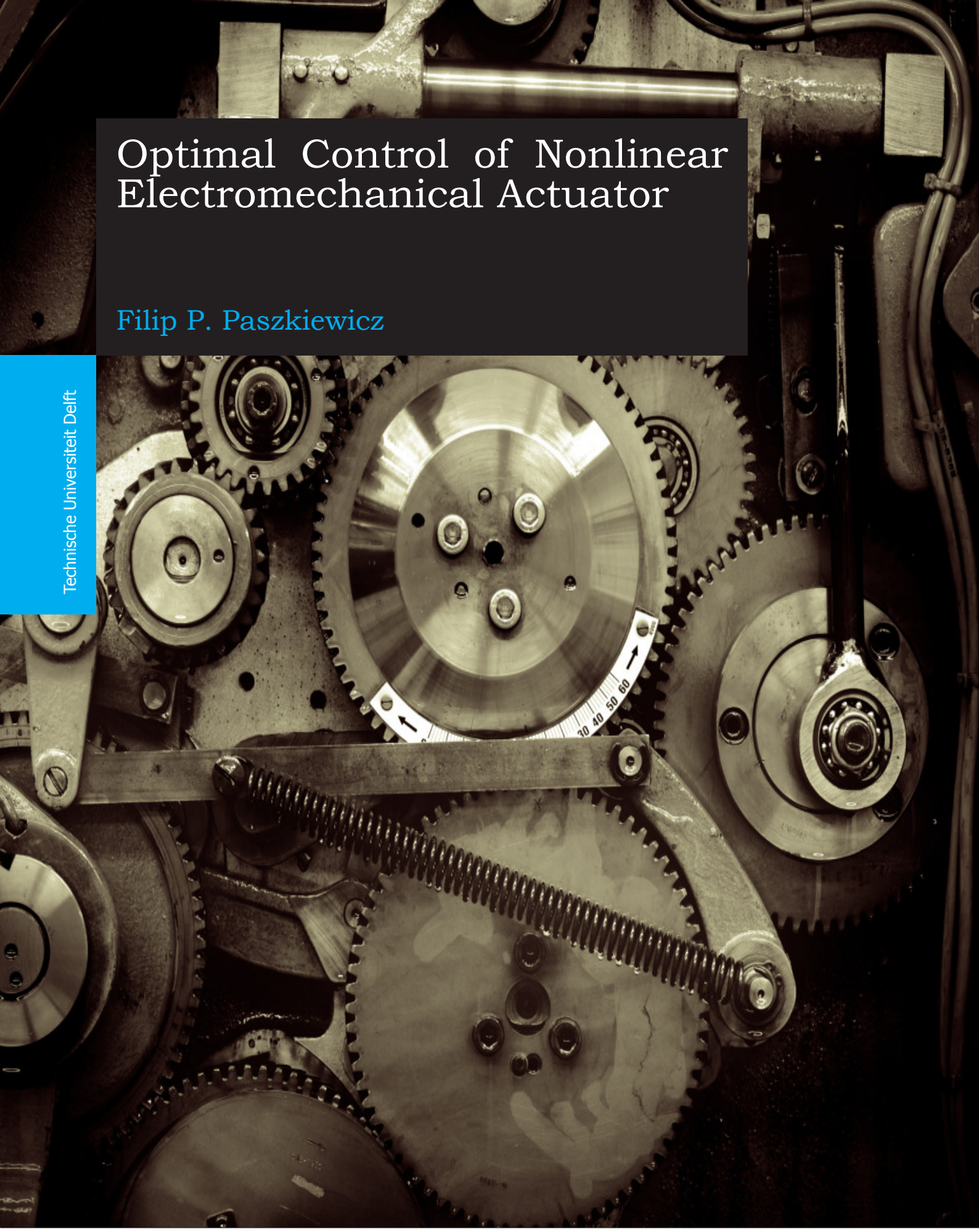


Optimal Control of Nonlinear Electromechanical Actuator

Filip P. Paszkiewicz

Technische Universiteit Delft

TU Delft Delft University of Technology

Challenge the future

Optimal Control of Nonlinear Electromechanical Actuator

by

Filip P. Paszkiewicz

in partial fulfilment of the requirements for the degree of

Master of Science
in Systems and Control

at the Delft University of Technology,
to be defended publicly on 20 Spetember 2016.

Supervisors:	*Dr. Ir.* Manuel Mazo,	TU Delft
	Ir. France Niewold,	Moog Inc.
Thesis committee:	*Dr. Ir.* Manuel Mazo Jr.	
	Dr. Ir. Jens Kober	
	Dr. Ir. Erik-Jan van Kampen	

TU Delft Delft University of Technology

Preface

”The desired answer can only be found by asking the right questions, at the right time, in the right place.”

Filip P. Paszkiewicz
Delft, September 2016

Abstract

The work done in this thesis focused on controller software for the Generic Short Stroke Actuator of Moog B.V. The actuator is controlled by the FCS controller, which makes use of a switched PID controller. This allows arbitrary control over the movement of the Generic Short Stroke Actuator. The FCS controller can switch between a position and force feedback loop. For various safety reasons the FCS controller has both force limits placed on it. If those limits are breached during operation, the FCS controller switches from position PID to force PID. Under certain conditions the even of a switch drives the actuator into a region of instability.

The following thesis deals with finding the conditions for which the FCS controller drives the system into unstable regions. With the problems found, a solutions will be sought after from the field of Hybrid and Optimal Control theory. Three different approaches are investigated to solve the problems found: Dwell-time, Back Calculation Anti-Wind Up,Model Predictive Control.

Dwell-time is a technique from the Hybrid Control theory field. A adequately found dwell-time guaranties the stability of any arbitrary switching system, like the FCS controller is. The Back Calculation Anti-wind up technique is a extension of PID control that deals with actuator saturation due to integrator wind up. Both of those techniques are improvements made to the FCS controller. Model Predictive Control comes form the filed of Optimal Control theory. Model predictive control tries to find optimal control input at each time step, taking into account any limits.

In order to be able to design the above controllers, an accurate mathematical model was developed using fist principles. The model was optimised and validated against measured data from the system plant. As the plant is non-linear, to analyse it locally, the non-linear equations are linearised to establish a state space linear time-invariant system model. As the controllers are implemented on a computer, the model needed to be discretized to enable design of discrete controllers.

Acronyms & Mathematics

Acronyms

FCS	Fokker Control Systems
GSSA	Generic Short Stroke Actuator
MITS	Moog Integrated Test Suit
MTC	Modular Test Controller
MSD	Moog Servo Drive
PLC	Programmable Logic Controller
PID	Proportional, Integral, Derivative
PM	Permanent Magnet
PMSM	Permanent Magnet Synchronous Motor
EM	Electromechanical Motor
SS	State Space
LTI	Linear Time-Invariant

Mathematics

A	matrix
$x(t)$	state vector
$f(\dots)$	function of …
R	a variable
$\mathcal{R}$	set of real numbers
$\mathcal{C}^1$	continuously differentiable
$\forall$	for all
$\dot{a}$	the derivative of a
…	a set of
$\lvert \dots \rvert$	absolute value
$\lVert \dots \rVert$	norm
$\rightarrow$	mapping
$\subset$	subset
$\leq$	smaller/bigger or equal
$<$	strictly smaller/bigger
$:=$	defined as
$\neq$	not equal to
$\in$	belongs to set
$\Rightarrow$ / $\rightrightarrows$	implies, if ... then
Δ	rate of change
A^T	transpose of A
$x(k+j\|k)$	prediction of $x(k+j)$ at time step k

Contents

List of Figures

List of Tables

1

Introduction

In the following thesis, research is conducted on precision control of a electromechanical actuator, proposed by engineers from Moog Inc. The project involves improvements on controlling the Generic Short Stroke Actuator (referred to as GSSA, throughout the thesis), which is used in testing and simulation, mainly for automotive and airspace applications. With the electromechanical actuator already designed and constructed, the focus is on software controlling the actuator. As a result all the research conducted in this thesis will be done to improve programs and controllers governing the actuator. No changes in the hardware design were made due to high cost of such approach.

1.1. Fokker, FCS and then Moog

Fokker was a Dutch aircraft manufacturer established in 1919 in Germany. In the 1970's Fokker Control Systems came as subsidiary of within the Fokker company. First products were control loading equipment used in flight simulators to replicate the forces that act on the pilot's controls. In the 1980's, the company extended their control system technologies to train and truck simulation industries. Later, in the 1990's, Fokker Control Systems, concentrated on motion platforms (also referred to as Stewart platforms or hexapods) used for flight, rail-based, vehicle, truck and car driving, and submarine simulation applications. The main Fokker company bankrupted in 1996, which resulted in Fokker Control Systems becoming an independent company, rather than a subsidiary, in 1997.

In 2005 Moog Inc. purchased Fokker Control Systems in Netherlands, with one office in Heerle and one in Nieuw Vennep. The company name was changed to Moog B.V. Moog is an American worldwide designer and manufacturer of motion and fluid controls, and control systems for applications in aerospace, defence, industrial and medical device markets. Their products and systems include mainly military and commercial aircraft flight controls, but also control of satellite positioning systems. Other products of Moog are thrust vector controls for space launch vehicles and controls for positioning shipment loads in trucks and automatic loading for transportation systems. Alternative uses are for civilian purposes like in industrial applications, including injection moulding machines for the plastics markets, metal forming, power generating turbines, simulators used to train pilots and certain medical applications.

Nowadays, the division of Moog B.V. in Nieuw Vennep focuses on the design and manufacturing of products and systems for aerospace and automotive testing, flight simulation and haptic/robotic applications. Examples of products are the G-seat, see Figure (1.1a), which is a replica of an ejection seat provided with haptic feedback to the pilot in the seat. It is used to get pilots familiar with forces working on their body during flight. The G-seat can be used in combination with a motion platform, creating a flight simulator, such as depicted in Figure (1.1b) The haptic technology can also be used for medical purposes, such as dental trainer simulators Figure (1.1c).

For many years Moog B.V. was predominantly using hydraulic actuators as they are very consistent and can generate big amounts of power and energy, required in macro-scale applications (aircraft and

(a) G-Seat

(b) Flight Simulator

(c) Dental Trainer

Figure 1.1: Three examples of products supplied by Moog: G-seat (a), flight simulator (b) and dental trainer (c). (Pictures from Moog's website.)

automotive simulations). In recent years though, Moog has been intensely investing into electromechanical actuators in efforts to make a transition from hydraulic to electric powered actuation. A result of those efforts is the research conducted in this thesis. With the power and energy output of electric motors increasing year on year, Moog introduces electromechanical actuators into an ever increasing range of simulation applications. Electrically powered motors have several advantages over hydraulically powered motors. Electricity is much easier to supply, over oil which needs pumps, valves and tubes to deliver it to a actuator. Electromechanical actuators only need a power source and they can be used within minutes from unpacking. Electricity is also much cleaner then oil, both in case of usage and if an accident occurs spillage. As a result of this electromechanical actuators are safer to work with, as in case of any malfunction a safety button is pressed and the actuator ceases any operation. In the case of hydraulic, in the worst case scenario, litres of leakage need to be cleaned up.

As a consequence of using electric actuators in new applications, new challenges arise, new limits are reached, and new solutions need to be developed. From the initiative of Moog B.V. engineers, this thesis does exactly that with research conducted in the new an exciting fields of hybrid and optimal control. In the case of this thesis, hybrid and optimal control are applied in the application of precision control for an electromechanical actuator.

1.2. The Project

What follows is the project proposed by MOOG B.V., where an electromechanical actuator is used for short stroke applications. With a new application came new challenges. In this thesis the goal is to solve those challenges either through hybrid or optimal control. Figure (1.2) most accurately depicts the controller system under investigation throughout this thesis, in its current form. Moog has applied the Fokker Control Systems (referred to as FCS) controller to the newly developed electromechanical actuator, which runs on a PC in the Moog Integrated Test Suite (referred to as MITS). That is connected to a Modular Test Controller (referred to as MTC). The MTC connects to the Moog Servo Drive (referred to as MSD) which directly connects to the electromechanical (referred to as EM) actuator. The EM actuator is discussed in detail in Section 3.1.1 and the MSD is described in full in Section 3.1.3.

As shown FCS controller has two control loops, which use PID controllers (although in general the D-term is not used, due to noise amplification nature). The FCS controller can work on two different feedback-loops, either the Position loop or the Force loop. This thesis focuses its research on improving the position loop. Under normal operational circumstances the FCS uses position feedback to follow a user's set reference signal like a sinusoidal wave, a step or a square wave. The controller can in principle follow any arbitrary motion. However, most of the time periodic signals are used due to the application. Examples of periodic signals are sinusoid, square and sawtooth wave. The uniqueness of the FCS controller comes from how limits are handled. Several levels of hard limits are placed on the actuator to either not damage the machine itself, or the load the machine is operating on. Four limits are placed on operation of the GSSA, which are enforced by the FCS controller:

1. Angle on the crank of the actuator

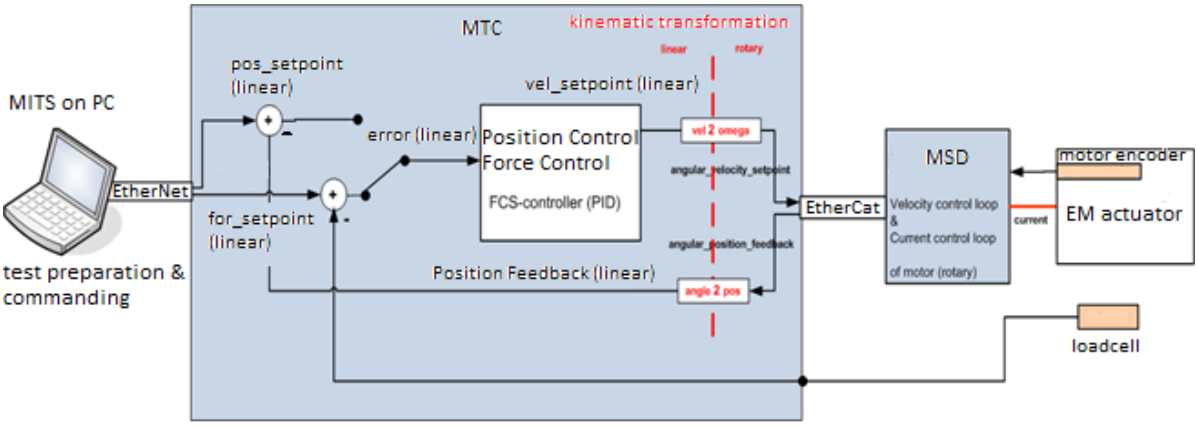


Figure 1.2: The Fokker Control Systems controller, with the Generic Short Stroke actuator

2. Available velocity of the load
3. Force exerted on the load
4. Position of the load (in the vertical axis)

If any of the limits are breached the FCS controller switches to a lower level control. In instant of a switch the limit that was validated becomes a constant reference set-point for the lower PI controller to follow. This thesis will focus on the operation of the position loop during a switch to force control, due to the force limits being violated.

Currently the switching between the PID controller of the position loop and the PID controller of the force loop is unstable. A switch from one control loop to the other results in the system becoming unstable and switching off. Preliminary investigation suggests that the problems must arise either from the nature of the switching (the switching is to fast or there are oscillations around the switching regions) or alternatively the control loops end in an unstable region after a switch occurs, and the PID controller drives the system into instability.

This leads to establishment of the following research question to be investigated throughout this thesis:

- Develop a controller that ensures stability and required performance of the Generic Short Stroke Actuator, under all operational conditions

1.3. Thesis arrangement

The rest of the thesis is arranged in four different chapters concerned with the plant model, investigated control schemes and the results of the thesis.

In Chapter (2) the physical principles that govern electric motors are explained and what types of electric motors are available. The difference between various electric motors is also explained. This is followed by a brief introduction into the hybrid and optimal control for electric motors.

In Chapter (3) the mathematical model of the plant based on first principle equations is given. Those equations are also linearised around an equilibrium point to give a state space linear time invariant system which is used for controller design. Parameters of the plant model are determined and validated using two independent sets of measurement data.

In Chapter (4) different potential control strategies that are investigated in this thesis are introduced. First the control problem is identified and subsequently three different control approaches are tested to find an answer to the stated research question. The three strategies investigated are:

Dwell Time control from Hybrid control field, PID anti-wind up techniques which elaborates in the existing PID control, and last Model Predictive Controller is designed as a type of Optimal Control strategy.

In Chapter (5) the findings of the thesis and the problems that arose during the research are discussed. On top, conclusions are drawn regarding the findings throughout the thesis and the research question is answered. The chapter is closed with suggestions regarding the future work beyond where this thesis concluded its research.

2

On Electric Motors, Hybrid Systems and Optimal Control

An electric motor is an electrical-mechanical machine that converts electrical energy into mechanical energy. The reverse is done by electric generators where mechanical energy is converged into electrical energy. In normal motoring mode, most electric motors operate through the interaction between an electric motor's magnetic field and winding currents to generate force within the motor. In certain applications, such as in the transportation industry with traction motors, electric motors can operate in both motoring and generating or braking modes to also produce electrical energy from mechanical energy.

2.1. Brief history

First ancestors of electric motors can be traced back to 1740s, to electrostatic devices made by *Andrew Gordon*. The theoretical principle behind production of mechanical force by the interactions of an electric current and a magnetic field, Ampère's force law, was discovered later by *André-Marie Ampère* in 1820. It wasn't till 1821 tough, when *Michael Faraday* did his famous experiment with permanent magnet (PM), current and mercury. Faraday dipped a wire, with a PM attached to it, in a pool of mercury. Once current was passed through the wire, it rotated around the magnet. Thus for the first time electric energy was converted to mechanical by the means of electromagnetism. In 1827, Hungarian physicist *Ányos Jedlik* started experimenting with electromagnetic coils. In 1828 Jedlik demonstrated the first device to contain the three main components of practical DC motors: the stator, rotor and commutator.

Those days at the small power end electric motors can be found in in objects as small as wristwatches. At the other end of the power spectrum (100's of Mega Watts), electric motors pump water uphill's for energy storage. Of course electric motors come in all shapes and sizes. Where motors of 12-15 MW power propulsion for cruise ships, motors of 12-15 W range can keep as cool in fans during summer. The flexibility of electric motors and the possibilities the different types of motors bring with them, make them the fastest growing alternative to hydraulics and petrol motors.[1]

2.2. Basic structure and working

At its simplest an electric motor consists of four parts: rotor, stator, windings, and commutator. Those four basic parts can be seen in Figure (2.2) In an electric motor the moving part is the rotor which turns the shaft to deliver the mechanical power. The stator is the stationary part of the motor's electromagnetic circuit and usually consists of either windings or permanent magnet. The stator core is made up of many thin metal sheets, called laminations. Laminations are used to reduce energy losses that would be present of a core made out of solid piece of metal would be used. The distance between the rotor and stator is called an air gap. The air gap has important uses, and is in general as small as possible. A large gap has a strong negative effect on the performance of an electric motor. Windings are wires that are laid in coils, usually wrapped around a laminated soft iron magnetic core so as to

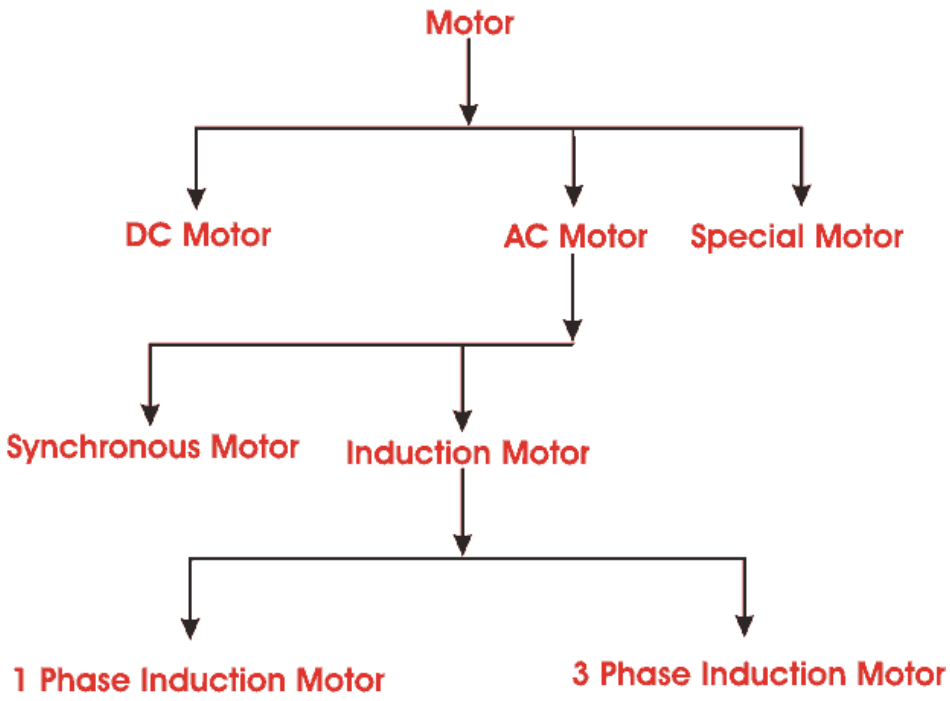


Figure 2.1: Division of electric motor types.

form magnetic poles when energised with current. Electric machines come in two possible configuration of magnet field poles: salient-pole machine and nonsalient-pole machine. Some motors have conductors which consist of thicker metal, such as bars or sheets of metal, usually copper, although sometimes aluminium is used. These are usually powered by electromagnetic induction. A commutator is a mechanism used to switch the input of most DC machines and certain AC machines consisting of slip ring segments insulated from each other and from the electric motor's shaft.

With its structure established we would like to now explain how an electric motor works. At its simplest any electric motor will have its stator produce a stationary or rotating, magnetic field. From physics we know that if we were to place a wire in a magnetic field, the wire will experience forces according to Lorentz law. The direction of such forces can be established according to the right-hand-rule. Now if a rotor is made of many windings of wire, which are wrapped around and connected in a circuit, forces of opposite direction are created and the rotor moves. According to Faraday's law of induction a wire with current passing through it, either direct or alternating, will *induce an electromotive force* referred to as induced EMF. The principle was first discovered by the already mentioned Michael Faraday and was later mathematically described by a *Scot James C. Maxwell*. All the physical laws just discussed are illustrated in Figure (2.3) for a better visual representation.

2.3. Types of motors

As electric motors are used for a vast array of tasks and due to the constant improvement in their technology, many types of motors exist adjusted for various tasks. Despite that few main categories can be identified. First of all, electric motors are divided by the type of current that drives them: *dc motors* or *ac motors*. Further dc motors are mainly divided into *brushed* and *brushless*. On the other hand, ac motors are either *synchronous* or *asynchronous* (usually referred to as induction motors). Induction motors come then mainly in either 1 phase configuration or 3 phase configuration. The last major category is special motors like Linear Induction motor (LIM), Stepper motor, Servo motor etc. with special features that were developed according to the needs of the industry or for a particular

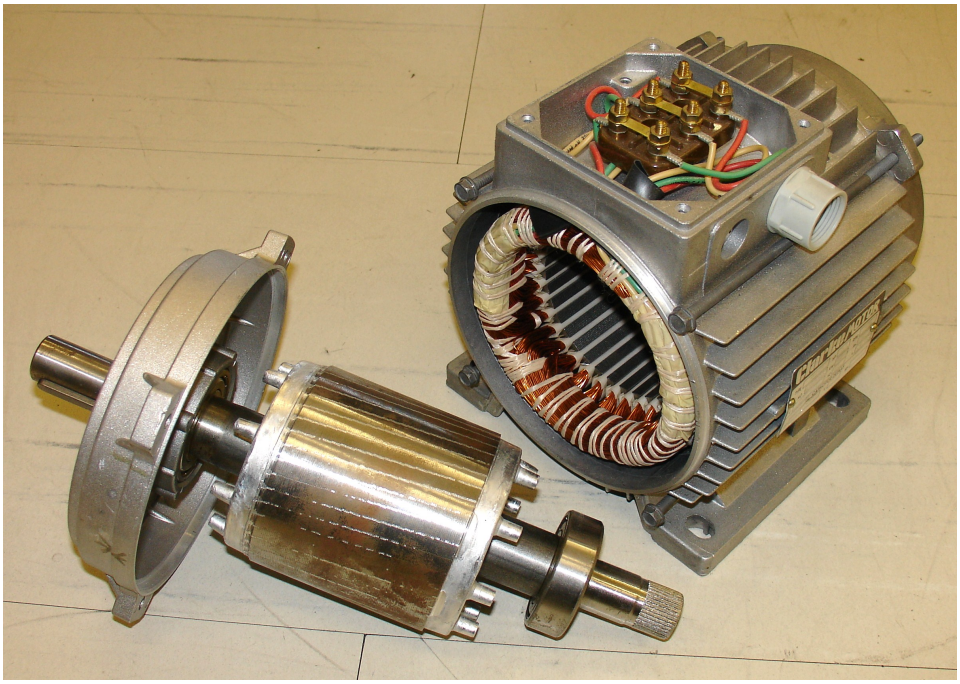


Figure 2.2: Basic structure of a electric motor

application like the use of hysteresis motor in hand watches because of its compactness. Figure (2.1) shows visually how the major types of motors are sub-divided.

2.3.1. DC motors

Historically the first motor was a *simple* dc-motor, which can be referred to as a *brushed DC electric motor*. Brushed DC motors can have a range of velocities, which are tuned by changing the operating voltage or the strength of the magnetic field. Depending on the connections configuration of the field to the power supply, the speed and torque characteristics of a brushed motor can be altered to provide steady speed or speed inversely proportional to the mechanical load. Brushed motors continue to be used in many applications like electrical propulsion, cranes, paper machines and steel rolling mills. The simplest version of a brushed DC motor is illustrated in Figure (2.4). The motor for simplicity has only two poles, created by a permanent magnet. Then windings are placed within the magnetic field. When a current passes through the coil wrapped around a soft iron core, the side of the positive pole is acted upon by an upwards force, while the other side is acted upon by a downward force. According to Fleming's *left hand rule*, the forces cause a turning effect on the coil, making it rotate. To make the motor rotate in a constant direction, direct current commutators make the current reverse in direction every half a cycle (in a two-pole motor) thus causing the motor to continue to rotate in the same direction.

The main problem with a brushed Dc motor is the brushes wear down due to mechanical movement and require regular replacement. *Brushless DC motors* using power electronic devices have replaced brushed motors in many applications. So what are the differences between brushed and brushless dc motors, and how does a brushless dc work? *Brushless DC electric motor* (BLDC for short), is a synchronous motors that is powered by a DC electric source via an integrated inverter power supply, which produces an AC electric signal to drive the motor. In this context alternating current does not imply a sinusoidal waveform, but rather a bi-directional current with no restriction on waveform. Additional sensors and electronics control the inverter output amplitude and waveform and frequency i.e. rotor speed. Brushless motors may be described as stepper motors; however, the term stepper motor tends to be used for motors that are designed specifically to be operated in a mode where they are frequently stopped with the rotor in a defined angular position. Two key performance parameters of brushless DC motors are the *motor constant*, Kt and the *BEMF constant*, Ke.

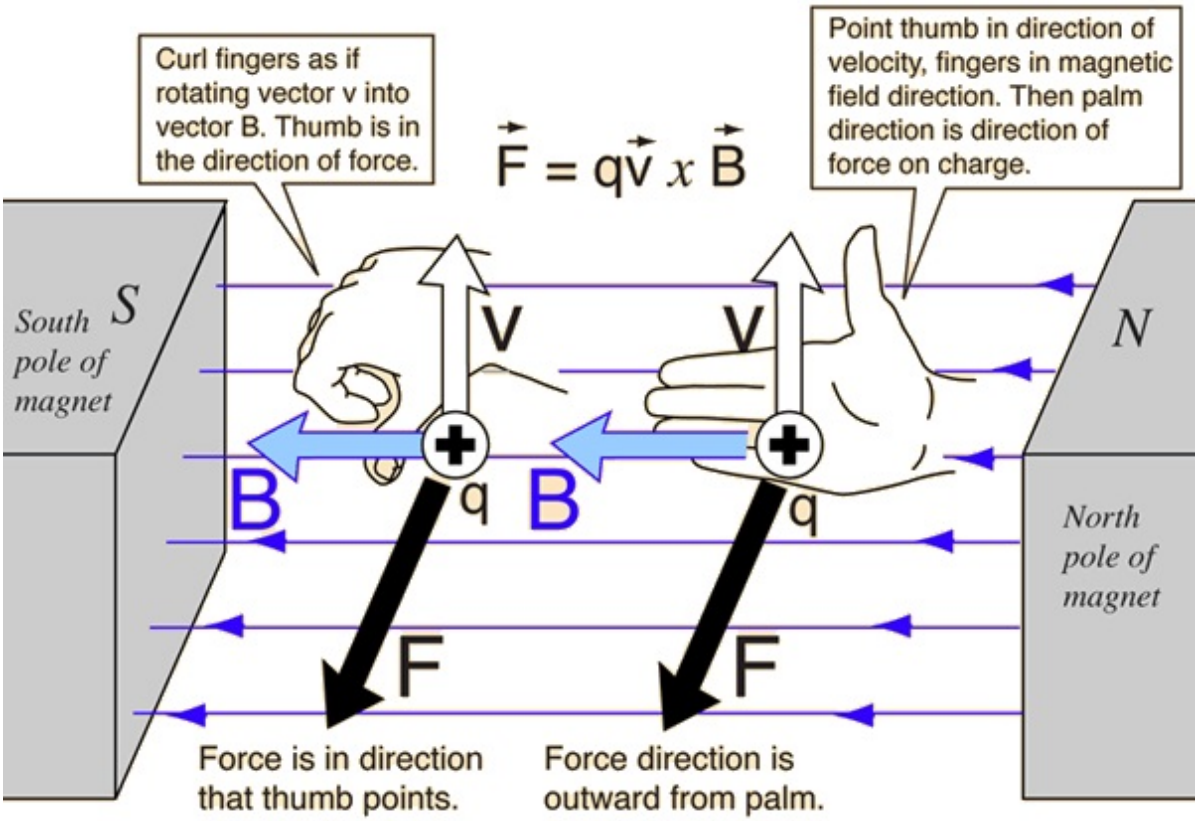


Figure 2.3: Illustration of the right hand rule, to find the direction of force in a magnetic field.

