in Figure 4.6. However, the initial step size has been increased to $30 g$. The purpose of the increased step size is to force the missile to operate at the extremes of the envelope. In Figure 4.9 one can see that when the angle-of-attack approaches the boundary of the operating domain the step response slows down and exhibits undershoot (to such an extent that the missile does not reach its commanded acceleration within the prescribed time).


Figure 4.8: Response to a series of steps for the closed loop nonlinear system. The red curve shows the commanded normal acceleration. The black curve corresponds to the conventional series expansion approach (First order $S E$ ). The blue curve corresponds to the enhanced velocity based approach (Enhanced $V B)$.


Figure 4.9: Angle-of-attack, $\alpha$, belonging to the response shown in Figure 4.8. As $\alpha$ approaches it minimum allowable value (of roughly -0.349 radians) the response of the conventional series expansion technique exhibits severe undershoot, as shown in Figure 4.8

### 4.4 Discussion

The static feedback coefficients calculated are, per definition, only intended to operate on the deviation-variables around each local equilibrium and a transformation of the controller output is therefore necessary. From (4.5) it becomes clear that in order to perform this transformation one must, to the already calculated controller effort $\Delta u$, add the controller output at equilibrium $u_{e}$.

$$
\begin{array}{r}
\Delta u=-K \Delta x=-K\left(x-x_{e}\right)=u-u_{e} \\
\Rightarrow u=\Delta u+u_{e} \tag{4.5}
\end{array}
$$

As presented in [1] there are several ways to address this problem. If the system includes pure integral action this additional control signal will inherently equal zero and does not need to be accounted for. If the system does not contain integral action one can calculate the value of this signal through the original non-linear system. However, such an approach can, depending on the system at hand, include fairly complex calculations. Furthermore, since this approach is based on the original non-linear model it becomes sensitive to modelling errors and, as stated in [1], seems to have little practical value. The seemingly most straightforward approach to solve this problem is to include integral action in each linear controller and so implicitly create this transformation through the feedback loop of the system. If such an approach is chosen, relying on integral action in the controller structure, one should keep in mind that the effect of the integral term is not immediate. In other words, proper tracking will not occur until the integral term has had sufficient time to correct for the lack of the aforementioned equilibrium control signal. This delay, meaning that belonging to the integral term, can be lessened by giving the integral part high weighting in the corresponding weighting matrix. However, depending upon the system and application at hand, this may not be realizable.

In Figure 4.8 it is easily observed that the conventional series expansion technique yields considerably worse performance compared to that of the enhanced velocity based technique. However, it remains to be proven whether this difference in performance stems from hidden coupling or is simply a result of other non-accounted factors. Due to a fairly complex structure, involving several partial derivatives (as shown in Section 3.1), the presence of hidden coupling terms is in practice hard to confirm analytically. However, as seen from the collected data in Figures $4.5 \mathrm{a}, 4.5 \mathrm{~b}, 4.8$ and 4.9 , when the gradient with respect to $\alpha$ becomes sufficiently high, the performance of the conventional series technique lessens noticeably. As such there is a strong indication that hidden coupling may be involved and in fact, as suggested, can be resolved by the use of the velocity based linearization technique.

## Chapter 5

## Modelling of aircraft dynamics

In this chapter follows a short description of the modelling of the aircraft dynamics including the current aircraft layout, linearization of governing equations and the extension of the resulting linear system to include servo dynamics.

### 5.1 Rigid body dynamics

At the core of the Ares Mars simulator lies the general rigid body dynamic equations, for details regarding these equations and their derivation the reader is referred to, for example, [23]. A short summary will however be given, starting by simply stating the relevant equations

$$
\begin{array}{r}
\dot{\bar{V}}=\frac{\bar{F}}{m}-\frac{\dot{m}}{m} \bar{V}-[\omega] \bar{V} \\
\dot{\bar{\omega}}=I^{-1} \bar{G}-I^{-1} \dot{I} \bar{\omega}-I^{-1}[\omega] I \bar{\omega} \tag{5.2}
\end{array}
$$

where $\bar{V}$ and $\bar{\omega}$ denote the speed- and angular rate vector respectively. $\bar{F}$ and $\bar{G}$ are the force and moment vectors containing all external forces and moments acting on the aircraft. These forces and moments are in turn generated by models of the engine and the aerodynamics, which are functions of system states, inputs, outputs and atmospheric properties. The parameters $m$ and $I$ denotes mass and the tensor of mass-moment-of-inertia respectively. The reference point is the aircraft center-of-gravity (cg). Furthermore $[\omega]$ denotes the skew-symmetric matrix $\left[\begin{array}{ccc}0 & -\omega_{z} & \omega_{y} \\ \omega_{z} & 0 & -\omega_{x} \\ -\omega_{y} & \omega_{x} & 0\end{array}\right]$, where $\omega_{x}, \omega_{y}$ and $\omega_{z}$ are pitch, roll and yaw rate respectively. Note that a left-multiplication with $[\omega]$ is equivalent to performing a cross-product with $\bar{\omega}$. Further note that since mass may, locally in time, be treated as constant the derivative terms of the mass and mass-moment-of-inertia are neglected during simulation. Furthermore, the system is extended with a set of kinematic equations describing the bearing of the aircraft in the inertial, or earth-fixed, frame of reference.

To summarize, the non-linear state-space representation of the rigid body dynamics may be written as

$$
\begin{align*}
\frac{\partial}{\partial t}\left[\begin{array}{c}
\bar{V} \\
\bar{\omega} \\
\bar{\Theta}
\end{array}\right] & =\left[\begin{array}{ccc}
-[\omega] & 0 & 0 \\
0 & -I^{-1}[\omega] I & 0 \\
0 & T & 0
\end{array}\right]\left[\begin{array}{c}
\bar{V} \\
\bar{\omega} \\
\bar{\Theta}
\end{array}\right]+\left[\begin{array}{cc}
\frac{1}{m} & 0 \\
0 & I^{-1} \\
0 & 0
\end{array}\right]\left[\begin{array}{l}
\bar{F} \\
\bar{G}
\end{array}\right] \\
\bar{\Theta} & =\left[\begin{array}{lll}
\phi, & \theta, & \psi,
\end{array}\right]^{T}  \tag{5.3}\\
T & =\left[\begin{array}{ccc}
1 & \sin (\phi) \tan (\theta) & \cos (\phi) \tan (\theta) \\
0 & \cos (\phi) & -\sin (\phi) \\
0 & \sin (\phi) / \cos (\theta) & \cos (\phi) / \cos (\theta)
\end{array}\right]
\end{align*}
$$

where $\bar{\Theta}$ denotes the Euler angle vector containing the Euler angles $(\phi, \theta, \psi)$. The matrix $T$ is a transformation matrix deriving from the predefined order of rotations belonging to the Euler angles.

### 5.2 Current aircraft layout

The current, and so far only, aircraft layout in Ares Mars is that of the open access ADMIRE model, [24]. It is, in terms of overall appearance, similar to that of the JAS 39-Gripen aircraft, meaning that available control surfaces include a single vertical rudder, two canards, leading edge flaps and inner and outer elevons, see Figure 5.1.


Figure 5.1: Principal overhead view of the ADMIRE aircraft model. Dark blue portions represent available control surfaces.

Other available control signals include pilot-leaver-angle (PLA) (which in extension relates to engine thrust). Note that since the motion of the system has six degrees of freedom (DOF) this aircraft has actuator redundancy, meaning that there are more ways to control the aircraft than what is principally needed. In order to resolve this redundancy one may, as discussed in [8], either turn to optimal control, such as LQ-control, or any form of control allocation, both of which will be further discussed in later sections. So far the standard for the Ares Mars simulator has been to apply both non-optimal control allocation and LQ-optimal control. Both canards and elevons have been coupled in pitch
and roll to create new control signals referred to as for instance; canard elevators and canard ailerons, depending on which motion the signal is to contribute to. To clarify, both canards and elevons have been coupled so that they operate symmetrically in pitch and anti-symmetrically in roll respectively.

### 5.3 Linearized flight mechanical model

Linearizing (5.3), by use of first order series expansion, about the equilibrium point $[\bar{V}, \bar{\omega}, \bar{\Theta}]^{T}=\left[\bar{V}_{0}, \bar{\omega}_{0}, \bar{\Theta}_{0}\right]^{T}$ yields the following set of equations:

$$
\begin{align*}
& \Delta \dot{V}_{x}=-\frac{\Delta T}{m}+\omega_{z_{0}} \Delta V_{y}+\Delta \omega_{z} V_{y_{0}}-\omega_{y_{0}} \Delta V_{z}-\omega_{y} \Delta V_{z_{0}}-\ldots  \tag{5.4}\\
& \ldots-g \cos \left(\theta_{0}\right) \Delta \theta \\
& \Delta \dot{V}_{y}=-\frac{\Delta C}{m}-\omega_{z_{0}} \Delta V_{x}-\Delta \omega_{z} V_{x_{0}}+\omega_{x_{0}} \Delta V_{z}+\omega_{x} \Delta V_{z_{0}}+\ldots  \tag{5.5}\\
& \ldots+g \cos \left(\phi_{0}\right) \cos \left(\theta_{0}\right) \Delta \phi-g \sin \left(\phi_{0}\right) \sin \left(\theta_{0}\right) \Delta \theta \\
& \Delta \dot{V}_{z}=-\frac{\Delta N}{m}+\omega_{y_{0}} \Delta V_{x}+\Delta \omega_{y} V_{x_{0}}-\omega_{x_{0}} \Delta V_{y}-\omega_{x} \Delta V_{y_{0}}-\ldots  \tag{5.6}\\
& \ldots-g \sin \left(\phi_{0}\right) \cos \left(\theta_{0}\right) \Delta \phi-g \cos \left(\phi_{0}\right) \sin \left(\theta_{0}\right) \Delta \theta \\
& \Delta \dot{\omega}_{x}=\frac{\Delta l}{I_{x}}+c_{11} \Delta \omega_{x}+c_{12} \Delta \omega_{y}+c_{13} \Delta \omega_{z}+\frac{I_{x y}}{I x} \Delta \dot{\omega}_{y}+\frac{I_{x z}}{I x} \Delta \dot{\omega}_{z}  \tag{5.7}\\
& \Delta \dot{\omega}_{y}=\frac{\Delta m}{I_{y}}+c_{21} \Delta \omega_{x}+c_{22} \Delta \omega_{y}+c_{23} \Delta \omega_{z}+\frac{I_{x y}}{I y} \Delta \dot{\omega}_{x}+\frac{I_{y z}}{I y} \Delta \dot{\omega}_{z}  \tag{5.8}\\
& \Delta \dot{\omega}_{z}=\frac{\Delta n}{I_{z}}+c_{31} \Delta \omega_{x}+c_{32} \Delta \omega_{y}+c_{33} \Delta \omega_{z}+\frac{I_{x z}}{I z} \Delta \dot{\omega}_{x}+\frac{I_{y z}}{I z} \Delta \dot{\omega}_{y}  \tag{5.9}\\
& \Delta \dot{\phi}=\Delta \omega_{x}+\tan \left(\theta_{0}\right) \sin \left(\phi_{0}\right) \Delta \omega_{y}+\tan \left(\theta_{0}\right) \cos \left(\phi_{0}\right) \Delta \omega_{z}+\ldots  \tag{5.10}\\
& \ldots+\tan \left(\theta_{0}\right)\left(\omega_{y_{0}} \cos \left(\phi_{0}\right)-\omega_{z_{0}} \sin \left(\phi_{0}\right)\right) \Delta \phi+\ldots \\
& \ldots+\sec { }^{2}\left(\theta_{0}\right)\left(\omega_{y_{0}} \sin \left(\phi_{0}\right)+\omega_{z_{0}} \cos \left(\phi_{0}\right)\right) \Delta \theta \\
& \Delta \dot{\theta}=\cos \left(\phi_{0}\right) \Delta \omega_{y}-\sin \left(\phi_{0}\right) \Delta \omega_{z}-\left(\omega_{y_{0}} \sin \left(\phi_{0}\right)+\omega_{z_{0}} \cos \left(\phi_{0}\right)\right) \Delta \phi  \tag{5.11}\\
& \Delta \dot{\psi}=\frac{\sin \left(\phi_{0}\right)}{\cos \left(\theta_{0}\right)} \Delta \omega_{y}+\frac{\cos \left(\phi_{0}\right)}{\cos \left(\theta_{0}\right)} \Delta \omega_{z}+\ldots  \tag{5.12}\\
& \ldots+\sec \left(\theta_{0}\right)\left(\omega_{y_{0}} \cos \left(\phi_{0}\right)+\omega_{z_{0}} \sin \left(\phi_{0}\right)\right) \Delta \phi+\ldots \\
& \ldots+\frac{\tan \left(\theta_{0}\right)}{\cos \left(\theta_{0}\right)}\left(\omega_{y_{0}} \sin \left(\phi_{0}\right)+\omega_{z_{0}} \cos \left(\phi_{0}\right)\right) \Delta \theta
\end{align*}
$$

Here the equations have been written out explicitly rather than in conventional matrix form. Expressions for coefficients $c_{11}$ to $c_{33}$ may be found in Appendix D. Note that

$$
\begin{align*}
\Delta \bar{F} & =[\Delta T, \Delta C, \Delta N]^{T}  \tag{5.13}\\
\Delta \bar{G} & =[\Delta l, \Delta m, \Delta n]^{T} \tag{5.14}
\end{align*}
$$

where $T$ denotes the force acting on the aircraft in the body-fixed coordinate system's x -direction and is defined as

$$
\begin{equation*}
T=\frac{\rho V^{2} S}{2} C_{T} \tag{5.15}
\end{equation*}
$$

where $\rho, V$ and $S$ denote local density of air, aircraft effective speed and reference surface respectively.