So how do brushed and brushless motors differ, does the seemingly simple difference of not using brushes carry a big impact with it? Brushed DC motors have been in commercial use since 1879. Brushless motors, on the other hand, did not become commercially viable until 1962. [2] Brushed DC motors develop a maximum torque when stationary, linearly decreasing as velocity increases. [3] Some limitations of brushed motors can be overcome by brushless motors, like higher efficiency and a lower susceptibility to mechanical wear. These benefits come at the cost of potentially less rugged, more complex, and more expensive control electronics. Which with developments do become cheaper and more widely available. Brushless motors offer several advantages over brushed DC motors, including high torque to weight ratio, more torque per watt increased reliability, reduced noise, longer lifetime elimination of ionising sparks from the commutator, and overall reduction of electromagnetic interference (EMI).

With no windings on the rotor, they are not subjected to centrifugal forces, and because the windings are supported by the housing, they can be cooled by conduction, requiring no airflow inside the motor for cooling. This in turn means that the motor's internals can be entirely enclosed and protected from dirt or other foreign matter. Brushless motor commutation can be implemented in software using a micro-controller or microprocessor computer, or may alternatively be implemented in analogue hardware, or in digital firmware using an FPGAs. Commutation with electronics instead of brushes allows for greater flexibility and capabilities not available with brushed DC motors, including speed limiting, "micro stepped" operation for slow and fine motion control, and a holding torque when stationary.

The maximum power that can be applied to a brushless motor is limited almost exclusively by heat, too much heat weakens the magnets and may damage the winding's insulation. When converting electricity into mechanical power, brushless motors are more efficient than brushed motors. This improvement is largely due to the brushless motor's velocity being determined by the frequency at which

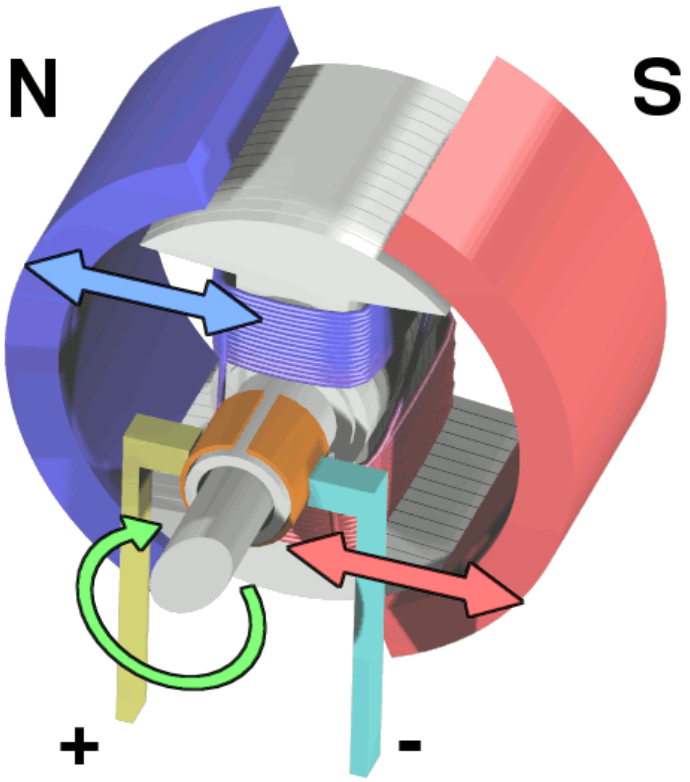


Figure 2.4: Basic structure of a dc power electric motor.

the electricity is switched, not the voltage. Additional increases, are due to the absence of brushes, which reduces mechanical energy that is lost due to friction. The enhanced efficiency is greatest in the no-load and low-load region of the motor's performance curve.

2.3.2. AC motors

An AC motor is an electric motor driven by an *alternating current* (AC). As with all motors, an AC motor will commonly consist of two basic parts: stator and rotor. The stationary stator has coils supplied with alternating current to produce a rotating magnetic field. The rotor is attached to the output shaft producing a second rotating magnetic field. The rotor magnetic field may be produced by permanent magnets, reluctance salience, or DC electrical windings. Less commonly, linear AC motors operate on similar principles as rotating motors, but have their stationary and moving parts arranged in a straight line configuration, producing linear motion instead of rotation. As stated before, two main types of AC motors are distinguished: synchronous and induction (asynchronous).

A synchronous electric motor is an AC motor in which, at steady state, the rotation of the shaft is synchronised with the frequency of the supply current; the rotation period is exactly equal to an integral number of AC cycles. Synchronous motors contain multiphase AC electromagnets on the stator of the motor that create a magnetic field which rotates in time with the oscillations of the line current. The rotor with permanent magnets or electromagnets turns in step with the stator field at the same rate and as a result, provides the second synchronised rotating magnet field of any AC motor.

Synchronous motors are available in sub-fractional sizes to high-horsepower industrial sizes. In the fractional horsepower range, most synchronous motors are used where precise constant speed is required. These machines are commonly used in analogue electric clocks, timers and other devices

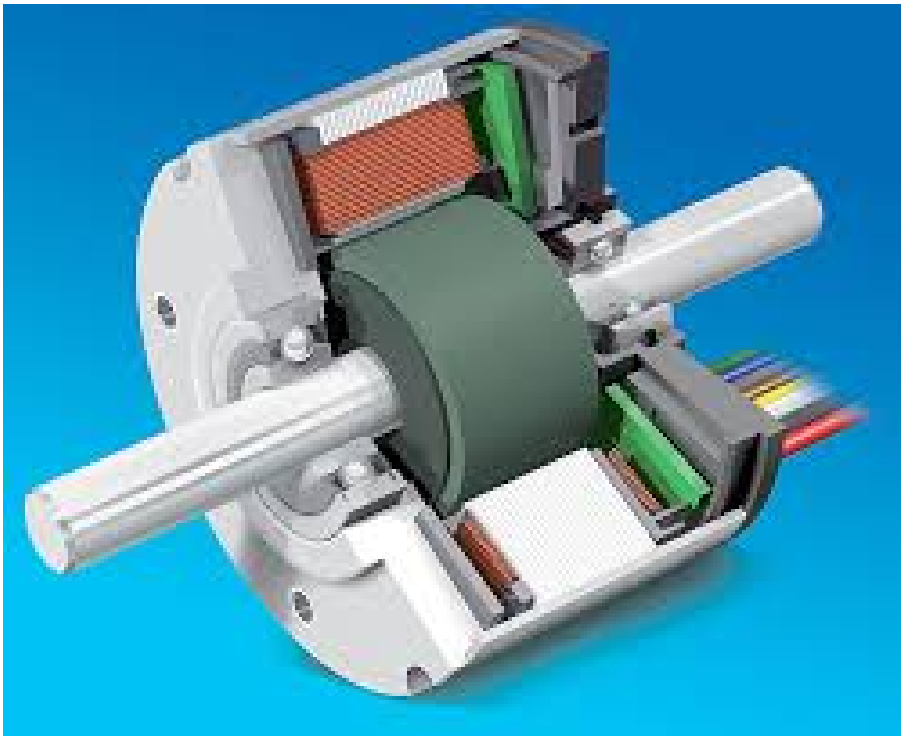


Figure 2.5: Basic structure of a synchronous motor, although we work with a permanent magnet motor the illustration uses a dc power supply in the rotor.

where correct time is required. In high-horsepower industrial sizes, the synchronous motor provides two important functions. First, it is a highly efficient means of converting AC energy to work. Second, it can operate at leading or unity power factor and thereby provide power-factor correction. Above a certain size, synchronous motors are not self-starting motors. This property is due to the inertia of the rotor; it cannot instantly follow the rotation of the magnetic field of the stator. Large motors operating on commercial power frequency include a **squirrel cage** induction winding which provides sufficient torque for acceleration and which also serves to damp oscillations in motor speed in operation. Once the rotor nears the synchronous speed, the field winding is excited, and the motor pulls into synchronisation.

An *induction* or *asynchronous* motor is an AC electric motor in which the electric current in the rotor needed to produce torque is obtained by electromagnetic induction from the magnetic field of the stator winding. An induction motor therefore does not require mechanical commutation, separate-excitation or self-excitation for all or part of the energy transferred from stator to rotor, as in universal, DC and large synchronous motors. An induction motor's rotor can be either wrapped type or squirrel-cage type. Three-phase squirrel-cage induction motors are widely used in industrial drives because they are rugged, reliable and economical. Single-phase induction motors are used extensively for smaller loads.

The synchronous motor and induction motor are the most widely used types of AC motor. The difference between the two types is that the synchronous motors rotate in exact synchronism with the line frequency. The synchronous motors do not rely on current induction to produce the rotor's magnetic field. By contrast, the induction motor requires *slip*, the rotor must rotate slightly slower than the AC current alternations, to induce current in the rotor winding. The cause of induced current in the rotor windings is the rotating stator magnetic field, so to oppose the change in rotor-winding currents the rotor will start to rotate in the direction of the rotating stator magnetic field. The rotor accelerates until the magnitude of induced rotor current and torque balances the applied load. Since rotation at synchronous speed would result in no induced rotor current, an induction motor always operates slower than its synchronous counterpart would.

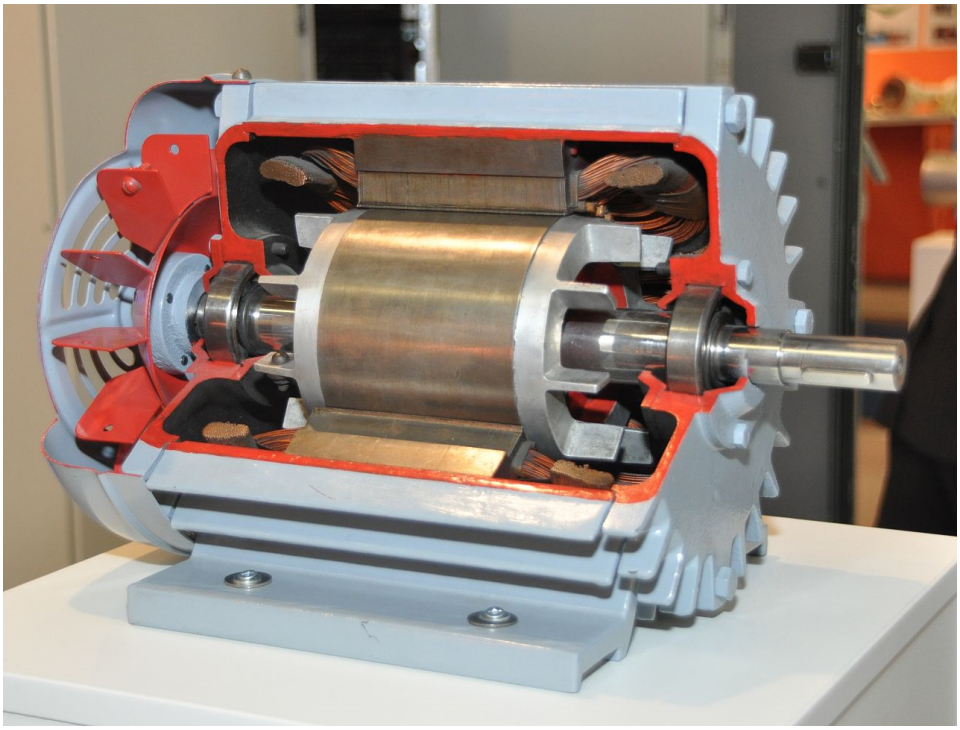


Figure 2.6: The basic structure of a Induction machine.

At this junction it has to be mentioned that the driving force behind the Generic Short Stroke Actuator is a Permanent Magnet Synchronous Motor. In the light of the above, it is known now that the motor that will be controlled is a AC synchronous motor, which either has a DC supply energising the coils making it act as a permanent magnet, or the motor simply has a surface mounted permanent magnet.

2.4. Hybrid Systems and Control

Hybrid systems theory describes both the continuous time behaviour of a system [4], described usually using differential equations and discrete part of the model which are usually software or drives. Hybrid theory has several applications in all aspects of mechanics and mechatronics.[5] To further itself hybrid system theory needs to better understand the interactions between physical processes, digital controllers and software. All the models now in use are either discrete or continuous time. This naturally leads to model inaccuracies, control shortcomings and as a result performance lower than expected. Alternative approaches consist of separating the analysis and design if the continuous and discrete parts and merging them in the final stage. Synthesis of such digital controllers encompasses PID, H_∞, IMC etc.

2.4.1. Dynamics models

A dynamic system can be described as the evolution of a *state* over *time*. The well-known simple mathematical description is:

$$\dot{x}(t) = f(x(t), u(t)) \tag{2.1}$$

The state variable at time t is given by $x(t)$. The state takes values in $\mathcal{X} \subseteq \mathbb{R}^n$ which evolve over time in $\mathcal{T} \subseteq \mathbb{R}$. The input $u(t)$, which can be either reference or command, or disturbances acting on the system, also evolves with time in $\mathcal{U} \subseteq \mathbb{R}^m$. It is possible to think of the state $x(t)$ as a summery up to time τ of all information from the past of the system that is needed in order to understand the behaviour of future states. A dynamical system can be defined as a relation between current state and an applied input and a state at a later time. For differential equation this means that we have a map ϕ from initial state, the initial time τ, the final time σ where $\tau \leq \sigma$ and a function $u : [\tau, \sigma] \to \mathcal{U}$ to the

state at time $x(\sigma)$. So:

$$x(\sigma) = \phi(\tau, \sigma, x, u) \tag{2.2}$$

Following classification of systems can be made based on:

- the state of the system
- on time
- the mechanism the drives the evolution

A hybrid system is a system with combination of discrete and continuous state and a combination of time-driven and event-driven evolution.

Time models

Continuous-time time-driven continuous-state systems were already described with equation (2.1). Their linearization has the most common form as:

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{2.3}$$

Where $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{m\times m}$ are real matrices. The transition map ϕ can be expressed in them as:

$$\phi(\tau, \sigma, x, u) = e^{A(\sigma-\tau)}x + \int_{\tau}^{\sigma} e^{A(\sigma-\theta)}Bu(\theta)\mathrm{d}\theta \tag{2.4}$$

Similar logic can be extended to time-driven discrete-time systems (sampled):

$$x(k+1) = f(x(k), u(k)) \tag{2.5}$$

where k is one time sample. Again the linear representation can be dirived as:

$$x(k+1) = \Phi x(t) + \Gamma u(t) \tag{2.6}$$

Subsequently an Automata is defined as $\sum = (\mathcal{Q}, \mathcal{U}, \phi)$ where: $mathcal Q$ a finite or countable set of discrete states, $\mathcal{U}$ a finite or coutable set of discrete inputs or the input alphabet, $\phi : \mathcal{Q} \times \mathcal{U} \to P(\mathcal{Q})$ is a partial transition function. When $\mathcal{Q}$ and $\mathcal{U}$ are finite we speak of a finite automata.

Event models

Automata or *finite state machines* are the most common models for event-driven systems. An automata is formally described as a set V for which a collection of all subsets of V is noted as $P(V)$. Additionally for two sets V and W a partial function from V to W is a mapping that is nor necessarily defined for all values of V, but only for subset D of V. Hence if f is a partial function from V to W, then there is a subset $D(f) \subseteq V$ such that f is a function from $D(f)$ to W
Subsequently an Automata is defined as $\sum = (\mathcal{Q}, \mathcal{U}, \phi)$ where:

- $\mathcal{Q}$ a finite or countable set of discrete states
- $\mathcal{U}$ a finite or coutable set of discrete inputs or the input alphabet
- $\phi : \mathcal{Q} \times \mathcal{U} \to P(\mathcal{Q})$ is a partial *transition function*

when $\mathcal{Q}$ and $\mathcal{U}$ are finite we speak of a finit automata.

2.4.2. Hybrid Models

Two hybrid system models are most important. The first ones are *Switched Systems*. They can be described using the common notation as:

$$\dot{x} = f_{q(t)}(x(t)) \tag{2.7}$$

Where $q : \mathcal{R}_+ \to \{1, ..., N\}$ is the switching signal that determines which vector field f_q, $i = 1, ..., N$ is active at time $t \in \mathcal{R}_+$. Switching may depend on time only, but may also be a function of the state $x(t)$ at time t, an external input and even have memory. In particular when the switching only depends in the state variable $x(t)$ at the present time t, one speaks of *discontinuous dynamic system* or *piecewise smooth system*. As an example:

$$\dot{x}(t) = f(x(t)) = \begin{cases} f_-(x(t)), & \text{if } \phi(x(t)) < 0 \\ f_+(x(t)), & \text{if } \phi(x(t)) > 0 \end{cases} \tag{2.8}$$

The choice of either of the two vector fields is active at any one time depends only upon the state at that time, no discrete state is necessary to describe the dynamics. Dynamic systems that switch between continuous state x and discrete state q can be described by *hybrid automata*.

Hybrid automata can be described as having two parts. The discrete part of the dynamics is modeled by means of graph whose vertices are called *modes*, and whose edges are *transitions*. The continuous state takes values in a vector spaces $\mathcal{R}$. For each mode there is a set of trajectories which are called *activities*, usually described using differential equations. Interaction between the discrete and continuous dynamics take place through *invartions* and *transitions relations*.

The frame work indicates the behaviour of the hybrid system: continuous phases separated by events art which discrete actions (re-initialization of the continuous state x anddiscrete state q) take place. It is obvious that these systems switch between many operating modes where each mode is governed by its own dynamical laws.

The bouncing ball

The most widely used example of a hybrid system is the bouncing ball. Here, the ball (thought of as a point-mass) is dropped from an initial height and bounces off the ground, dissipating its energy with each bounce. The ball exhibits continuous dynamics between each bounce, however as the ball impacts the ground, its velocity undergoes a discrete change modelled after an inelastic collision. A mathematical description of the bouncing ball follows. Let x_1 be the height of the ball and x_2 be the velocity of the ball. A hybrid system describing the ball is as follows:

When $x \in C = x_1 > 0$, flow is governed by $\dot{x_1} = x_2, x_2 = -g$ where g is the acceleration due to gravity. These equations state that when the ball is above ground, it is being drawn to the ground by gravity.

When $x \in D = x_1 = 0$, jumps are governed by $x_1^+ = x_1, x_2^+ = -\gamma x_2$, where $0 < \gamma < 1$ is a dissipation factor. This is saying that when the height of the ball is zero, which is during the impact with ground, its velocity is reversed and decreased by a factor of γ. Effectively, this describes the nature of a inelastic collision.

The bouncing ball is an especially interesting hybrid system, as it exhibits Zeno behaviour. Zeno behaviour has a strict mathematical definition, but can be described informally as the system making an infinite number of actions, in a finite amount of time (for a mathematical definition Appendix (A.2.2)). In this example, each time the ball bounces it loses energy, making the subsequent jumps (impacts with the ground) closer and closer together in time.

2.4.3. Hybrid Control for PSMS

As hybrid control is a new field in control theory, not a lot of research can be found on the application of hybrid control for permanent magnet synchronous motors. The research so far was focused on either speed or torque control, with very little done on position control.

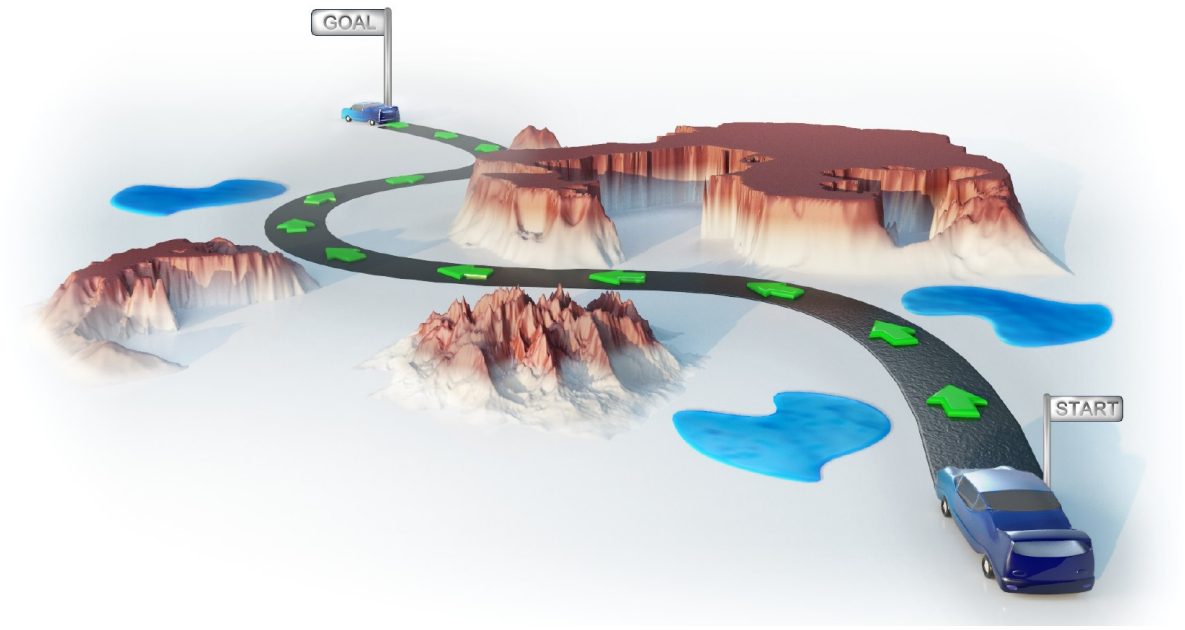


Figure 2.7: Car driving as a optimal control problem.

Within hybrid control two main subcategories can be distinguished. The first is the already mentioned switched systems. Examples of switched control can be found in Florent et al. [6–8]. The focus of the research was switching between two torque controllers, so as to take into account the dynamics of the power inverter. The second one is Sliding Mode Control. Following Filippov solution, two systems never fully switch but remain in the region between the two subsystems, effectively "slid" along it. Thus we obtained so called *sliding mode*. For more on sliding modes the reader is referred to Appendix (A.2.1) An example of sliding mode control used for speed and position control of pmsm is presented in [9, 10], and in [11] sliding mode control is compered against a neuro-fuzzy controller.

The major obstacle is that none of the mentioned papers deals with limits or considers the PMSM as a driving force of an actuator, leaving out the essential change in dynamics due to the kinematic transformation that takes place between a linear and angular axis.

2.5. Optimal Control

Optimal control theory is widely thought of as an extension of the calculus of variations. Calculus of variations is a field of mathematics that deals with finding a maximum or minimum of a function. Optimal control uses mathematical optimisation to find optimal control laws for dynamic systems. The method is largely accredited to the work of Lev Pontryagin and Richard Bellman in the 1950s. Optimal control is a sub-field within the filed of classical control theory. [12–14]

Optimal control at its core, deals with the problem of finding a control law for dynamic system such that a certain minima is found, which satisfies optimality criterion. A control problem always includes a cost functional, that is a function that is a combination of system state, system outputs and control variables. An optimal control is a set of differential equations describing the evolution of system states and the influence of control variables, that minimize the cost functional. The optimal control can be derived using *Pontryagin's maximum principle* (a necessary condition also known as Pontryagin's minimum principle), or by solving the *Hamilton–Jacobi–Bellman* equation (a sufficient condition).

The most common example of a Optimal control problem is driving a car. The driver chooses to drive in such fashion as to optimise a objective: the travel time, the travel distance, the travel safety. This objective comes with limitations: maximum speed of the car, speed limits, a minimum distance that has to be travelled, amount of petrol used. Figure (2.7) gives a visualisation.

A proper cost functional is a mathematical expression which gives the travelling time as a function of the speed, geometrical considerations, and initial conditions of the system. It is often the case that the constraints are interchangeable with the cost function. In general terms a cost function can be expressed as follows:

$$J = \Phi\left(x(t_0), t_0, x(t_f), t_f\right) + \int_{t_0}^{t_f} \mathcal{L}\left(x(t), u(t), y(t)\right) dt$$

which can be subject to equality constraints:

$$h(x, u, y) = \mathcal{C}$$

or inequality constraints:

$$h(x, u, y) < \mathcal{K}$$

where $x(t)$ is a state vector, $u(t)$ is the control variable vector, $y(t)$ is a output vector, $\mathcal{C}$ is a set of equality constraints, and $\mathcal{K}$ is a set of inequality constraints.

At this point it is important to empathise, from classical optimisation theory, that unless a function is convex, no global minima exists and any solution found is only locally optimal.

2.5.1. Optimal Control for PMSM

Many examples of optimal control strategies for PMSM can be found. Most of them deal with either current control or torque control. All applications exploit the structure of the controller and usually focus on issues with the power inverter. Different cost functions are also used depending on the approach. For examples on optimal control for PMSM the reader is proposed [15–18].

The problem with the explored methods is, that they make use of their own structure in many cases. Limits are also only explicitly approach in only one example.

3

Plant and Model

3.1. Generic Short Stroke Actuator

The *Generic Short Stroke Actuator* (throughout the rest off this work we refer to it as GSSA) is an upgrade on the Short Stroke Actuator. The purpose of the upgrade was of to make the actuator more generic and applicable to a wider range of applications. Thus, the GSSA became part of Moog's family of Linear Electric Actuators, for short stroke applications. The main uses of GSSA are in plastic machinery, metal forming machinery, test and simulation, and power generation. GSSA belongs to family of electromechanical actuators, which for Moog is a relatively new technology in comparison to the predominantly used hydraulic actuators. Subsequently a significantly bigger amount of research is still going into it.

3.1.1. What is a GSSA?

In the previous chapter the permanent magnet synchronous motor was described. Now we would like to go more into detail of what makes the GSSA actuator, and discuss the other parts that make it. Additionally, dynamic equations are used to describe mathematically the physical concepts described prior.

The GSSA consists of four main parts: permanent magnet synchronous motor, scale up gear box, slider-crank system which transforms the rotational movement of the electric motor to linear motion, and a spring with a load attached to it. Figure (3.1) presents how the whole of GSSA operates as one actuator. We would like to describe each of those parts separately next.

Permanent Magnet Synchronous Motor

The GSSA was already established to be driven by a PMSM. the synchronous motor used is a true surface mounted permanent magnet (we refer to it as PM), rather then a dc supplied circuit. Permanent magnets, which are mounted on the surface of the rotor, appear magnetically *round*. Therefore, the motor torque is the result of the reactive force between a magnet on the rotor and the electromagnet on the stator. Compared with DC, induction, and synchronous reluctance machines; PMSM have a lower weight-to-torque and weight-to-power ratio, higher efficiency, excellent controllability in the full torque-speed operating envelope, ruggedness and reliability, and lower cost of maintenance.

The *permanent magnet synchronous motor* can be thought of as a cross between an AC induction motor (ACIM) and a brushless DC motor. Sometimes PMSM is called a brushless DC motor due to similarity in torque-speed characteristics. They have rotor structures similar to BLDC motors which contain permanent magnets. However, their stator structure resembles that of its ACIM cousin, where the windings are constructed in such a way as to produce a sinusoidal flux density in the airgap of the machine. As a result, they perform best when driven by sinusoidal waveforms. However, unlike their ACIM relatives, PMSM motors perform poorly with open-loop scalar V/Hz control, since there is no rotor coil to provide mechanical damping in transient conditions. Field Oriented Control is the most popular control technique used with PMSMs. As a result, torque ripple can be extremely low, on par with that of ACIMs. However, PMSM motors provide higher power density for their size compared to ACIMs. This

is because with an induction machine, part of the stator current is required to *induce* rotor current in order to produce rotor flux. These additional currents generate heat within the motor. However, the rotor flux is already established in a PMSM by the permanent magnets on the rotor.[19]

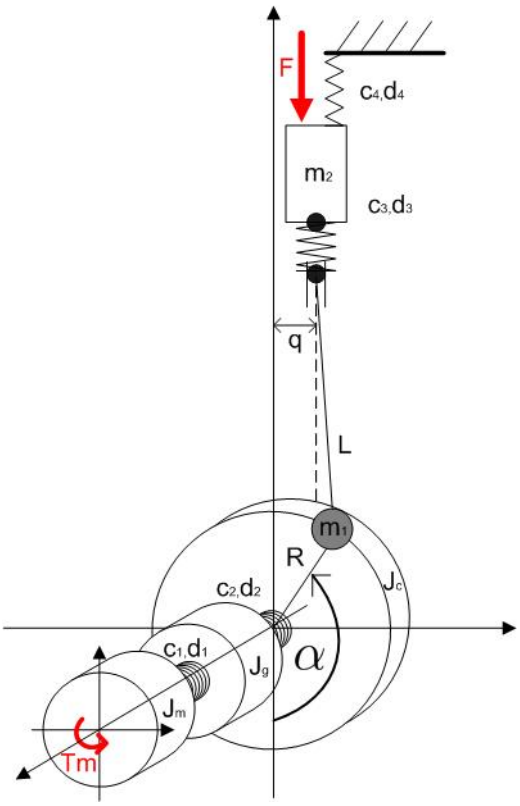


Figure 3.1: Individual parts of the GSSA. A PMSM connected to a gearbox with a crank-shaft at the end.

As mentioned before, synchronous motors can move with a synchronous or constant speed when at steady state. This is achieved by placing a constant magnetic field in a Rotating Magnetic Field (or RMF). In a PMSM the RMF is achieved by an ac supply, which usually has three phases. By applying sinusoidal wave forms, we can rotate an RMF at a constant speed based on a frequency of the ac signal and the number of poles the motor has. Figure (3.2) illustrates the working principle. If a PM is placed in such RMF and given an initial speed, the poles of the PM will *lock* with the poles of the RMF. This will result in the PM turning at the same speed as the RMF. The issue is giving and initial speed to the PM, because of that PMSMs are not self-starting. Those issues can be easily resolved thought, as described in Section 2.3.1.

Step-up Gearbox

A transmission is a machine that consists of a power source and a power transmission system, which provides controlled application of the power. Often the term transmission refers simply to the gearbox that uses gears and gear trains to provide speed and torque conversions from a rotating power source to another device. The most common use is in motor vehicles, where the transmission adapts the output of the internal combustion engine to the drive wheels. Such engines need to operate at a relatively high rotational speed, which is inappropriate for starting, stopping, and slower travel. The transmission reduces the higher engine speed to the slower wheel speed, increasing torque in the process. We would like to do the same just with an electromechenical actuator.

Figure (3.3) illustrates and example of a simple gearbox. For an ideal gearbox $P_{in} = P_{out}$ at *steady state*. An *ideal* gearbox is one that is both lossless **and** stores no kinetic energy. Emphasis put on steady state as the gearbox can also store energy as rotational kinetic energy in its wheels and other

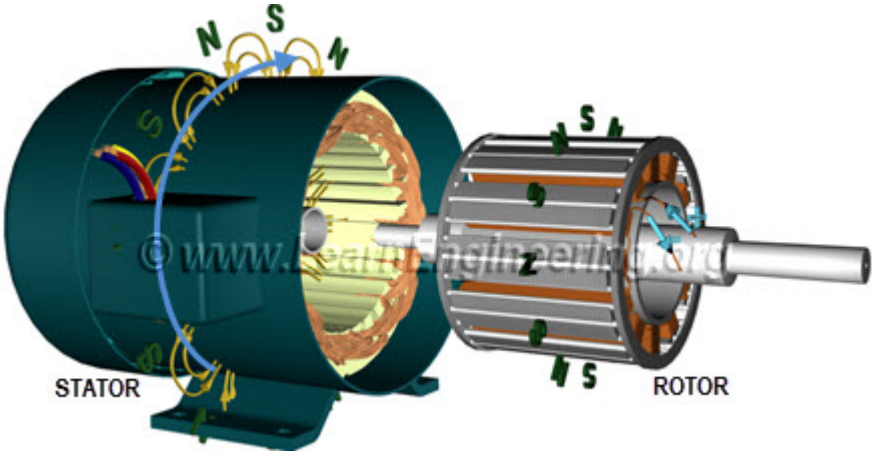


Figure 3.2: Synchronous speed rotors. A rotating magnetic field locks with the static field of a PM.

spinning parts. This statement assumes that energy stored by the gearbox is insignificantly small compared to the energy transfers through the gearbox. Therefore, we can assume that the power equation holds and that we are dealing with an ideal gearbox.

With the above established we can safely say that $\omega_{out} = Tr^{-1}\omega_{in}$, where $Tr > 1$ is the gear ratio. Meaning that the input spins faster then the output. Subsequently torques are related by the reciprocal ratio $\tau_{out} = Tr\tau_{in}$. As the equation for torque is $\tau = I\dot{\omega}$, if substituted into the torque equality we have $Tr\tau_{in} = Tr^{-1}I\dot{\omega}_{in}$, which becomes

$$\tau_{in} = Tr^{-2}I\dot{\omega}_{in}$$

So now, if we think of the system as a *black box*: as seen from the input, the load responds to torque as though there were no gearbox there, but now with moment of inertia $I_{tot} = Tr^{-2}I$. normally the inertia of a gearbox is also important and therefor the equation of inertia with a gearbox is:

$$I_{tot} = Tr^{-2}I + I_{gearbox} \tag{3.1}$$

A gearbox can also be a step down, were the torque is reduced and the output shaft spins faster then the input shaft. In such case all the equation presented just need to be reversed.

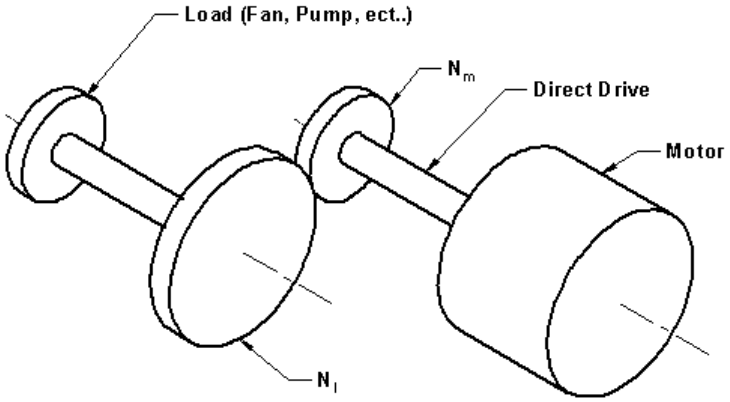


Figure 3.3: Step up ideal gearbox.

Crank-Shaft

A crankshaft is a mechanical part able to perform a conversion between rotational and linear motion. An illustrative example is a reciprocating engine, it translates reciprocating motion of the piston into

rotational motion; whereas in a reciprocating compressor, it converts the rotational motion into reciprocating motion. In order to do the conversion between two motions, the crankshaft has *crank throws* or *crankpins*, additional bearing surfaces whose axis is offset from that of the crank, to which the *big ends* of the connecting rods from each cylinder attach. The crankshaft we worked with is presented in Figure (3.1). It works on opposite principle where a rotational movement is converted into linear movement of a mass.

3.1.2. Mathematical Model and Equations

With the physical structure of GSSA established; in order to be able to design the best possible controller, a either liner or non-linear a model in either time or frequency domain is needed. To establish such model, we decided to use ordinary differential equations to develop a state space (in the thesis abbreviated as SS) model.[20] From above sections it is easy to say intuitively that the model must have to main parts. An electrical due to the electric circuitry in the motor, and mechanical due to the spinning shaft of the motor and the moving load. Additionally the dynamic of both the gearbox and the crank-shaft need to be taken into account for a complete model.

The PMSM operates, like most synchronous motors, on three-phase ac current. The currents are in three axes: a, b and c. Those currents are called the *balanced* set of phase currents. Control of PMSMs is done in the *d-q frame* though, where *direct* axis is allied with the magnetic field of the PM. The *quadrature* axis is positioned at 90^o *electric* degrees to the other axis and in line with the motor shaft. As the control is done in the d-q frame the modelling also needs to be in the same frame. In order to connect abc frame with d-q frame one can use the well known *Park's Transformation*. [1] Park's transformation enables the switch go between abc frame and d-q frame, and vice versa as follows:

$$\begin{bmatrix} d \\ q \\ 0 \end{bmatrix} = T \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{3.2}$$

where T is:

$$T = \frac{2}{3} \begin{bmatrix} \cos\theta & \cos\theta - \frac{2\pi}{3} & \cos\theta + \frac{2\pi}{3} \\ -\sin\theta & -\sin\theta - \frac{2\pi}{3} & -\sin\theta + \frac{2\pi}{3} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{3.3}$$

With park's transformation in place, the focus can be on deriving the dynamic equations in the d-q frame as switching between d-q and abc frame becomes trivial.

In order to establish the dynamical current equations, *Kirchhoff's second (Voltage)* rule applies: "The sum of the emfs in any closed loop is equivalent to the sum of the potential drops in that loop". Subsequently the sum of emfs around any given loops can be summed up to:

$$V_{in} = V_{inductance} + V_{resistance} \tag{3.4}$$

The motor also experiences a back EMF effect (BEMF for short). The term BEMF, is most commonly used to refer to the voltage that occurs in electric motors where there is relative motion between the armature of the motor and the magnetic field from the motor's field magnets, or windings.[21] From Faraday's law, the voltage is proportional to the magnetic field, length of wire in the armature, and the speed of the motor. This effect is not due to the motor's inductance and is a completely separate effect. Regardless the sum of emfs with the BEMF taken into account becomes:

$$V_{in} = V_{inductance} + V_{resistance} + V_{bemf}. \tag{3.5}$$

If the values of potential droops around the loop and separate the d and the q axis are substituted for, the following dynamic equations are achieved:

$$\begin{aligned} V_d &= L_d \dot{i_d} + R i_d + L_q \frac{p}{2} \dot{\theta} i_q \\ V_q &= L_q \dot{i_q} + R i_q - L_d \frac{p}{2} \dot{\theta} i_d + p\lambda\dot{\theta} \end{aligned} \tag{3.6}$$

were V_d and V_q are voltages supplied in d and q axis respectively. Similarly L_d, L_q, i_d and i_q represent individual inductance and currents. Resistance R is constant throughout the circuit. BEMF is generally a non-linear effect, but it can be assumed linear for simplicity of the model and its value is exactly proportional to the flux created by the PM, λ. It is worth reminding at this point that as the PM for the stator is used, it appears magnetically "round", and the motor torque is the result of the reactive force between the magnets on the rotor and the electromagnets of the stator. This results in the optimum torque angle being 90 degrees, which is obtained by regulating the d-axis current to zero in a typical FOC application. Therefor in the future we will assume $i_d = 0$. The reasoning is explained later when deriving the linear open loop function of the system.

With the electric part of the PMSM established the spinning crank of the motor can be described next. To derive that equation, Newton's Law of Conservation of Momentum applies: "The vector sum of the external forces F on an object is equal to the inertia of that object multiplied by the acceleration vector of the object". Subsequently the sum of themomentum around the motor are:

$$\tau_{motor} = \tau_{crank} + \tau_{load} + F_{friction} \tag{3.7}$$

were τ_{motor} is the electric torque produced by the motor, and $F_{friction}$ are the frictional losses in the PMSM and the gearbox. If the torque and force values are substitute for, the non-linear equation is as follows:

$$\frac{3p}{2}[\lambda i_q + (L_d - L_q) i_q i_q] = J_t \ddot{\theta} + (b_m + b_g)\dot{\theta} + \tau_{load} \tag{3.8}$$

were λ is again the flux due to the PM, b_m and b_g are friction constants for the motor and the gearbox respectively. Friction is a highly non-linear phenomenon, but again as in the case of BEMF for the purposes of simplicity it is assumed to be a constant. The J_t term stands for the total in the system. As discussed already in the section about the gearbox (see equation 3.1), the total inertia of the system with the gearbox affecting the load can be found as:

$$J_t = J_{motor} + J_{gearbox} + \frac{J_{load}}{Tr^2} \tag{3.9}$$

were J_{motor} and $J_{gearbox}$ are the inertias of the crank and gearbox respectively. The load inertia is J_{load} and Tr stands for transmission ratio, or simply gearbox ratio. The same simplification can be done with the viscous friction in the system, subsequently $b_a = b_g + b_m$. Similarly the flux due to PM can be related to both the BEMF effect and the torque produced by the motor. Subsequently,a new constant is introduce to have a greater clarity in the mathematical model. The BEMF constant is $K_e = p\lambda$ and the torque constant is $K_t = \frac{3p}{2}\lambda$.

The whole system is put together in a more standard notation for *state space* representation. More information on SS form for a PMSM can be found in [21–26]. Subsequently the equations are found to be:

$$\begin{aligned}
\dot{i_d} &= \frac{V_d}{L_d} - \frac{R}{L_d} i_d - \frac{L_q}{L_d}\frac{p}{2}\dot{\theta} i_q \\
\dot{i_q} &= \frac{V_q}{L_q} - \frac{R}{L_q} i_q + \frac{L_d}{L_q}\frac{p}{2}\dot{\theta} i_d - \frac{K_e}{L_q}\dot{\theta} \\
\ddot{\theta} &= \frac{3p}{2}\frac{[(L_d - L_q) i_q i_q]}{J_t} + \frac{K_t}{J_t} i_q - \frac{b_a}{J_t}\dot{\theta} - \frac{\tau_{load}}{J_t}.
\end{aligned} \tag{3.10}$$

As it is easy to tell the model must be a, at least, 4th order system. the sates of the system are clearly i_d, i_q, θ and $\dot{\theta}$. Which are the currents in the d-q frames, and the angular position and velocity of the crank of the PMSM. The system becomes simplified to some extend when it is taken into consideration that $i_d = 0$.

As presented in Figure (3.4) it can be seen that the mass is connected to the end of the crank with a spring, and then further stabilised with another spring. The connecting spring has a certain stiffness and damping factor, where the stabilising spring has small stiffness and negligible damping. Subsequently, the system can be thought as a *two degree of freedom spring-mass system*, where the masses are

connected to each other through a *spring-damper*, the first mass has a certain viscous damping factor, and the second mass is connected to a stationary surface through a spring. In thisparticular case only the first mass is actuated (by the electric motor producing torque). In order to included the dynamics of the *spring-mass*, we need to extend the system by additional two states. Normally such two degrees of freedom system would be modelled as [27]:

$$m_1\ddot{x}_1 + (c_1 + c_2)\dot{x}_1 - c_2\dot{x}_2 + k_2x_1 - k_2x_2 = F$$
$$m_2\ddot{x}_2 - c_2\dot{x}_1 + c_2\dot{x}_2 - k_2x_2 + (k_2 + k_3)x_2 = 0$$

where in the case of the given system F is the driving force, and x_2 and $\dot{x}_2$ are the the position and velocity of the second, unactuated mass. In order to adapt the above equations to the case of the GSSA with a mass connected to it we need to make a few substitutions. The position of the first mass is the linear position of the rod of the GSSA, which is related to the angle as $x_1 = f(\theta)$ and the velocity is related as $\dot{x}_1 = f(\dot{\theta})$. We discuss those functions shortly. The dampers are $c_1 = b_a$ and $c_2 = b_l$, the springs are $k_2 = k_b$ and $k_3 = k_s$. In the case of masses, angular inertia is the first mass $m_1 = J_a$ and the second mass is the load attached atop GSSA $m_2 = M$. The force that drives the first mass becomes the motor torque, $F = \tau$.

If the equations of the PMSM (3.10) are combined with equations for a two degree of freedom spring-mass system the full dynamic description becomes:

$$\begin{aligned}
\dot{i}_d &= \frac{V_d}{L_d} - \frac{R}{L_d}i_d - \frac{L_q}{L_d}\frac{p}{2}\dot{\theta}i_q \\
\dot{i}_q &= \frac{V_q}{L_q} - \frac{R}{L_q}i_q + \frac{L_d}{L_q}\frac{p}{2}\dot{\theta}i_d - \frac{K_e}{L_q}\dot{\theta} \\
\ddot{\theta} &= \frac{3p}{2}\frac{[(L_d - L_q)i_q i_q]}{J_a} + \frac{K_t}{J_a}i_q - \frac{b_a}{J_a}\dot{\theta} - \frac{b_a + b_l}{J_m}f(\dot{\theta}) + \frac{b_l}{J_m}\dot{x} - \frac{k_b}{J_m}f(\theta) + \frac{k_b}{J_m}x - \frac{\tau_{load}}{J_m} \\
\ddot{x} &= \frac{b_l}{M}f(\dot{\theta}) - \frac{b_l}{M}\dot{x} + \frac{k_b}{M}f(\theta) - \frac{k_b + k_s}{M}x.
\end{aligned} \tag{3.11}$$

where b_l and k are the damping factor and stiffness introduced by the spring, M is the mass of the load, and $x, \dot{x}, \ddot{x}$ refer to the linear position, velocity and acceleration, respectably. It is important to not at this juncture that $f(\theta) = constant \times \theta$ and $f(\dot{\theta}) = constant \times \dot{\theta}$ under linearization of the model. Important to mention at this point is the change from J_t to J_m. Term J_m refers to the combined inertia of the motor with the gearbox. This true, because in equations (3.11) two masses are present which need to be separately. When considering dynamics of the angular velocity of the motor (3.10), the whole weight moved by the motor is considered. That constitutes both the weight of the motor crank as well as the shaft, spring and lead combined.

With the dynamics of the system firmly establish, all that is left is to find the kinematic translation from the angular movement of the PMSM into linear movement of the load placed at the end of the crank-shaft system, as Figure (3.1) reminds us. First, it is noted that:

$$x = l\cos(\beta) - r\cos(\theta) \tag{3.12}$$

where x is the linear position of the mass. The offset q can be found as

$$q = l\sin(\beta) - r\cos(\theta)$$

Simple rearranging allows to find the angle β, between the leaver and the third virtual spring

$$\sin(\beta) = \frac{r\sin(\theta) - q}{l}$$

Substituting for $\sin(\beta)^2$ in equation (3.12) and using trigonometric identities, the transformation form angular motion to linear motion is described as:

$$x_a = l\sqrt{1 - \left(\frac{r\sin(\theta) - q}{l}\right)^2} - r\cos(\theta) \tag{3.13}$$

By differentiating equation (3.13) the relation between angular and linear velocity is found. Unsurprisingly, the two velocities are connected through a non-linear function of the following form:

$$\dot{x}_a = \frac{r\sin(\beta-\theta)}{\cos(\beta)}\dot{\theta} = \left(\frac{r\sin(\beta)\cos(\theta)}{\cos(\beta)} - r\sin(\theta)\right)\dot{\theta}$$

$$\therefore \dot{x} = K_\theta(\theta)\dot{\theta} \tag{3.14}$$

Where

$$K_\theta(\theta) = r\sin(\theta) - \frac{r^2\sin(\theta)\cos(\theta) - rq\cos(\theta)}{l\sqrt{1-\left(\frac{r\sin(\theta)-q}{l}\right)^2}}. \tag{3.15}$$

Now if the same logic is followed as before with one further differentiation kinematic transformation between angular and linear acceleration is found to be:

$$\therefore \ddot{x}_a = K_{\dot{\theta}}(\theta)\ddot{\theta}. \tag{3.16}$$

Where

$$\begin{aligned} K_{\dot{\theta}} = &\left(r\sin\theta + \frac{r^2\sin(\theta)\cos(\theta) - rq\cos(\theta)}{l\sqrt{1-\left(\frac{r\sin(\theta)-q}{l}\right)^2}}\right) + \\ &\dot{\theta}\left(r\cos\theta - \frac{r^2\cos\theta^2 + r^2\sin\theta^2 - rq\sin\theta}{l\sqrt{1-\left(\frac{r\sin(\theta)-q}{l}\right)^2}} - \frac{r^4\cos\theta^2(r\sin\theta-q)^2}{\left(l\sqrt{1-\left(\frac{r\sin(\theta)-q}{l}\right)^2}\right)^3}\right) \end{aligned} \tag{3.17}$$

With the position, velocity and acceleration of the load established, the last desired knowledge is about the force acting on the load.

From the dynamics equations (3.11) and using the established kinematic transformations, force acting on the load is found to be:

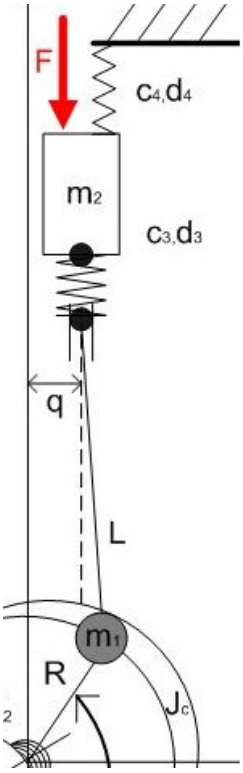
$$F = M\ddot{x} = b_l\dot{x}_a - b_lM\dot{x} + k_bMx_a - k_b + k_sMx \tag{3.18}$$

where $\dot{x}_a$ and x_a can be found using equations (3.14) and (3.13), respectably. The above follows again form the idea, that the mass atop the shaft is a two mass system. Again Figure (3.4) for reference. For such am system the force acting on it can be found by considering the forces in the spring and the dumper, due to the relative position of the two masses to each other. This is a improvement on the current approach, which approximated force on the load as the internal forces of the connecting spring. It was felt that the current approach is to simple and using a full description, with the data provided, achieves a more accurate approximated. Despite the increased complexity of calculation.

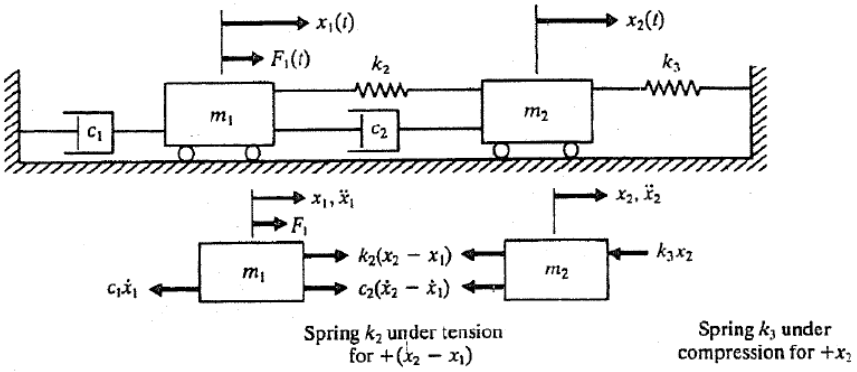
3.1.3. Internal control, MTC and Full Model

In previous section we established and discussed the GSSA. What parts make it and we proposed a dynamic-kinematic model for the plant. Now we would like to discuss the Moog Servo Drive and the subsequent mathematical model the arises from it.

A common practise is to control PMSMs with cascade control. [28, 29] Which is often used when there are several measurement signals and only one control signal. As seen in most works, the secondary loop is the current loop. The electric part of an electric motor operating fastest, this loop understandably has the highest bandwidth. The current loop is put in series with the primary velocity loop which has a lower bandwidth, and which controls the angular velocity of the crank shaft of the motor. In series with those two loops in most cases a position loop is implemented. In this project the differences from standard practise comes in two ways. First the most outer loop uses the linear position of the load as feedback rather than angle of the crank as it is usually done. Second the position



(a) Crank-Shaft



(b) Two mass system

Figure 3.4: Kinematic transformation of the crank-shaft. System modelled as one mass driving another.

loop is interchangeable with the force loop. This was discussed in the section 3.1.2 and illustrated in Figure (3.5).

As is illustrated in Figure (3.5), the first step is to derive the dynamic equations of the current loop. The feedback loop is made of PID type controller in series with the electrical circuits of the PMSM. PID control is by far the most common way of using feedback in natural and man-made systems. PID controllers are commonly used in industry and a large factory may have thousands of them, in instruments and laboratory equipment. In engineering applications, the controllers appear in many different forms: as a standalone controller, as part of hierarchical, distributed control systems, or built into embedded components. Most controllers do not use derivative action as it amplifies unwanted noise(Further discussion on PI control can be found in the section (4.2.1)).

Subsequently Moog is also using a PI controller for the internal current loop. PI controllers are linear controllers,in order to be put in seires with the system, the nonlinear dynamic equations (3.11) need to be linearised. In order to achieve a linear system of the current equations the Jacobian with respect to the states was found. The system additionally has to be linearised around an equilibrium point. For the purposes of simplicity and inability to find a more appropriate linearization standard practise is followed and zero is chosen as an equilibrium. This results in the dynamic matrix A to be:

$$\begin{bmatrix} \dot{i_d} \\ \dot{i_q} \end{bmatrix} = A_i = \begin{bmatrix} \frac{-R}{L_d} & \frac{-L_q \theta p}{2L_d} \\ \frac{L_d \theta p}{2L_q} & \frac{-R}{L_q} \end{bmatrix}_{(0,0,0)} = \begin{bmatrix} \frac{-R}{L_d} & 0 \\ 0 & \frac{-R}{L_q} \end{bmatrix} \begin{bmatrix} i_d \\ i_q \end{bmatrix} \tag{3.19}$$

Similarly finding the Jacobian matrix with respect to the inputs gives the input matrix B:

$$B_i = \begin{bmatrix} \frac{1}{L_d} & 0 \\ 0 & \frac{1}{L_q} \end{bmatrix}. \tag{3.20}$$

As the output of the current loop are the states of the loop, the output matrix C becomes just identity

and with no feedthrough term matrix D is zero:

$$C_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad D_i = 0. \tag{3.21}$$

The exact structure of the PI controller was unknown and initial investigation did not render and promising results (see section 3.1.4), is was decided to approximate the PI controller as a first order lag filter. Subsequently the *transfer function* (TF) can be represented in the following form:

$$PI_{current} = \frac{sci_1 + ci_2}{sci_3 + ci_4} \tag{3.22}$$

were ci_x is a constant of the TF. To put the filter in series with the given state space system, the TF is transferred into SS. This allows us to put the two systems in series. Two SS systems can be connected in series in the following way.[30] The new matrix A is:

$$A_{new} = \begin{bmatrix} A_1 & 0 \\ B_2C_1 & A_2 \end{bmatrix} \tag{3.23}$$

subsequently the B, C and D matrices can be found as:

$$B_{new} = \begin{bmatrix} B_1 \\ B_2D_1 \end{bmatrix} \quad C_{new} = \begin{bmatrix} D_2C_1 & C_2 \end{bmatrix} \quad D_{new} = D_2D_1. \tag{3.24}$$

If the proposed method is used, then the series interconnection between the electric circuits and PI controller can established, as seen in Figure (3.5). It is assumed that i_d and i_q axes are controlled by a separate PI controller, although both PI controllers are assumed to be the same. Subsequently, two feedback loops and the system in series become:

$$A_{ser-i} = \begin{bmatrix} \frac{-ci_4}{c_3 i} & 0 & 0 & 0 \\ 0 & \frac{-ci_4}{c_3 i} & 0 & 0 \\ \frac{-ci_1 ci_4 - ci_2 ci_3}{L_d ci_3^2} & 0 & \frac{-R}{L_d} & 0 \\ 0 & \frac{-ci_1 ci_4 - ci_2 ci_3}{L_q ci_3^2} & 0 & \frac{-R}{L_q} \end{bmatrix}. \tag{3.25}$$

It can be noticed that the current loop additional states are the error between reference and feedback in the d and q axes, e_{id}, e_{iq}. The additional states represent Subsequently following from equation (3.25) and using equations (3.24), the subsequent system becomes as follows:

$$B_{ser-i} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \frac{c1_i}{L_d ci_3} & 0 \\ 0 & \frac{c1_i}{L_q ci_3} \end{bmatrix} \quad C_{ser-i} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad D_{ser-i} = 0. \tag{3.26}$$

Lastly, to complete the current set of equations, the influence of the feedback loop on the dynamics of the system, needs to be included in the dynamic equations. The standard equation resulting from a feedback interconnection is found as:

$$H_{feedback} = \frac{Forward}{1 + Loop}. \tag{3.27}$$

At this point it the feedback is assumed to be a negative unity feedback. Therefore for a SS system the feedback interconnection results in a new A matrix as follows:

$$A_{loop-i} = \left[A_{ser-i} - C_{ser-i} * I * B_{ser-i} \right] \tag{3.28}$$

if the numbers are filled, the A matrix of the current feedback loop becomes:

$$A_{loop-i} = \begin{bmatrix} \frac{-ci_4}{c_3 i} & 0 & -1 & 0 \\ 0 & \frac{-ci_4}{c_3 i} & 0 & -1 \\ \frac{-ci_1 ci_4 - ci_2 ci_3}{L_d ci_3^2} & 0 & \frac{-R}{L_d} - \frac{ci_1}{L_d ci_3} & 0 \\ 0 & \frac{-ci_1 ci_4 - ci_2 ci_3}{L_q ci_3^2} & 0 & \frac{-R}{L_q} - \frac{ci_1}{L_q ci_3} \end{bmatrix}. \tag{3.29}$$

This dives the complete set of dynamic equations for the current loop.

Referring back to Figure (3.5) it is seen that with the current feedback loop established, velocity feedback loop in found in the same fashion. The first step is to combine the established current loop, with momentum equations of the crank.