The linear forces $C$ (y-direction) and $N$ (z-direction) are defined similarly. $l$ in turn denote the roll moment acting on the aircraft about the body-fixed coordinate system's x -axis and is defined as

$$
\begin{equation*}
l=\frac{\rho V^{2} S b}{2} C_{l} \tag{5.16}
\end{equation*}
$$

where $b$ denotes wing span and equally reference length in the lateral direction. The moments $m$ (y-axis) and $n$ (z-axis) are, again, defined similarly. Note that both $C_{T}$ and $C_{l}$ are nonlinear functions of several parameters, all of which are presented in Table 5.1

Table 5.1: Names and notation for all variables of which the coefficients of forces and moments are functions.

| Parameter | Notation/symbol |
| :---: | :---: |
| Angle-of-attack | $\alpha$ |
| Sideslip angle | $\beta$ |
| Mach number | $M$ |
| Canard elevator deflection | $\delta_{c e}$ |
| Canard aileron deflection | $\delta_{c a}$ |
| Elevator inner deflection | $\delta_{e i}$ |
| Elevator outer deflection | $\delta_{e o}$ |
| Aileron inner deflection | $\delta_{a i}$ |
| Aileron outer deflection | $\delta_{a o}$ |
| Rudder deflection | $\delta_{r}$ |
| Leading edge deflection | $\delta_{l e}$ |
| Angle-of-attack rate | $\dot{\alpha}$ |
| Sideslip angle rate | $\dot{\beta}$ |
| Pitch rate | $p$ |
| Roll rate | $q$ |
| Yaw rate | $r$ |
| Pitch angular acc. | $\dot{p}$ |
| Roll angular acc. | $\dot{q}$ |
| Normal load factor | $n_{z}$ |
| Canard elevator rate | $\dot{\delta}_{c e}$ |
| Canard aileron rate | $\dot{\delta}_{c a}$ |
| Elevator inner rate | $\dot{\delta}_{e i}$ |
| Elevator outer rate | $\dot{\delta}_{e o}$ |
| Aileron inner rate | $\dot{\delta}_{a i}$ |
| Aileron outer rate | $\dot{\delta}_{a o}$ |
| Rudder rate | $\dot{\delta}_{r}$ |
| Pilot lever angle | $P L A$ |
|  |  |

The forces and moments are therefore linearized accordingly

$$
\begin{align*}
\Delta T & =\frac{d T}{d \alpha} \Delta \alpha+\frac{d T}{d \beta} \Delta \beta+\cdots+\frac{d T}{d \dot{\delta}_{r}} \Delta \dot{\delta}_{r}+\frac{d T}{d P L A} \Delta P L A  \tag{5.17}\\
\Delta l & =\frac{d l}{d \alpha} \Delta \alpha+\frac{d l}{d \beta} \Delta \beta+\cdots+\frac{d l}{d \dot{\delta}_{r}} \Delta \dot{\delta}_{r}+\frac{d l}{d P L A} \Delta P L A \tag{5.18}
\end{align*}
$$

Evaluating derivative terms and inserting expressions for $\Delta \alpha, \Delta \beta$ etc. finally yields a system of the form

$$
\begin{align*}
A_{1} \Delta \dot{x} & =A_{2} \Delta x+B \Delta u \\
\Delta x & =[\Delta \bar{V}, \Delta \bar{\omega}, \Delta \bar{\Theta}]^{T}  \tag{5.20}\\
\Delta u & =[\Delta \bar{F}, \Delta \bar{G}]^{T}
\end{align*}
$$

Note that derivatives of aerodynamic coefficients are calculated as central derivatives from tabulated data. The inputs of the flight mechanical system are hence all control surface deflections along with their respective rates (with the exception of leading edge flaps). The states include all linear and angular velocities, as described in Section 5.1, along with the introduced Euler angles. A summary is given below.

$$
\begin{align*}
\text { States }: & x=[\bar{V}, \bar{\omega}, \bar{\Theta}]^{T} \\
\bar{V}= & {\left[V_{x}, V_{y}, V_{z}\right]^{T} } \\
\bar{\omega}= & {\left[\omega_{x}, \omega_{y}, \omega_{z}\right]^{T} }  \tag{5.21}\\
\bar{\Theta}= & {[\phi, \theta, \psi]^{T} } \\
\text { Inputs }: & u=\left[\delta_{c e}, \delta_{c a}, \delta_{e i}, \delta_{a i}, \delta_{e o}, \delta_{a o}, \delta_{r}, \delta_{l e}, \ldots\right. \\
& \left.\dot{\delta}_{c e}, \dot{\delta}_{c a}, \dot{\delta}_{e i}, \dot{\delta}_{a i}, \dot{\delta}_{e o}, \dot{\delta}_{a o}, \dot{\delta}_{r}, \delta_{P L A}\right]^{T}
\end{align*}
$$

In order to create full freedom for the user, the coupling between canards and elevons are broken. This is simply done by post-multiplying the linear input matrix $B$ with the inverse of the connectivity/coupling matrix

$$
\begin{array}{r}
M=\left[\begin{array}{cccc}
J & 0 & \ldots & 0 \\
0 & J & & \\
\vdots & & \ddots & \\
0 & & & J
\end{array}\right]  \tag{5.22}\\
J=\left[\begin{array}{cc}
1 & -1 \\
1 & 1
\end{array}\right]
\end{array}
$$

This results in the new system

$$
\begin{array}{r}
A_{1} \Delta \dot{x}=A_{2} \Delta x+\hat{B} \Delta u \\
\hat{B}=B M^{-1} \tag{5.23}
\end{array}
$$

The new inputs to the system are

$$
\begin{array}{r}
u=\left[\delta_{r c}, \delta_{l c}, \delta_{\text {rie }}, \delta_{l i e}, \delta_{\text {roe }}, \delta_{l o e}, \delta_{r}, \delta_{l e}, \ldots\right. \\
\left.\dot{\delta}_{r c}, \dot{\delta}_{l c}, \dot{\delta}_{r i e}, \dot{\delta}_{l i e}, \dot{\delta}_{\text {roe }}, \dot{\delta}_{l o e}, \dot{\delta}_{r}, \delta_{P L A}\right]^{T} \tag{5.24}
\end{array}
$$

All subscripts are explained in Table 5.2.
Table 5.2: Abbreviations of control signal subscripts.

| Subscript/abbreviation | Name/parameter |
| :---: | :---: |
| rc | right canard |
| lc | left canard |
| rie | right inner elevon |
| lie | left inner elevon |
| roe | right outer elevon |
| loe | left outer elevon |
| r | rudder |
| le | leading edge flaps |
| PLA | pilot leaver angle |

### 5.4 Servo dynamics

As shown in (5.24) both control surface deflections and control surface rates are inputs to the linearized flight mechanical model. However, in a strict physical sense, these are not allowed to vary independently. A servo model is therefore necessary in order to couple deflections and rates. In order to properly extend the existing model with the servo dynamics the linearized servo dynamics should, preferably, be of (at least) second order. However, it is not uncommon practice to appreciate servo dynamics with a simple first order model, and a small description regarding the necessary approach for both first- and second order servo dynamics will be given. Note that $\delta_{c}$ denotes commanded deflection.

If the linear servo models were of second order the extended model, now including deflections and rates, would take the form that is shown in Figure 5.2. $\tilde{A}$ and $\tilde{B}$ denote the extended, or new, system matrices. The lower square block matrix of $\tilde{A}$ will be diagonal given that there are no couplings between actuators.


Figure 5.2: Extended state space model with a linear second order servo. Black dots represent either scalar (potentially) non-zero values or blocks of the same. Note that the tall matrix beneath $\delta$ and $\dot{\delta}$ is the portion of the old input matrix $B$ that concerns deflections and rates (not PLA).

If the linear servo model instead were of first order there would be no present information on how the control surfaces accelerate in response to commanded deflection. The rows and columns crossed with red lines in Figure 5.3 must subsequently be left out from the augmented system matrix $\tilde{A}$. However, if the columns containing information about the effects of control surface rates were to be left out completely then information would inevitably be lost and the resulting model would suffer from problems with accuracy.


Figure 5.3: Extended state space model with a linear first order servo. Information about the effects of control surface rates are intentionally left out.

A simple solution to this problem is to replace all rates, by means of the linear first order servo model, and add the resulting coefficients at appropriate locations, as shown in Figure 5.4.


Figure 5.4: Extended state space model with a linear first order servo. Information about the effects of control surface rates have been added in both the state matrix $\tilde{A}$ and the input matrix $\tilde{B}$. Note that $B_{d o t}$ denote the portion of the old input matrix $B$ that only concern control surface rates. Furthermore, $A_{s}$ and $B_{s}$ denote state- and input matrix for the linear servo model. A superscript in the form of an asterisk has been added to $\delta$ and $\delta_{c}$ to indicate that the values contained in the column below are not the same as in Figure 5.2.

By employing this approach information is retained. However, the commanded deflections will now directly influence states in an nonphysical manner. This is, unfortunately, an unavoidable consequence deriving from the nonphysical first order servo model. After the extension with the servo model, regardless of its order, the new inputs to the system becomes

$$
\begin{equation*}
u=\left[\delta_{r c c}, \delta_{l c c}, \delta_{\text {riec }}, \delta_{l i e c}, \delta_{\text {roec }}, \delta_{l o e c}, \delta_{r c}, \delta_{l e c}, \delta_{P L A}\right]^{T} \tag{5.25}
\end{equation*}
$$

where the addition $c$ to the subscripts denotes commanded.
The current servo model in Ares is a simple first order plus dead time (FOPDT) model. Unfortunately, LQ-regulation techniques do not accommodate for exponential (conventional) delay terms and the servo model must be linearized. This is done by a Padé approximation to yield a model of the form

$$
\begin{equation*}
G_{\text {servo }}=\frac{-s+c_{1}}{c_{2} s^{2}+c_{3} s+c_{4}} \tag{5.26}
\end{equation*}
$$

Here, the model is represented in transfer function form (or in terms of the Laplace variable $s$ ) rather than in the previous state space form. This is simply to highlight the presence of the right half plane (RHP) zero that derives from the linearization of the delay term. Note that this zero in fact is necessary to approximate the delay, the existence of such does not invalidate the above described procedure. However, when implementing one needs to make sure that the state space matrices corresponding to (5.26) are modified such that the input matrix $C_{\text {servo }}$ is the unity matrix. This is achieved by performing a change of basis with the new base matrix $C_{\text {servo }}^{-1}$.

### 5.5 Validation

Equilibrium values from two different trim conditions are inserted into the linearized equations derived in Section 5.3. These trim points are $\left[M, H_{\text {true }}\right]=$ $[0.75,1100]$ and $\left[M, H_{\text {true }}\right]=[0.22,3000]$, where $M$ is the Mach number and $H_{\text {true }}$ denotes true altitude. These points are chosen to represent conditions in which control authority is high and low respectively. Trim is achieved with fixed level canards ( $\delta_{r c}=\delta_{l c}=0$ ) and open loop control from pilot stick deflection to elevons, i.e. all elevons operate symmetrically in pitch. The pilot model takes the form of simple proportional feedback acting on pitch rate. The aircraft is initialized at zero bank and yaw angle (with no roll or yaw rate present) from which the pilot simply trims the plane by adjusting the stick deflection until the aircraft reaches equilibrium. Two linear systems are hence created. The response to a step of $-1^{\circ}$ (or roughly -0.0175 rad ) in pilot stick deflection is calculated and compared between the nonlinear simulator and the derived linear systems. The results are shown in Figures 5.5-5.7.


Figure 5.5: Elevator deflection, $\delta_{e}$, as a function of time. Note that the signal $\delta_{e}$ consists of all available elevons operating as one. As expected, the curves overlap at almost all instances. However one can distinguish a small undershoot in the response of the linear system deriving from the linearized servo model.


Figure 5.6: (a): Pitch rate $q$ (or equally $w_{y}$ ). (b): Euler angle $\theta$. (c): Speed in body fixed x-direction, $V_{x}$. (d): Speed in body fixed z-direction, $V_{z}$. Trim point [0.22 3000].


Figure 5.7: (a): Pitch rate $q$ (or equally $w_{y}$ ). (b): Euler angle $\theta$ (about the inertial frame y-axis). (c): Speed in body fixed x-direction, $V_{x}$. (d): Speed in body fixed z-direction, $V_{z}$. Trim point [0.75 1100].

As expected agreement between non-linear and linear simulations lessen when control authority is high (given the same step in pilot stick deflection). However, in addition, there seems to be some coupling causing the pitch rate to deviate at approximately 0.3 seconds. In Figures $5.8 \mathrm{a}-5.8 \mathrm{c}$ one can see that there is a slight response in both $\phi, \psi$ and $V_{y}$. However, there are no inertial couplings and there appear to be no added disturbances in the simulator. Unfortunately, the number of signals within Ares are in the order of several hundred, meaning that tracking the source of the apparent coupling would be extremely time consuming.


Figure 5.8: (a): Euler angle $\phi$. (b): Euler angle $\psi$. (c): Speed in body fixed y-direction, $V_{y}$. Trim point [0.75 1100]

In [8] the authors make use of the ADMIRE model (ver. 3.4h, 2003) when giving an example of the possible benefits of control allocation for a simplified flight controller. They choose their states as $[\alpha, \beta, p, q, r]^{T}$. Their inputs have, similarly to the earlier approach in Ares, been coupled such that canards and elevons operate symmetrically. Their inputs are therefore $\left[\delta_{c}, \delta_{l e}, \delta_{r e}, \delta_{r}\right.$ ], where $c$ denotes canard, le denotes left elevon, re denote right elevon and $r$ denotes rudder. They perform their linearization about trimmed values at 0.22 Mach and 3000 m . Their resulting state space matrices $A$ and $B$ are given in (5.28) and (5.30). In comparison, when trimmed values from the same operating point are inserted in to the linearized flight mechanical model from Section 5.3, one receives the state space matrices shown in (5.27) and (5.29). The later has been given the subscript $A$ denoting Ares. Note that the Euler angles, engine related signals and leading edge flaps have been excluded. Furthermore, the input matrix, $B_{A}$, has been rearranged to yield the same order of control signals as in [8]. The states in common are $p, q$ and $r$ (or equally $\omega_{x}, \omega_{y}$ and $\omega_{z}$ ) and as such the lower right $3 \times 3$ block matrix of $A$ and the lower $3 \times 4$ portion of $B$ may be compared.

$$
\begin{align*}
A_{A} & =\left[\begin{array}{cccccc}
-0.052 & 0 & 0.012 & 0 & -16.312 & 0 \\
0 & -0.118 & 0 & 16.490 & 0 & -69.733 \\
-0.129 & 0 & -0.601 \\
0 & -0.138 & 0 & 0 & 68.784 & 0 \\
-0.004 & 0 & 0.035 \\
0 & 0.009 & 0 & \begin{array}{cc}
-1.010 & 0 \\
0 & -0.503 \\
-0.096 & 0
\end{array} & 0.559 \\
A & =\left[\begin{array}{ccccc}
-0.543 \\
0 & 0.014 & 0 & 0.978 & 0 \\
0 & -0.118 & 0.222 & 0 & -0.966 \\
2.622 & -0.003 \\
0 & 0.708 & \left.\begin{array}{llll}
-0.997 \\
0 & 0 & -0.506 & 0 \\
-0.094 & 0 & -0.213
\end{array}\right]
\end{array}\right] \\
B_{A} & =\left[\begin{array}{cccc}
-1.297 & -0.169 & -0.169 & 0 \\
0 & 0.812 & -0.812 & 2.073 \\
-1.012 & -6.251 & -6.251 & 0 \\
0 & -4.546 & 4.546 & 1.342 \\
1.708 & -1.379 & -1.379 & 0 \\
0 & -0.306 & 0.306 & -0.909
\end{array}\right] \\
B & =\left[\begin{array}{cccc}
0.007 & -0.087 & -0.087 & 0 \\
0 & 0.012 & -0.012 & 0.029
\end{array}\right] \\
\left.\begin{array}{ccccc}
0 & -4.242 & 4.242 & 1.487 \\
1.653 & -1.274 & -1.274 & 0.002 \\
0 & -0.281 & 0.281 & -0.882
\end{array}\right]
\end{array}\right. \tag{5.27}
\end{align*}
$$

Note that all values have been rounded off to three decimal points. With the matrices from [8] as basis the deviation (of all non-zero values) ranges from roughly $0.47 \%$ to $9.74 \%$ with an average of about $5.62 \%$. The largest deviation belongs to the input matrix $B_{A}$, but this difference may stem from the control technique applied when performing the trim in question.

Given the results presented it should be concluded that the linearized models compare well to the nonlinear simulator.

## Chapter 6

## Tuning of weighting matrices

One of the apparent drawbacks of LQ-control is the inability to explicitly control where the closed loop poles are placed, instead one has to manually tune the weighting matrices $Q$ and $R$ until a suitable location is found. Several attempts have been made at finding ways to automatically tune these matrices including heuristic population-based optimization schemes such as particle swarm, evolution algorithm and bees algorithm optimization, e.g. [16], [17]. These have to various degrees been successfully implemented, however there seems to be an (understandable) lack of representation of higher order MIMO systems. The developed method aims at being applicable for a large variety of possible aircraft models, for some of which decoupling of lateral and longitudinal motion may not be possible. The system that describes the motion of the aircraft will therefore have, at a minimum, six dynamic equations relating to linear and angular velocities respectively. Furthermore there should be no restriction to the manageable number of control signals (other than that it must be finite). It is therefore necessary to find a more systematic approach to the automatic tuning problem. One such an approach may be to solve the inverse LQ-problem. To clarify, given that the designer chooses a set of poles an eigenstructure assignment approach could yield a feedback matrix $K$. This feedback matrix could then be analyzed through the inverse LQ-problem to generate values of weighting matrices $Q$ and $R$. The approach is illustrated in Figure 6.1.


Figure 6.1: Given a set of eigenvalues (spectrum) of the closed loop state matrix, conventionally denoted $A_{c}$, a static feedback matrix $K$ is generated through an eigenstructure assignment procedure. This feedback matrix $K$ is then associated with a set of weighting matrices $Q$ and $R$ such that $K$ is LQ-optimal.

Note that even though the eigenstructure is assigned, the system might not behave according to specifications, depending on how the structure in question is chosen. It might very well be that the poles are chosen based on a rule of thumb or perhaps experience from previous design iterations. Regardless, the functional relationship between the system poles and the transient behaviour, sensitivity functions and/or other design outputs is in general difficult to appreciate. Therefore the design process might not be completed after generating the feedback matrix $K$ and the weighting matrices $Q$ and $R$ become relevant starting points for further design iterations. The advantage of this approach is that, given that the feedback matrix $K$ is LQ-optimal, the system is guaranteed at least 60 degrees of phase margin and infinite gain margin (however only one-half gain reduction tolerance) [15]. Furthermore, the weighting associated with the generated matrices $Q$ and $R$ grants insight in to the implied priorities of the eigenstructure assignment.

### 6.1 The inverse LQ-problem

The inverse LQ-problem may, somewhat informally, be defined as; given the stabilizing state feedback matrix $K$ find the real valued set of weighting matrices $\Omega=\{Q, R\}$ that generates $K$ as a solution to the LQ-problem. In [9] the author presents a method of finding such a set of equivalent $Q$ 's assuming $R$ is fixed at unity. He consequently derives and proposes the relation

$$
\begin{equation*}
Q=K^{T} K-D A-A^{T} D \tag{6.1a}
\end{equation*}
$$

where

$$
\begin{equation*}
D=\left(B^{+T} K-\left(I-B^{+T} B^{T}\right) L\right) \tag{6.1b}
\end{equation*}
$$

and $A$ and $B$ are the conventional system matrices such that $\dot{x}=A x+B u$ and the superscript + denotes the Moore-Penrose pseudoinverse. $L$ is an arbitrary symmetric matrix of appropriate size. As can be seen in (6.1b), the freedom in the choice of $L$ is lost, and the problem becomes bijective, when the input matrix $B$ has a left inverse. Further note that this relation does not guarantee the necessary positive semi-definiteness of $Q$.

> Note: Simply put, if $B$ has a true left inverse such that $\left(I-B^{+T} B^{T}\right)=0$, the set of equivalent matrices $Q$ collapse (into a single point) and hence one specific feedback matrix $K$ maps to one specific weighting matrix $Q$, and vise versa (bijective). Further note that if this is the case and $Q$ does not fulfill the necessary requirements, i.e. is not symmetric and positive semi-definite, then $K$ is simply not LQ-optimal.

By definition, any sum of positive semi-definite matrices remains positive semi-definite. A sufficient, but not necessary, condition for the positive semidefiniteness of matrix $Q$ can therefore be broken down into the definiteness of the three separate terms on the right hand side of (6.1a), i.e. $K^{T} K,-D A$, and $-A^{T} D$. The term $K^{T} K$ will, regardless of the value of $K$, be positive definite (the proof is trivial and is therefore omitted). However the definiteness of state matrix $A$ may vary depending on the nature of the system in question, i.e. whether all included states-, only some- or none are stable.

Therefore to predict the definiteness of the matrix products $-D A$ and $-A^{T} D$ is, in general, exceedingly difficult. Suppose, for the sake of argument, that the matrices $A$ and $A^{T}$ were known to be negative definite, then the matrices $-A$ and $-A^{T}$ would inherently be positive definite. However the same could not necessarily be said about the products $-D A$ and $-A^{T} D$, for such to be true it must hold that both $A$ and $A^{T}$ commute with the matrix $D$ and $D$ is likewise positive definite. By inspection it may be concluded that, by an appropriate choice of $L$, $D$ could in fact be made (at least) positive semi-definite (e.g. $L=-\lambda I$, where $I$ is the unity matrix of the same size as matrix $A$ and $\lambda$ is an arbitrary number chosen sufficiently large). However, as the matrix $I-B^{+T} B^{T}$ per definition is rank deficient, it is clear that the choice of $L$ only has limited effect on the structure of $D$ and no guarantee could be made that either $A$ or $A^{T}$ commutes with $D$ for all possible values of $A$.

Note: Through the property of subadditivity we know that $\operatorname{rank}\left(I-B^{+T} B^{T}\right) \leq \operatorname{rank}(I)-\operatorname{rank}\left(B^{+T} B^{T}\right)=n-\operatorname{rank}\left(B^{+T} B^{T}\right)$, where $n$ is the number of system states. Therefore, the term $I-B^{+T} B^{T}$ will only have full rank if (and only if) $\operatorname{rank}\left(B^{+T} B^{T}\right)=0$. This is clearly only fulfilled when $B=\overline{0}$. Therefore, any non-zero input matrix $B$, regardless of shape and values contained, will cause the term $I-B^{+T} B^{T}$ to be rank deficient.

Note that a brute force approach, of simple trial and error of the elements of $L$, will quickly become infeasible with an increasing number of system states. As such, the problem (of solving (6.1a)-6.1b)) must be approached analytically to, if such exist, yield an implicit solution in terms of $L$ and $Q$, i.e. $Q=f(L)$. Note that (6.1a) could be rearranged to yield a Sylvester equation in $D$ which could be solved efficiently by use of, for example, the Kronecker product, as shown in (6.1a*)-(6.1a**)

$$
\begin{gather*}
A^{T} D+D A=K^{T} K-Q  \tag{*}\\
\Leftrightarrow\left(I \otimes A^{T}+A^{T} \otimes I\right) \operatorname{vec}(D)=\operatorname{vec}\left(K^{T} K-Q\right) \tag{**}
\end{gather*}
$$

However, as the state matrix $A$ might be singular due to inclusion of controller states (i.e. tracking errors) the solution might not be unique, as suggested by the Sylvester-Rosenblum-theorem, see for example [25].

Note: Excluding some of the mathematical formality the Sylvester-Rosen-blum-theorem could simply be stated as: "The equation $\mathcal{A} X+X \mathcal{B}=\mathcal{C}$ has a unique solution in $X$ for every $\mathcal{C}$ if and only if $\mathcal{A}$ and $-\mathcal{B}$ do not share any eigenvalues". Note that if the state matrix $A$ has been augmented with integral states (controller states) some of the eigenvalues inevitably takes the value zero. As such the matrices $-A$ and $A^{T}$ consequently shares these eigenvalues and a unique solution is no longer guaranteed, i.e. the matrix $\left(I \otimes A^{T}+A^{T} \otimes I\right)$ is non-invertible.

One therefore has to turn to solution methods including reduced row echelon forms or alike. Introducing the following definitions: $\operatorname{vec}(X)=X_{v e c}$, $(I \otimes X+X \otimes I)=X_{\text {Sylv }}$ and $I \otimes X=X_{\text {Kron }}$, the overall system (that guarantees the necessary definiteness and symmetry of $Q$ and $L$ ) becomes

$$
\begin{align*}
\left(A^{T}\right)_{\text {Sylv }} D_{v e c} & =\left(K^{T} K\right)_{v e c}-Q_{v e c}  \tag{6.2}\\
D_{v e c} & =\left(I-B^{+T} B^{T}\right)_{K r o n} L_{v e c}-\left(B^{+T} K\right)_{v e c}  \tag{6.3}\\
S L_{v e c} & =\overline{0}  \tag{6.4}\\
S Q_{v e c} & =\overline{0}  \tag{6.5}\\
Q & \geq 0 \tag{6.6}
\end{align*}
$$

where $S$ is a matrix containing all necessary symmetry conditions on the elements of $Q$ and $L$. To give a brief example, the first row of $S$ would take the form $[0,1 \ldots-1,0 \ldots 0]$, where the negative one appears at the $(\mathrm{n}+1)^{\prime}$ 'th position such that the first equation of (6.4) reads $L_{12}-L_{21}=0$. Note that $S$ will have in total $\frac{n(n-1)}{2}$ rows. Further note that among the challenges, of efficiently solving (6.2) - (6.6), lies the need to express (6.6) linearly in terms of $Q_{v e c}$. One possible linear and sufficient condition, that ensures positive semi-definiteness of $Q$, is that all diagonal elements are non-negative and that $Q$ simultaneously is diagonally dominant. Such restrictions in the structure of $Q$ may however result in a lack of viable solutions. In contrast one could create a set of polynomial conditions on the elements of $Q$ by use of the Schur complement, however these might, as implied, make (6.2) - (6.6) analytically hard to solve.

Note: Given the partitioning $M=\left[\begin{array}{cc}\mathcal{A}^{T} & \mathcal{B} \\ \mathcal{C}\end{array}\right]$, the Schur complement of the block $\mathcal{A}$ is defined as $\mathcal{M} / \mathcal{A}=\mathcal{C}-\mathcal{B}^{T} \mathcal{A}^{-1} \mathcal{B}$. It can be proven that the matrix $\mathcal{M}$ is positive definite if and only if both $\mathcal{A}$ and $\mathcal{M} / \mathcal{A}$ are positive definite. Assuming that $\mathcal{A}$ is invertible and that $\mathcal{B}$ has one column the Schur complement $\mathcal{M} / \mathcal{A}$ will take the form of a rational polynomial of degree $\frac{\mathcal{O}(i+1)}{\mathcal{O}(i)}$ (in the elements of matrix $\mathcal{M}$ ), where $\mathcal{A} \in \mathcal{R}^{i \times i}$. However, assuming that the determinant of the matrix $\mathcal{A}$ is positive, this Schur condition for positive definiteness may be rewritten as a polynomial of degree $\mathcal{O}(i+1)$. The same principles may then be reiterated for successively smaller portions of $M$ each giving rise to a polynomial of the above form.

Given all stated above there seems to be no simple approach to guarantee a viable analytic solution given arbitrary $A, B$ and $K$.

Noting that (6.1) is linear in $L$, another approach to finding a solution may be to reformulate the problem as a semi-definite program which could be solved by, for example, modern interior-point methods. In practice this could be achieved by use of the function feasp in the Matlab LMI toolbox or by use of the open access Matlab-based modeling language YALMIP, [22]. In contrast the problem could likewise be solved as a constrained non-linear optimization problem as presented in [18]. The degrees of freedom, relating to the choice of $L$, are here diminished by the choice of a suitable objective function, $f(Q(L))$. Note that the choice $L=0$ causes (6.1) to reduce to the ARE and the freedom in the variable $L$ is instead inherited in the set of equivalent matrices $P$ (by introducing the necessary optimality condition $K=R^{-1} B^{T} P$ ). For convenience, and without loss of generality, this choice $(\mathrm{L}=0)$ is hereby assumed throughout.

The optimization problem thus becomes

$$
\begin{array}{cl}
\min _{P} & f(Q(P)) \\
\text { s.t. } & P=P^{T} \\
& K=B^{T} P \\
& Q=\left(A-\frac{1}{2} B K\right)^{T} P+P\left(A-\frac{1}{2} B K\right)  \tag{6.7}\\
& P \geq 0 \\
& Q \geq 0
\end{array}
$$

In [18] the authors choose the condition-number of the block matrix $\left[\begin{array}{ll}Q & 0 \\ 0 & R\end{array}\right]$ as the objective function $f(Q(P))$. As such they ensure low loss of significant digits when performing operations including $Q$ and $R$ (note that they do not assume that $R$ is fixed at unity). In [19] the authors instead suggest $\bar{\sigma}(Q-\operatorname{diag}(Q))$ as the objective function, where $\bar{\sigma}(X)$ denotes the maximum singular value of the matrix $X$. This choice ensures that $Q$ is maximally diagonal and hence that the weighting implied by $Q$ is easily interpreted. In both [18] and [19] the authors prove that the resulting optimization problem is convex, however in the former case with the necessity of LMI constraints. Note that the constraint $Q \geq 0$ can be rewritten as $\min (\lambda(Q)) \geq 0$ and likewise for $P$, where $\lambda(X)$ denotes the set of eigenvalues of the matrix $X$. As pointed out in [19] however, depending on formulation, the problem may not be differentiable at all instances and a non-gradient based search method might be necessary.

### 6.1.1 Solution to the inverse LQ-problem

The solution to the inverse problem is found by solving problem (6.7) by means of non-linear optimization with $\bar{\sigma}(Q-\operatorname{diag}(Q))$ as the objective function. The constraints $Q \geq 0$ and $P \geq 0$ are rewritten in the form $\min (\lambda(Q)) \geq 0$ and $\min (\lambda(P)) \geq 0$ respectively. The implementation is performed by use of the Matlab function fmincon with the default interior-point algorithm. The problem is convex, as proven in [19], and the initial values of all elements in $P$ are arbitrarily set to one. The matrix $P$ is further vectorized (denoted $P_{\text {vec }}$ ) and the symmetry condition $P=P^{T}$ is rewritten as $S P_{v e c}=0$, where the matrix $S$ contains all necessary symmetry conditions on all elements of $P$. The optimality condition $K=B^{T} P$ may be rewritten in terms of $P_{v e c}$ as $K_{v e c}=\left(I \otimes B^{T}\right) P_{v e c}$, where $\otimes$ denotes the Kronecker product. Combining these conditions on $P_{\text {vec }}$ yields

$$
\left[\begin{array}{c}
S  \tag{6.8}\\
I \otimes B^{T}
\end{array}\right] P_{v e c}=\left[\begin{array}{c}
0 \\
K_{v e c}
\end{array}\right]
$$

to which one may find the associated set of solutions by use of the reduced row echelon form to in turn yield $P_{v e c}$ as a linear function of the free variables in the same (denoted $P_{\text {free }}$ ), i.e. those instances in $P_{\text {vec }}$ which are not necessary to uphold the given equality constraints. The minimization problem therefore
becomes

$$
\begin{align*}
\min _{P_{\text {free }}} & \bar{\sigma}(Q(P)-\operatorname{diag}(Q(P))) \\
\text { s.t. } & Q=\left(A-\frac{1}{2} B K\right)^{T} P+P\left(A-\frac{1}{2} B K\right)  \tag{6.9}\\
& P=V_{0}+V P_{\text {free }} \\
& \min (\lambda(Q)) \geq 0 \\
& \min (\lambda(P)) \geq 0
\end{align*}
$$

where $V_{0}$ and $V$ derive from the aforementioned reduced row echelon form of (6.8). A Matlab script of the described procedure may be found in Appendix E.

### 6.2 Eigenstructure assignment

For any given LTI MIMO-system, with cross-coupling between states, the location of the poles do not uniquely define the systems transient behaviour. Rather the behaviour depends on both poles (eigenvalues) and eigenvectors of the closed loop system matrix $A_{c}$. If $A_{c}$ is diagonalizable, i.e. has full rank of the eigenspace, then the solution to the initial value problem becomes

$$
\begin{equation*}
x=e^{A_{c} t} x_{0}=X e^{\lambda t} X^{-1} x_{0} \tag{6.10}
\end{equation*}
$$

where the column vectors of $X$ are the eigenvectors of $A_{c}$ and $\lambda$ is a diagonal matrix with diagonal entries equal to the eigenvalues of $A_{c}$. From (6.10) it becomes apparent that the poles of the closed loop system only uniquely defines the system transient response when $X-\operatorname{diag}(X)=0$, i.e. all states are decoupled. Therefore, a simple pole placement technique may be insufficient. It is known, [20], that when employing constant output feedback, a maximum of $\max (m, n)$ eigenvectors may be assigned with $\min (m, n)$ free entries in each vector, where $n$ and $m$ are the dimensions of states and control signals respectively. However, some of this inherent freedom must be used to ensure that there in fact is a viable solution to the inverse LQ-problem as described above. As presented in [18] a solution to the inverse problem exists if and only if

$$
\begin{array}{r}
T^{*}(j \omega) T(j \omega)-I \geq 0 \\
T(j \omega)=I+K(I j \omega-A)^{-1} B \tag{6.12}
\end{array}
$$

where the superscript $*$ denotes the conjugate transpose. The remaining freedom in the choice of $K$, if such exist, may be used to enforce optimality in some other sense, e.g. such that $A_{c}$ is maximally diagonal.