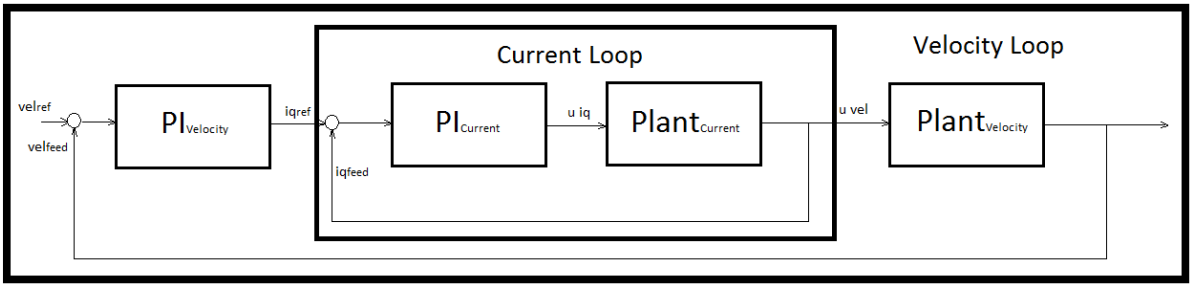


Figure 3.5: The inner feedback loop ensures desired current behaviour and the outer loop is thevelocity loop.

Therefore, first the dynamic nonlinear equations of PMSM's crank are linearised around steady state point at origin. Which gives the matrix A as:

$$\begin{bmatrix}\dot{\theta}\\ \ddot{\theta}\end{bmatrix} = A_v = \begin{bmatrix}0 & 1\\ 0 & -\frac{b_m+b_g}{J_t}\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\end{bmatrix} \tag{3.30}$$

with the currents in d and q axes acting as inputs the remaining matrices look as follows:

$$B_v = \begin{bmatrix}0 & 0\\ 0 & \frac{K_t}{J_t}\end{bmatrix} \quad C_v = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} \quad D_v = 0. \tag{3.31}$$

this gives the velocity sub-system, with θ and $\dot{\theta}$ as states.

If the crank dynamics are included into the current loop. This rendered the following dynamic A matrix:

$$A_{sys} = \begin{bmatrix} \frac{-ci_4}{c_3 i} & 0 & -1 & 0 & 0 & 0\\ 0 & \frac{-ci_4}{c_3 i} & 0 & -1 & 0 & 0\\ \frac{-ci_1 ci_4 - ci_2 ci_3}{L_d ci_3^2} & 0 & \frac{-R}{L_d} - \frac{ci_1}{L_d ci_3} & 0 & 0 & 0\\ 0 & \frac{-ci_1 ci_4 - ci_2 ci_3}{L_q ci_3^2} & 0 & \frac{-R}{L_q} - \frac{ci_1}{L_q ci_3} & 0 & \frac{-K_e}{L_q}\\ 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & \frac{K_t}{J_t} & 0 & -\frac{b_a}{J_t} \end{bmatrix}. \tag{3.32}$$

Again as with the current loop the rest of the matrices can be found as:

$$B_{sys} = \begin{bmatrix} 1 & 0\\ 0 & 1\\ \frac{c1_i}{L_d ci_3} & 0\\ 0 & \frac{c1_i}{L_q ci_3}\\ 0 & 0\\ 0 & 0 \end{bmatrix} \quad C_{sys} = \begin{bmatrix}0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\end{bmatrix} \quad D_{sys} = 0. \tag{3.33}$$

again the system become larger as the dynamics of θ and $\dot{\theta}$ are now included in the model as well.

With the dynamics of the angular velocity of the motor added, all that is left is to connect the velocity PI controller. As before the PI controller is in series with the rest of the system and the same methodology was used thus far can be applied. The PI velocity controller naturally is assumed to have the same form as the PI current controller, and is to look as:

$$PI_{velocity} = \frac{scv_1 + cv_2}{scv_3 + cv_4} \tag{3.34}$$

Subsequently, if that controller in a SS form is included in with the already established system the new A matrix becomes as follows:

$$A_{ser-v} = \begin{bmatrix} \frac{-cv_4}{cv_3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{-ci_4}{c_3 i} & 0 & -1 & 0 & 0 & 0 \\ \frac{-cv_1cv_4 - cv_2cv_3}{cv_3^2} & 0 & \frac{-ci_4}{c_3 i} & 0 & -1 & 0 & 0 \\ 0 & \frac{-ci_1ci_4 - ci_2ci_3}{L_d ci_3^2} & 0 & \frac{-R}{L_d} - \frac{ci_1}{L_d ci_3} & 0 & 0 & 0 \\ \frac{-ci_1(cv_1cv_4 - cv_2cv_3)}{L_q ci_3 cv_3^2} & 0 & \frac{-ci_1ci_4 - ci_2ci_3}{L_q ci_3^2} & 0 & \frac{-R}{L_q} - \frac{ci_1}{L_q ci_3} & 0 & \frac{-K_e}{L_q} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & \frac{K_t}{J_t} & 0 & -\frac{b_a}{J_t} \end{bmatrix}. \tag{3.35}$$

From the matrix (3.35) it can be easily concluded that one more state were added to the dynamic matrix A. with the new state in the system being the error between reference and feedback of velocity, the remaining states are the errors in both the currents, both the currents, and the angular velocity and position. The remaining system matrices are as follows:

$$B_{ser-v} = \begin{bmatrix} 1 \\ 0 \\ \frac{cv_1}{cv_3} \\ 0 \\ \frac{cv_1 ci_1}{L_q cv_3 ci_3} \\ 0 \\ 0 \end{bmatrix} \quad C_{ser-v} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad D_{ser-v} = 0. \tag{3.36}$$

With 3.36 matrices established the whole forwarded dynamic of the velocity loop was found.

All that is left, is to incorporate the velocity feedback into the dynamical system equations. Using again principle (3.28) the A matrix of the feedback velocity is found to be of the following form:

$$A_{vel} = \begin{bmatrix} \frac{-cv_4}{cv_3} & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & \frac{-ci_4}{c_3 i} & 0 & -1 & 0 & 0 & 0 \\ \frac{-cv_1cv_4 - cv_2cv_3}{cv_3^2} & 0 & \frac{-ci_4}{c_3 i} & 0 & -1 & 0 & \frac{-c1_v}{c3_v} \\ 0 & \frac{-ci_1ci_4 - ci_2ci_3}{L_d ci_3^2} & 0 & \frac{-R}{L_d} - \frac{ci_1}{L_d ci_3} & 0 & 0 & 0 \\ \frac{-ci_1(cv_1cv_4 - cv_2cv_3)}{L_q ci_3 cv_3^2} & 0 & \frac{-ci_1ci_4 - ci_2ci_3}{L_q ci_3^2} & 0 & \frac{-R}{L_q} - \frac{ci_1}{L_q ci_3} & 0 & \frac{-K_e}{L_q} - \frac{ci_1 cv_1}{L_q ci_3 cv_3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & \frac{K_t}{J_t} & 0 & -\frac{b_a}{J_t} \end{bmatrix}. \tag{3.37}$$

The rest of the matrices remain again unchanged. The last peace of dynamics that needs to be added is the dynamics associated with the two mass spring damper system and the effect that has on the whole system. As discussed in section 3.1.2 two new states need to be introduced to in order to accommodate the dynamics of the moving mass atop the shaft. The final open loop state space matrix

for GSSA, with the PLC (MSD) included is:

$$A_{GSSA} = \begin{bmatrix} \frac{-cv_4}{cv_3} & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & \frac{-ci_4}{c_3 i} & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ \frac{-cv_1 cv_4 - cv_2 cv_3}{cv_3^2} & 0 & \frac{-ci_4}{c_3 i} & 0 & -1 & 0 & \frac{-c1_v}{c3_v} & 0 & 0 \\ 0 & \frac{-ci_1 ci_4 - ci_2 ci_3}{L_d ci_3^2} & 0 & \frac{-R}{L_d} - \frac{ci_1}{L_d ci_3} & 0 & 0 & 0 & 0 & 0 \\ \frac{-ci_1(cv_1 cv_4 - cv_2 cv_3)}{L_q ci_3 cv_3^2} & 0 & \frac{-ci_1 ci_4 - ci_2 ci_3}{L_q ci_3^2} & 0 & \frac{-R}{L_q} - \frac{ci_1}{L_q ci_3} & 0 & \frac{-K_e}{L_q} - \frac{ci_1 cv_1}{L_q ci_3 cv_3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{K_t}{J_m} & \frac{-k_b qr}{J_m l\sqrt{1-\frac{q^2}{l^2}}} & \frac{-b_a}{J_m} - \frac{qr(b_a+bl)}{J_m l\sqrt{1-\frac{q^2}{l^2}}} & \frac{k_b}{J_m} & \frac{b_l}{J_m} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & \frac{k_b qr}{Ml\sqrt{1-\frac{q^2}{l^2}}} & \frac{b_l qr}{Ml\sqrt{1-\frac{q^2}{l^2}}} & -\frac{k_b+k_s}{M} & -\frac{b_l}{M} \end{bmatrix} \quad (3.38)$$

With the dynamics of the system firmly established, to complete the model of the GSSA the kinematic transformation from angular frame to linear frame has to be included. Equations (3.12)-(3.17) show the nonlinear transformation between the angular and linear frame. These equations need to be linearised so they can be included in the SS system.

Now the two outputs of the whole system, due to the positioning of the sensors are the linear position of the end of the shaft, and the forces acting on the load. The linearisation can be found the same way as it was done previously in the case of the dynamic nonlinear equation. The system output matrix C is found by calculating a Jacobian matrix of the nonlinear kinematic equations. As the feedback loops we are interested in are of position and force, we shell have two outputs and the output matrix C for the complete system linearised around zero becomes:

$$C = \begin{bmatrix} 0 & 0 & \frac{qr}{l\sqrt{1-\frac{q^2}{l^2}}} & 0 & 0 & 0 \\ 0 & 0 & \frac{k_b qr}{l\sqrt{1-\frac{q^2}{l^2}}} & \frac{b_l qr}{l\sqrt{1-\frac{q^2}{l^2}}} & -b_l & \end{bmatrix}.$$

and if included, the error states e_{id}, e_{id} and e_v the GSSA output C matrix become:

$$C_{GSSA} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \frac{qr}{l\sqrt{1-\frac{q^2}{l^2}}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{k_b qr}{l\sqrt{1-\frac{q^2}{l^2}}} & \frac{b_l qr}{l\sqrt{1-\frac{q^2}{l^2}}} & -(k_b + k_s) & -b_l \end{bmatrix} \quad (3.39)$$

To obtain the linear position of the mass the angular movement needs to be transfomed to linear movement. Subsequently it is only natural that reference signals to the GSSA need to be transformed from linear to angular movement. As the FCS uses the error of the linear position, the command signal it sends is also in the linear framework. Consequently, the single needs to be transformed into the angular framework, which results in the following input matrix B:

$$B_{GSSA} = \begin{bmatrix} \frac{l\sqrt{1-\frac{q^2}{l^2}}}{rq} \\ 0 \\ \frac{cv_1 l\sqrt{1-\frac{q^2}{l^2}}}{cv_3 rq} \\ 0 \\ \frac{cv_1 ci_1 l\sqrt{1-\frac{q^2}{l^2}}}{L_q cv_3 ci_3 rq} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (3.40)$$

With no feedfroward term introduced the $D_{GSSA} = [0 \quad 0]^T$ remains unchanged. With that said, the GSSA system SS linear representation is complete.

This concludes the finding full GSSA model. As the Figures (3.5) and Figure (1.2) show the GSSA and the MSD always come as one package. Installation and usage of a GSSA requires to have the MSD. Therefore, the combination of MSD with PMSM creates a whole system which is the GSSA, and then a load is added on top. In the rest of this thesis's research, any controller that was designed is tested using the (3.38)-(3.40) LTI system.

3.1.4. Parameter Optimisation and Validation

With the model established in the previous section it is necessary to find the numerical vales of the parameters in order to be able to perform full system analysis. Additionally, it makes the design of precise optimal controllers possible. Naturally estimating the whole system in one go might be very difficult and troublesome, therefore first validation of the inner current loop is done, and with the values of the inductances and internal resistance the inertias, friction values, and other values are found. Last step is to validate the kinematic transformation from angular to linear frame.

There are two major approaches to modelling. Either through the *grey-box method* or *black-box method*. The later leaves us with not knowledge of the internal working of the system, but it very easy to preform which means it might be a good initial guess if needed. Engineers from Moog provided us with initial values for all parameters and those will be the initial starting points.

First, the values of all parameters received from Moog engineers were tested to see how well the model preforms. Table (3.1) shows the initial parameters that were provided. The focus is first on the most inner loop, which is the current loop, to make sure that the resistance and inductances vales are accurate. At the same time the constants of the PI current controller are tested. Fortunately from the initial tests it is clear that the initial values are very promising. The model is able to predict the outcome of the current loop with great accuracy. As the initial run is so successful further optimisation improve upon it, for better estimation.

In the light of the current loop tests, the velocity loop is tested as well, to test what results it renders. As Figure (3.6) illustrates the results are unsatisfactory. This leads to the conclusion that either the model is incorrect or the parameter values provided are incorrect. It being more likely that the parameters were obtain to fit a different model than the one established for the purposes this thesis, the model is kept and the parameters are optimised. The initial parameters are kept as initial starting point for the optimisation.

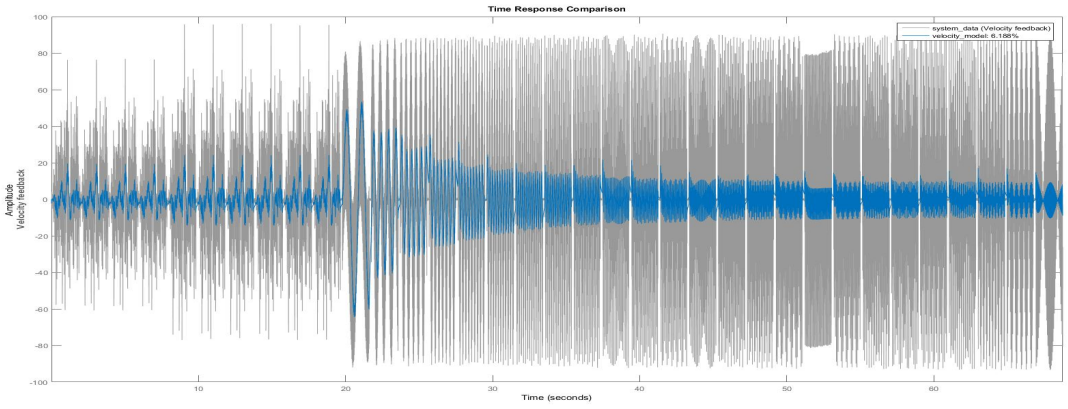


Figure 3.6: First testing of initial values for the velocity loop.

With the initial grey box test not giving any promising results, black box method is explored for

Parameter list		
Names of System Parameters	Initial guess	Optimised values
Load Mass (M)	10 kg	10 kg
Internal Resistance (R)	0.378 Ω	0.378 Ω
d-axis inductance (L_d)	3.427 mH	3.427 mH
q-axis inductance (L_q)	3.334 mH	3.334 mH
motor pole count (p)	6	6
BEMF constant (K_e)	1.13 $V/(rad/s)$	0.0169 $V/(rad/s)$
torque constant (K_t)	1.39 Nm/A	191.3 Nm/A
total Inertia (J_t)	0.0139 kgm^2	5.7×10^{-3} kgm^2
PMSM with Gearbox Inertia (J_m)	11.42 kgm^2	6.4×10^{-4} kgm^2
viscus friction (b_a)	0.2 Nms	3×10^{-4} Nms
spring damping (b_l)	0.1 Nms	0.001 Nms
spring constant (k_b)	50 Nm	48.75 Nm
spring constant (k_s)	5 Nm	4.875 Nm
torque load (Tl)	0.01 Nm	1.36×10^{-4} Nm
rod length	90 mm	80 mm
radius	23 mm	22.5 mm
offset	10 mm	9 mm
gearbox ratio	27.5	27.5
ci_1	19	80.3
ci_2	10555	11017
ci_3	1	1
ci_4	555	263
cv_1	12	7.81
cv_2	0.12	4.69
cv_3	1	1
cv_4	100	100.4

Table 3.1: Table of parameters. On the left the initial values. On the right the parameters found through optimisation.

better estimation. Figure (3.7) shows the two different attempts to find a dynamic model for the velocity loop. Two different approaches were tried. The first was to estimate a TF with the most optimal number of poles and zeros. The second was to find a SS system with the right order. As it can be seen from the figure, finding a TF was only about unsuccessful. On the other hand, the SS model shows to be very promising. In the case the grey box model is not optimal, the SS model found through the black box approximation can be used for testing and research purposes.

It was explored if the grey- box model can be optimised, using the dynamical equations found in the Section (3.1.3). To do so first the inner current loop is investigated. The estimation of the parameters of this loop are done first, this makes the overall model estimation easier handle. To do this MATALB was used (all MATLAB code can be found in appendix C). From Matlab the grey-box identification toolbox was used. To be able to optimise the model a sine-sweep on the physical system was performed, with frequencies ranging between 0.1Hz to 50 Hz. The limits where chosen based on the operational range of GSSA. Additionally, pink noise was also added with frequencies up to 50Hz, for a

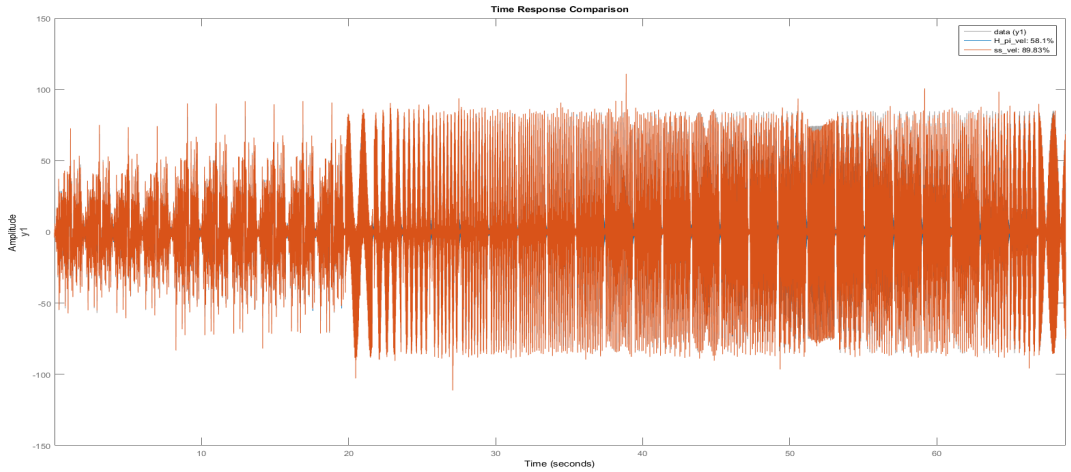


Figure 3.7: Attempting to find a system model through black box approach.

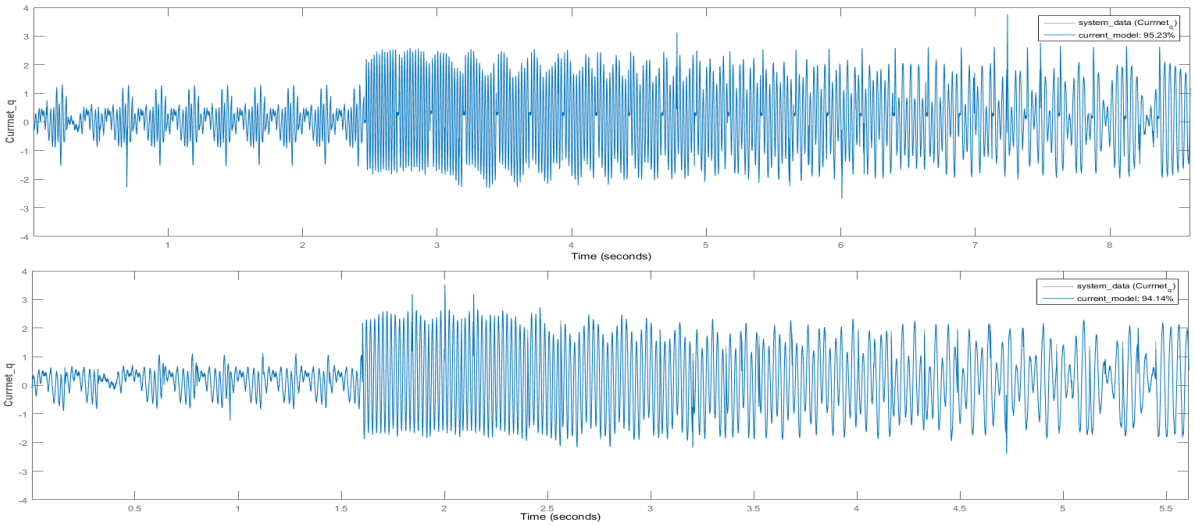


Figure 3.8: Model estimation with optimised parameters, top. Model validation on optimised parameters, bottom.

better model approximation. Due to equipment limitations, finding a step response was not possible. With over $40,000$ input-output pairs the optimisation is run. Least squares optimisation algorithm was used, which is the default search algorithm for the optimisation toolbox.

Figure (3.8) illustrates the results of optimisation as well as the validation of the same model. It is easy to see that not a great improvement was possible in comparison with the initial values. Such outcome was expected as the original tests were already very successful. A more accurate model might difficult to find as the current loop has a very high bandwidth, the sampling for optimisation was run at 8KHz as the equipment could not handle a higher sampling frequency. That makes noise quite dominant in the system.

Estimation of the velocity loop was done in the same fashion as previously the current loop. The inputs and corresponding outputs of the feedback loop were measured. Again a sine-sweep, between 0.1Hz and 50Hz, was used with pink noise up to 50Hz frequencies. With the velocity loop the sampling was done at 1KHz, as the feedback loop has a bandwidth of approximately 100Hz. The results of optimising with over $60,000$ input-output points are shown Figure (3.9). Clearly the optimisation gave a much better estimated system than the initial one, the test run on the validation set confirms

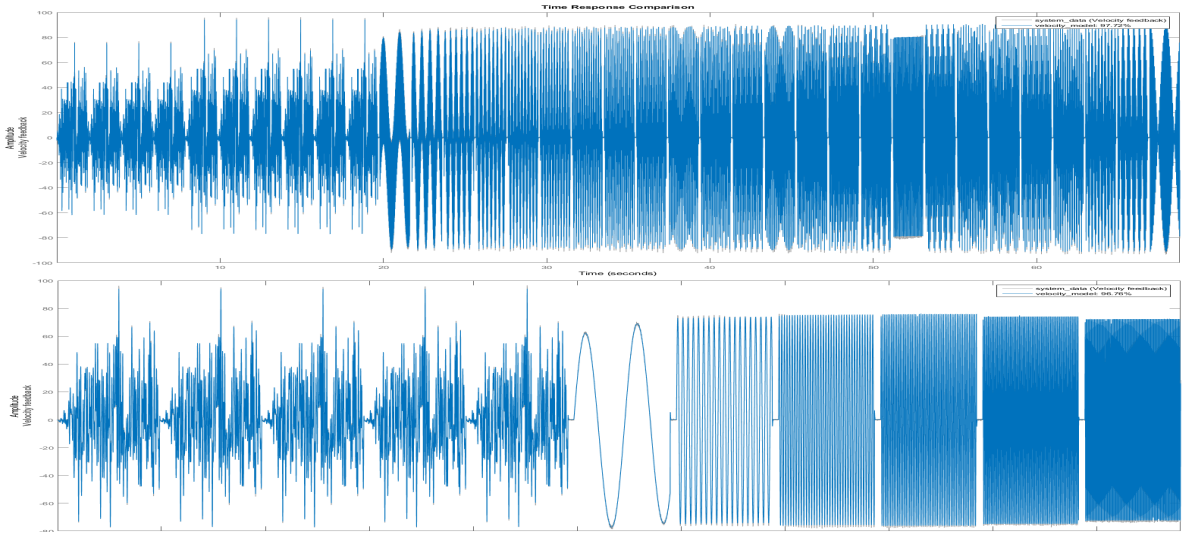


Figure 3.9: Model estimation with optimised parameters, top. Model validation on optimised parameters, bottom.

Time [s]	isdref [A]	isqref [A]	isd [A]	isq [A]
-0.404938	0	-2.398441792	-0.089262024	-2.313161135
-0.3990005	0	-2.093308449	0.064921565	-2.262845993
-0.393063	0	-2.502043724	0.014605123	-2.766954899
-0.3871255	0	-2.453521252	0.00265748	-2.382480621
-0.381188	0	-3.017095566	0.028082125	-3.042263746
-0.3752505	0	-3.001567841	0.005642373	-2.955726147
-0.369313	0	-3.27239418	0.041283358	-3.350580215
-0.3633755	0	-3.536242247	0.032091882	-3.519293547
-0.357438	0	-3.376008272	-0.021745048	-3.28682065
-0.3515005	0	-3.704526901	-0.019934298	-3.757477045

Figure 3.10: The values in the d-axis are negligible in comparison to the q-axis.

good results. With the model being able to predict with an accuracy in the high 90%, the inaccuracy is attributed to noise or model inaccuracies.

Lastly, the position loop is closed, to validate the force loop. Subsequently, the first test is with the initial values supplied by Moog. The initial results are quite promising, but to improve the results the same methodology is followed as previously. The optimisation is done much the same way as with all other loops. Using equations (3.38)-(3.40), the initial values supplied by Moog are optimised, with the help of the MatLab optimisation toolbox and 100 000 I/O pairs. The results of the closed position loop can be seen in Figure (3.11). The same figure illustrates the parameters found for the position loop, not surprisingly, are also valid for the force loop. For this optimisation the P gain for both force and position was set to 1 and the I gain to zero. This was done for the easy of modelling. As follows, the complete model of the FCS with the GSSA was found, with both complete dynamics and kinematics.

With the dynamic model complete further improvements are researched. From the structure of the dynamic A matrix it can be seen that the matrix is not in its minimal realisation. The A matrix can be reduced by removing to states that are redundant, which are the current in the d-axis i_d and as a logical extension the error on that current value. The matrix reduction is in line with the original assumptions that the value of current i_d is always zero, as the actual value is negligible compered to i_q. This is due to lack of field weakening, in the motor. The crank of the PMSM is assumed to never spin fast enough for nonlinear occur. We proved that experimentally which is shown in Figure (3.10).

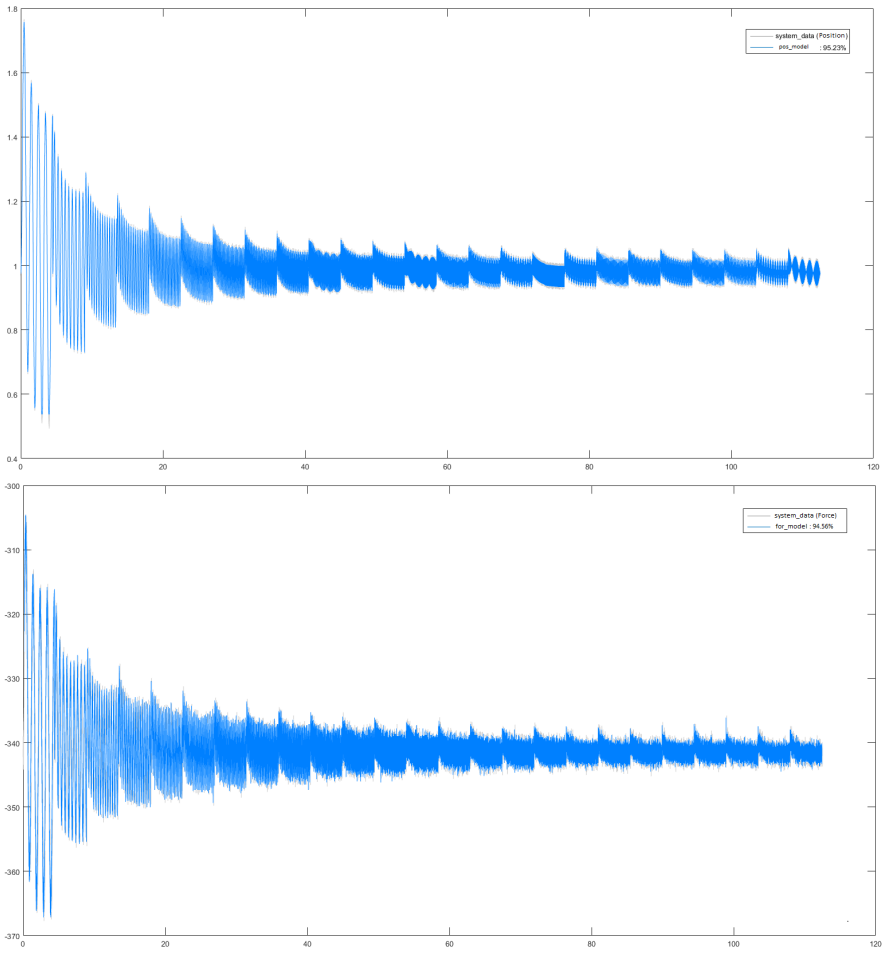


Figure 3.11: Force and Position loop validation.

It follows that the reduced A matrix in its minimal realisation form is:

$$\begin{bmatrix} \dot{e_\theta} \\ \dot{e_{i_q}} \\ \dot{i_q} \\ \dot{\theta} \\ \ddot{\theta} \\ \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} \frac{-cv_4}{cv_3} & 0 & 0 & 0 & -1 & 0 & 0 \\ \frac{-cv_1cv_4-cv_2cv_3}{cv_3^2} & \frac{-ci_4}{c_3i} & -1 & 0 & \frac{-c1_v}{c3_v} & 0 & 0 \\ \frac{-ci_1(cv_1cv_4-cv_2cv_3)}{L_qci_3cv_3^2} & \frac{-ci_1ci_4-ci_2ci_3}{L_qci_3^2} & \frac{-R}{L_q}-\frac{ci_1}{L_qci_3} & 0 & \frac{-K_e}{L_q}-\frac{ci_1cv_1}{L_qci_3cv_3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{K_t}{J_m} & \frac{-k_bqr}{J_ml\sqrt{1-\frac{q^2}{l^2}}} & \frac{-b_a}{J_m}-\frac{qr(b_a+bl)}{J_ml\sqrt{1-\frac{q^2}{l^2}}} & \frac{k_b}{J_m} & \frac{b_l}{J_m} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{k_bqr}{Ml\sqrt{1-\frac{q^2}{l^2}}} & \frac{b_lqr}{Ml\sqrt{1-\frac{q^2}{l^2}}} & -\frac{k_b+k_s}{M} & -\frac{b_l}{M} \end{bmatrix} \begin{bmatrix} e_\theta \\ e_{i_q} \\ i_q \\ \theta \\ \dot{\theta} \\ x \\ \dot{x} \end{bmatrix} \tag{3.41}$$

which with the parameters field the reduced system will be used for all simulation using *Simulink*, as well as designing new controllers.

3.2. System analysis

To conclude this chapter in this last section the system is analysed. A side from the two inner PI control loops, the analysis on the system is done with no further control applied. By analysing the plant in open loop, a base is made which can be used for designing optimal controllers for the plant. Additionally, it gives an indication on what performance can be expected from the system. The focuses of the analysis is both in the time and frequency domain.

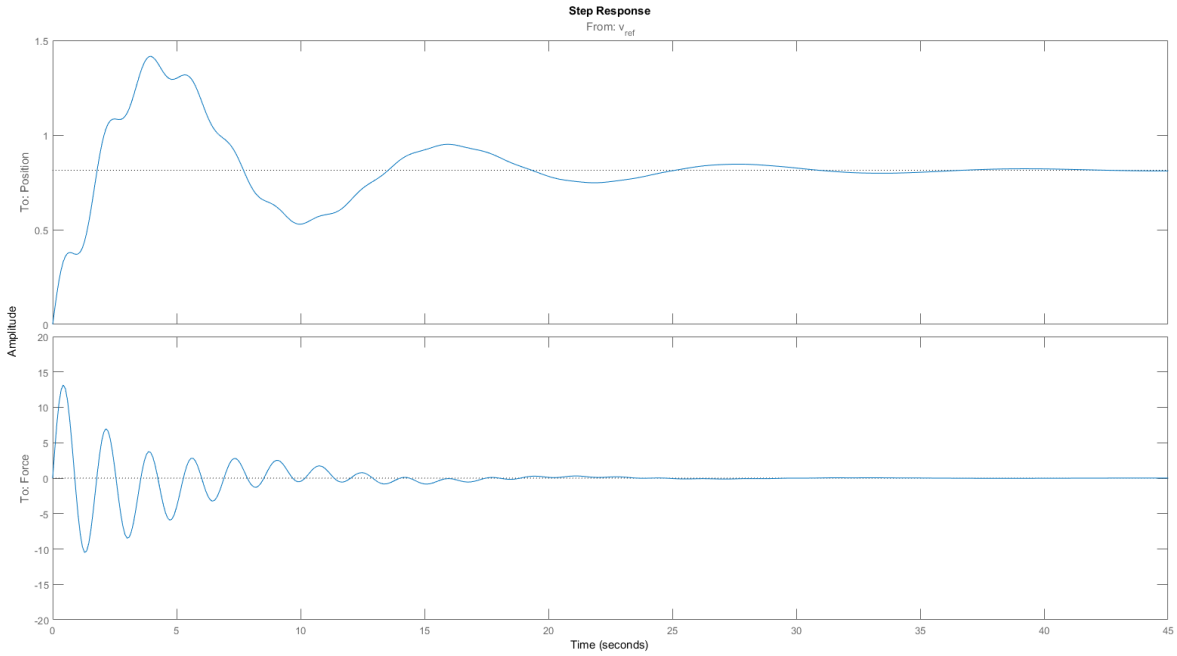


Figure 3.12: System response to a unitary step input in linear velocity.

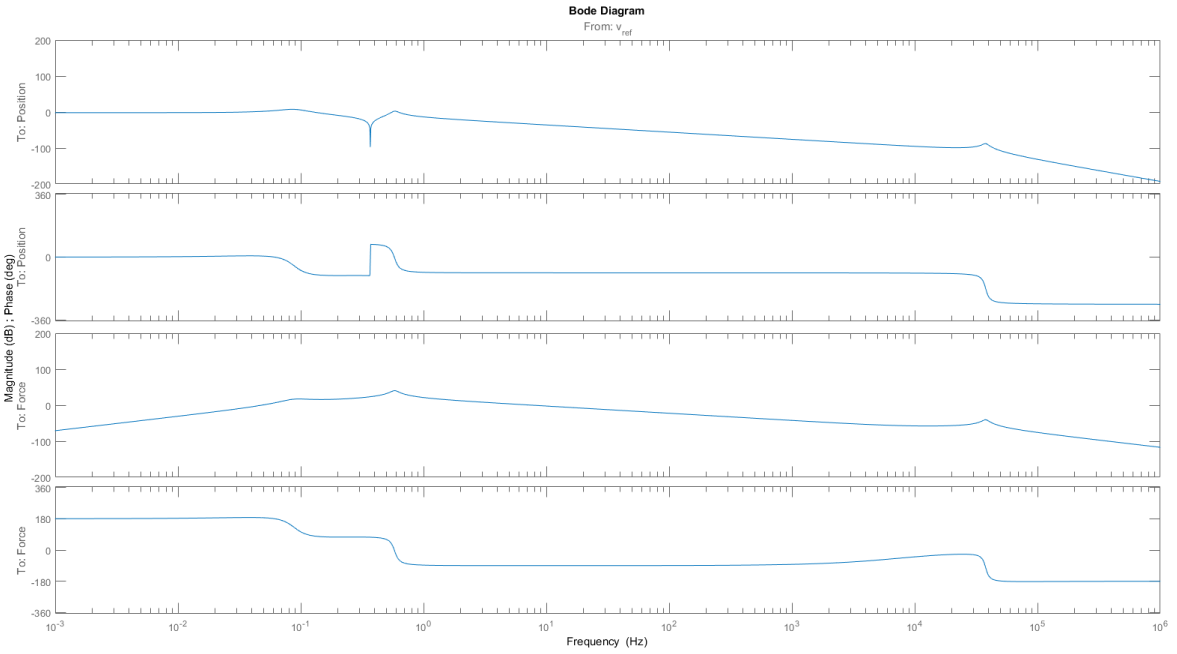


Figure 3.13: Bode plot of the GSSA

In Figure(3.12) it can be seen the response of the system plant to a unit step input. From the figure it can be seen that the position has a steady state different from 1, in response to a unit step in liner velocity. This indicates that potential controllers will need an integrator as the system does not inherently have one. The force on the other hand has a value of 0 Newton's which might be counter intuitive as the springs are under tension in order to position the masses. To remember at this point is the fact, that the system seeks a energy equilibrium, which understandable results in, effectually zero forces acting on the mass. The system is also slow needing 29.7 seconds to fully settle and is clearly not well damped. This behaviour is expected, though after inspecting plant poles. The two dominant pole pairs $0.2309 \pm 3.647j$ and $0.1223 \pm 0.5405j$, have an imaginary part to them. That explain the

highly oscillatory behaviour. The other poles are situated at 137, which is the only real pole, and $1,222 \pm 23,737j$. Subsequently, they do not have much effect on the system behaviour.

Plot shows that around the $0.1Hz$ to $1Hz$ area, two anti-resonances occur and one resonance. This behaviour is due to the double mass-spring system [31]. The first anti-resonance is due to the viscous damping in the electric motor. The following resonance and anti-resonance pair, is the result of two masses connected through a spring. Due to damping the anti-resonance is not sharp. Much further down the spectrum, around $3,700Hz$, the dynamics of the electric circuit can be observed. The system is clearly of second order with rather low ζ, which is shown by the peak. The lower plot shows the input to the second output (force acting on the load). The bottom bode plot, shows all the behaviour as the first one, with the exception of low frequencies. The force on the load does not experience the dynamics associated with the viscous damping of the motor. It does though experience the softer spring at the top for the system, which connects the load to a stationary surface. The low bandwidth, also gives an indication to the long settling time of the plant.

Lastly we look at the parameters resulting from the optimisation. Table (3.1) contains both the initial and optimised system parameters. Over all the parametrisation was successful. The parameters found, though are very different then expected. The BEMF is much lower than expected and the torque is vastly higher. A big difference can also be noticed between the initial inertia of the electric motor with the gearbox and the optimised value. Significant change is also visible when it comes to the parameter values of the current PI controller, which is the suspect causing the rest of the irregularities.

4

GSSA Control

The following chapter focuses on designing controllers using the model developed in the previous chapter. As discussed prior, three main controllers are implemented to deal with the problem stated at the begging of this thesis. The three controllers under investigation are: Dwell-Time control for Switched Systems, Back Calculation Anti-wind Up for PID and Model Predictive Control.

4.1. Problem

The original problem presented was concerned the system becoming unstable under certain conditions during switching. The FCS switches between the position feedback loop to the force feedback loop if the force limits are active, and are validated. If the force limits are active the closed position loop switches to the closed force loop, both loops have independently tuned PI controllers.

The position loop might be subjected to any standard periodic signal like sinusoid, square wave, sawtooth, etc. During the operation, the forces acting on the load are monitored. If the forces on the load exceed the limits imposed by the operator, a switch occurs and the force limit becomes a constant input reference for the PI force controller to follow. As long as the limits are active the PI force controller will keep the force at the upper or lower limit, not letting the force value change and as a result not letting the load move further. While the force loop is active, the force that would be subjected on the load if the position reference would be followed, is still monitored. If those forces return within the operational limits, the loops switch again and the position feedback loop is allowed to act on the periodic reference again.

In the plant at hand, some switching conditions cause the GSSA emergency off switch feature to be activated. The two possible cause are either too fast switching between the controllers or actuator saturation. In the first case, the switching is fast enough to destabilise the system by switching from one loop to the other before any of them can stabilise after the switch. In the second case, when a switch occurs, the difference between the reference of one loop and the other is large. That causes saturation of the actuator and leads eventually to instability.

The GSSA as all real life actuators has operational limitations. The PMSM can only have that much current, and in turn can produce only a limited amount of torque. This problem is known as actuator saturation. During sudden changes in the reference signal, which are big for the plant, the integrator in a PI controller might *wind up*. A wind up is when the integrator acts on a very large error, expecting a very large control signal. Such control signal cause the actuator output to saturate. This might lead to damage done to the actuator, as well as increased operational costs.

To see how the control signal changes in behaviour between a unsaturated and saturated case the reader is referred to Figure (4.1). Clearly behaviour as presented in the Figure can cause delays, which in turn lead to hindered performance or even instability. In Figure (4.2) the behaviour of the plant is

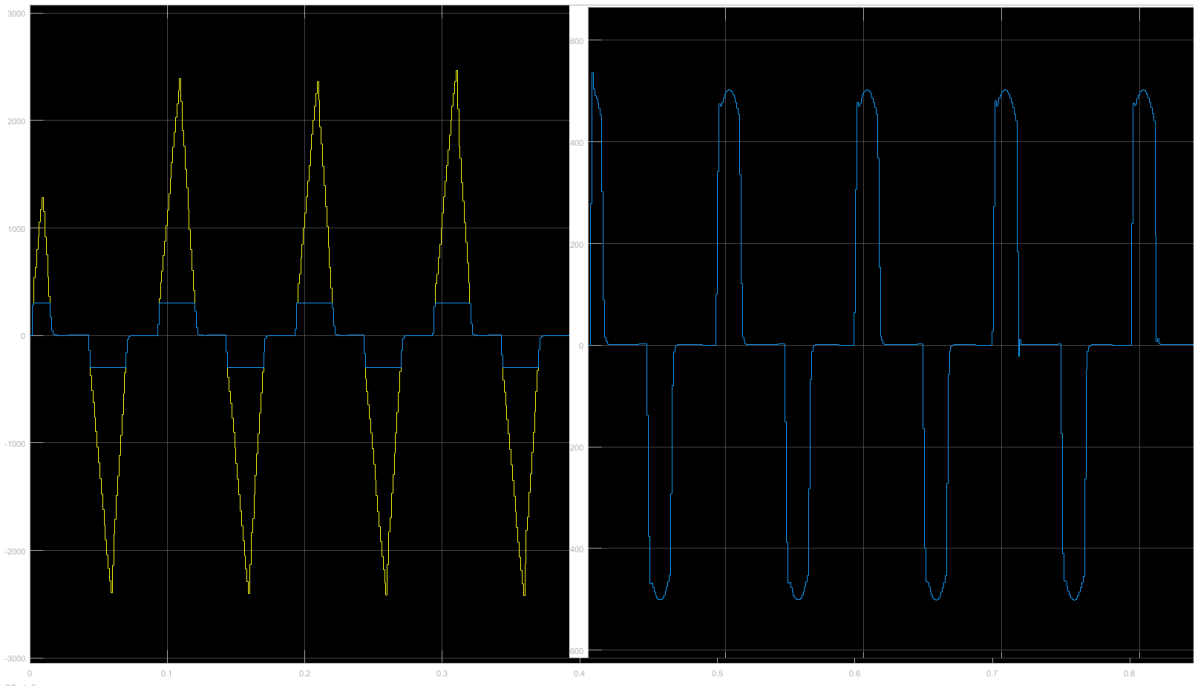


Figure 4.1: On the right the saturated controller is presented. On the left the system under the same signal, but with out saturation (during normal operation).

presented when under saturation. Clearly the performance is hindered, and no longer on an acceptable level.

A constantly saturated actuator not only put dangerous amount of stress on the mechanical system itself, which might result in damaging the system, but also introduces a big delay in the system. The delay will make the system respond to any further changes in the reference signal very slow and gives an unacceptable performance of the plant.

To make sure the issue is solved and to ensure that further problems like that do not arise in the future the following three solutions were researched:

- Switching Systems with Dwell Time
- Anti-Wind up
- MPC

The first keeps the current structure of the FCS, but adds a condition that no switch can be made between the two feedback loops unless a specified, minimal amount of time has passed. That time is called *Dwell Time*, this was discussed in section (4.2.3). The second approach goes back to control basics. A well-known method to deal with integrator wind up, are anti-wind up techniques. The Back Calculation is investigated in section (4.2.6). The last approach is a most recent control schema out of the three, which is most difficult academically to design. The *Model Predictive Control* is best known for being able to handle, both plant constraints and output constraints. MPC in researched in Section (4.3).

4.2. PID and Switched Control

The first solution to be explored, comes from the well known PID type controllers. PID type controllers are by far the most commonly used controllers in the world. Almost all controllers implemented in the industry today are PID control algorithm or it's many variations. PID controllers are efficient, reliable and most importantly very easy to operate. Designing a full PID algorithm takes time, experience and

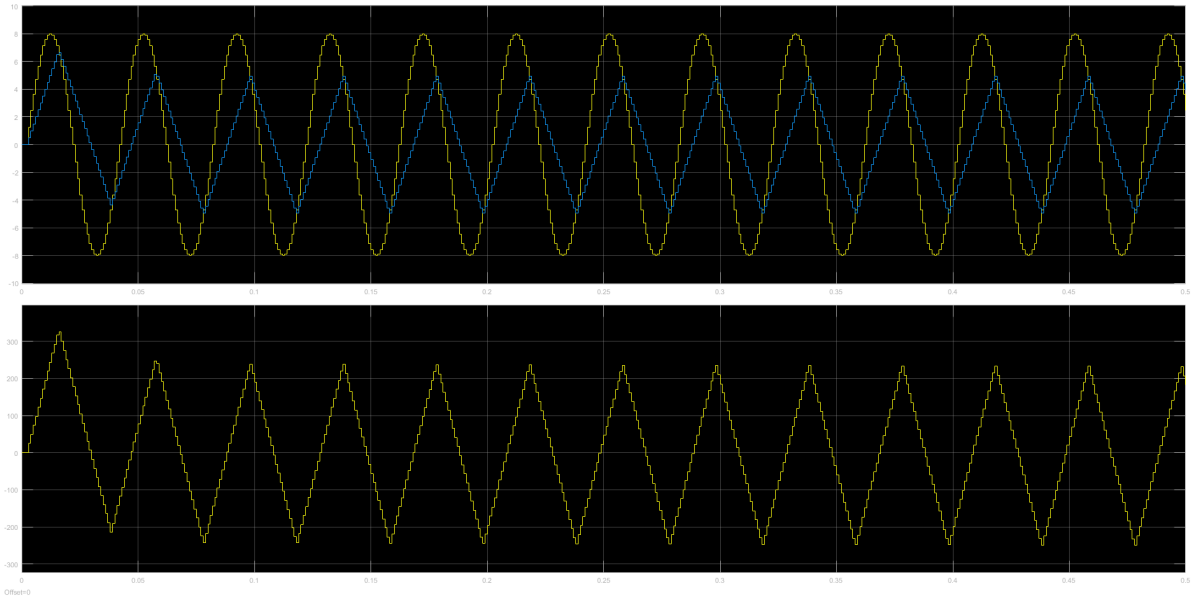


Figure 4.2: Plant behaviour with a saturated control input.

for most knowledge. Tuning a PID controller is a very easy task and is done by thousands of engineers every day. All that because algorithm can be either vied as operating on several rules of thumb, or be approached analytically.

4.2.1. PID Control

PID controllers work with the, very simple yet incredibly powerful, principle of *feedback*. Application of the feedback principle has led to major breakthroughs in communication, instrumentation, and control. At is simplest the principle of feedback can be expressed as [32]:

Theorem 4.2.1. *Increase the manipulated variable when the process variable is smaller than the stepoint, and decrease the manipulated variable when the process variable is larger than the setpoint.*

This type of feedback is called *negative feedback*, as it is always trying to oppose whatever the system is doing, to make it follow our setpoint. Negative feedback is highly useful as it steers the process variable close to wanted setpoint despite disturbances, and variation of the process variables. More on the idea of feedback can be found in [20].

Now as mentioned before the main interest is in the PI version of the PID algorithm. So the proportional and integral parts of the algorithm are used. The derivative of the error is not used as it amplifies noise, which is of course undesired and unwanted. In most text books the PI algorithm can be found in time domain as:

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau\right) \tag{4.1}$$

where u is control variable and e is the control error. The control variable is thus sum of two variables: the P-term and the I-term. The control parameters are K and the integral time T_i. Equation (4.1) can also be expressed in the frequency domain, by taking the Laplace transform of it:

$$u(s) = \left(K + \frac{K}{sT_i}\right)e(s) \tag{4.2}$$

giving us the TF from error to control signal.

In ideal scenario, where no sensor noise is present, the over all loop gain K should be high. The higher the closed loop gain the more insensitive the control loop becomes to load disturbance. It also ensures that the process output y is close to the setpoint y_{ref}. On the other hand large closed loop

gain, results in high sensitivity to sensor noise if present, and proportional control always has a steady-state error.

To over come the some of the problems of just proportional control, the integral action is introduced. The main function of I-term is to get rid of the steady state error. It follows from equation (4.2) that:

$$u_0 = \left(K + \frac{K}{T_i}t\right)e_0$$

where $e_0$is a constant error and u_0 is a constant control signal. As long as $e_0 \neq 0$, this clearly contradicts the assumption that the control signal u_0 is constant. A controller with integral action will always give zero steady-state.

4.2.2. Switched Control

In its most general form a *hybrid system* can be described as

$$\begin{cases} x \in C & \dot{x} = f(x) \\ x \in D & x^+ = g(x) \end{cases} \tag{4.3}$$

In the above representation the state x can evolve either according to differential equation (sometimes referred to as differential inclusion) $\dot{x} = f(x)$, while in set C; or according to difference equation (or difference inclusion) $x^+ = g(x)$, while in set D. The notation $\dot{x}$ represents the velocity of x and x^+ represents the value of state after instantaneous change.

The behaviour of a dynamical system described by a differential equation is the *flow* and by difference equation the *jump*. This leads to:

- $C \subset \mathbb{R}^n$ is the *flow*
- $f(*) : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is the flow map with $C \subset domF$
- $D \subset \mathbb{R}^n$ is the *jump* set
- $g(*) : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is the flow map with $D \subset domG$

With the above in mind a hybrid system has the notation $\mathcal{H} = (C, F, D, G)$. One of the main types of hybrid systems, are *switched systems*.

Continuous-time systems with discrete switching events are referred to as *switched systems*. One can arrive at a switched system from a hybrid system, by neglecting the discrete dynamics and instead consider all possible switching patterns from a certain class. This makes switching systems very different form hybrid systems especially in the analysis stage. Considerable amount of research has gone into the field of switched systems in the recent years [33]. The reasoning being that switched systems can considerably outperform single degree controllers [34], [35].

A nonlinear dynamic equation is at its simplest presented as:

$$\dot{x} = f(x, t) \tag{4.4}$$

with initial conditions x_0 at time t_0 has a solution $x : [t_0, t_1] \rightarrow \mathcal{R}$ as long as x is continuous and differentiable on (t_0, t_1). Note, if $f(*)$ is continuous in t and x, any solution x will be continuously differentiable. If for all initial conditions one can have a unique solution trajectory, then well-posedness is considered. From [36] the definition for a system to be local Lipschitz can be found as:

Theorem 4.2.2. *For a piecewise continuous function $f(t,x)$ in t, which satisfy Lipschitz condition then there exists an $L > 0$ and $r > 0$ such that*

$$L||x - y|| \geq ||f(t,x) - f(t,y)||$$

For all x and y in a neighbourhood $B := \{x \in \mathcal{R}^n | ||x - x_0|| < r\}$ of x_0 and for all $t \in [t_0, t_1]$. Then there exists a $\delta > 0$ such that the equation

$$\dot{x} = f(t,x) \text{ with } x(t_0) = x_0$$

has a unique solution over $[t_0, t_0 + \delta]$

Such system can be said that it is separated into two by a hyper-surface defined by C_+ or C_-. Consequently as long as the system is in either of them, no ambiguity can be found in how the system evolves. Such case is when the dynamics of both f_+ and f_- steer towards either C_+ or C_-, otherwise no solution can be found. An alternative definition for the sliding behaviour mentioned in [37] is based on the *equivalent control definition* of sliding modes, which is essential due to Utkin [38]. This definition is related to "switching control subsystems".

Suppose that a continuous state space is partitioned into a finite or infinite number of *operating regions* by means of a family of *switching surfaces*, or *guards*. In each of these regions, a continuous-time dynamical system is given. Whenever a system trajectories hits a switching surface the state jumps to a new value specified by a *reset map*. Such a map has its domain as the union of the switching surfaces and range as the whole state space. In summery the system is specified by:

- The family of switching surfaces and the resulting operational regions;
- The family of continuous-time subsystems, one for each operating region;
- Reset map

For a family f_p, $p \in \mathcal{P}$ of functions from $\mathbb{R}^n$ to $\mathbb{R}^n$, where $\mathcal{P}$ is some index set, a family of systems is:

$$\dot{x} = f_p(x), \quad p \in \mathcal{P} \tag{4.5}$$

Evolving on $\mathbb{R}^n$. The functions f_p are assumed to be sufficiently regular, at least Lipschitz. The easiest case to think about is when all these systems are linear:

$$f_p = A_p x, \quad A_p \in \mathbb{R}^{n \times n}, p \in \mathcal{P} \tag{4.6}$$

and the index set $\mathcal{P}$ is finite: $\mathcal{P} = \{1, 2, \ldots, m\}$.

A piecewise constant function σ: $[0, \infty) \to \mathcal{P}$ can be used to define a switched system generated by the above family. Such function σ, which is called a *switching signal*, has finite number of discontinuities which quite intuitively are called *switching times*. The role of the switching signal is to specify at each time instant t, the index $\sigma(t) \in \mathcal{P}$ of the *active subsystem*.

Therefor the GSSA with the FCS controler can be though of as a *linear switching system*. Both the position loop and the force loop have different dynamics, as they have a different PI controller and they use a different output to close the loop. Subsequently the system can be represented in a switch form as:

$$\dot{x} = \begin{cases} A_p x + B_p u \to y_f < \left|\epsilon_f\right| \\ A_f x + B_f u \to y_f > \left|\epsilon_f\right| \end{cases} \tag{4.7}$$

where are index set is only $\mathcal{P} = \{1, 2\}$. The switching controller has to switch between the force and position dynamics.

There are two approaches to switching: *autonomous* and *controlled*. By autonomous switching it is inferred that no control is held over the mechanism triggering the discrete events and in controlled switching an operator has full control over the switching signal and switches the system in order to achieve desired behaviour. Our system of course switches in a autonomous manner, according to the set limits placed on its outputs.

4.2.3. Switching Time

Very often precise switching has a structure and constrains to it. As such the tool to use for stability analysis, relies on *multiple Lyapunov functions*. In other words, if a switched system (4.7) is considered with $\mathcal{P} = \{1, 2\}$ and it is assumed that both of the subsystems are globally asymptotically stable, the stability of the switched system is investigated using V_1 and V_2, which are Lyapunov functions for the two subsystems.

With the assumption that no common Lyapunov function can be found, stability properties greatly depend on the switching signal. For t_i where $i = 1, 2, \dots$, switching times, if values of V_1 and V_2 coincide at each switching time then V_σ is a continuous Lyapunov function and asymptotic stability follows. Otherwise when V_p only decreases when subsystem p is active then V_σ is a discontinuous function. To claim asymptotic stability for such case, the values of V_p is looked at the beginning of each switch were $\sigma = p$. the values have to decrease over time for asymptotic stability.

Theorem 4.2.3. *Let (4.5) be a finite family of globally asymptotically stable systems, and let V_p, $p \in \mathcal{P}$ be a family of corresponding radially unbounded Lyapunov functions. Suppose that there exists a family of positive definite continuous functions W_p, $p \in \mathcal{P}$ with the property that for every pair of switching times $(t_i, t_j), i < j$ such that $\sigma(t_i) = \sigma(t_j) = p \in \mathcal{P}$ and $\sigma(t_k) \neq p$ for $t_i < t_k < t_j$ the following must be true:*

$$V_p(x(t_j)) - V_p(x(t_i)) \leq -W_p(x(t_i)) \tag{4.8}$$

Then the switched system (4.7) is globally asymptotically stable.

For the proof of the above the reader is referred to [39]. First stability of the origin in the sense of Lyapunov is shown. Let m be the number of elements in $\mathcal{P}$. Without loss of generality, it is assumed that $\mathcal{P} = 1, 2, \dots, m$. Consider the ball around the origin of an arbitrary given radius $\epsilon > 0$. Let $\mathcal{R}_m$ be a set of the form $x : V_m(x) \leq c_m, c_m > 0$, which is contained in this ball. For $i = m - 1, \dots, 1$, let $\mathcal{R}_i$ be a set of the form $x : V_i \leq c_i, c_i > 0$, which is contained in the set $\mathcal{R}_{i+1}$. Denote by δ the radius of same ball around the origin which lies in the intersection of all nested sequences of set constructed in this way for all possible permutations of $1, 2, \dots, m$ Suppose that the initial condition satisfies $\|x(0)\| \leq \delta$. If the first k values of $sigma$ are distinct, where $k \leq m$, then by construction $\|x(t_k)\| \leq \epsilon$. After that, the value of σ will start repeating, and the condition (4.8) guarantees that the state trajectory will always belong to one of the above stets. Figure (4.3) illustrates this argument for the case $m = 2$.

To show asymptotic stability, observe that due to the finiteness of $\mathcal{P}$ there exists an index $q \in \mathcal{P}$ that has associated with it an infinite sequence of switching times $t_{i1}, t_{i2}, \dots$ such that $\sigma(t_{ji}) = q$. The sequence $V_q(x(t_{i1})), V_q(x(t_{i2})), \dots$ is decreasing and positive, and therefor has a limit $0 \leq c$. Subsequently, the following:

$$\begin{aligned} 0 = c - c &= \lim_{j\to\infty} V_q(x(t_{i_{j+1}})) - \lim_{j\to\infty} V_q(x(t_{i_j})) \\ &= \lim_{j\to\infty} [V_q(x(t_{i_{j+1}})) - V_q(x(t_{i_j}))] \\ &\leq \lim_{j\to\infty} [-W_q(x(t_{i_j}))] \leq 0 \end{aligned} \tag{4.9}$$

. Thus $W_q(x(t_{i_j})) \to 0$ as $j \to \infty$. Then W_q has to be positive definite. In view of radial unboundedness of $V_p, p \in \mathcal{P}$, an argument similar to the one used earlier to prove Lyapunov stability shows that $x(t)$ stays bounded. Therefore, $x(t_{i_j})$ must converge to zero as $j \to \infty$. It now follows from the Lyapunov stability property that $x(t) \to 0$ as $t \to \infty$.

Dwell time

Most straight forward to invoke stability for a switched system is to introduce a *dwell time*. Dwell time is a number $\tau_d > 0$ lets us introduce a class of restricted switching times $t_1, t_2, \dots$ with the property $t_{i+1} - t_i > \tau_d$. It is well-known that for a family (4.6) which is asymptotically stable the system (4.7) is asymptotically stable if τ_d is large enough. Under suitable assumptions, a big enough τ_d can grantee asymptotic stability for nonlinear systems too.