In order to place the poles of a MIMO-system one may employ a Lyapunov based technique accordingly

$$
\begin{array}{r}
A X-X \lambda=B G \\
K=G X^{-1} \tag{6.14}
\end{array}
$$

where $X$ and $\lambda$ are defined as in (6.10) and $G$ is an arbitrary auxiliary matrix. The question that naturally arises is then, how does one choose $G$ such that (6.11) is fulfilled? Given that $X$ is linear in $G$ the feedback matrix $K$ becomes nonlinear in the same due to the inversion of $X$ in (6.14) (given that $X$ is not diagonal, in which case pole placement may be employed). This further implies that $T(j \omega$ ) is not only frequency dependent (as implied by the functional relationship to $w$ ) but also nonlinear in $G$. This means that the criteria for LQ-optimality is determined by a nonlinear, frequency dependent, matrix inequality. Similar to the constraint $Q \geq 0$, (6.11) may be rewritten as $\min \left(\lambda\left(T^{*}(j \omega) T(j \omega)-I\right)\right) \geq 0$. Even though this may simplify the procedure to some degree the problem still remains ambiguous in the sense that there are infinitely many feedback matrices $K$ which satisfy (6.11), or otherwise put, there are an infinite number of feasible structures of the function $f(j \omega)=\min \left(\lambda\left(T^{*}(j \omega) T(j \omega)-I\right)\right)$.

Given a randomly chosen system, that may or may not be stable, along with a predetermined and already proven LQ-optimal feedback matrix $K$, the term $\min \left(\lambda\left(T^{*}(j \omega) T(j \omega)-I\right)\right)$ can be seen to take the shape of a rational polynomial with a denominator of an even degree, see Figure 6.2. Note that this is purely an empirical observation.


Figure 6.2: The term $\min \left(\lambda\left(T^{*}(j \omega) T(j \omega)-I\right)\right)$, plotted over the interval $\omega \in$ [ $-60,60$ ], for a randomly created LTI-system with a predetermined LQ-optimal feedback matrix $K$.

A first approach may then be to choose the elements of $G$ such that $K$ solves the inverse eigenvalue problem (places the poles) and such that one minimizes some suitable measure of correlation between the term $\min \left(\lambda\left(T^{*}(j \omega) T(j \omega)-I\right)\right)$ and a beforehand chosen, non-negative and proper, rational polynomial. However by doing so one might risk unnecessary pole-zero cancellations which in practice could increase model dependency. A second and perhaps more reasonable approach is to allow an arbitrarily weighted LQ-algorithm to create a feasible structure of $\min \left(\lambda\left(T^{*}(j \omega) T(j \omega)-I\right)\right.$ which may be used in the same sense as described above. As such one can ensure that if pole-zero cancellations do exist it is no worse than that caused by other valid choices of weighting matrices $Q$ and $R$. Note that for simplicity $\omega$ may be discretized and made finite however at the expense of some accuracy, as will be further discussed in Section 6.3. The optimization problem of finding an LQ-optimal feedback matrix $K$, that solves a given inverse eigenvalue problem, so becomes

$$
\begin{equation*}
\min _{G} \sqrt{\frac{1}{k} \sum_{i=1}^{k}\left(Y_{r}\left(j \omega_{i}\right)-Y\left(j \omega_{i}, G\right)\right)^{2}} \tag{6.15a}
\end{equation*}
$$

$$
\begin{array}{r}
Y\left(j \omega_{i}, G\right)=\min \left(\lambda\left(T^{*}\left(j \omega_{i}, G\right) T\left(j \omega_{i}, G\right)-I\right)\right) \\
T\left(j \omega_{i}, G\right)=I+K(G)\left(I j \omega_{i}-A\right)^{-1} B \\
K=G X^{-1}(G)  \tag{6.15b}\\
A X-X \lambda=B G
\end{array}
$$

where the subscript $r$ denotes reference and indicates association with the predetermined LQ-optimal feedback matrix (here denoted $K_{r}$ ). Note that the correlation measure is here taken to be the the root-mean-square over a finite interval in $\omega\left(\omega \in\left\{\omega_{1}, \omega_{2} \ldots \omega_{k}\right\}\right)$. Further note that assigning the poles, as previously mentioned, does not uniquely define a solution to the eigenstructure assignment problem, or otherwise stated, problem (6.15) might not be bijective. It is however analytically hard to predict which elements in $G$ are necessary to achieve a small enough value of the objective function (6.15a) (such that (6.11) is upheld).

### 6.2.1 Solution to the eigenstructure assignment

The solution to the eigenstructure assignment problem is found by solving (6.15) by means of nonlinear optimization with an interior-point algorithm (in similar fashion to that of problem (6.7)). Note that convexity of problem (6.15) is not guaranteed, the solution may therefore alter according to the choice of initial values. Given that the optimization variable $G$ is merely a transformed version of $K$ in the eigenspace of $A_{c}(G=K X)$ it is reasonable to assume that low
controller effort may be gained from choosing low initial values of $G$. The initial values of all elements in $G$ are therefore arbitrarily chosen to be 0.01 (as $K$ is undefined at $G=0$ ). A Matlab script of the procedure can be found in Appendix F.

### 6.3 A small example

An LTI-system is created with random values of all elements in matrices $A$ and $B$ (the only constraint is that the pair $(A, B)$ be controllable and that the matrix $A$ does not share any eigenvalues with those desired by the eigenstructure assignment). All states are assumed to be measured perfectly (without disturbances) and the system therefore takes the form

$$
\begin{align*}
\dot{x} & =A x+B u \\
y & =x \tag{6.16}
\end{align*}
$$

The dimensions of the state- and input vector are $6 \times 1$ and $4 \times 1$ respectively. A set of LQ-optimal poles for the system are created by solving an LQ-problem with a randomly chosen (positive semi-definite) state weighting matrix $Q$ along with the input weighting matrix $R=I_{4 \times 4}$. The eigenstructure assignment, as described in this section, is applied. In Figures 6.3-6.4s one can see the results of the procedure, i.e. the linearly interpolated graph to the function $Y\left(j \omega_{i}, G\right)$ from ( 6.15 b ) and the resulting pole locations in comparison to those desired.


Figure 6.3: The function $Y(j \omega)$ plotted over the interval $\omega \in[-60,60]$.


Figure 6.4: Location of poles for the LQ-optimal eigenstructure assignment (blue) in comparison to the desired pole placement (red). The small axes provide a close-up of each unique pole pair, note the scales. One can see that the complex conjugate poles deviate the most from their desired locations.

The resulting weighting matrix $Q_{e a}$, where the subscript ea denotes eigenstructure assignment, from the solution to the inverse LQ-problem becomes

$$
Q_{e a}=\left[\begin{array}{cccccc}
3.84 & -5.74 & -4.61 & -10.38 & -1.36 & 5.56  \tag{6.17}\\
-5.74 & 11.20 & 8.62 & 17.62 & 2.39 & -7.68 \\
-4.61 & 8.62 & 7.21 & 14.03 & 1.95 & -4.85 \\
-10.38 & 17.62 & 14.03 & 31.10 & 4.24 & -15.42 \\
-1.36 & 2.39 & 1.95 & 4.24 & 0.66 & -1.90 \\
5.56 & -7.68 & -4.85 & -15.42 & -1.90 & 14.20
\end{array}\right]
$$

In comparison the randomly created weighting matrix $Q$ is

$$
Q=\left[\begin{array}{llllll}
1.80 & 1.34 & 1.45 & 0.95 & 1.12 & 1.28  \tag{6.18}\\
1.34 & 3.31 & 1.28 & 0.97 & 1.76 & 1.55 \\
1.45 & 1.28 & 2.28 & 1.14 & 1.17 & 1.08 \\
0.95 & 0.97 & 1.14 & 1.88 & 0.88 & 0.88 \\
1.12 & 1.76 & 1.17 & 0.88 & 2.22 & 1.09 \\
1.28 & 1.56 & 1.08 & 0.88 & 1.09 & 2.65
\end{array}\right]
$$

Note that all values have been rounded off to two decimal points. Further note that, in this instance, the resulting matrix $Q$ is not maximally diagonal. The possibility of gaining a maximally diagonal $Q$ seems to depend largely on the quantitative success of the eigenstructure optimization, i.e. how well the solution has managed to uphold (6.11). In this particular case the resulting function $Y(j \omega)$, at large absolute values of $\omega$, becomes slightly negative (in the
order of $10^{-6}$ ), as shown in Figure 6.5. This seemingly small violation of the constraints appears to cause conflicts between the symmetry constraints of the matrix $P$ and the optimality constraint

$$
\begin{equation*}
K=R^{-1} B^{T} P \tag{6.19}
\end{equation*}
$$

As such symmetry of the matrix $P$ has to be enforced through optimization and the new objective function has hence been set as

$$
\begin{equation*}
f(P)=\sum_{i=1}^{n}\left(\sum_{j=1}^{n}\left(\left|P_{j i}-P_{i j}\right|\right)\right) \tag{6.20}
\end{equation*}
$$

where $n$ is the number of states, in this instance six.


Figure 6.5: The function $Y(j \omega)$ plotted over the interval $\omega \in[-5000,5000]$. The provided annotation shows that when the absolute value of $\omega$ is sufficiently large, the function $Y(j \omega)$ becomes negative.

Unfortunately this approach does not explicitly guarantee the necessary symmetry of $P$ and as such a final measure has to be taken in the form

$$
\begin{equation*}
P_{f}=\frac{1}{2}\left(P^{T}+P\right) \tag{6.21}
\end{equation*}
$$

where the subscript $f$ denotes final, and $P_{f}$ is then used in order to find $Q$ (or equivalently $Q_{e a}$ ). Because of this constraint violation the poles shift slightly from their desired positions, as shown in Figure 6.4. From simple trial and error it becomes clear that both too few and too many samples taken in $\omega$, as a rule, results in greater violation of (6.11). Too few samples renders the optimization excessively relaxed and optimal solutions may be found in structures of $Y(j \omega)$ far from what is sought after. Too many samples however increases complexity and seem to make the problem more non-convex. As such the optimization may end in local optima that greatly violates (6.11). Noteworthy is that the allowable violation seem to increase both with absolute values of $\omega$ and with decreased sizes of state matrices, allowable in this instance referring to results with seemingly reasonable deviations in pole locations. It should here be stated
that when the set of discretized values of $\omega$ increases the computational cost rises as well and eventually the optimization becomes so slow that the added time consumption may invalidate any potential gain in accuracy.

Note: It should further be noted that (for larger systems) even when comparably few samples in $\omega$ are taken the optimization still takes a fair amount of time and the practicality of this approach could, with reason, be questioned.

The resulting feedback matrix $K_{e a}$, where the subscript ea denotes eigenstructure assignment becomes

$$
K_{e a}=\left[\begin{array}{cccccc}
2.89 & -6.87 & 6.14 & 24.92 & 12.39 & -0.02  \tag{6.22}\\
-31.81 & 31.62 & 29.17 & 36.29 & -5.51 & -14.82 \\
3.87 & 14.83 & -2.31 & 16.51 & 2.81 & -8.66 \\
20.43 & -12.69 & -2.90 & -0.54 & 9.61 & 17.27
\end{array}\right]
$$

In comparison the LQ-optimal feedback matrix $K$ used to create the desired set of poles is

$$
K=\left[\begin{array}{cccccc}
3.24 & -7.30 & 5.78 & 24.94 & 12.53 & 0.40  \tag{6.23}\\
-31.88 & 31.64 & 29.29 & 36.39 & -5.58 & -14.88 \\
4.77 & 14.12 & -3.01 & 15.95 & 2.88 & -7.79 \\
20.87 & -12.61 & -3.14 & -0.51 & 9.65 & 17.07
\end{array}\right]
$$

Note that, as before, all values have been rounded off to two decimal points.

## Chapter 7

## Actuator redundancy

For both reasons of safety (e.g. fault tolerance) and performance it is sometimes practice to include more control signals than are inherently needed. As discussed in Section 5.2 this is the case for the current aircraft model. A necessary question then becomes, how does one specify which control signal is to be used at any given moment? Or otherwise stated, how does one prioritize between all available controls? Two ways of solving these questions are discussed in detail in [8], the first method being optimal control and the later what is referred to as control allocation.

### 7.1 Optimal control

Given a performance index, or equally objective function, a suitable algorithm is used to find a solution to an associated optimization problem (either off- or online). Notable examples include H-infinity control, LQ-control and model predictive control. Given that the optimization problem is well-defined the solution defines the optimal priority between controls and no further design choices are necessary.

Among the drawbacks of optimal control is the necessity to define an appropriate objective function, as in the special case of LQ-control in which weighting matrices must be defined. Furthermore, inclusion of constraints may make the problem computationally expensive, as with explicit model predictive control. In reality constraints are bound to be present, for instance in the shape of limits on the controller output. For the aircraft at hand these constraints may also come in the form of limits on load factors as to avoid undue stress on both components and pilot. These constraints may be resolved afterwards in a nonoptimal fashion, e.g. by use of rate limiters or alike, however at the expense of designed performance.

### 7.2 Control allocation

Given that the pair $(A, B)$ is controllable and that the input matrix $B$ is rank deficient there exists a factorization of $B$ such that

$$
\begin{equation*}
B=B_{1} B_{2} \tag{7.1}
\end{equation*}
$$

where $B_{1}$ has full column rank and $B_{2}$ inherits the rank deficiency from $B$. The system may consequently be rewritten as

$$
\begin{array}{r}
\dot{x}=A x+B_{1} v \\
v=B_{2} u \tag{7.3}
\end{array}
$$

In [8] the authors refer to signal $v$ as the virtual control input and points out that it may be interpreted as the total control effort deriving from all actuators. In another sense $v$ may be viewed as the raw input to the system in question. For the flight mechanical model presented in Section 5.3 this signal $v$ would refer to external forces and moments acting on the aircraft, $B_{1}$ would then be a partial unity matrix stating how forces and moments enter the system. $B_{2}$ would in turn be a matrix stating how all available control signals translate into forces and moments, i.e. a matrix describing the control surface effectiveness at the given operating point. System (7.2) may be stabilized, or controlled, by any preferable method, such as LQ-control. Equation (7.3) must then be solved in order to portion the necessary raw signal onto the real, or actual, input signals of the original system. Note that since $B_{2}$ is rank deficient there exists an infinite number of solutions, i.e. an infinite number of ways to create the necessary controller effort. A solution may be found both in an optimal and non-optimal sense. An example of a non-optimal approach is to simply state how control signals are to co-operate based on a rule of thumb or perhaps designers intuition. One could combine signals, for example saying that some signals are to be symmetric, anti-symmetric or operate in a proportional fashion. If these combinations are linear in nature the columns of $B_{2}$ inevitably decreases until, if the procedure is repeated, there is no inherent freedom left and the resulting remains of $B_{2}$ no longer exhibits rank deficiency.