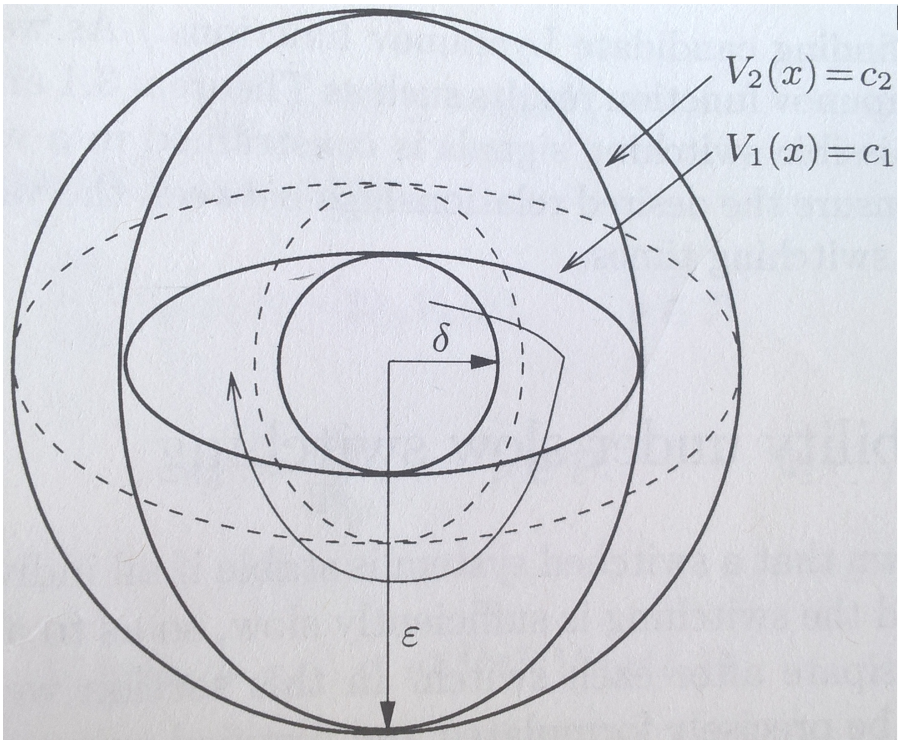


Figure 4.3: Proving Lyapunov stability in Theorem 4.2.3

Let's assume that all systems in (4.5) are globally exponentially stable. Then for each $p \in \mathcal{P}$ with some positive constants a_p, b_p and c_p, a Lyapunov equation V_p is:

$$a_p|x|^2 \leq V_p(x) \leq b_p|x|^2 \tag{4.10}$$

and

$$\frac{\partial V_p}{\partial x} f_p(x) \leq -c_p|x|^2 \tag{4.11}$$

Combination of the above renders:

$$\frac{\partial V_p}{\partial x} f_p(x) \leq -2\lambda_p V_p(x), \quad p \in \mathcal{P}$$

where

$$\lambda_p := \frac{c_p}{2b_p}, \quad p \in \mathcal{P} \tag{4.12}$$

This implies that:

$$V_p(x(t_0 + \tau_d)) \leq e^{-2\lambda_p \tau_d} V_p(x(t_0))$$

provided that $\sigma(t) = p$ for $t \in [t_0, t_0 + \tau_d)$.

Taking into consideration that our case is $\mathcal{P} = 1, 2$, the calculations can be simplified a bit. The assumption is that σ takes values 1 on $[t_0, t_1)$ and 2 on $[t_1, t_2)$, where $t_{i+1} - t_i \geq \tau_d$, $i = 0, 1$.So from the above inequality it can be concluded that:

$$V_2(t_1) \leq \frac{b_2}{a_1} V_1(t_1) \leq \frac{b_2}{a_1} e^{-2\lambda\tau_d} V_1(t_0)$$

and furthermore

$$V_1(t_2) \leq \frac{b_1}{a_2} V_2(t_2) \leq \frac{b_1}{a_2} e^{-2\lambda\tau_d} V_2(t_1) \leq \frac{b_1 b_2}{a_1 a_2} e^{-2(\lambda_+ \lambda_2)\tau_d} V_1(t_0). \tag{4.13}$$

It is now a matter of simple computation to find an explicit lower bound on τ_d which guarantees that the hypotheses of Theorem (4.2.3) are satisfies, implying that the switched systems (4.5) are globally asymptotically stable. It is even sufficient to ensure that:

$$V_1(t_2) - V_1(t_0) \leq -\gamma |x(t_0)|^2$$

for some $\gamma > 0$. Considering (4.13), this can be true when:

$$\left(\frac{b_1 b_2}{a_1 a_2} e^{-2(\lambda_1+\lambda_2)\tau_d} - 1\right) V_1(t_0) \leq -\gamma |x(t_0)|^2.$$

The above holds by the virtue of (4.10), when:

$$\left(\frac{b_1 b_2}{a_1 a_2} e^{-2(\lambda_1+\lambda_2)\tau_d} - 1\right) a_1 \leq -\gamma.$$

As γ can be any arbitrary positive number, all that is required is:

$$\frac{b_1 b_2}{a_2} e^{-2(\lambda_1+\lambda_2)\tau_d} < a_1$$

which can be rewritten in the form:

$$-2(\lambda_1 + \lambda_2)\tau_d < \log \frac{a_1 a_2}{b_1 b_2}$$

or putting the dwell time on one side:

$$\tau_d > \frac{1}{2(\lambda_1 + \lambda_2)} \log \frac{a_1 a_2}{b_1 b_2} \tag{4.14}$$

which gives the lower desired bound on the dwell time.

In essence all that was used in the above is the fact that there exists a positive constant μ such that:

$$V_p(x(t_0 + \tau_d)) \leq \mu V_q(x(t_0)) \quad \forall x \in \mathbb{R}^n, \quad \forall p, q \in \mathcal{P} \tag{4.15}$$

If the above inequality does not hold globally in the state space for all $\mu > 0$, then only local asymptotic stability can be established.

4.2.4. Finding the Dwell Time

With methodology established in the previous section, it is now needed to apply the theory to find and test a desired dwell time for the system. The two system that the switching will occur between is of course the position PI loop and the force PI loop. In order to achieve those PI loops, two different PI controllers are connected in series with the respective SISO systems, either position or force. Then the loop around the new system is closed with a unitary, negative feedback gain.

The new SISO systems with the PI controller and the feedback closed around them are described in a SS LTI form as:

$$\begin{aligned} x(k+1) &= A_p x(k) \\ y(k) &= C_p x(k) \end{aligned} \tag{4.16}$$

for the position loop, and the force loop:

$$\begin{aligned} x(k+1) &= A_f x(k) \\ y(k) &= C_f x(k). \end{aligned} \tag{4.17}$$

Due to the PI controller being introduced an additional state was introduced, as error from either the position or force feedback. From Equations (4.16) and (4.17), it is possible to tell that the SISO

systems are discrete. Naturally, the PI controllers are also discrete in such case. As a result the two PI controllers used were:

$$PI_p = \frac{51z - 50.99}{z - 1}$$

and

$$PI_f = \frac{12z - 12}{z - 1}$$

for position feedback loop and force feedback loop respectively. Both controllers were discretized using $1ms$ sampling time, and the integrators were approximated with the *forward Euler* rule.

In order to be able to guarantee stability, by the use of dwell time, Lyapunov candidate equations are needed for both the force PI loop and the position PI loop. As the systems are SS and LTI for both a *Quadratic Lyapunov* function was chosen of the form:

$$V(x) = x^T P x. \tag{4.18}$$

From [40] it is know, that in order for $V(x)$ to be a sound Lyapunov function, matrix $P > 0$ and:

$$\Delta V(x) = -x^T Q x \tag{4.19}$$

were matrices $P > 0$ and $Q > 0$ must satisfy the discrete Lyapunov equation:

$$A^T P A - P + Q = 0. \tag{4.20}$$

Converse theorem states that if A is Hurwitz, then there exists a quadratic Lyapunov function $V(x) = xTPx$ that proves the stability of A, i.e., there exists $P > 0$ and $Q > 0$ that satisfies the Lyapunov equation.

In order to ensure that difference Equation (4.19) is negative-definite, $\Delta V(x) < 0$, a trivial positive-definite, symmetric, orthogonal matrix was chosen. Subsequently $Q = I_8$, as there are now eight states. With the Q matrix picked MATLAB was used to see if a positive-definite matrix P can be found, which is the solution to Equation (4.20), for both position (4.16) and force (4.17) loops. Which, according to Equation (4.18), would give $V_p(x)$ and $V_f(x)$ respectively.

Proceeding, two solutions were found for the individual systems giving P_p and P_f. From Lyapunov theorem it can be concluded that as $P_p > 0$ and $P_f > 0$, and is obviously $Q > 0$, both loops are globally asymptotically stable. Subsequently, MATLAB gave the following solution:

$P_p = 1.0e + 09\times$

$$\begin{bmatrix} 0.0000052184 & -0.0022088833 & -0.0002612434 & 0.0000000227 & 0.0000160638 & 0.0000000010 & 0.0000013619 & 0.00001363079 \\ -0.0022088833 & 1.08414150 & 0.1281892869 & -0.0000111146 & -0.0067781021 & -0.0000004695 & 0.0001017172 & -0.0067834728 \\ -0.0002612434 & 0.1281892869 & 0.01515720 & -0.0000013142 & -0.0008011305 & -0.0000000555 & 0.0000118786 & -0.0008021060 \\ 0.0000000227 & -0.0000111147 & -0.0000013142 & 0.0000000011 & 0.0000000696 & 0 & -0.0000000010 & 0.0000000696 \\ 0.0000160638 & -0.0067781021 & -0.0008011305 & 0.0000000696 & 0.0000761658 & 0.0000000029 & 0.0000040783 & 0.0000437068 \\ 0.0000000010 & -0.0000004695 & -0.0000000555 & 0 & 0.0000000029 & 0.0000000010 & -0 & 0.0000000029 \\ 0.0000013619 & 0.0001017172 & 0.0000118786 & -0.0000000010 & 0.0000040783 & -0 & 0.0000386751 & -0.00000049970 \\ 0.00001363070 & -0.0067834728 & -0.0008021060 & 0.0000000695 & 0.0000437068 & 0.0000000029 & -0.0000005000 & 0.0000491120 \end{bmatrix} \tag{4.21}$$

for position Lyapunov function $V_p(x)$, and:

$P_f = 1.0e + 19\times$

$$\begin{bmatrix} 0.0000102423 & 0.0000003701 & 0.0000000597 & -0.0000000001 & 0.0000198748 & 0.0000000008 & -0.0000085821 & -0.0032007366 \\ 0.0000003701 & 0.0000000140 & 0.0000000022 & -0 & 0.0000007183 & 0 & -0.0000003101 & -0.0001156827 \\ 0.0000000597 & 0.0000000022 & 0.0000000003 & -0 & 0.0000001159 & 0 & -0.0000000500 & -0.0000186701 \\ -0.0000000001 & -0 & -0.0000000000 & 0 & -0.0000000002 & -0.0000000000 & 0 & 0.0000000357 \\ 0.0000198748 & 0.0000007183 & 0.0000001159 & -0.0000000002 & 0.0000385662 & 0.0000000015 & -0.0000166532 & -0.0062108901 \\ 0.0000000008 & 0 & 0.0000000000 & -0 & 0.0000000015 & 0.0000000000 & -0.0000000006 & -0.0000002517 \\ -0.0000085821 & -0.0000003101 & -0.0000000500 & 0 & -0.0000166532 & -0.0000000006 & 0.0000071910 & 0.0026819144 \\ -0.0032007366 & -0.0001156827 & -0.0000186701 & 0.0000000357 & -0.0062108901 & -0.0000002517 & 0.0026819144 & 1.0002298103 \end{bmatrix} \tag{4.22}$$

for position Lyapunov function $V_f(x)$.

With the Lyapunov functions established, a dwell time can be found such as to guarantee stable switching between the two systems. First a set of states before and after the switch were found. This enables finding a value of Lyapunov function for both position and force loop. As both loops are LTI

systems the lower bound for the dwell-time is found for only one switching instant, and without the loss of generality I can be assumed that results hold for all switching instances.

To begin find the lower bound on dwell time, first the values of Lyapunov functions were found. For the position loop before the switch and for the force loop after the switch. The functions have the following values:

$$Vp = 1.9884e + 07 \tag{4.23}$$

and

$$Vp = 2.7771e + 16. \tag{4.24}$$

Those values, according to Equation (4.10) lead to the bounds a_p and b_p to be found as:

$$a_p = 0.02 \quad b_p = 51 \tag{4.25}$$

and the limits of V_f were calculated as:

$$a_p = 1.413e - 11 \quad b_p = 7.079e + 10. \tag{4.26}$$

Now using the difference of Lyapunov functions, the values for c_p and c_f are chosen so as to give high stability margins λ which will be discussed next. As a result, c_p and c_f were chosen so as to satisfy Equation (4.11):

$$c_p = 150,000 \quad c_f = 15e + 14 \tag{4.27}$$

and subsequently the stability margins were found according to Equation (4.12) as:

$$\lambda_p = 1,471 \quad \lambda_f = 10,595. \tag{4.28}$$

Now the last thing to do is to find the lower bound on for stable switching between the two PI loops. This can be done by substituting Equations (4.25-4.28) into inequality (4.14), and the lower bound is found as:

$$\tau_d > 0.0048. \tag{4.29}$$

Now system can be tested for stability with any value of dwell time higher then $4.8ms$, and stability is guaranteed.

4.2.5. Testing of Switching Systems

As a lower bound on the dwell time, which gives big stability margins λ_p and λ_f (the reader is send to Appendix(A.1) for more information), was found it can be simulated on the system. If the plant becomes unstable due to fast switching, the introduction of dwell time guarantees to stabilise switching. Subsequently, the system was subjected to a sinusoidal wave of $25Hz$ and an amplitude of $8mm$. Such reference signal was chosen as it is in the middle of the plant operating frequency. With an allowable amplitude between $0mm$ to $8.5mm$ the greatest forces are experienced in the upper limits. As a result signals with high amplitudes are used.

The top part of Figure (4.4) presents the results of simulating the plant with active force limits and an actuator that is saturated by switching. Dwell-time was introduced into the controlling algorithm to prevent performance deterioration. In the light of the previous chapter a dwell-time of $5ms$ was used As is quite clear from dwell-time does not solve the issue. Subsequently, suggesting that switching is not too fast for the system to handle. The bottom part of the graph presents the behaviour of the plant if the actuator would not saturate, and the behaviour of the plant that is desired.

4.2.6. Integrator Wind up

The integrator windup is a well know and studied problem [20, 32]. Already in the days when controllers were still analogue, engineers had to face and solve this problem. The integrator windup comes from the controller requiring a still higher control input, but the control variable reached its saturation point. After all a valve can only be full open, twice fully open is not possible. An electric motor can only produce that much voltage or current, subsequently it can only deliver so much torque. Once saturation is reached any closed loop system becomes effectively an open loop. If integral action is still

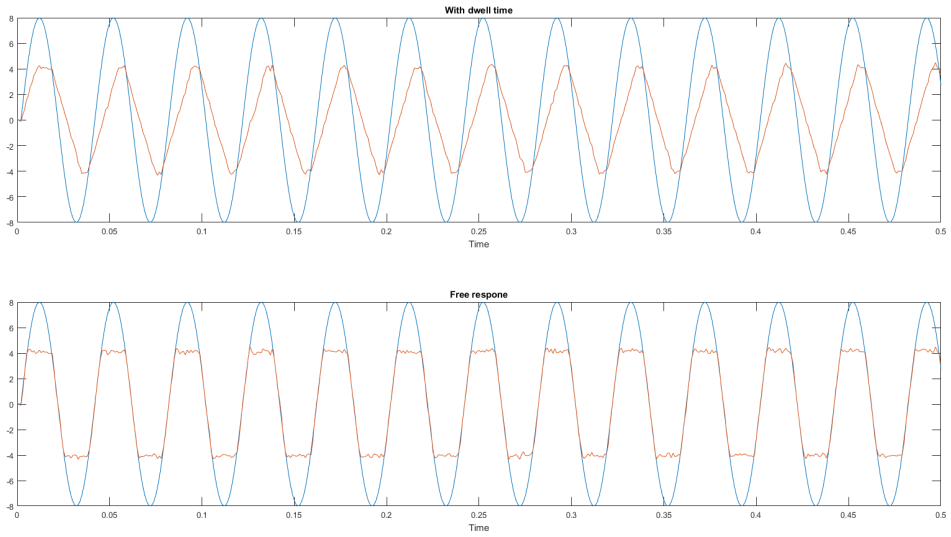


Figure 4.4: Plant response to a sinusoidal wave of $25Hz$ and amplitude of $8mm$, with force limits active and saturated actuator output. (top graph - saturated actuator, bottom graph - free response)

used the error continues to integrate, although it brings no effect. Subsequently the system is significantly delayed if the error was to change sign. Additionally, an open loop system is easy to destabilise.

One of the main reasons for actuator saturation and subsequently the integrator windup, is large change in setpoint. Such large change in setpoint can occur during a switching between to controllers with different setpoints. Exactly like the FCS controller does, between the force and position loops. The position controller has some given setpoint and based on that setpoint certain amount of force is required. The force controller has a setpoint equal to the force limit. When the difference between the force resulting from the position setpoint and the force limit is to large, motor torque output saturates (see Figure 4.5). During the saturation the error keeps being integrated which introduces a delay in the system once the system is again within limits. The sum of delays, from consecutive switches renders the system unstable.

The most well known approach to deal with actuator saturation and integrator windup, is back-calculation. Back-calculation works by recomputing the integral, when the output saturates, so that the new value gives an output at the saturation limit. It is advantageous not to reset the integrator instantaneously, but dynamically with a time constant T_s.

As shown in Figure (4.6), the system has an extra feedback path that is generated by measuring the control signal and forming an error signal (e_s) as the difference between current control input (v) and maximum allowable control signal (u). Error is fed to the input of the integrator through gain $1/T_s$. With no saturation $e_s = 0$, with saturation and $e_s \neq 0$ the normal feedback loop is broken. The new feedback path around the integrator takes over and the integrator output is driven to zero. As $e_s = v - u$, it follows that:

$$v = u + \frac{KT_s}{T_i}e. \tag{4.30}$$

Since the signals e and u have the same sign, it follows that v is always larger than u in magnitude. This prevents the integrator from winding up.

The rate at which the controller output is rest is dependant on the time constant T_s. The smaller T_s the faster the integrator will reset. Small reset times are not advised with PID controllers as spurious errors can cause saturation, which can accidentally reset the integrator. This is due to the derivative term. In this thesis only PI controllers are used. As a result, the problem does not apply.

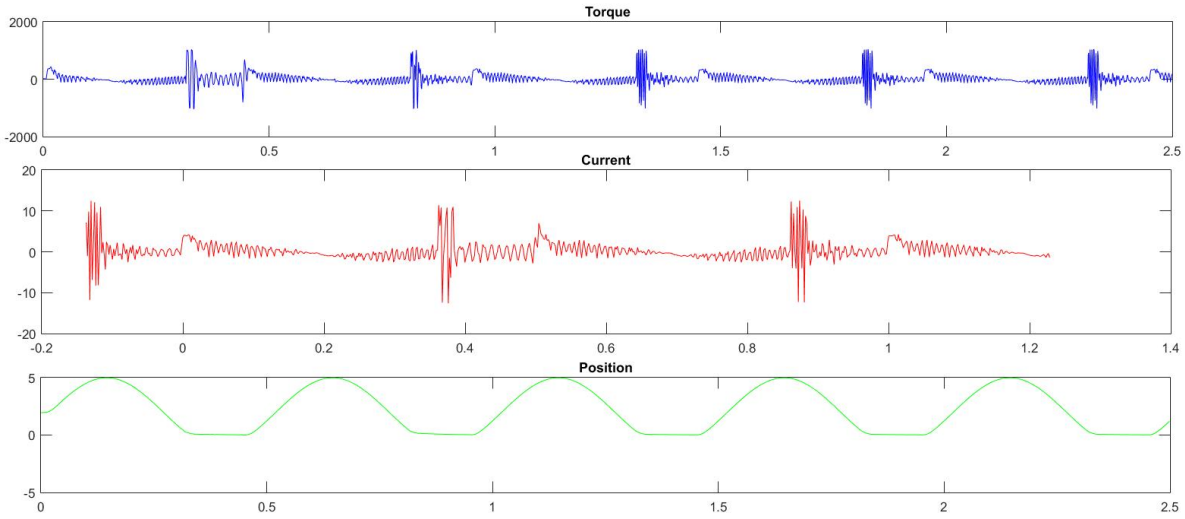


Figure 4.5: The lowest graph shows how the position is limited. During the times of limitation the two top graphs show how both current i_q and torque saturate and oscillate.

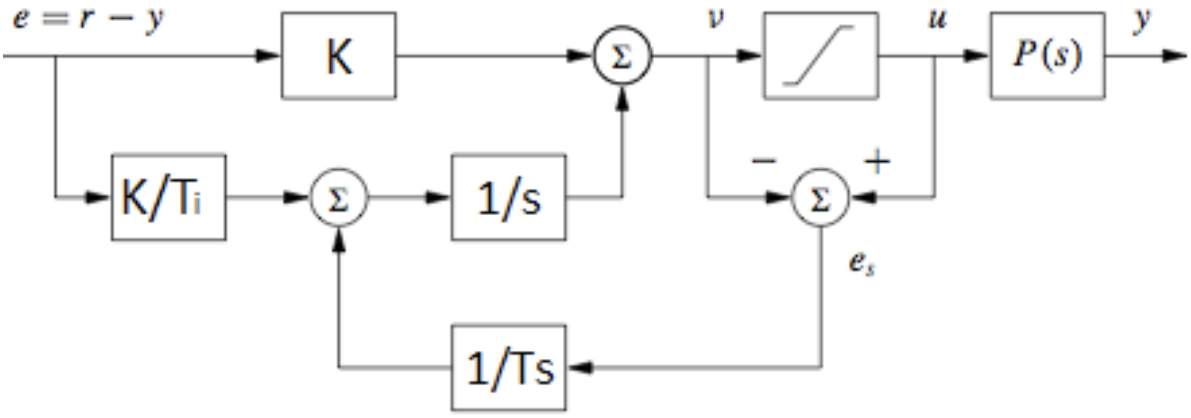


Figure 4.6: Block diagram of the anti-windup solution

4.2.7. Testing anti-windup

With the workings of the anti-windup is to be tested one the model. The arrangement for integrating an anti-wind-up strategy in a PI controller, presented in Figure (4.6), was implemented in Simulink. The error between the input and output of the saturation block was fed into the integrator line. This feedback has a gain parameter which is used for the purposes of tuning. To recreate the problem, system control input was set to a value that would normally saturate the actuator.

Three different simulations were run with, three different values of force limit. For the purposes of this simulations a sinusoidal wave was used as it is the most common type of reference used for the GSSA. The sinusoid had a frequency of $25Hz$. A sinusoid of this frequency was used as it is in the middle of the operational range of the GSSA (between $0.01Hz$ to $50Hz$). It was felt that the middle of the range is the best starting point. The amplitude of the sinusoid was set to $8mm$. This value was chosen due to the limit of the GSSA being $8.5mm$. The greatest forces and subsequently the biggest control effort is around the operational limits. As a result, it was felt that any solution that works for the limits must work elsewhere.

First the rest time on the integrator was set to one second as initial value was available. Unsurprisingly the results were not satisfactory. Figure (4.7) shows the behaviour of the plant when the control input is started and the force limit is set to the middle of range of forces **that saturate the control**

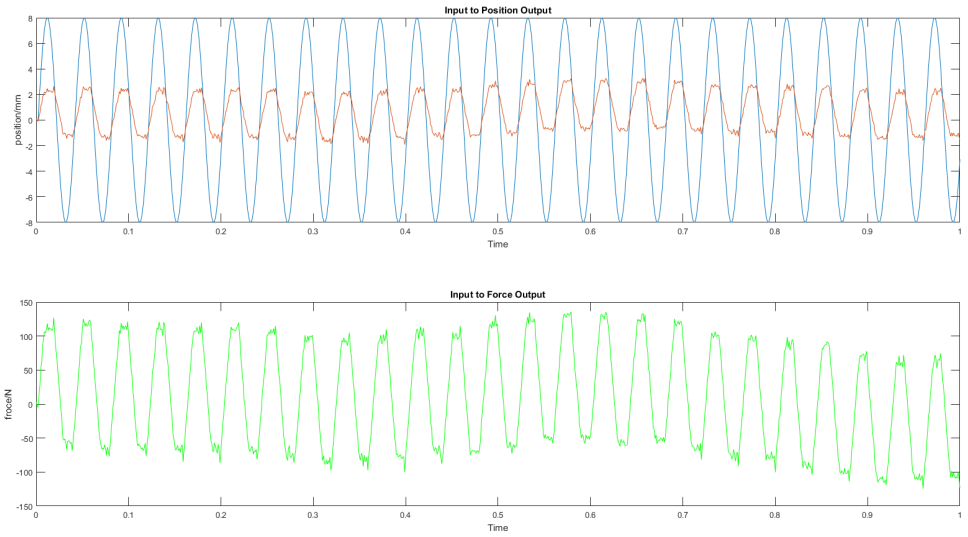


Figure 4.7: Response of the system to a 200 Newton force limit which saturates the control input.

input, which is a $200N$. The reset time was readjusted to $T_s = 5ms$. Quite clearly a satisfactory response can be observed. The system is stable and the performance is improved when put next to the lack of the anti-wind up strategy.

Further the same arrangement was tested, the reference signal was kept the same, and the integrator reset time was kept unchanged. The only difference came in the force limit. The limit was increased and now the force was limited to $50N$. The simulation results are presented in Figure (4.8). An improvement in the system output can be observed initially, but a trend away from the reference is clear the longer the simulation goes on. Lastly the force limit was change to $350N$, close to the upper bound of the force saturation range. The results are illustrated in Figure (4.9). Quite clearly the system response is not even close to following the reference.

From the performed simulations it is clear that one reset time for situation of saturation of the GSSA is not feasible. One rest time T_s is not even feasible for different force limits for the same signal

4.3. Model Predictive Control

Model Predictive Control (usually referred to as MPC) is a control approach originated in the 1950's. It first emerged in factories, when human engineers started changing inputs of manufacturing plants, because they knew that in a minute, hour, or day the process will experience a change of same sort. They wanted to act ahead to minimise the influence of this change on the manufacturing process. That is how the first predictive control was born.

For a long time prediction control was lost to various research labs, at big chemical companies. The academic community dismissed predictive control because: it seemed not very interesting, in the '60 and '70 academia was obsessed with control for SS whereas the process industry uses impulse and step models mainly, and lastly most importantly, predictive control did not originate from any University theories, but practical, hands on industrial experience.

The tides first started to change in the late 1970's when the first pioneers of MPCs emerged. In 1978 Richalet *et al.* [41] [42] Has published the first MPC algorithm which was called *Model Predictive Heuristic Control*(MPHC). Richalet emphasised that MPHC can be applied to control problems not suitable for simple PID control. He did not care much about optimal solutions or constraints thought. Independently, in the early '80, Cutler and Ramaker developed a different approach[43], till this day

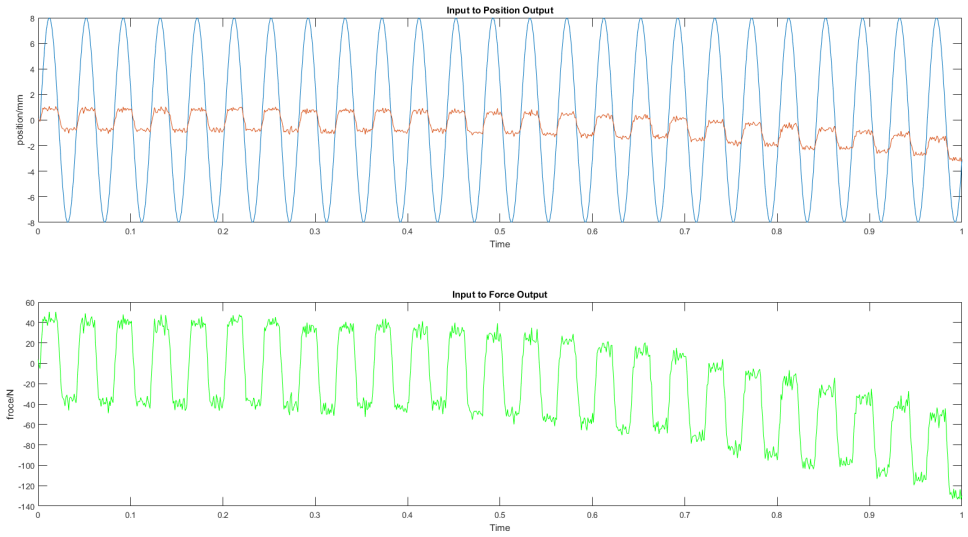


Figure 4.8: Response of the system to a 50 Newton force limit which saturates the control input.

known as *Dynamic Matrix Control* (DMC). DMC was developed with both constraints and optimality in mind. The DMC approach is still used those days in industrial applications.