In [8] the authors prove that, if the system in question is linear and performance indexes are constant, using LQ-control along with optimal control allocation (with a quadratic performance index) yields the same result as using LQ-control on a corresponding non-truncated system (i.e. with the original input matrix $B)$. The authors further conclude that the handling of constraints may, to some degree, be made simpler with the use of control allocation. They hence propose the optimization problem

$$
\begin{array}{r}
w=\arg \min _{u \in U} u^{T} W u \\
U=\arg \min _{\underline{u} \leq u \leq \bar{u}}\left(B_{2} u-v\right)^{T} W_{v}\left(B_{2} u-v\right) \tag{7.5}
\end{array}
$$

In terms of words, the process may be described as: Among those allocations that minimize the difference between the real and desired control effort, choose the one that minimizes the performance index in (7.4). The authors stress that this approach is not equivalent to including constraints in a non-allocated optimal control problem. Note that, as shown in [8], the set of solutions to this optimization problem may take a fairly complex shape even for comparably low order systems.

### 7.2.1 Control allocation and servo dynamics

When extending the model with the linear servo dynamics the new input matrix $\tilde{B}$ will take the form

$$
\begin{equation*}
\tilde{B}=\left[\tilde{B}_{1}, \tilde{B}_{2}\right] \tag{7.6}
\end{equation*}
$$

as described in Section 5.4, where $\tilde{B}_{2}$ refers to control signals not belonging to control surfaces (e.g. engine related) and

$$
\tilde{B}_{1}=\left[\begin{array}{ccc}
\tilde{B}_{11} & \ldots & \tilde{B}_{1 m}  \tag{7.7}\\
\vdots & \ddots & \vdots \\
\tilde{B}_{n 1} & \ldots & \tilde{B}_{n m} \\
B_{s} & \ldots & 0 \\
\vdots & \ddots & \vdots \\
0 & \ldots & B_{s}
\end{array}\right]
$$

where $B_{s}$ is the input matrix for the linear servo system, $n$ is the dimension of states (of the former non-extended system) and $m$ is the number of control surfaces. Note that the elements $\tilde{B}_{x x}$ will always equal zero apart from the special case when the order of the linear servo model is lower than the highest order derivative of the control surface deflections. From the structure in (7.7) it is apparent that $\tilde{B}_{1}$ will have full column rank and so the possibility of performing control allocation vanishes. Otherwise stated, the rank deficiency of the former input matrix $B$ is hidden in the new state matrix $\tilde{A}$. The question then becomes, how does one perform control allocation while taking the servo dynamics into account?

If the effects of the control surface rates on the system are sufficiently small, in comparison to control surface deflections, a simple (and perhaps crude) method may be to simply remove control surface rates from the given input signals (with no adjustments made). The system could then be fitted with fictional delays on the control surface deflections (in order to simulate the inherent delay of actuator movement). Note however that this is not a true augmentation of the system and so no new states relating to the delay appears in the system description, one is merely keeping track of the delayed response when trimming for the non-delayed system.

Note: If the delay terms are linearized, by use of for example a Padé approximation, and the system augmented with subsequent states, one inevitably ends up with the same dilemma as when extending the system with linear servo dynamics, i.e. the resulting input matrix has full rank and control allocation is no longer possible.

If such an approach is applied, the input matrix $B$ remains rank deficient effectively allowing the designer to use control allocation techniques. Note that the resulting phase margin of the closed loop system is inevitably decreased and so the requirement for accurate modeling of other internal parts increase as such. In order to lessen the effects of this approach, the resulting control signals may be sped $u p$ using, for example, either a Smith predictor or a suitable lead filter. The later could potentially be in the form of a low-pass filtered inverse of the servo dynamics. Note however that this method is only appropriate if the servo model does not exhibit any notch behaviour or alike, no substantial overshoot (i.e. is well damped) and there exist no strong non-linearities in the same. Simply put, the servo system must be predictable and well-behaved. In Figure 7.1 one may see the working principles of a Smith predictor.


Figure 7.1: Working principles of a Smith predictor. The Smith predictor compares the modelled system response, with and without delay, and makes the appropriate compensation

An other possible solution is to truncate the system before the inclusion of servo dynamics, perform control law design with forces and moments as virtual inputs, and then implement a dynamic control allocation scheme. In other words, perform one additional control law design that is equality constrained, e.g. with LQ-control. However, when constraining the LQ-problem an analytical solution is in general hard to find. As such, most approaches aim at solving a discretized finite time horizon estimation of the initial problem, simpler put, the problem is converted to that of MPC (to which all associated challenges follow suit, such as robust online optimization). The benefit of simplified handling of constraints may therefore be lost and so invalidate one of the main premises of the control allocation approach. Furthermore the optimal virtual control signal is likely time-varying, as in the case of LQ-control, the problem is therefore only truly linear in frozen time or likewise if the equality constraint is discretized and made finite (i.e. the virtual control is sampled at a finite number of instances in time).

Further solutions include designing an outer feedback loop for the servo that tracks the optimal virtual control. However, several difficulties arise, including that (for reason of stability) the outer loop should be slower than the inner, effectively meaning that the servo dynamics has to be slowed down. Furthermore, the optimal virtual control may be time-varying in such a way that either zero steady state error may not be guaranteed or introduce the necessity of higher order integral control, which may violate the criterion of a slower outer loop. In excess to that one must either be able to directly measure the control surface rates, turn to pseudo derivatives or augment the existing servo model with an observer, e.g. linear time-derivative tracker, which inevitably would introduce further delay.

## Chapter 8

## Results

A program has been written in Matlab that incorporates many, but not all, of the above mentioned procedures and techniques. In this chapter the program and its functionality will be covered. Moreover, a summary will be given of which procedures/techniques were implemented along with a brief explanation as to why some where excluded. Lastly, the results from both linear and nonlinear simulations will be put forth.

### 8.1 Program description

Within the Ares simulator lies a module that calculates central derivatives of aerodynamic coefficients. Furthermore, the module outputs trim conditions and trim accuracy along with mass and inertial data. These are saved in a simple text file, which is initially read and sorted by the program. The program then creates a family of linear systems in accordance with Chapter 5. When calling the program the user has the possibility of choosing whether or not to use virtual signals, i.e. whether or not to apply control allocation as described in Chapter 7. If the user chooses virtual controls a new input matrix is created of the form

$$
B_{1}=\left[\begin{array}{cccc}
1 & 0 & \ldots & 0  \tag{8.1}\\
0 & 1 & \ldots & 0 \\
\vdots & & \ddots & \vdots \\
0 & 0 & \ldots & 1 \\
0 & 0 & \ldots & 0 \\
\vdots & & \ddots & \vdots \\
0 & 0 & \ldots & 0
\end{array}\right]
$$

where the upper portion is a $6 \times 6$ unity matrix. The matrix relating the virtual and original control signals is hence created as

$$
\begin{equation*}
B_{2}=B_{1}^{+} B \tag{8.2}
\end{equation*}
$$

where + denotes pseudo inverse. In this particular case the pseudo inverse is chosen as the Moore-Penrose pseudo inverse. As outlined in Chapter 7 the introduction of virtual control signals hinders the adding of proper servo dynamics and as such the original input matrix $B$ is simply truncated to remove control surface rates. Fortunately the aerodynamic model of the Admire model is simple enough such that the influence of the rates are mostly zero (with the exception of canards). Furthermore, if virtual controls are chosen a graphical user interface (GUI) is subsequently started that allows the user to couple the original control signals if he/she so wishes. The purpose of the GUI is simply to make the interaction with the program as user friendly as possible. In Figure 8.1 one may see how this GUI is designed


Figure 8.1: Design of the graphical user interface. The user may simply press on whichever surface he/she wishes to apply constraints to.

Note that for every new constraint that is applied, the effective redundancy decreases and eventually the system is no longer overactuated. If such is the case, when the GUI is closed (by pressing solve), the program automatically changes the approach to optimal control, the original input matrix $B$ is restored and the servo dynamics are subsequently added.

The original intent of the program was to further allow the user to choose between either LQ-optimal eigenstructure assignment, as described in Chapter 6, or a straightforward LQI approach, as described in Chapter 2. However, as outlined in Section 6.3, the time consumption of the LQ-optimal eigenstructure assignment rapidly increases with the size of the state matrices $A$ and $B$. Furthermore, the loci of LQ-optimal poles is, unfortunately, much narrower than anticipated. This means that in order to successfully make use of the proposed eigenstructure assignment one has to have extensive insight into the LQ-optimal root loci of the system in question. Even though the movement of the poles are known (to some extent), see for example [15], the problem is evident. What is interesting to note is however that when a set of LQ-optimal poles have been chosen, they seem to remain sufficiently optimal for comparably large deviations in system matrices $A$ and $B$ (implying that the proposed eigenstructure assign-
ment can be solved with moderate deviations in pole locations). In Figure 8.2 one can see the results from a similar trial to that of the example in Section 6.3. In this trial however, one randomly chosen element in each of the matrices $A$ and $B$ have successively been decreased by five percent for a total of five times, resulting in a maximal deviation of roughly $22.62 \%$.


Figure 8.2: Location of poles for the LQ-optimal eigenstructure assignment (blue) in comparison to the desired pole placement (red). The small axes provide a close-up of the poles that deviate the most.

The method is nonetheless deemed to impractical to be incorporated in the program. Moreover, when extending the system with the linear servo dynamics the order of the system increases to such an extent that, not only is the process heavily time consuming, but the eigenstructure assignment risks failing due to the added complexity. It should here be mentioned that the Matlab function place solves for the feedback matrix $K$ by optimizing for numerical stability, i.e. minimizes the condition number for the eigenvector matrix $X$ of the closed loop state matrix $A-B K$. In practice, this means that the place function may yield systems where the strength of the coupling between states are far from acceptable. As such no pole placement or eigenstructure assignment techniques are currently incorporated in the presented program.

Therefore, regardless of the choice concerning virtual control signals, an LQoptimal feedback matrix $K$ is generated for each linear system created. The user must promptly input which states and outputs he/she wishes to stabilize and/or control through set point tracking (LQI-control). Furthermore, the user has the possibility of defining weighting matrices $Q$ and $R$. If no such choice is made, the program automatically creates these matrices by applying Bryson's rule. This however means that the user consequently must define maximum limits for all control signals and/or states (depending on if both matrices are left empty or not).

The user further has the possibility to apply alpha-shifted LQ-control, as described in Chapter 2. Before the process of creating the feedback matrices $K$, the user is therefore prompted if he/she wishes to trim the initial alpha-shift and integral values (initial referring to the first trim point of those input to the program). If the user chooses to do so, a series of additional windows are opened, one of which contains two continuous knobs. In this window the user
can simply trim the integral weighting and alpha-shift values (by turning the knobs) while keeping track of linear transient responses and pole/zero locations in the z-plane. Note here that all systems are discretized, using a zero-order hold method, with a sample time of $1 / 60$ seconds. This is to take into consideration the update frequency in the digital Ares Mars simulator.

While it would be beneficial for the alpha-shift and integral values to alter according to the position in the operating domain, as with weighting matrices $Q$ and $R$, performing automated tuning of such values implies yet further optimization procedures effectively slowing down the process. Moreover, as can be seen in Figures 8.3a-8.3b, the transient response remains fairly consistent over large portions of the subsonic envelop.

(a) Transient response of roll rate $p\left(\omega_{x}\right)$.

(b) Transient response of pitch rate $q\left(\omega_{y}\right)$.

Figure 8.3: Linear transient response for roll- and pitch rate in the envelope 0.4-0.9 Mach and $1000-4000 \mathrm{~m}$ (true altitude). Note that each figure contains 641 graphs which, mostly, overlap. The red line represents commanded values.

For the responses in Figures 8.3a-8.3b, the weighting matrices $Q$ and $R$ were chosen as

$$
\begin{align*}
Q=\operatorname{diag}( & {\left[3.77 \times 10^{-6}, 4.00,1.11 \times 10^{-3}, 1.01 \times 10^{-1}, \ldots\right.}  \tag{8.3}\\
& \left.\left.\ldots 4.05 \times 10^{-1}, 5.25,150,15 \times 10^{3}, 150\right]\right) \\
R=\operatorname{diag}([ & 4.00 \times 10^{-2}, 4.00 \times 10^{-2}, 8.16 \times 10^{-4}, 8.16 \times 10^{-4}, \ldots  \tag{8.4}\\
& \left.\left.\ldots 8.16 \times 10^{-4}, 8.16 \times 10^{-4}, 1.00 \times 10^{-4}\right]\right)
\end{align*}
$$

Furthermore, the state matrix $A$ was altered (alpha-shifted) as

$$
\begin{align*}
& A=A+\alpha  \tag{8.5}\\
& \alpha=\operatorname{diag}([0,0,0,3.02,2.03,0]) \tag{8.6}
\end{align*}
$$

Note that the Euler angles have been removed from the linear state space systems in these simulations. The grid applied takes the shape of that shown in Figure 8.4


Figure 8.4: Portioning of the envelope. Note that each vertex and each center of all squares represent one operating point (equally equilibrium- or trim point)

As the response remains fairly consistent over large portions, it may in practice only be necessary to perform a limited number of trimming sessions (in critical portions of the envelope). Such portions may be instances where the solution to the dynamic equations undergo sudden changes (bifurcation), such as supersonic transitioning, or where control surface authority is low or in risk of reversal.

Once feedback matrices for all trim points have been calculated all values are linearly interpolated to yield a piecewise linear surface as shown in Figure 8.5


Figure 8.5: Element $(1,5)$ from the feedback matrix $K$ plotted over the flight envelop. All red dots represent interpolated values at intermediate points in the grid, they are calculated and shown simply to verify that the interpolation has been successful. As anticipated, the strength of the feedback lessens when control authority increases (i.e. when speed is high and altitude is low). Note that this picture is merely illustrative and does not relate to the responses shown in Figures 8.3b and 8.3a

During simulation the flight control systems module (FCS-module) calculates the aircraft's current position in the grid in terms of center-point and angle about such center-point. Center-point refers to all trim points located at the center of each square sub-region shown in Figure 8.4. These points are all given a linear index starting from one, at the lowest value of the first scheduling parameter (in this instance Mach number), and from there successively increases to the highest value of the second scheduling parameter (in this instance true altitude). Furthermore, all triangular sub-regions surrounding these center-points are likewise given a linear index starting from one at the lowest such region and successively increasing in counter clockwise direction (CCW), These indices are are hence used when reading and updating the values of the interpolated feedback coefficients from the matrix $K$, the principle is shown in Figure 8.6


Figure 8.6: Partitioning of the envelope. Each center-point is given a linear index ranging from one to $n$. Furthermore each triangular sub-region about these center-points are given a linear index ranging from one to four (CCW).

### 8.2 Linear simulation results

The program is called with states one through six as states to be stabilized, i.e. all linear and angular velocities. Since the Euler angles are merely the integrals of the angular rates in the linearized systems they are, for simplicity, removed. The states to be controlled through set point tracking are $V_{x}, \omega_{x}$, $\omega_{y}$ and $\beta$ (speed in x-direction, roll rate, pitch rate and side slip angle). The matrix $R$ is created by use of Bryson's rule. The matrix $Q$ is chosen manually and all instances relating to tracking errors are tuned using the built-in GUI. The resulting values are

$$
\begin{align*}
& Q=\operatorname{diag}([0.1,50,0.1,30,600,50,220,500,6000,1])  \tag{8.7}\\
& R=\operatorname{diag}\left(\left[0.82 \times 10^{-3}, 0.82 \times 10^{-3}, \ldots, 0.1 \times 10^{-3}\right)\right. \tag{8.8}
\end{align*}
$$

Additionally, the state matrix $A$ is altered accordingly (alpha-shifted)

$$
\begin{align*}
A & =A+\alpha  \tag{8.9}\\
\alpha & =\operatorname{diag}([0,0,0,3,0,2]) \tag{8.10}
\end{align*}
$$

The scheduling parameters are, again, Mach number and true altitude. The lower and upper limits of the operating domain are chosen as 0.3 and 0.9 Mach and 1000 and 7000 meters true altitude respectively. The operating grid takes the form of that shown in Figure 8.4 where each square sub-region has dimensions 0.1 Mach and 1000 meters true altitude, which gives 85 trim points in total. Furthermore, both right elevons, left elevons and canards are made to operate symmetrically. To clarify, the new inputs to the system are $\delta_{l e}, \delta_{r e}$,
$\delta_{c e}, \delta_{r}$ and $\delta P L A$, where $l e$, re and ce denotes left elevon, right elevon and canard elevon respectively. Note that, as a result, canards cannot directly affect lateral motion. The linear transient responses of $V_{x}, \omega_{x}, \omega_{y}$ and $\beta$ are shown in Figures 8.7a-8.7d.


Figure 8.7: Linear transient responses for $V_{x}, \omega_{x}, \omega_{y}$ and $\beta$ in the envelope 0.3-0.9 Mach and 1000-7000m (true altitude). Each figure contains 85 graphs. Note that no step has been made in $\beta$.

### 8.3 Nonlinear simulation results

The resulting interpolated values of the feedback matrix $K$, from the procedure described in Section 8.2, are implemented in the Ares Mars simulator and the aircraft is initialized in trim at 0.8 Mach at 1100 meters. Due to the current trimming procedure there is a sudden change in controlled parameters at the beginning of the simulation. As such, there is an immediate transient response when the simulation starts. To clarify, the pilot first trims the aircraft using open loop control (in which stick deflection directly translate to control surface deflections). When the simulation begins the aircraft switches to rate mode (stick deflection translates to commanded angular rates) and the pilot must consequently shift the stick to neutral position. Due to these transients the aircraft initially ends up with negative pitch and as such starts to loose altitude. The pilot must therefore, shortly after the shift to neutral, make a slight negative step in elevator stick deflection to level out the aircraft. In Figure 8.8 one can see the response in pitch rate during these initial maneuvers.


Figure 8.8: Response in pitch rate during the initial maneuvering (with the purpose of leveling the aircraft).

At ten seconds into the simulation the pilot makes a doublet step command in pitch rate (note that the rate of the stick deflection is slightly limited to simulate human movement). The results in both pitch rate, pitch angle and altitude are shown in Figures 8.9a-8.9c.


Figure 8.9: Nonlinear transient response for $\omega_{y}, \theta$ and $H_{\text {true }}$. The red curve in subfigure (a) shows the commanded pitch rate.

In Figures 8.10a and 8.10b one can see the corresponding control surface deflections and pilot lever angle.


Figure 8.10: Control surface deflections and pilot lever angle during the doublet in pitch rate

## Chapter 9

## Conclusions

A program for creating control law parameters for generic fighter aircraft has been created. The program is based on gain scheduled LQ-control with possible setpoint tracking and a prescribed degree of stability (alpha-shift). The possibility of automated tuning has been investigated by solving the inverse LQ-problem coupled to an eigenstructure assignment. The resulting method is however, currently, of limited practical use due to the seemingly narrow loci of LQ-optimal poles and the extensive time consumption associated with the necessary non-convex optimization. Furthermore, the existence of hidden coupling during conventional gain scheduling procedures has been brought to light and a possible remedy in the form of enhanced velocity based linearization (EVBL) has been considered. Note that due to the limited time budget of this project the EVBL approach has not been implemented in the Ares simulator and as such the possible presence of hidden coupling, for the controller structure employed in Section 8, has not been evaluated. It should however be stressed that, given the results presented, hidden coupling may indeed pose serious issues in the implementation of gain scheduled controller schemes and that velocity based linearization, in this context, is a viable alternative to conventional series expansion. It should further be concluded that hidden coupling may be present even when employing advanced control techniques such as LQ-control and that advantages of EVBL, in the implementation flight control systems, should be further considered. Moreover, the benefits and disadvantages of LQ-optimal control allocation has been studied. Regarding such it is to be concluded that one of the main advantages is the ability to explicitly handle constraints in a less strict manner than that of constrained LQ-control. However, similar to constrained LQ-control (and explicit MPC), the problem still requires solving a comparably complex optimization problem that may result in a heavily faceted polytopic set of solutions. To avoid this one would instead have to turn to online numerical optimization in which case MPC may be a viable option. Furthermore, the introduction of virtual controls causes possible loss in model accuracy as it effectively hinders the inclusion of linearized servo dynamics. As a possible alternative to the proposed LQ-optimal eigenstructure assignment one may look to, for example, partial pole placement techniques, see e.g. [21].

## Bibliography

[1] D. J. Leith W. E. Leithead. "Survey of gain-scheduling analysis and design", Int. Journal of Control, vol. 73, no. 11, pp. 1001-1025, 2000.
[2] D. J. Leith W. E. Leithead, "Gain-scheduled and nonlinear systems: Dynamic analysis by velocity-based linearization families", Int. Journal of Control, vol. 70, no. 2, pp. 289-317, 1998.
[3] H. Lhachemi, D. Saussié, G. Zhu, "Hidden Coupling Terms Inclusion in Gain-Scheduling Control Design: Extension of an Eigenstructure Assignment-Based Technique", IFAC-PapersOnLine, vol. 49, no. 17, pp. 403-408, 2016.
[4] H. Lhachemi, D. Saussié G. Zhu, "An enhanced velocity-based algorithm for safe implementations of gain-scheduled controllers", Int. Journal of Control, vol. 90, no. 9, pp. 1973-1989, 2017.
[5] I. Kaminer, A. M. Pascoal, P. P. Khargonekar, E. E. Coleman, "A velocity algorithm for the implementation of gain-scheduled controllers", Automatica, vol. 31, no. 8, pp. 1185-1191, 1995.
[6] L. Reberga, D. Henrion, J. Bernussou, F. Vary. "LPV Modeling of a Turbofan Engine", Proc. of the 16th IFAC World Congress, Prague, Czech Republic, IFAC Proceedings Volumes, vol. 38, no. 1, pp. 526-531, 2005.
[7] R. T. Reichert, "Dynamic scheduling of modern-robust-control autopilot designs for missiles", IEEE Control Systems Magazine, vol. 12, no. 5, pp. 35-42, 1992.
[8] O. Härkegård, S. T. Glad. "Resolving actuator redundancy: optimal control vs. control allocation", Automatica, vol. 41, no. 1, pp. 137-144, 2005.
[9] J. L. Casti. "A Note on the General Inverse Problem of Optimal Control Theory", IIASA Research Memorandum. IIASA, Laxenburg, Austria: RM-74-008, 1974.
[10] N. Wiener, Extrapolation, Interpolation, and Smoothing of Stationary Time Series: With Engineering Applications, Cambridge, MA: MIT Press; 1942.
[11] A. C. Hall, The Analysis and Synthesis of Linear Servomechanisms, Cambridge, MA: MIT Press; 1943.
[12] R. E. Kalman,. "Contributions to the Theory of Optimal Control", Bol. Soc. Mat. Mexicana, vol. 5, pp. 102-119, 1960.
[13] Y. S. Ledyaev, "On Analytical Solutions of Matrix Riccati Equations", Proc. Steklov Inst. Math, vol. 273, pp. 214-228, 2011.
[14] F. Jabbari, "Extensions to LQR" [Internet], University of California Irvin [updated 2014 March; cited 2020 Jul 15]. Available from: http://mae2.eng.uci.edu/~fjabbari//me270b/chap5.pdf
[15] M. S. Triantafyllou, F. S. Hover, "Maneuvering and Control of Marine Vehicles" [Internet], Massachusetts Institute of Technology Cambridge, Massachusetts USA [updated 2003 Nov 5; cited 2020 Jul 15]. Available from: https://ocw.mit.edu/courses/mechanical-engineering/2-154-maneuvering-and-control-of-surface-and-underwater-vehicles-13-49-fall-2004/lecture-notes/
[16] M. A. Sen, M. Kalyoncu, "Optimal Tuning of a LQR Controller for an Inverted Pendulum Using the Bees Algorithm", Journal of Automation and Control Engineering, vol. 4, no. 5, 2016.
[17] K. Hassani, W. S. Lee, "Optimal Tuning of Linear Quadratic Regulators Using Quantum Particle Swarm Optimization", Proc. of the Int. Conference of Control, Dynamic Systems, and Robotics, Ottawa, Ontario, Canada, no. 59, 2014.
[18] M. C. Priess, R. Conway, J. Choi, J. M. Popovich Jr., C. Radcliffe. "Solutions to the Inverse LQR Problem with Applications to Biological Systems Analysis", IEEE, vol. 23, no. 2, 2015.
[19] F. Aioun, A. Heniche and H. Bourles, "Maximally-diagonal solution to the inverse LQR problem", Proc. of 1994 33rd IEEE Conference on Decision and Control, Lake Buena Vista, FL, USA, IEEE, vol. 2, pp. 1445-1446, 1994.
[20] K. Sobel, S. Joshi, "Eigenstructure Assignment for Fault Tolerant Flight Control Design" [Internet], Nasa Office of Equal Opportunity; 2002. NASA grant number NAG 2028 [cited 2020 Jul 25]. Available from: https:// ntrs.nasa.gov/search.jsp?R=20020023592
[21] K. C. Cheok, "Simultaneous linear quadratic pole placement (LQPP) control design", IFAC 15th Triennial World Congress, Barcelona, Spain, IFAC Proceedings Volumes vol. 35, no. 1, pp. 289-293, 2002.
[22] J. Löfberg, "YALMIP : a toolbox for modeling and optimization in MATLAB", 2004 IEEE International Conference on Robotics and Automation, New Orleans, LA, pp. 284-289, 2004.
[23] R. Nelson, Flight stability and automatic control, second edition, New York:McGraw-Hill;1998.
[24] L. Forsell, U. Nilsson, "ADMIRE The Aero-Data Model In a Research Environment Version 4.0, Model Description" [Internet], FOI - Swedish Defence Research Agency, Systems Technology; 2005. [cited 2020 Jul 15]. Available from: https://www.foi.se/rest-api/report/FOI-R--1624--SE
[25] O. Nevanlinna, "Sylvester equations and polynomial separation of spectra", OaM, vol. 13, no. 3, pp. 867-885, 2019.

## Appendices

## Appendix A

## Non-linear missile dynamics

Values of all coefficients that are not defined may be found i [7].

## Atmospheric properties

$Q$ : dynamic pressure $\left[l b / f t^{2}\right]$
$p_{0}$ : free stream static pressure $\left[l b / f t^{2}\right]$
M: Mach number [-]

$$
\begin{equation*}
Q=0.7 p_{0} M^{2} \tag{A.1}
\end{equation*}
$$

## Aerodynamic coefficients

$C_{n}$ : normal force coefficient [-]
$C_{m}:$ pitching moment coefficient [-]

$$
\begin{gather*}
C_{n}=\operatorname{sign}(\alpha)\left(a_{n}|\alpha|^{3}+b_{n}|\alpha|^{2}+c_{n}|\alpha|\right)+d_{n} \delta  \tag{A.2a}\\
c_{n}=-0.17\left(2-\frac{M}{3}\right)  \tag{A.2b}\\
C_{m}=\operatorname{sign}(\alpha)\left(a_{m}|\alpha|^{3}+b_{m}|\alpha|^{2}+c_{m}|\alpha|\right)+d_{m} \delta  \tag{A.2c}\\
c_{m}=0.051\left(\frac{8 M}{3}-7\right) \tag{A.2d}
\end{gather*}
$$

Forces and moments
$F_{l}$ : Normal/lift force $[l b]$
$F_{x}$ : Tangential force [lb]
$M_{l}$ : Pitching moment $[f t-l b]$

$$
\begin{gather*}
F_{l}=Q S C_{n}(\alpha, M, \delta)  \tag{A.3a}\\
M_{l}=Q S d C_{m}(\alpha, M, \delta)  \tag{A.3b}\\
F_{x}=Q S C_{a} \tag{A.3c}
\end{gather*}
$$

## Governing dynamics

States
$\alpha$ : angle of attack [rad]
$q$ : pitch rate $[\mathrm{rad} / \mathrm{s}]$
$V_{m}$ : missile velocity $[\mathrm{ft} / \mathrm{s}]$
$\delta$ : tail fin deflection [rad]

Inputs
$\overline{\delta_{c}}$ : commanded tail fin deflection $[\mathrm{rad}]$
Outputs
$\overline{\eta: \text { normal acc. }[g]}$

Miscellaneous
$\tau$ : tail fin time constant $[1 / s]$
m: mass [slugs]
$g$ : gravitational acc. $\left[f t / s^{2}\right]$

$$
\begin{gather*}
\dot{\alpha}=\frac{\cos (\alpha)}{m V_{m}} F_{l}+q  \tag{A.4a}\\
\dot{q}=\frac{M_{l}}{I}  \tag{A.4b}\\
\dot{V}_{m}=\frac{F_{x}}{m} \cos (\alpha)-\left|\frac{F_{l}}{m} \sin (\alpha)\right|  \tag{A.4c}\\
\dot{\delta}=-\frac{1}{\tau} \delta+\frac{1}{\tau} \delta_{c}  \tag{A.4d}\\
\eta=\frac{F_{l}}{m g} \tag{A.4e}
\end{gather*}
$$

## Appendix B

## Linearized missile dynamics

Functions $f_{1}$ to $f_{4}$ correspond to the right hand sides of (A.4a) to (A.4d) in that given order. Function $h$ similarly corresponds to the right hand side of (A.4e). Note that sine and cosine are abbreviated $s$ and $c$ respectively. The linearization of aerodynamic coefficients can be found in the Matlab script in Appendix C.