Both the early variations and the ones that came latter, over time share some common features. All MPCs use an explicit internal model to predict the future behaviour of the plant. All predictive controllers find the optimum control law by optimising a cost function which uses the already mentioned model. Lastly all controllers work use the receding horizon idea. Predictive control uses the receding horizon principle. This means that after computation of the optimal control sequence, only the first control sample will be implemented, subsequently the horizon is shifted one sample and the optimisation is restarted with new information due measurements and feedback. Figure (4.10) explains the idea of receding horizon. At time k the future control sequence $u(k|k), \ldots, u(k+Nc-1|k)$ is optimised such that the performance-index $J(u,k)$ is minimised subject to constraints. At time k the first element of the optimal sequence $(u(k) = u(k|k))$ is applied to the real process. At the next time instant the horizon is shifted and a new optimisation at time $k+1$ is solved.

As model predictive control at it simplest can be summarised as finding the minimal, optimal solution to a quadratic programming problem, a link to Switching PI control can be made. Due to the structure of quadratic programming problems, it can be argued that MPC is a control approach which depending on the constraints active chooses a sufficient control gain. Deatailes can be found in Section (4.3.4).

4.3.1. MPC for PMSM control

Permanent magnet synchronous motors are very popular in the industry, as they can deliver consistent performance, are highly efficient, and can provided high accuracy. No surprise then, that as model predicative control has been successfully applied to many other industrial uses it will eventually make its way into power electronics. The transition has been slow and hesitant, since MPC is "traditional" a rather slow control strategy. As it was developed initially for chemical plants, sampling frequencies were in the order of hours or days. For PMSM's or other electric motors such sample times are absolutely unthinkable. Regardless much research was done to adapt MPC for the use on PMSM and other servo drives.

Although the foundations of MPC came over 25 years ago as presented in [44], or other examples discussed in previous chapter, only now it gained popularity. From the past few years the reader is referred to and to the references therein [45–48]. With all the research done, all of it focuses on replacing the most commonly used PI cascade control. The new control techniques focused on di-

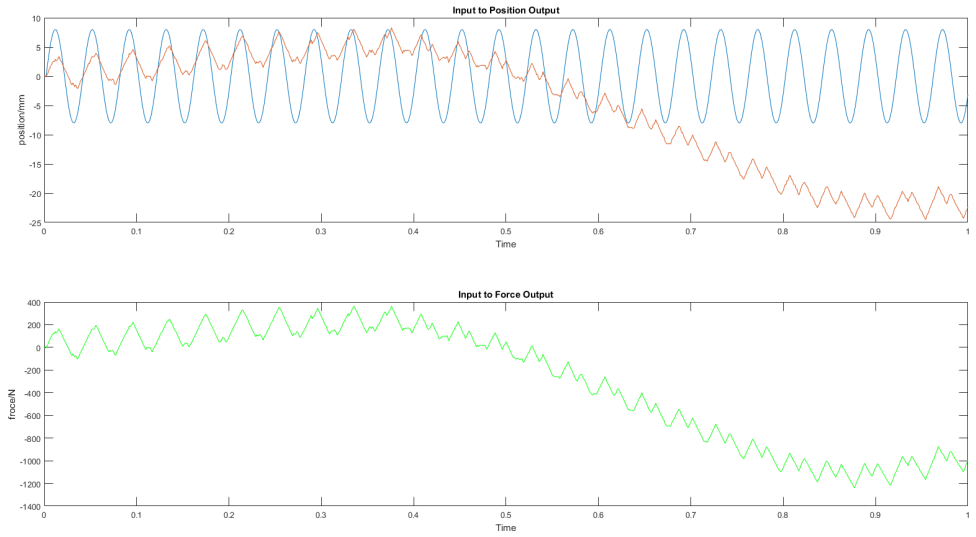


Figure 4.9: Response of the system to a 350 Newton force limit which saturates the control input.

rectly controlling PMSM's, either through velocity control, current control, or torque control. Several approaches are used like GPC, MDC, or more novel approaches such as model predictive direct torque control. The handy cap of such research is that, if applied for industrial use, PLC's would have to be reprogrammed with a much more complicated control approach.

Instead, in the following section, the focus is on designing a MPC which is a higher level controller over the traditional cascade control, yet brings the benefits of standard MPC by naturally handling constraints that a plan might have, either on its input, output, or states. Subsequently, system is treated as a whole, PMSM with the PLC. Additionally, it is a novel problem of controlling position and forces not of the PMSM itself, but a load that is actuated by the electric motor. As the load adds new dynamics, the control problem changes and is made more difficult.

4.3.2. Formulation of the MPC problem

In order to design a MPC first the optimisation problem is formulated. This involves choosing the right cost function to optimises and the right model for predictions, as well as establishing what limitations are imposed on the plant. Based on that information, the right optimisation algorithm must be chosen.

Cost Function

As the limitations imposed on the GSSA are mainly on the control input and the system outputs, the chose was made to work with the Generalised Prediction Control (GPC) like algorithm. Developed by *Clarke et al* [49, 50] in his later years. In GPC the cost function is formulated as follows:

$$J(\Delta u, k) = \sum_{j=N_m}^{N} \| \left(\hat{y}_p(k+j|k) - r(k+j)\right) \left(\hat{y}_p(k+j|k) - r(k+j)\right)^T \|_Q + \sum_{j=1}^{N_u} \|\Delta u(k+j-1|k)\Delta u(k+j-1|k)^T\|_R \quad . \tag{4.31}$$

with:

- $r(k)$ is the reference trajectory
- $y(k)$ is the output signal

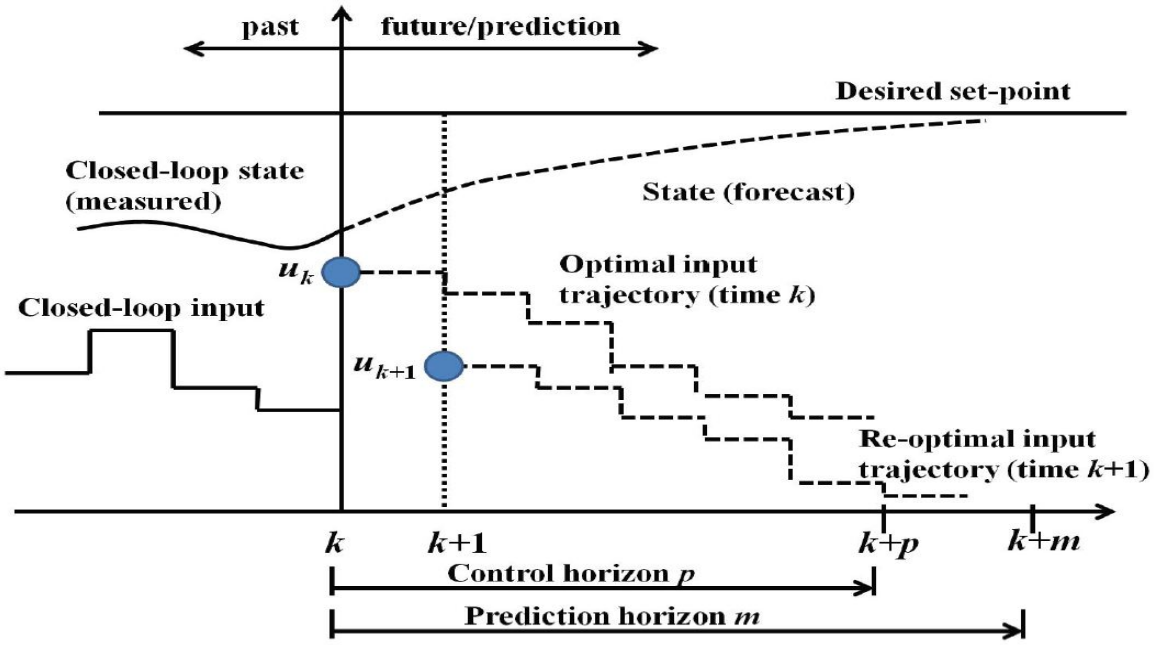


Figure 4.10: The Receding Horizon Principle

- $\hat{y_p}(k) = P(q)y(k)$ is the weighted process output signal
- $\Delta u(k)$ is the control increment signal
- N_m is the minimum cost-horizon
- N is the prediction horizon
- N_u is the control horizon
- Q is the weight on the control signal
- R is the weight on the control signal
- $P(q)$ is a polynomial with desired closed-loop poles

where $\hat{y}_p(k+j|k)$ is the prediction of $\hat{y}_p(k+j)$, based on knowledge up to time k, the increment input signal is $\Delta u(k) = u(k) - u(k-1)$ and $\Delta u(k+j) = 0$ for $j \geq N_u$. R determines the trade-off between tracking accuracy (first term) and control effort (second term).

Optimisation Model

The model ,developed for GSSA, is in the standard IO (input/output) LTI from. In discrete SS form system is defined as:

$$\begin{aligned} \dot{x}(k+1) &= A_o x(k) + B_o u(k) \\ y(k) &= C_o x(k) \end{aligned} \tag{4.32}$$

where $x(k)$ is the state, $u(k)$ is the control input, $y(k)$ is the output and d is a constant disturbance. In order to apply the GPC like scheme the input signal $u(k)$ itself is not used, but the input increment of the signal instead, defined as:

$$\Delta u(k) = u(k) - u(k-1) = (1 - q^{-1})u(k) = \Delta(q)u(k)$$

where $\Delta(q) = (1 - q^{-1})$ is denoted as the increment operator. Using the increment of the input signal implies that the model keeps track of the actual value of the input signal. The model needs to integrate the increments of the inputs to calculate the output corresponding with the input signals actually

applied to the process.

Subsequently, a change from the IO model to an IIO (Increment-input-output) model must be made. The relationship between the two models is simple:

$$A_i = \begin{bmatrix} I & C_o \\ 0 & A_o \end{bmatrix} \quad B_i = \begin{bmatrix} 0 \\ B_o \end{bmatrix} \quad B_i = \begin{bmatrix} I & C_o \end{bmatrix} \tag{4.33}$$

the new state of the system are:

$$x_i(k) = \begin{bmatrix} y(k-1) \\ \Delta x_o(k) \end{bmatrix}$$

where $\Delta x_o(k) = x_o(k) - x_o(k-1)$ is the increment of the original state. Then the new LTI state space realisation, is given by:

$$\begin{aligned} \dot{x}(k+1) =& A_i x_i(k) + B_i \Delta u(k) \\ y(k) =& C_i x_i(k) \end{aligned} \tag{4.34}$$

An interesting observation in comparing the IO model with the corresponding IIO model is the increase of the number of states with the number of outputs of the system. As can been seen from the state matrix A_i of the IIO model this increase in the number of states compared to the state matrix A_o of the IO model is related to the integrators required for calculating the actual process outputs on the basis of input increments. The additional eigenvalues of A_i are all integrators: eigenvalues $\lambda_i = 1$. The proof can be found in [51].

Constraints

In practice, industrial processes are subject to constraints. Specific signals must not violate specified bounds due to safety limitations, environmental regulations, consumer specifications and physical restrictions. As already established in previous sections, the GSSA has physical restriction as only a limited amount of torque can be produced by the system. On top of that input and output limits, are in place to ensure the safety of the system and the user.

Careful tuning of the controller parameters may keep these values away from the bounds. However, because of economical motives, the control system should drive the process towards the constraints as close as possible, without violating them, what in the case of the GSSA gives a wider operational range.

Therefore, predictive control employs a more direct approach by modifying the optimal unconstrained solution in such a way that constraints are not violated. This is achieved this by using the optimisation technique of quadratic programming (shortened to QP).

In the case of the GSSA the only input is limited as not to saturate the PMSM torque and both the outputs have a limits imposed, again to protect both the system and the user:

$$\begin{aligned} u_{min} \leq& u(k) \leq u_{max}, \quad \forall k \\ pos_{min} \leq& y_1(k) \leq pos_{max}, \quad \forall k \\ For_{min} \leq& y_2(k) \leq For_{max}, \quad \forall k. \end{aligned}$$

Under no circumstances the constraints may be violated, and the control action has to be chosen such that the constraints are satisfied. However, since in generally the future values of inputs and outputs are not known, often in the optimisation at time k, constraints of the form are used:

$$\psi(k+j|k) \leq \Psi(k+j) \tag{4.35}$$

where:

$$\psi(k+j|k) = \begin{bmatrix} u(k+j|k) \\ -u(k+j|k) \\ y_1(k+j|k) \\ -y_1(k+j|k) \\ y_2(k+j|k) \\ -y_2(k+j|k) \end{bmatrix}, \quad \Psi(k+j) = \begin{bmatrix} u_{max} \\ u_{min} \\ pos_{max} \\ pos_{min} \\ For_{max} \\ For_{min} \end{bmatrix}$$

this way all the limitations of the system are ready to be used in the optimisation problem.

The capability of predictive control to cope with signal constraints is probably the main reason for its popularity. It allows operation of the process within well-defined operating limits. As constraints are respected, better exploitation of the permitted operating range is feasible. In many applications of control, signal constraints are present, caused by limited capacity of liquid buffers, valves, saturation of actuators and the more. By minimising performance index, subject to these constraints,the best possible control signal within the set of admissible control signals was obtained.

To obtain a tractable optimisation problem, the input signal should be structured. In other words, the degrees of freedom in the future input signal $[u(k|k), u(k+1|k), \dots, u(k+N-1|k)]$ must be reduced. This can be done in various ways. The most common and most frequently used method is by using a control horizon.

$$u(k+j|k) = u(k+N_u-1|k) \quad \forall \quad j \geq N_u$$

When control horizon is used, the input signal is assumed to be constant from a certain moment in the future, denoted as control horizon N_u.

4.3.3. Formulating the optimisation problem

It is recalled that the cost function (4.31) for the optimisation problem was:

$$J(\Delta u, k) = \sum_{j=N_m}^{N} \| \left(\hat{y}_p(k+j|k) - r(k+j)\right) \left(\hat{y}_p(k+j|k) - r(k+j)\right)^T \|_Q + \sum_{j=1}^{N_u} \|\Delta u(k+j-1|k) \Delta u(k+j-1|k)^T\|_R$$

This can be rewritten in the following form:

$$J(\Delta u, k) = \|\mathcal{Z}(k) - \mathcal{T}(k)\|_{\mathcal{Q}}^2 + \|\Delta\mathcal{U}(k)\|_{\mathcal{R}}^2 \tag{4.36}$$

where:

$$\mathcal{Z}(k) = \begin{bmatrix} \hat{y}(k+H_m|k) \\ \vdots \\ \hat{y}(k+H_p|k) \end{bmatrix} \quad \mathcal{T}(k) = \begin{bmatrix} \hat{r}(k+H_m|k) \\ \vdots \\ \hat{r}(k+H_p|k) \end{bmatrix}$$

$$\Delta\mathcal{U}(k) = \begin{bmatrix} \hat{u}(k|k) \\ \vdots \\ \hat{u}(k+H_m-1|k) \end{bmatrix}$$

and the weighting matrices are given by:

$$\mathcal{Q} = \begin{bmatrix} Q(H_m) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & Q(H_p) \end{bmatrix} \quad \mathcal{R} = \begin{bmatrix} R(0) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & R(H_u-1) \end{bmatrix}$$

Now if it is assumed that the whole state vector is given and an observer is not used just yet, the matrix $\mathcal{Z}(k)$ can be found in the following way. Using the model (4.34) it can be predicted by iteration. Consequently, the following is obtained:

$$\begin{aligned} \hat{x}(k+1|k) &= A_i x(k) + B_i \hat{u}(k) \\ &\vdots \\ \hat{x}(k+H_p|k) &= A_i \hat{x}(k+H_p-1) + B_i u(k+H_p-1) \\ &= A_i^{H_p} x(k) + A_i^{H_p-1} B\hat{u}(k|k) + \dots + B\hat{u}(k+H_p-1) \end{aligned} \tag{4.37}$$

In the first line the prediction of the control signal $\hat{u}(k|k)$ was used rather then $u(k)$, as the control signal is not know prior the start.

Now the assumption is made that the control signal will only change at times $k, k+1, \dots, k+H_u+1$ and will remain constant after that. Subsequently, $\hat{u}(k+i|k) = \hat{u}(k+H_u-1)$ for $H_u \leq i \leq H_p - 1$. Later, the concern is with the rate of change of the control signal rather then just the control signal it self. So in order to make predictions for $\Delta\hat{u}(k+i|k) = \hat{u}(k+i|k) - \hat{u}(k+i-1|k)$, and as at time k the value of $u(k-1)$ is known:

$$\begin{aligned}
\hat{u}(k|k) &= \Delta\hat{u}(k|k) + u(k-1) \\
&\vdots \\
\hat{u}(k+H_u-1|k) &= \Delta\hat{u}(k+H_u-1|k) + \cdots + \Delta\hat{u}(k|k) + u(k-1)
\end{aligned}$$

and hence:

$$\begin{aligned}
\hat{x}(k+1|k) =& Ax(k) + B[\Delta\hat{u}(k|k) + u(k-1)] \\
&\vdots \\
\hat{x}(k+H_u|k) =& A^{H_u}x(k) + (A^{H_u-1} + \cdots + A + I)B\Delta\hat{u}(k|k) \dots \\
&+ B\Delta\hat{u}(k+H_u-1|k) \\
&+ (A^{H_u-1} + \cdots + A + I)Bu(k-1) \\
\hat{x}(k+H_u+1|k) =& A^{H_u+1}x(k) + (A^{H_u} + \cdots + A + I)B\Delta\hat{u}(k|k) \dots \\
&+ (A+I)B\Delta\hat{u}(k+H_u-1|k) \\
&+ (A^{H_u} + \cdots + A + I)Bu(k-1) \\
&\vdots \\
\hat{x}(k+H_p|k) =& A^{H_p}x(k) + (A^{H_p-1} + \cdots + A + I)B\Delta\hat{u}(k|k) \dots \\
&+ (A^{H_p-H_u} + \cdots + A + I)B\Delta\hat{u}(k+H_u-1|k) \\
&+ (A^{H_p-1} + \cdots + A + I)Bu(k-1)
\end{aligned}$$

The above predictions on $\hat{x}$ can be put together in a matrix-vector for for both transparency and ease of understanding:

$$\begin{bmatrix} \hat{x}(k+1|k) \\ \vdots \\ \hat{x}(k+H_u|k) \\ \hat{x}(k+H_u+1|k) \\ \vdots \\ \hat{x}(k+H_p|k) \end{bmatrix} = \begin{bmatrix} A \\ \vdots \\ A^{H_u} \\ A^{H_u+1} \\ \vdots \\ A^{H_p} \end{bmatrix} x(k) + \underbrace{\begin{bmatrix} B \\ \vdots \\ \sum_{i=0}^{H_u-1} A^iB \\ \sum_{i=0}^{H_u} A^iB \\ \vdots \\ \sum_{i=0}^{H_p} A^iB \end{bmatrix} u(k-1) +}_{past}$$

$$\underbrace{\begin{bmatrix} B & \dots & 0 \\ AB+B & \dots & 0 \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{H_u-1} A^iB & \dots & B \\ \sum_{i=0}^{H_u} A^iB & \dots & AB+B \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{H_p} A^iB & \dots & \sum_{i=0}^{H_p-H_u} A^iB \end{bmatrix} \begin{bmatrix} \Delta\hat{u}(k|k) \\ \vdots \\ \Delta\hat{u}(k+H_u-1|k) \end{bmatrix}}_{prediction} . \tag{4.38}$$

The last thing to find are the predictions of future inputs, which can be find as follows:

$$\begin{aligned}
\hat{y}(k+1|k) &= C\hat{x}(k+1|k) \\
\hat{y}(k+2|k) &= C\hat{x}(k+2|k) \\
&\vdots \\
\hat{y}(k+H_p|k) &= C\hat{x}(k+H_p|k)
\end{aligned}$$

or again in matrix-vector form:

$$\begin{bmatrix} \hat{y}(k+1|k) \\ \vdots \\ \hat{y}(k+H_p|k) \end{bmatrix} = \begin{bmatrix} C & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & C \end{bmatrix} \begin{bmatrix} \hat{x}(k+|k) \\ \vdots \\ \hat{x}(k+H_p|k) \end{bmatrix}. \tag{4.39}$$

Subsequently, if equation (4.38) is substituted into equation (4.39), output prediction matrix $\mathcal{Z}(k)$ becomes:

$$\begin{bmatrix} \hat{y}(k+1|k) \\ \vdots \\ \hat{y}(k+H_p|k) \end{bmatrix} = \begin{bmatrix} CA \\ \vdots \\ CA_i^{H_u} \\ CA_i^{H_u+1} \\ \vdots \\ CA^{H_p} \end{bmatrix} x(k) + \begin{bmatrix} B \\ \vdots \\ \sum_{i=0}^{H_u-1} A^i B \\ \sum_{i=0}^{H_u} A^i B \\ \vdots \\ \sum_{i=0}^{H_p} A^i B \end{bmatrix} u(k-1) + \tag{4.40}$$

$$\begin{bmatrix} B & \dots & 0 \\ AB+B & \dots & 0 \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{H_u-1} A^i B & \dots & B \\ \sum_{i=0}^{H_u} A^i B & \dots & AB+B \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{H_p} A^i B & \dots & \sum_{i=0}^{H_p-H_u} A^i B \end{bmatrix} \begin{bmatrix} \Delta\hat{u}(k|k) \\ \vdots \\ \Delta\hat{u}(k+H_u-1|k) \end{bmatrix}.$$

This leaves the $\mathcal{Z}(k)$ matrix in the form:

$$\mathcal{Z}(k) = \Psi x(k) + \Upsilon u(k-1) + \Theta \Delta\mathcal{U}(k) \tag{4.41}$$

with the vectors and matrix Ψ, Υ, Θ being defined as:

$$\Psi = \begin{bmatrix} CA \\ \vdots \\ CA_i^{H_u} \\ CA_i^{H_u+1} \\ \vdots \\ CA^{H_p} \end{bmatrix} \quad \Upsilon = \begin{bmatrix} CB \\ \vdots \\ C\sum_{i=0}^{H_u-1} A^i B \\ C\sum_{i=0}^{H_u} A^i B \\ \vdots \\ C\sum_{i=0}^{H_p} A^i B \end{bmatrix}$$

$$\Theta = \begin{bmatrix} CB & \dots & 0 \\ C(AB+B) & \dots & 0 \\ \vdots & \ddots & \vdots \\ C\sum_{i=0}^{H_u-1} A^i B & \dots & CB \\ C\sum_{i=0}^{H_u} A^i B & \dots & C(AB+B) \\ \vdots & \ddots & \vdots \\ C\sum_{i=0}^{H_p} A^i B & \dots & C\sum_{i=0}^{H_p-H_u} A^i B \end{bmatrix}. \tag{4.42}$$

Now with the help of the already defined vector $\mathcal{T}(k)$ and now derived vectors Ψ and Υ, a "trucking error" can be defined as:

$$\mathcal{E}(k) = \mathcal{T}(k) - \Psi x(k) - \Upsilon u(k-1). \tag{4.43}$$

This is a trucking error in the sense of the difference between future target trajectory and a system response where $\Delta\mathcal{U}(k) = 0$. If $\mathcal{E}(k)$ becomes 0, then it can implied that it must be true that $\Delta\mathcal{U}(k) = 0$. With this said the cost function can be defined as:

$$\begin{aligned} J(k) =& \|\Theta\Delta\mathcal{U}(k) - \mathcal{E}(k)\|_{\mathcal{Q}}^2 + \|\Delta\mathcal{U}(k)\|_{\mathcal{R}}^2 \\ =& [\Delta\mathcal{U}(k)^T\Theta^T - \mathcal{E}(k)]\mathcal{Q}[\Theta\Delta\mathcal{U}(k) - \mathcal{E}(k)] + \Delta\mathcal{U}(k)^T\mathcal{R}\Delta\mathcal{U}(k) \\ =& \mathcal{E}(k)^T\mathcal{Q}\mathcal{E}(k) - 2\Delta\mathcal{U}(k)^T\Theta^T\mathcal{Q}\mathcal{E}(k) + \Delta\mathcal{U}(k)^T[\Theta^T\mathcal{Q}\Theta + \mathcal{R}]\Delta\mathcal{U}(k). \end{aligned}$$

It can be said that the cost function is defined in the form:

$$J(k) = const - \Delta\mathcal{U}(k)^T\mathcal{G} + \Delta\mathcal{U}(k)^T\mathcal{H}\Delta\mathcal{U}(k) \tag{4.44}$$

where the following can be define:

$$\mathcal{G} = 2\Theta^T \mathcal{Q}\mathcal{E}(k) \tag{4.45}$$

and

$$\mathcal{H} = \Theta^T \mathcal{Q}\Theta + \mathcal{R}. \tag{4.46}$$

It is important to realise at this point that both $\mathcal{G}$ and $\mathcal{H}$ do not deepened on $\Delta\mathcal{U}$.

Now to formulate the optimisation problem as a QP optimisation. First, constraints of the system are considered. Those can be expressed in the general form:

$$F\begin{bmatrix}\mathcal{U}(k)\\1\end{bmatrix} \leq 0 \tag{4.47}$$

$$G\begin{bmatrix}\mathcal{Z}(k)\\1\end{bmatrix} \leq 0 \tag{4.48}$$

where it can defined $\mathcal{U}(k) = [\hat{u}(k|k)^T, \dots, \hat{u}(k+H_u-1|k)^T]^T$, analogously to $|delta\mathcal{U}(k)$. As as the defined cost function deals with the rate of change in control signal, all the constraints should be expressed in $\Delta\mathcal{U}(k)$.

If it is assumed that F is:

$$F = [F1, F2, \dots, F_{H_u}, f]$$

as each F_i is of size $q \times m$, and f has size $q \times 1$, the (4.47) becomes:

$$\sum_{i=1}^{H_u} F_i \hat{u}(k+i-q|k) + f \leq 0.$$

Since

$$\hat{u}(k+i-1|k) = u(k-1) + \sum_{j=0}^{i-1} \Delta\hat{u}(k+j|k)$$

(4.47)is written as:

$$\sum_{j=1}^{H_u} F_j \Delta\hat{u}(k|k) + \sum_{j=2}^{H_u} F_j \Delta\hat{u}(k+1|k) + \dots$$

$$+ F_{H_u}\Delta\hat{u}(k+H_u-1|k) + \sum_{j=1}^{H_u} F_j u(k-1) + f \leq 0.$$

Now to define $F_i = \sum_{j=i}^{H_u} F_j$ and $F = [F_1, \dots, F_{H-u}]$, the (4.47) matrix is put in the form:

$$F\Delta\mathcal{U}(k) \leq -F_1 u(k-1) - f \tag{4.49}$$

where the right-hand side of the inequality is a vector, which is known at time k. Subsequently, the constraints on the control signal were expressed in terms of the rate of change in the control signal.

With on limits converted the limits on the system output are next. With either full state measurement or by using an observer, (4.41) can be used to express (4.48) as:

$$G\begin{bmatrix}\Psi x(k) + \Upsilon u(k-1) + \Theta\Delta\mathcal{U}(k)\\1\end{bmatrix} \leq 0$$

if G matrix is divided in a similar fashion to F, $G = [\Gamma, g]$ becomes, where g is the last column of G. As follows the inequality is arranged as:

$$\Gamma[\Psi x(k) + \Upsilon u(k-1)] + \Gamma\Theta\Delta\mathcal{U}(k) + g \leq 0$$

or

$$\Gamma\Theta\Delta\mathcal{U}(k) \leq -\Gamma[\Psi x(k) + \Upsilon u(k-1)] - g \tag{4.50}$$

which lest us express the constraints on the system output in the terms of rate of change of the control signal.

Both sets of limits can now be combined into one, which gives the limits for the QP optimisation problem as:

$$\begin{bmatrix} F \\ \Gamma\Theta \end{bmatrix} \Delta\mathcal{U}(k) \leq \begin{bmatrix} -F_1 u(k-1) - f \\ -\Gamma[\Psi x(k) + \Upsilon u(k-1)] - g \end{bmatrix}. \tag{4.51}$$

If full state measurement is not available (as is the case in most real life applications), an observer can be sued and $x(k)$ becomes $\hat{x}(k|k)$.

With the limits for the optimisation established, the optimisation problem can be formulated in a way to find the optimal change in the control signal, with the help of equation (4.44) is:

$$\begin{aligned} \min_{\Delta\mathcal{U}} \quad & \Delta\mathcal{U}(k)^T \mathcal{H} \Delta\mathcal{U}(k) - \Delta\mathcal{U}(k)^T \mathcal{G} \\ s.t. \quad & \begin{bmatrix} F \\ \Gamma\Theta \end{bmatrix} \Delta\mathcal{U}(k) \leq \begin{bmatrix} -F_1 u(k-1) - f \\ -\Gamma[\Psi x(k) + \Upsilon u(k-1)] - g \end{bmatrix} \end{aligned} \tag{4.52}$$

This optimisation problem can be put in a more standard form:

$$\begin{aligned} \min_{\theta} \quad & \frac{1}{2}\theta^T \Phi \theta + \phi^T \theta \\ s.t. \quad & \Omega\theta \leq \omega \end{aligned} \tag{4.53}$$

where the following are defined:

$$\theta = \Delta\mathcal{U}, \quad \Phi = 2\mathcal{H}, \quad \phi = \mathcal{G}.$$

This is the standard formulation of a quadratic programing optimisation problem. The QP problem types are well known and several standard algorithms exists which can solve a well posted QP problem very fast. If the Hessian of our cost function (that is Φ) is *positive definite* or *semi-positive definite*, then the MPC needs to solve a *convex* optimisation problem. Convexity of the problem guaranties that a minimal solution exists. If that solution is within the convex quadratic surface, constrained by hyperplanes due to inequality constraints, then the optimisation problem is *feasible*.