$$
\begin{array}{r}
\frac{\partial f_{1}}{\partial \alpha}=\frac{Q_{e} S}{m V_{m e}}\left(-s\left(\alpha_{e}\right) C_{n e}+\left.c\left(\alpha_{e}\right) \frac{\partial C_{n}}{\partial \alpha}\right|_{e c}\right)  \tag{B.1}\\
\frac{\partial f_{1}}{\partial q}=1 \\
\frac{\partial f_{1}}{\partial V_{m}}=c\left(\alpha_{e}\right) \frac{0.7 p_{0} S}{m \operatorname{sos}^{2}}\left(C_{n e}+\left.V_{m e} \frac{\partial C_{n}}{\partial V_{m}}\right|_{e c}\right) \\
\frac{\partial f_{1}}{\partial \delta}=c\left(\alpha_{e}\right) \frac{Q_{e} S}{m V_{m e}} d_{n}
\end{array}
$$

$$
\begin{array}{r}
\frac{\partial f_{2}}{\partial \alpha}=\left.\frac{Q_{e} S d}{I} \frac{\partial C_{m}}{\partial \alpha}\right|_{e c}  \tag{B.2}\\
\frac{\partial f_{2}}{\partial q}=0 \\
\frac{\partial f_{2}}{\partial V_{m}}=\frac{0.7 p_{0} S d}{I \operatorname{sos}^{2}}\left(2 V_{m e} C_{m e}+\left.V_{m e}^{2} \frac{\partial C_{m}}{\partial V_{m}}\right|_{e c}\right) \\
\frac{\partial f_{2}}{\partial \delta}=\frac{Q_{e} S d}{I} d_{m}
\end{array}
$$

$$
\begin{array}{r}
\frac{\partial f_{3}}{\partial \alpha}=\frac{Q_{e} S}{m}\left(-s\left(\alpha_{e}\right)\left(C_{a}+\frac{T}{Q_{e} S}\right)-\ldots\right. \\
\left.\ldots-\zeta_{1}\left(c\left(\alpha_{e}\right) C_{n e}+\left.s\left(\alpha_{e}\right) \frac{\partial C_{n}}{\partial \alpha}\right|_{e c}\right)\right) \\
\zeta_{1}=\operatorname{sign}\left(C_{n e}\right) \operatorname{sign}\left(s\left(\alpha_{e}\right)\right) \\
\frac{\partial f_{3}}{\partial q}=0 \\
\frac{\partial f_{3}}{\partial V_{m}}=\frac{0.7 p_{0} S}{m s o s^{2}}\left(2 V _ { m e } \left(c\left(\alpha_{e}\right) C_{a}-\ldots\right.\right. \\
\left.\left.\ldots-\left|s\left(\alpha_{e}\right) C_{n e}\right|\right)-\zeta_{2} V_{m e}^{2}\left|s\left(\alpha_{e}\right)\right| \frac{\partial C_{n}}{\partial V_{m}} l_{e c}\right) \\
\zeta_{2}=\operatorname{sign}\left(C_{n e}\right) \\
\frac{\partial f_{3}}{\partial \delta}=-\zeta_{2} \frac{Q S}{m}\left|s\left(\alpha_{e}\right)\right| d_{n} \\
\frac{\partial f_{4}}{\partial \alpha}=0, \frac{\partial f_{4}}{\partial q}=0, \frac{\partial f_{4}}{\partial V_{m}}=0, \frac{\partial f_{4}}{\partial \delta}=-\frac{1}{\tau}  \tag{B.4}\\
\frac{\partial f_{1}}{\partial \delta_{c}}=0, \frac{\partial f_{2}}{\partial \delta_{c}}=0, \frac{\partial f_{3}}{\partial \delta_{c}}=0, \frac{\partial f_{4}}{\partial \delta_{c}}=\frac{1}{\tau} \\
\frac{\partial f_{1}}{\partial T}=0, \frac{\partial f_{2}}{\partial T}=0, \frac{\partial f_{3}}{\partial T}=\frac{c\left(\alpha_{e}\right)}{m}, \frac{\partial f_{4}}{\partial T}=0 \\
\frac{\partial h}{\partial \alpha}=\left.\frac{Q S}{m g} \frac{\partial C_{n}}{\partial \alpha}\right|_{e c} \\
\frac{\partial h}{\partial q}=0 \\
\frac{\partial h}{\partial V_{m}}=\frac{0.7 p_{0} S}{m g s o s^{2}}\left(2 V_{m e} C_{n e}+\left.V_{m e}^{2} \frac{\partial C_{n}}{\partial V_{m}}\right|_{e c}\right) \\
\frac{\partial h}{\partial \delta}=\frac{Q S}{m g} d_{n}
\end{array}
$$

## Appendix C

## Missile guidance system script

```
%% Linearisation
clear all; %#ok<*CLALL>
close all;
clc
%a=(AOA) [rad]
% q = pitch rate [rad/s]
% Vm = missile velocity [ft/s]
% n = normal acceleration [-] (g)
% Conversion
m2ft = 3.2808398950131; % [ft/m]
%ft2m=1/m2ft; %[m/ft]
d2r = pi/180; %[rad/degree]
r2d = 1/d2r;
% sqft2sqm = ft2m^2; %[m^2/ft^2]
% slug2kg = 14.5939029; %[kg/slug]
% lb_sqft2kg_sqm = 0.04214011; %[(kg/m^2)/(lb/ft^2)]
%% Model parameters
% Imperial
sos = 316*m2ft; %[ft/s] (speed of sound)
S = 0.44; %[ft^2]
m=13.98; %[slugs]
p0 = 973.3; %[lb/ft^2]
d = 0.75; %[ft]
l=182.5; %[slugs/ft^2]
g = 9.82*m2ft; %[ft/s^2]
% SI
% sos = 316; %[m/s] (speed of sound)
%S = 0. 44*sqft 2sqm; % [m^2]
%m=13.98*slug2kg; %[kg]
% p0 = 973.3*lb_sqft2kg_sqm; %[kg/m^2]
%d=0.75*ft 2m; %[m]
```

```
%l=182.5*(slug2kg/sqft2sqm); %[kg/m^2]
% Tail fin actuator time constant
tau = 1/150;
% Aero-data parameters
a_n = 0.000103;
b_n = -0.00945;
d_n = -0.034;
a_m = 0.000215;
b_m = -0.0195;
d_m = -0.206;
Ca = -0.3;
%% Operating domain grid
a = linspace( - 20, 20,20)*d2r;
Vm = linspace (2,4,20)*sos;
i = 0;
for ai = a
    i = i + 1;
    j = 0;
    for Vmi = Vm
        j = j + 1;
        Clear A B C D
        e_n = -0.17;
        e_m = 0.051;
        M = Vmi/sos;
        Qp = 0.7*p0*M^2;
        C_n = e_n* (2-M/3);
        c_m = e_m*(8*M/3-7);
        % Areo. coeff. without \Delta-term
        Cna(i,j)=sign(ai)*(a_n*abs(ai*r2d)^3 + ...
            b_n*abs(ai*r2d)^2 + c_n*abs(ai*r2d));
        Cma(i,j) = sign(ai)*(a_m*abs(ai*r2d)^3 + ...
            b_m*abs(ai*r2d)^2 + c_m*abs(ai*r2d));
        % Equilibrium tail fin deflection (same as commanded)
        \Delta(i,j) = -Cma(i,j)/d_m; %#ok<*SAGROW>
        Cn(i,j) = Cna(i,j) + d_n*\Delta(i,j);
        Cm(i,j) = Cma(i,j) + d_m*\Delta(i,j);
        % Equilibrium engine thrust
        T(i,j) = (Qp*S)*((abs(Cn(i,j)*sin(ai))/\operatorname{cos}(ai)) - Ca);
        % Equilibrium pitch rate & normal acceleration
        q(i,j) = - cos(ai)*((Cn(i,j)*Qp*S)/(m*Vmi));
        n(i,j)=(Cn(i,j)*Qp*S)/(m*g);
        %Test (to validate eq. calculations)
        a_dot (i,j) = (cos(ai)/(m*Vmi))*Qp*S*Cn(i,j) + q(i,j);
        q_dot (i,j) = (Qp*S*Cm(i,j)*d)/l;
        Vm_dot (i,j) = (Qp*S/m)*( (Ca + ...
                T(i,j)*(1/(Qp*S)))*\operatorname{cos}(ai) - abs(Cn(i,j)*sin(ai)));
```

```
% Areo. coeff. derivatives
dCn_da = ...
    sign(ai*r2d)*(2*b_n*r2d*abs(ai*r2d)*sign(ai*r2d) - ...
    e_n*r2d*sign(ai*r2d)*(Vmi/(3*sos) - 2) + ...
    3*a_n*r2d*abs(ai*r2d)^2*sign(ai*r2d));
dCm_da = ...
    sign (ai*r2d)*(2*b_m*r2d*abs(ai*r2d)*sign(ai*r2d) + ...
    e_m*r2d*sign (ai*r2d)* ((8*Vmi)/(3*sos) - 7) + ...
    3*a_m*r2d*abs(ai*r2d)^ 2*sign(ai*r2d));
dCn_dVm = - (e_n*abs(ai*r2d)*sign(ai))/(3*sos);
dCm_dVm = (8*e_m*abs(ai*r2d)*sign(ai))/(3*sos);
dCn_d\Delta = d_n;
dCm_d\Delta = d_m;
% Jacobian linearisation
df1_da = ((Qp*S)/(m*Vmi))*(-sin(ai)*Cn(i,j) + ...
    cos(ai)*dCn_da);
df1_dq = 1;
df1_dVm = cos(ai)*((0.7*p0*S)/(m*sos^2))*(dCn_dVm*Vmi + ...
    Cn(i,j));
df1_d\Delta = cos(ai)*((Qp*S)/(m*Vmi))*dCn_d\Delta;
df2_da=((Qp*S*d)/l)*dCm_da;
df2_dq = 0;
df2_dVm=((0.7*p0*S*d)/(I*\mp@subsup{\operatorname{sos}}{}{\wedge}2))*(dCm_dVm*Vmi^2 + ..
    Cm(i,j)*2*Vmi)
df2_d\Delta}=((Qp*S*d)/l)*dCm_d\Delta
df3_da}=((Qp*S)/m)*(-sin(ai)*(Ca + T(i,j)*(1/(Qp*S)) ..
    ) - sign(Cn(i,j))*sign(sin(ai))*(dCn_da*sin(ai) + ...
    Cn(i,j)*\operatorname{cos(ai)) );}
df3_dq = 0;
df3_dVm=(0.7*p0*S)/(m*sos^2)*(Ca*Cos(ai)*2*Vmi - ..
    2*Vmi*abs(Cn(i,j)*sin(ai)) - ...
    sign(Cn(i,j))*(Vmi^2)*dCn_dVm*abs(sin(ai)) )
df3_d\Delta = - sign(Cn(i,j))*abs(sin(ai))*((Qp*S)/m)*dCn_d\Delta;
df4_da = 0;
df4_dq=0;
df4_dVm = 0;
df4_d\Delta = - (1/tau);
df1_d\Delta_c = 0;
df2_d\Delta_c = 0;
df3_d\Delta_c = 0;
df4_d\Delta_c = (1/tau);
df1_dT = 0;
df2_dT = 0;
df3_dT = cos(ai)/m;
df4_dT = 0;
dy_da = ((Qp*S)/(m*g))*(dCn_da);
dy_dq = 0;
dy_dVm}=((0.7*p0*S)/(m*g*sos^2))*(dCn_dVm*Vmi^2 + . . .
        Cn(i,j)*2*Vmi);
dy_d\Delta = (Qp*S*(dCn_d\Delta))/(m*g);
A = [df1_da, df1_dq, df1_dVm, df1_d\Delta;
```

```
                    df2_da, df2_dq, df2_dVm, df2_d\Delta;
                df3_da, df3_dq, df3_dVm, df3_d\Delta;
                df4_da, df4_dq, df4_dVm, df4_d\Delta];
            B = [df1_d\Delta_c, df2_d\Delta_c, df3_d\Delta_c, df4_d\Delta_c;
                    ]';
            C = [eye(size(A));
                dy_da, dy_dq, dy_dVm, dy_d\Delta];
    D = zeros(size(C,1),size(B,2));
    sys{i,j} = ss(A,B,C,D);
    % Augmented system for LQI-control
    Ai = [A, zeros(size(C(end,:)))';
        C(end,:), 0];
        Bi = [B;zeros(1,size(B,2))];
        Ci = [C(end,:),0;
        zeros(size(C(end-1,:))),1];
    Di = zeros(size(Ci,1),size(Bi,2));
    sysi{i,j} = ss(Ai,Bi,Ci,Di);
    % Cost matrices Q & R (only 2 states to be penalized; n ...
            and integral(nc - n))
        Q = diag([10^1.5 10^2.5]);
        R = diag([1 0.01]);
    L{i,j} = lqr(Ai,Bi,Ci'*Q*Ci,R);
        L{i,j} = lqi(ss(A,B,C(end,:),0),Ci'*Q*Ci,R); % Built ..
        in Matlab command for LQI.
        end
end
for i = 1:length(a)
    for j = 1:length(Vm)
        Lda(i,j) = L{i,j}(1,1); % Feedback coefficient for ...
                tain fin deflection (\Delta_C) to angle-of-attack (alpha)
            Ldq(i,j) = L{i,j}(1,2); % Feedback coeff. ...
            LdVm(i,j) = L{i,j}(1,3); % Feedback coeff. ...
            Ld\Delta(i,j) = L{i,j}(1,4); % Feedback coeff. ...
            Ldin(i,j) = L{i,j}(1,5); % Feedback coeff....
            LTa(i,j) = L{i,j}(2,1); % Feedback coefficient for ...
                (engine thrust) T to angle-of-attack (alpha)
            LTq(i,j) = L{i,j} (2,2); %Feedback coeff. ...
            LTVm(i,j) = L{i,j}(2,3); % Feedback coeff....
            LT\Delta (i,j) = L{i,j} (2,4); % Feedback coeff. ...
            LTin(i,j) = L{i,j}(2,5); % Feedback coeff....
    end
end
[amesh,Vmmesh] = meshgrid(a,Vm);
%% Plotting of coefficients over operating domain
```

```
% Figure(1)
surf(amesh,Vmmesh,Lda')
%
% Figure(2)
surf(amesh,Vmmesh,Ldq')
%
% Figure(3)
% surf(amesh,Vmmesh,LdVm')
%
% Figure(4)
surf(amesh,Vmmesh,Ld\Delta')
Figure (5)
surf(amesh,Vmmesh, Ldin')
% Figure(6)
% surf(amesh,Vmmesh,LTa')
%
% Figure(7)
surf(amesh,Vmmesh,LTq')
Figure(8)
surf(amesh,Vmmesh, LTVm')
% Figure(9)
surf(amesh,Vmmesh,LT\Delta')
%
% Figure(10)
% surf(amesh,Vmmesh,LTin')
% Figure(12)
% surf(amesh,Vmmesh,Cn')
%% Test (to see if a random sample of 10 linear systems from ...
    'sys{}' are stabilized and tracks reference properly)
for i = 1:10
    aa = 1 + round((length (a) -1)*rand (1, 1),0);
    bb}=1+\operatorname{round}((length(Vm)-1)*rand (1, 1),0)
    Open_sys = sys{aa,bb};
    A =Open_sys.A; B = Open_sys.B; C = Open_sys.C(end,:); D = ...
        Open_sys.D (end,:);
    LL = L{aa,bb}(:, 1:end-1);
    Closed_sys = ss(A-B*LL,B,C,D);
    s = tf('s');
    Ki = L{aa,bb} (:, end);
    I = Ki/s;
    Closed_sys_Integral = feedback(Closed_sys*I,1);
    % Plott step responses
    Figure(100);
    step(Closed_sys_Integral,linspace(0,10,5000))
    hold on
end
```


## Appendix D

## Coefficients of linearized equations

$$
\begin{align*}
& c_{11}=\frac{I_{x z} \omega_{y_{0}}-I_{x y} \omega_{z_{0}}}{I_{x}}  \tag{D.1}\\
& c_{12}=\frac{\left(I_{y}-I_{z}\right) \omega_{z_{0}}+I_{x z} \omega_{x_{0}}+2 I_{y z} \omega_{y_{0}}}{I_{x}}  \tag{D.2}\\
& c_{13}=\frac{\left(I_{y}-I_{z}\right) \omega_{y_{0}}-I_{x y} \omega_{x_{0}}-2 I_{y z} \omega_{z_{0}}}{I_{x}}  \tag{D.3}\\
& c_{21}=\frac{\left(I_{z}-I_{x}\right) \omega_{z_{0}}-I_{y z} \omega_{y_{0}}-2 I_{x z} \omega_{x_{0}}}{I_{y}}  \tag{D.4}\\
& c_{22}=\frac{I_{x y} \omega_{z_{0}}-I_{y z} \omega_{x_{0}}}{I_{y}}  \tag{D.5}\\
& c_{23}=\frac{\left(I_{z}-I_{x}\right) \omega_{x_{0}}+I_{x y} \omega_{y_{0}}+2 I_{x z} \omega_{z_{0}}}{I_{y}}  \tag{D.6}\\
& c_{31}=\frac{\left(I_{x}-I_{y}\right) \omega_{y_{0}}+I_{y z} \omega_{z_{0}}+2 I_{x y} \omega_{x_{0}}}{I_{z}}  \tag{D.7}\\
& c_{32}=\frac{\left(I_{x}-I_{y}\right) \omega_{x_{0}}-I_{x z} \omega_{z_{0}}-2 I_{x y} \omega_{y_{0}}}{I_{z}}  \tag{D.8}\\
& c_{33}=\frac{I_{y z} \omega_{x_{0}}-I_{x z} \omega_{x_{0}}}{I_{z}} \tag{D.9}
\end{align*}
$$

## Appendix E

## Inverse LQ-algorithm script

```
%% [Q,P,PerfInd,out,Kopt,X] = inv_lqr(A, B, K,output,optgoal)
% A = state matrix 'A'
% B = input matrix 'B'
% K = state feedback matrix 'K'
% output = 1/0 (i.e. 'yes' or 'no')
% if 1 the program will output warnings before
% returning with empty variables 'Q' and 'P'.
% General info:
    This program solves the inverse LQR-problem
    given 'A','B' and 'K'. The program first makes
    inital checks to verify that 'K' indeed could
    be optimal for some 'Q', note that 'R' is
    fixed at unity! The Lyapunov matrix 'P' is
    solved for from 'K = B'*P' using the reduced
    row echelon form of 'B''. This creates a
    relation of the type 'P = V0 + Vn.*Pfree' which is
    represented in the script in function handle
    form. The state weighting matrix 'Q' is then
    solved for from 'Q = - (A-B*0.5*K)'*P + P*(A-B*0.5*K)',
    which is the algebraic Riccati equation (ARE).
    Note that this relation implies an infinite set
    of feasable 'Q's. To reduce this set, and to enforce
    the necessary positive semi-definitness of 'Q',
    the problem of finding 'Q' is reformulated as a
    nonlinear optimization problem in the free variables
    of 'P' ('Pfree'). The opt. goal is to create a 'Q'
    which is as diagonal as possible. This is achieved
    by defining the objective function as:
    f(x)}=\operatorname{max}(\operatorname{svd}(abs(Q(Pfree)-diag(Q(Pfree))))).
    the only constraint is nonlinear and of the form:
    min(eig(Q(Pfree))) \geqa, where 'a' is chosen in the
    script. Further note that 'Pfree', for simplicity,
    takes the name 'X'.
function [Q,P,PerfInd,out,Kopt,X] = ...
    inv_lqr(Ain,Bin,K,output,optgoal)
global VarStr n m A B
A = Ain;
B = Bin;
n = size(A,1);
```

```
m = size(B,2);
if isa(K,'double')
    if m<n
    ok = 1;
    for run = 1:2
    %Quick check!
    I = real (eig(A-B*K))>0;
    if sum(I) > 0
        if output == 1
                disp('The feedback matrix ''K'' is not stabilizing!')
                disp('It can therefore not be LQ-optimal!')
                pause(5)
                clc
        end
        P = [];
        Q = [];
        PerfInd = inf;
        out = [];
        return
    end
    [RB,p] = rref([B',K]);
    ind = 1:n;
    np = ind(not(ismember(ind,p)));
    Vn=RB(:,1:0.5*end);
    VO = RB (:,0.5*end+1:end);
    Vn_col_sum = sum(abs(Vn), 2);
    row0 = find(Vn_col_sum==0);
    if sum(abs(V0(row0)))>0
        if output == 1
            disp('The desired feedback matrix ''K'' cannot be ...
                created by means of any real matrix ''P''!')
            pause
                clc
        end
        P = [];
        Q = [];
        PerfInd = inf;
        out = [];
        return % Optimal control allocation?
    end
    Vn(row0,:) = [];
    V0(row0,:) = [];
    Vn = [Vn;zeros(length(np),n)];
    V0 = [V0;zeros(length(np),n)];
    for i = 1:length(np)
        if np(i)>1
        pre = ind(1:np(i)-1);
    else
        continue
    end
    if np(i)<n*n
        post = ind(np(i):end);
    else
        continue
```

```
    end
    ind = [pre,ind(end),post];
    ind(end) = [];
end
Vn = Vn(ind,:);
V0 = V0(ind,:);
V0_vec = v0(:);
Vn_block = kron(eye(n,n),Vn);
%Symmetry
LinearInd = reshape(1:n*n,n,n);
Sym_mat = [];
for i = 1:n
    for j = 1:n
        Sym_vec = zeros(1,n*n);
        Sym_vec(LinearInd(i,j)) = Sym_vec(LinearInd(i,j)) + 1;
        Sym_vec(LinearInd(j,i)) = Sym_vec(LinearInd(j,i)) - 1;
        Sym_mat = [Sym_mat;Sym_vec]; %#ok<*AGROW>
        end
end
[Sym_mat,p] = rref(Sym_mat);
ind = 1:n*n;
np = ind(not(ismember(ind,p)));
for i = 1:length(np)
    if np(i)>1
        pre = ind(1:np(i)-1);
    else
        continue
    end
    if np(i)<n*n
        post = ind(np(i):end);
    else
        continue
    end
    ind = [pre,ind(end),post];
    ind(end) = [];
end
Sym_mat = Sym_mat(ind,:);
if run == 1
    An = [Vn_block;Sym_mat];
    bn = [V0_vec;zeros(n*n,1)];
else
    An = Vn_block;
    bn = V0_vec;
end
if length(np) == n + n*(n-1)/2
    %null
else
    disp('Symmetry constraints are invalid! Please check code!')
    pause(3)
    clc
    Q = [];
    P = [];
    PerfInd = inf;
    out = [];
```

```
| 165 
    [AV,p] = rref([An,bn]);
    AVn = AV (:, 1:n*n);
    AVO = AV (:, end);
    ind = 1:n*n;
    np = ind(not(ismember(ind,p)));
    AVn_col_sum = sum(abs(AVn),2);
    row0 = find(AVn_col_sum==0);
    if sum(abs(AV0(row0)))\not=0
        if output == 1
            disp('Warning! Conflict between constraints. ...
                    Symmetry is enforced through optimization!')
                disp('Q may therefore not be maximally diagonal!')
                pause(3)
                clc
        end
        ok = 0;
    end
    AVn(row0,:) = [];
    AV0 (row0,:) = [];
    AVn = [AVn;zeros(length(np),n*n)];
    AVO = [AV0;zeros(length(np),1)];
    for i = 1:length(np)
    if np(i)>1
        pre = ind(1:np(i)-1);
    else
        continue
    end
    if np(i)<n*n
        post = ind(np(i):end);
    else
        continue
    end
    ind = [pre,ind(end),post];
    ind(end) = [];
end
AVn = AVn(ind,:);
AVO = AVO(ind,:);
AVn = - (AVn-eye (n*n));
VarStr = [];
nonp = length(np);
for i = 1:nonp
    VarStr = [VarStr,' ','X(',num2str(i),')'];
end
VarStr = ['[',VarStr,']'];
P1 = @(X) reshape (AVO + ...
    sum(AVn(:,np) . *repmat(eval(VarStr),n*n,1), 2),n,n);
    Ac}=A-B*0.5*K
    Q = @(X) Ac'*P1 (X) + P1 (X)*Ac;
    if strcmp(optgoal,'svd')
```

```
        fun = @(X) max(svd(abs(Q(X)-diag(diag(Q(X))))));
else
    fun = @(X) sum(sum(abs((Q(X)-diag(diag(Q(X)))))));
end
if run == 2
    fun = @(X) sum(sum(abs(P1(X) - P1(X)')));
end
options = optimoptions('fmincon','Display','iter',...
'MaxFunctionEvaluations',1e7,'MaxIterations',500,...
'ConstraintTolerance',1e-9,'OptimalityTolerance',1e-9,...
'FunctionTolerance',1e-12);
a = 0;
nonlcon = @(X) min_eig_con(X,Q,P1,a);
e = 1e-2;
if nonp}=
        if ((ok==1)&&(run==1))|((ok==0)&&run==2)
            x0 = ones(nonp,1);
            [X,\neg,\neg,out] = ...
                    fmincon(fun,x0,[],[],[],[],[],[],nonlcon,options);
            P = P1(X');
            if run == 2
                    P = 0.5*(P + P');
            end
            Q = -(Ac'*P + P*Ac);
            Ktest = lqr(A,B,Q,eye(size(B,2)));
            PerfInd = sum(sum(abs(Ktest-K)))/numel(B);
        end
else
    P = reshape(AV0,n,n);
    Q = - (Ac'*P + P*AC);
    try
            Ktest = lqr(A,B,Q,eye(size(B,2)));
            PerfInd = sum(sum(abs(Ktest-K)))/numel(B);
        catch
            PerfInd = inf;
        end
        out = [];
end
if (output == 1)&&((ok==1)|(run==2))
        if PerfInd > e
            disp('Warning! The feedback matrix ''K'' has been ...
                    recreated within a higher tolerance than what ...
                    might be acceptable!')
        end
end
if ok == 1
    break
end
end
else
    Q = K'*K - (pinv(B)'*K*A + A'*pinv(B)'*K);
    try
        [Ktest,P] = lqr(A,B,Q,eye(m));
        PerfInd = sum(sum(abs(Ktest-K)))/numel(B);
```

```
                    out = [];
                X = [];
        catch
            PerfInd = inf;
            Q = [];
            P = [];
            out = [];
            X = [];
        end
    end
    Kopt = [];
elseif isa(K,'function_handle')
    Q = @(X) K(X)'*K(X) - (pinv(B)'*K(X)*A + A'*pinv(B)'*K(X));
    fun = @(X) Perf(X,Q,K);
    options = optimoptions('fmincon','Display','iter',...
    'MaxFunctionEvaluations',1e7,'MaxIterations',1500,...
    'ConstraintTolerance',1e-9,'OptimalityTolerance',1e-9,...
    'FunctionTolerance',1e-12);
    try
        K(ones(1, numel(B)));
        R = rand(size(B));
        RR = R'*R;
        x0 = 0.01* ' '*RR;
        x0 = x0(:);
    catch
        count = 0;
        while true
            count = count + 1;
            try
                    K(ones(1, count));
                    x0 = 0.01*ones(1,count);
                    break
                catch
                    %null
                end
            end
    end
    a = 0;
    nonlcon = @(X) min_eig_con2(X,Q,a);
    [X,\neg,\neg,out] = fmincon(fun,x0,[],[],[],[],[],[],nonlcon,options);
    Q = 0.5*(Q(X)'+Q(X));
    Kopt = K(X);
    [Ktest,P] = lqr(A,B,Q,eye(m));
    PerfInd = sum(sum(abs(Ktest-K(X))))/numel (B);
else
    disp('Unrecoginzed class of input ''K''')
    Q = [];
    P = [];
    PerfInd = inf;
    out = [];
    Kopt = [];
    return
end
end
function [c,ceq] = min_eig_con(X,Q,P1,a)
c(1) = max(eig(0.5*(Q(X)+Q(X)'))) +a;
C(2) = -min(eig(0.5*(P1 (X)+P1(X)'))) +a;
ceq = [];
```

```
346 end
347
function [c,ceq] = min_eig_con2(X,Q,a)
c(1) = - min(eig(0.5*(Q(X)+Q(X)'))) +a;
ceq = [];
end
function val = Perf(X,Q,K)
global m A B
try
Ktest =@(X) lqr(A,B,Q(X), eye(m));
    val = sum(sum(abs(Ktest(X)-K(X))))/numel(B);
catch
    val = 1e6;
    end
    end
```


## Appendix F

## LQ-optimal eigenstructure assignment script

```
function [Q,Kreal,w_val,Xopt] = MyPlace(Ain,Bin,P,x0opt,output,N)
global VarStr m n A B w %#ok<*NUSED>
A = Ain;
B = Bin;
n = size(A,1);
m= size(B,2);
if (output == 1)&&(m>1)
    disp('Warning! As the system has multiple inputs an ...
            optimization has to be performed in order to find K.')
    disp('This means that the process of finding K might take
            time! The task of finding the Hessian in the matlab ...
            function')
        disp('fmincon(...) runs in polynomial time, O(N^2), where N ...
            is the number of optimization variables (equal to ...
            numel(B)).')
        disp('If the program is "hot started" (given a near optimal ...
            starting point) the time consumption will be ...
            substantially lower!')
        pause
        clc
end
Lambda = eigdiag(P);
I = eye(n,n);
T = kron(I,A) + kron(-Lambda',I);
if rank(T)==n^2
    XG = T\kron(I,B);
else
    Kreal = [];
    return
end
VarStr = [];
for i = 1:numel(B)
    VarStr = [VarStr,' ','X(',num2str(i),')']; %#ok<*AGROW>
end
```

```
VarStr = ['[',VarStr,']'''];
V = @(X) reshape(XG*eval(VarStr),n,n);
K = @(X) reshape(eval(VarStr),size(B'))/V(X);
Kfun = K;
Klqr = lqr(A,B,eye(n),eye(m));
if isempty(x0opt)
    x0 = 0.01*ones(numel (B),1);
else
    x0 = x0opt;
end
syms w
if rank(A) == n
    start = 0;
else
    start = 0.01;
end
MaxIt = 200;
if m>1
    ItStr = 'iter';
else
    ItStr = 'off';
end
w_val = [linspace(start,50,N),100,150,200];
options = optimoptions('fmincon','Display',ItStr,...
'MaxFunctionEvaluations',1e7,'MaxIterations',MaxIt, . . 
'algorithm','interior-point','FunctionTolerance',1e-12,...
'StepTolerance',1e-9);
fun = @(X) optfun(X,K,V,w_val,Klqr,P,1,0);
[X,fval] = fmincon(fun,x0,[],[],[],[],[],[],[],options); ...
    %#ok<*ASGLU>
Kreal = K(X);
Xopt = X;
[Q,\neg,PerfInd,\neg,\neg,\neg] = inv_lqr(A,B,K(X),0,'svd');
if PerfInd < inf %change
    Kreal = K(X);
    Xopt = X;
    return
end
% Last resort
try
    [Q,\neg,PerfInd,\neg,Kreal,X] = inv_lqr(A,B,Kfun,0,'svd');
    if PerfInd < inf %change
        Xopt = X;
        return
    end
catch
    if output == 1
        disp('No feasable solution found!')
        Kreal = [];
        Xopt = [];
        return
```

```
    end
end
end
function val = optfun(X,K,V,w_val,Klqr,P,a,b) %#ok<*INUSL, *INUSD>
global A B n m
if m>n
    c = 0;
else
    c = 1;
end
if a>0
    T = @(X,w) eye(m,m) + K(X)*((eye(n,n)*1i*w - A)\B);
    TT =@(X,w) ctranspose(T (X,w))*T (X,w) - eye (m,m);
    fpp = @(X,w) min(eig( TT(X,w) ));
    Tlqr = @(w) eye(m,m) + Klqr*((eye (n,n)*1i*w - A)\B);
    TTlqr = @(w) ctranspose(Tlqr(w))*Tlqr(w) - eye(m,m);
    flqr = @(w) min(eig( TTlqr(w) ));
    %fpoly = @(w) 1/(w^2 + 1);
    count = 0;
    for i = w_val
            count = count + 1;
            eig_real(count) = fpp(X,i);
            eig_des(count) = flqr(i);
        end
else
    eig_real = 0;
    eig_des = 0;
end
Ac = @(X) A - B*K (X);
Off_diag = @(X) sum(sum(abs(Ac(X)-\operatorname{dag}(\operatorname{diag}(AC(X))))));
val = a*rms(abs( eig_real - c*eig_des )) + b*off_diag(X);
end
function Lambda = eigdiag(P)
if length(P)\not=numel(P)
        warning('''P'' must be a vector!')
        Lambda = [];
        return
end
Base = diag(real(P));
I = imag(P);
ind = find(I\not=0);
I (I==0) = [];
if sum(I)\not=0
    warning('The complex poles must conjugate!')
    Lambda = [];
    return
end
for i = 1:length(I)
    col_shift = mod(1:length(I),2) - 1*not(mod(1:length(I),2));
```

```
156 Base(ind(i),ind(i)+col_shift(i)) = I(i); %#Ok<*FNDSB>
157 end
Lambda = Base;
end
```