The field of convex optimisation is vast. The interested reader is referred to [52, 53] for further information on convex optimisation and to [54] for convex optimisation in control design.

4.3.4. Solving the MPC optimisation problem and Controller structure

Quadratic programming optimisation problems, depending on their structure can be solved through a number of optimisation algorithms. If the function is quadratic with linear constraints the modified simplex algorithm can be used. If the constraints are not linear, based on their structure either Cutting plane and Ellipsoid algorithms can be used or Lagrange Multiplayer and SQP are available. That said the two most widely used and implemented are the *active set method* [52] and now taking over (last 10 years) *interior point method* [55, 56].

Several software programs exist that utilise either one or both of the optimisation algorithms. One such program is MatLab and its Optimisation Toolbox and by extension the MPC Toolbox, also used in this thesis.

Active Set method

Active-set methods are two-phase iterative methods that provide an estimate of the active set at the solution. In the first phase (the feasibility phase or phase 1), the objective is ignored while a feasible

point is found for the constraints $Ax = b$ and $f \leq Dx$. In the second phase (the optimality phase or phase 2), the objective is minimised while feasibility is maintained. For efficiency, it is been official if the computations of both phases are performed by the same underlying method. The two-phase nature of the algorithm is reflected by changing the function being minimised from a function that reflects the degree of infeasibility to the quadratic objective function. For this reason, it is helpful to consider methods for the optimality phase first.

Given a feasible point x_0, active-set methods compute a sequence of feasible iterates $\{x_k\}$ such that $x_{k+1} = x_k + \alpha_k p_k$ and $\varphi(x_{k+1}) \leq \varphi(x_k)$, where p_k is a nonzero search direction and k is a non-negative step length. Active-set methods are motivated by the main result of Farkas' Lemma, which states that a feasible x must either satisfy the first-order optimality conditions or be the starting point of a feasible descent direction

Interior Point method

Optimisation problem can roughly be distinguished into two classes: interior and exterior point methods. The interior point methods operate in the interior of the feasible region, while exterior point methods try to obtain a solution from outside the feasible region. So, interior point methods (IPMs) are not new. In fact, already in the 1960s some interior point methods for nonlinear programming, e.g. barrier methods, were proposed and analyzed. In the 1960s and 1970s, due to the fast computer developments, researchers became interested in the complexity of methods. A method was called polynomial if the number of arithmetic operations required by the method to solve the problem, is bounded from above by a polynomial in the problem size. Such a polynomial method was considered to be an efficient method.

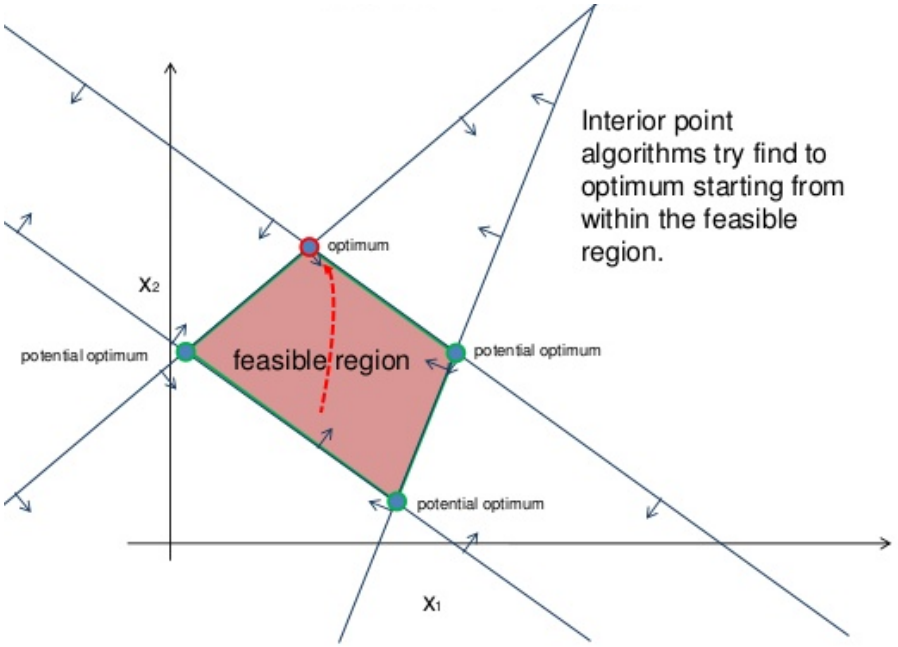


Figure 4.11: Graphical representation of the Interior point algorithm.

In 1979, Khachian published the first polynomial algorithm for LP. This ellipsoid method was based on some nonlinear programming techniques developed by Shor and Yudin and Nemirovsky. Contrary to the simplex method, the ellipsoid method does not use the combinatorial structure of LP. Although this first polynomial method has many important theoretical consequences, i t soon appeared that in practice this method is hopelessly slow. Consequently, the quest changed to find a polynomial method for LP also being efficient in practice. Karmarkar answered this question in an epoch-making work in 1984. He proposed a new polynomial method and claimed that this so-called *projective method* could solve large scale linear programming problems as much as 100 times faster than the simplex method.

This claim was received with much scepticism. Nowadays, it has become clear that Karmarkar's work has opened a new research field, the field of interior point methods, which has yielded many theoretical and practical jewels.

Interior-point methods solve problems iteratively such that all iterates satisfy the inequality constraints strictly. They approach the solution from either the interior or exterior of the feasible region but never lie on the boundary of this region. Each iteration of the method is computationally expensive but can make significant progress towards the solution. These particular characteristics are what separates interior-point methods from other methods. To set up the equations enabling us to design the interior-point methods the general theory on constrained optimisation is used, by defining a Lagrangian function and setting up Karush-Kuhn-Tucker (KKT) conditions for the QP's that is to be solved. The KKT-conditions, or optimality conditions, are conditions that must be satisfied for a vector x to be a solution of a given QP.

MPC controller structure

Now that it was established how to formulate a MPC problem and what are the possibly methods of solving the QP optimisation with the MPC, the controller structure is discussed inn detail.

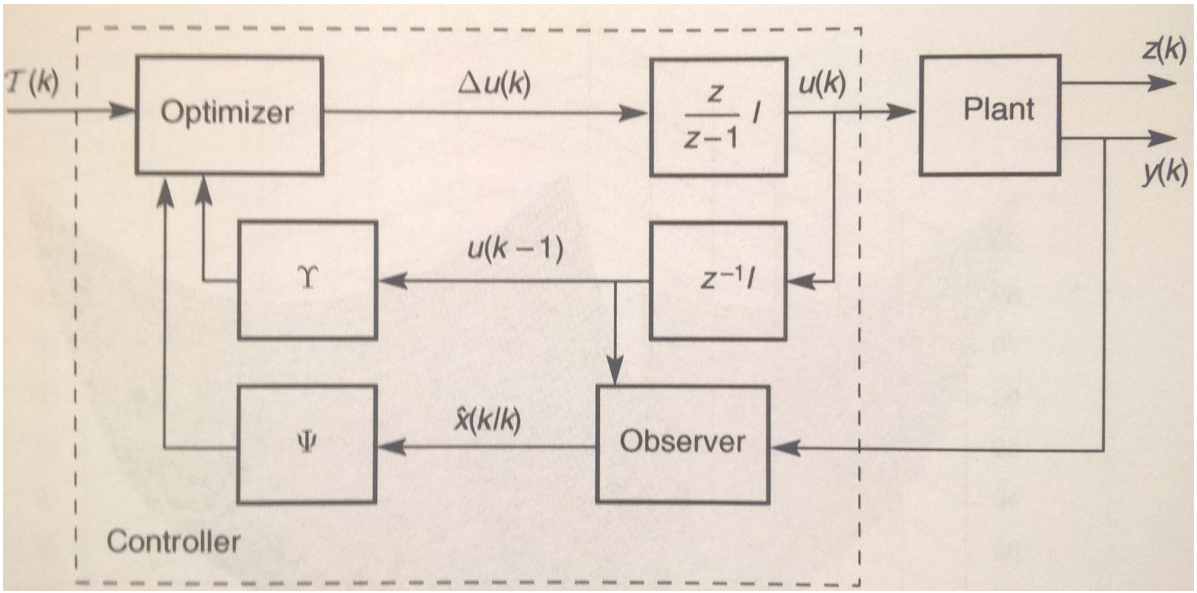


Figure 4.12: Structure of a MPC controller with active constraints

The structure of a MPC controller can be found in Figure (4.12). As presented the MPC controller is essentially an optimiser which is supplied with system data and based on them tries to find the most optimal control signal whilst considering the past and predicting the future. The the presented schematic it is indicated that an observer is used, as an observer has to be used due to the lack of full state measurement. At this juncture, it is interesting that similarities between a advanced MPC controller and simple PI control can be found. If the assumption is made that the set of active constraints is constant and the knowledge of that active set prior the optimisation time step is known, in principle it can be said that the optimisation is subject to equality constraints. Subsequently, the solution to the optimisation problem K_{MPC} would be a constrained LTI control law. For a fixed set of equality constraints.

In reality the set of active constraints changes over time. If one would precompute the solutions to all that possibly combinations of active sets of constraints, the MPC would come down to choosing the right LTI control law for the set of constraints that is active. In that respect a MPC controller can be thought of as a gain scheduler, that is trying to find the right PI controller for the currently imposed limits. Intuitively one can say that such reasoning is dangerous as the possible number of active

constrain combinations can become quite big, quite quickly. To be exact 2^q where q is the number of rows of our Ω matrix. Subsequently, such approaches to solving a MPC optimisation problem are very rarely attempted and methods like the already discussed active set or interior point are used.

4.3.5. MPC Stability

Predictive control design does not give an a prior guaranteed stabilising controller, on contrary to methods like L_Q, H_∞ or $l_1 - control$. Such methods have built-in stability properties. Subsequently, the stability of the predictive controller needs to be analysed. Two main cases can be distinguished, namely the unconstrained and constrained case. It is very important to notice that a predictive controller that is stable for the unconstrained case is not automatically stable for the constrained case.

Unconstrained

If no limits are active, the stability problem becomes one of a LTI SS output-feedback system. To analyse such, first the whole system description with the observer and controller included in the system description. So to be able to find all the dynamics of the complete system. First the state vector needs to be defined as containing the actual states and the estimation of those states, $x = [x \quad \hat{x}]^T$. To establish the whole system description, the equations defining all the states are needed.

First the system states are shown, with the dynamics introduced by the MPC controller added to them. If no constraints are active, the MPC controller becomes a linear controller with a static gain. The MPC control law can be defined us:

$$\Delta u(k) = K_{MPC}\mathcal{E}(k)$$

where

$$K_{MPC} = [1, 0, \dots, 0]\mathcal{H}^{-1}\Theta^T Q$$

using the augmented IIO system model (4.34) that takes $\Delta u(k)$ as system inputs, the control law can be substituted in to achieve the LTI SS control system:

$$\dot{x}(k+1) = A_i x(k) + B_i K_{MPC}\mathcal{E}(k) \tag{4.54}$$

were equation 4.43 gives the error, $\mathcal{E}(k) = \mathcal{T}(k) - \Psi x(k) - \Upsilon u(k-1)$. Subsequently, the system equation becomes:

$$\dot{x}(k+1) = A_i x(k) + B_i K_{MPC}(\mathcal{T}(k) - \Psi x(k) - \Upsilon u(k-1))$$

if expanded:

$$\dot{x}(k+1) = (A_i - B_I K_{MPC}\Psi)x(k) + B_i K_{MPC}\mathcal{T}(k) - B_i K_{MPC}\Upsilon u(k-1). \tag{4.55}$$

Now equation (4.55) gives the dynamics of the closed loop SS system with the MCP controller. To have full system description the dynamic equations of the observer need to be found. The estimations of the states can be found as follows:

$$\begin{aligned}
\dot{\hat{x}}(k+1|k) &= A\hat{x}(k|k) + B\Delta u(k) + L(y - \hat{y}(k|k)) \\
\hat{y}(k|k) &= C\hat{x}(k|k) \\
y(k) &= Cx(k) \\
\Delta u(k) &= K_{MPC}\mathcal{E}(k) = K_{MPC}(\mathcal{T}(k) - \Psi\hat{x}(k|k) - \Upsilon u(k-1)).
\end{aligned}$$

If the equations are substituted into their right places the resultant observer equations become:

$$\dot{\hat{x}}(k+1|k) = A\hat{x}(k|k) + BK_{MPC}\big(\mathcal{T}(k) - \Psi\hat{x}(k|k) - \Upsilon u(k-1)\big) + L\big(Cx(k) - C\hat{x}(k|k)\big)$$

after expansion the equation becomes:

$$\dot{\hat{x}}(k+1|k) = (A - LC - BK_{MPC}\Psi)\hat{x}(k|k) + LCx(k) + BK_{MPC}\mathcal{T}(k) - BK_{MPC}\Upsilon u(k-1). \tag{4.56}$$

Equation (4.55) and (4.56) can be combined into matrix form to create the system equations. As only the state matrix is necessary for stability analysis, it is the focus. Now, combining the two equations the following state matrix can be found:

$$\begin{bmatrix} \dot{x}(k+1) \\ \dot{\hat{x}}(k+1|k) \end{bmatrix} = \begin{bmatrix} A & -BK_{MPC}\Psi \\ LC & A - LC - BK_{MPC}\Psi \end{bmatrix} \begin{bmatrix} x(k) \\ \hat{x}(k|k) \end{bmatrix} \tag{4.57}$$

As the above matrix is not in its most pleasant form the following coordinate change can be defined:

$$\begin{bmatrix} x \\ \tilde{x} \end{bmatrix} = \begin{bmatrix} I & 0 \\ I & -I \end{bmatrix} \begin{bmatrix} x \\ \hat{x} \end{bmatrix} = T \begin{bmatrix} x \\ \hat{x} \end{bmatrix}$$

with a very elementary calculation it can be shown that matrix in equation (4.57) becomes:

$$T \begin{bmatrix} A & -BK_{MPC}\Psi \\ LC & A - LC - BK_{MPC}\Psi \end{bmatrix} T^{-1} = \begin{bmatrix} A - BK_{MPC}\Psi & -BK_{MPC}\Psi \\ 0 & A - LC \end{bmatrix}.$$

The coordinate gives the new system expression as:

$$\begin{bmatrix} \dot{x}(k+1) \\ \hat{x}(k+1|k) \end{bmatrix} = \begin{bmatrix} A - BK_{MPC}\Psi & -BK_{MPC}\Psi \\ 0 & A - LC \end{bmatrix} \begin{bmatrix} x(k) \\ \hat{x}(k|k) \end{bmatrix}. \tag{4.58}$$

The new matrix is an upper triangular matrix, with its eigenvalues on the diagonal. As the dynamics of a system, and by extension the eigenvalues, do not change under coordinate it can be inferred that the new matrix describes the system exactly like the old one.

With the eigenvalues, or poles, on the diagonal of the state matrix it can be concluded that if the controller stabilises the system, and the observe has a gain that results in stability the following can be sated with no loss of generality:

- the eigenvalues of $(A - BK_{MPC}\Psi)$ must be strictly inside of the unit circle
- the eigenvalues of $(A - LC)$ must be strictly inside of the unit circle

In conclusion the above are stated as necessary and sufficient conditions for a unconstrained MPC controller to be nominal stable. The above conditions are no longer feasible if one or more constraints are activated. Then the MPC becomes an non-linear controller.

Constrained

The case of proving stability for constraint MPC controllers is far from trivial. The approach most commonly used and well known, is the principle of infinite horizon. It was first originally introduced in [57], where it was presented how an infinite-horizon constrained control problem can be approximated by finite-receding-horizon control with terminal constraint. It has been well known that making the prediction horizon infinity $N = \infty$, guaranties stability for controlled systems, the problems arose on how to deal with constraints in such situation. The work of Rawlings and Musk was the breakthrough that was needed. The solution is to parameterize the predictive control problem in terms of finite number of parameters, even if the prediction horizon is infinity, so as to still be able to solve the optimisation problem over a finite-dimension space. As it turns out it is even possible to keep the problem quadratic.

With the above in mind an infinite horizon is introduced into the MPC optimisation problem. To be able to prove that infinite horizon guarantees stability for an MPC constrained case a few things need to be discussed first. Foremost an assumption needs to be made.

Assumption 1. *The following matrix:*

$$M = \begin{bmatrix} uI - A & -B \\ -C & -D \end{bmatrix} \tag{4.59}$$

does not drop rank for all $1 \leq \|u\|$.

Assumption (1) guarantees that the M matrix is full rank. Additionally, it is assumed that a steady state of the system $(x_{ss}, u_{ss}) = (x_{ss}, 0)$, is unique. Lastly, the redder is reminded about the Krasovskii-LaSalle principle [58], for asymptotic stability. If a function $V(k)$ can be found such that:

- $0 < V(k), \forall x \neq x_{ss}$
- $\Delta V(k) = V(k) - V(k-1) \leq 0, \forall x$

additionally, $V(k) = \Delta V(k) = 0$ has to for $x = x_{ss}$, and when the set $\Delta V(k) = 0$ contains no trajectory of the system with the expectation of the trivial trajectory $x(k) = x_{ss}$ for $0 \leq k$. It can be concluded that, according to the Krasovskii-LaSalle principle the steady-state $x(k) = x_{ss}$ is asymptotically stable.

Now it is possible to show that a controller with an infinite horizon $N = \infty$ guaranties asymptotic stability of the system. To do so the following theorem is proposed:

Theorem 4.3.1. *For an SS LTI system with the following description:*

$$x(k+1) = Ax(k) + B\Delta u(k)$$
$$y(k) = Cx(k)$$

which satisfies Assumption (1). A set $\Delta\tilde{u}$ can be defined:

$$\Delta\tilde{u} = \begin{bmatrix} \Delta u(k|k) \\ \vdots \\ \Delta u(k+N_u-1|k) \end{bmatrix},$$

where $N_u \in \mathbb{Z}^+ \cup \infty$. The performance index can defined as:

$$\min_{\Delta u} \quad J(\Delta u, k)$$
$$J(\Delta u, k) = \sum_{j=N_m}^{\infty} \|(\hat{y}(k+j|k)\|_Q^2 + \sum_{j=1}^{N_u} \|\Delta u(k+j-1|k)\|_R^2.$$

The predictive control law minimising this performance index results in a stabilising controller.

Proof. First let us define the function:

$$V(k) = \min_{\Delta u} \quad J(\Delta u, k) \geq 0 \tag{4.60}$$

and the optimiser $\Delta\tilde{u}^*(k)$ becomes:

$$\Delta\tilde{u}^*(k) = \arg\min_{\Delta u} J(k)$$

where the vector $\Delta\tilde{u}^*(k)$ is:

$$\Delta\tilde{u} = \begin{bmatrix} \Delta u^*(k|k) \\ \vdots \\ \Delta u^*(k+N_u-1|k) \end{bmatrix}, \quad \text{for} \quad N_u < \infty$$

and

$$\Delta\tilde{u} = \begin{bmatrix} \Delta u^*(k|k) \\ \Delta u^*(k+1|k) \\ \vdots \end{bmatrix}, \quad \text{for} \quad N_u = \infty$$

where $\Delta u^*(k+j|k)$ means the optimal input value to be applied a time $k+j$ as calculated at time k. Let $\Delta u^*(k)$ be the optimal performance signal at time $k+j$ when the optimal input sequence $\Delta u^*(k)$ computed at time k is applied. So then it can be said:

$$V(k) = \sum_{j=N_m}^{\infty} \|(\hat{y}^*(k+j|k)\|_Q^2 + \sum_{j=1}^{N_u} \|\Delta u^*(k+j-1|k)\|_R^2$$

Based on $\Delta u^*(k)$ and the stabilising modification method a vector can be constructed $\Delta u_{sub}(k)$. This vector can be seen as a suboptimal input sequence for the $k+1$ optimisation sample step. The goal is to construct $\Delta u_{sub}(k+1)$ such that:

$$J(\Delta u_{sub}, k+1) \leq V(k).$$

Now the following must be computed:

$$V(k+1) = \min_{\Delta u} J(\Delta u_{sub}, k+1)$$

to find:

$$V(k+1) = \min_{\Delta u(k+1)} J(\Delta u, k+1) \leq J(\Delta u_{sub}, k+1) \leq V(k).$$

For the infinite horizon case a suboptimal control sequence can be defined as follows:

$$\Delta \tilde{u}_{sub} = \begin{bmatrix} \Delta u^*(k+1|k) \\ \vdots \\ \Delta u^*(k+N_u-1|k) \\ u_{ss} \end{bmatrix}, \quad \text{for} \quad N_u < \infty,$$

and

$$\Delta \tilde{u}_{sub} = \begin{bmatrix} \Delta u^*(k+1|k) \\ \Delta u^*(k+2|k) \\ \vdots \end{bmatrix}, \quad \text{for} \quad N_u = \infty,$$

and find:

$$V_{sub}(k+1) = J(\Delta u_{sub}, k+1).$$

If found this becomes equal to:

$$V_{sub}(k) = \sum_{j=N_m}^{\infty} \|(\hat{y}^*(k+j|k)\|_Q^2 + \sum_{j=1}^{N_u} \|\Delta u^*(k+j-1|k)\|_R^2$$

and so:

$$V_{sub}(k+1) \leq V(k).$$

No it can be observed:

$$V(k+1) = \min_{\Delta u(k+1)} J(\Delta u, k+1) \leq J(\Delta u_{sub}, k+1)$$

subsequently it can be said that $V(k+1) \leq V(k)$. Using the fact that $\Delta V(k) = V(k) - V(k-1) \leq 0$ and that, based on Assumption (1), the set $\Delta V(k) = 0$ contains no trajectory of the system with the exception of the trivial trajectory $x(k) = x_{ss}$ for $0 \leq k$, the Krasovskii-LaSalle principle states that the steady-state is asymptotically stable. □

As is proven above, as far as stability is concerned, it makes no difference whether $N_u = \infty$ or $N_u < \infty$. For $N_u = \infty$ the predictive controller becomes equal to the optimal LQ-solution (Bitmead, Gevers and Wertz [59]). Before the work of Rawlings and Muske, as discussed already, the issue of constrains was a big obstacle in the use of infinite horizon. Now because of the finite number of control actions that can be applied, the constraint-handling has become a finite-dimensional problem instead of an infinite-dimensional problem.

4.3.6. Designing a MPC

First, iterating again through the cost function and all the tuning parameters:

$$J(\Delta u, k) = \sum_{j=N_m}^{N} \|(\hat{y}_p(k+j|k) - r(k+j))\|_Q^2 + \sum_{j=1}^{N_u} \|\Delta u(k+j-1|k)\|_R^2 \tag{4.61}$$

with:

- N_m = minimum cost-horizon
- N = prediction horizon

- N_u = control horizon
- Q, R = weighting factor
- $P(q)$ = tracking filter (observer)

The above parameters tuning parameters can be thought of as control dials on a computer. In that sense they are very similar to gain factors in PID type controllers. Although MPC tuning is far from being automated, the same as with the Zigler-Nicholas tuning method for PID, some rules of thumb are available for MPC as well.

Discretization of the Plant

As the MPC makes predictions at discrete time intervals, the model which it will use to make those predictions also needs to operate on discrete interval samples. Subsequently developed continuous time plant needs to be discretized, to fit its purposes. Several methods exist to turn a continuous time signal/function into a discrete equivalent. The three most commonly used are:

- Zero Order Hold (Zoh),
- Forward rectangular rule (Foh),
- The trapezoidal rule (Tustin).

Where the given system was discretized by using the Zoh method.

In [60], Astrom and Wittenmark present how to achieve a discrete plant from a continuous system description. The use of matrix exponentials is necessary and the input is assumed to be piecewise constant. Subsequently the system matrices can be found as:

$$\Phi_{DZ} = e^{A_{GSSA} \times h} \quad \Gamma_{DZ} = \int_0^h e^{A_{GSSA} s} ds \times B_{GSSA}$$

where is the sampling period between one time interval and the next, $h = t_{k+1} - t_k$. Subsequently, the discrete SS system takes the form:

$$\begin{aligned} x(k+1) &= \Phi_{DZ} x(k) + \Gamma u_{DZ} u(k) \\ y(k) &= C_{DZ} + D_{DZ} u(k) \end{aligned} \tag{4.62}$$

where $C_{DZ} = C_{GSSA}$ and $D_{DZ} = D_{GSSA}$. As a result (4.62) is a discrete LTI SS system.

The general rule of sampling a SS system is to use a sampling period anywhere between four and ten times the bandwidth of the continuous-time system. Even higher if not all system dynamics were captured. As the MPC controller is to operate at $1KHz$, this is the sampling frequency used to discretise the plant. As the bandwidth of the continuous SS is around $0.1Hz$, the chosen sampling frequency is more than suitable. A $1KHz$ sampling frequency gives a sampling period of $1ms$, which can be found by $h = \frac{1}{f} = 0.001$.

The reader is referred to Figure (4.13), for the comparison between the response to a unit step input of discrete and continuous systems. Quite obviously both systems have the same response, proving that all the dynamics of the continuous-time system were captured by the discrete representation. Important to mention is that the discrete system has different poles from the continuous one. The poles are:

$$\lambda(\Phi) = \begin{bmatrix} 0.0000009 + 0.000005i \\ 0.0000009 - 0.000005i \\ 0.8718464 \\ 0.9997593 + 0.003646i \\ 0.9997593 - 0.003646i \\ 0.9998776 + 0.000541i \\ 0.9998776 - 0.000541i \end{bmatrix}$$

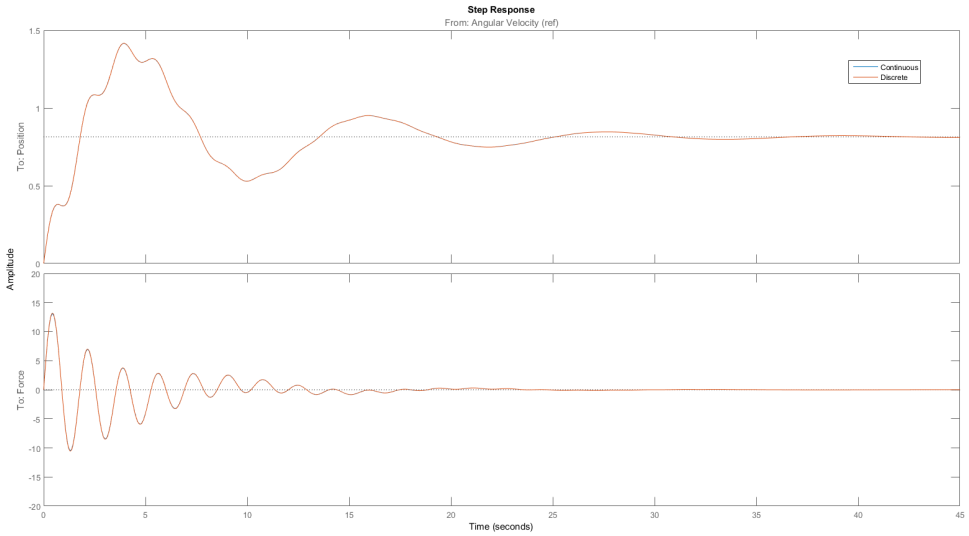


Figure 4.13: Step response of the discrete LTI SS system.

the difference is caused by how discrete poles are expressed. In the *z-domain* which is the discrete equivalent of the LaPlace domain, stable poles are contained in a unit circle around the origin. Subsequently, a pole is stable if it has a value between one and minus one, $-1 \leq p \leq 1$. That same is true for the imaginary part of the pole.

Initial tuning

As using the GPC like performance index is used, the initial settings were discussed by Clarke and Mothadi [61] in 1989 and letter refined. Subsequently, the initial setting for the summation parameters (N_m, N, N_u) and the signal weighting parameters ($P(q)$, λ) were discussed.

First the initial values for the summation parameters. The minimum cost horizon is easiest to find as starting point is always $N_m = 1$. What needs to be kept in mind is that if the system experiences any delays or inverse responses due to non-minimum zeros, N_m needs to be extended by the duration of the event.

The prediction horizon needs to be chosen so as to include all the crucial dynamics of $y(k)$ due to the response to $\Delta u(k)$. Subsequently N is closely related to the step response of the open loop system and its stability. The control horizon, is usually set smaller than the prediction horizon as $N = N_u$ results in very aggressive control action. What is essential done with N_u, is to force the increment on the control signal to become zero:

$$\Delta u(k+j) = 0 \quad for \quad N_u \leq j$$

which is of course equivalent to the control signal becoming constant:

$$u(k+j) = constant \quad for \quad N_u \leq j$$

if $N_u << N$ is kept, that will result in a smooth control signal.As presented in previous section keeping the control horizon small, also decreases the computational effort. In an optimisation problem where inequality constraints are present, with $N_u < N$ the number of parameters in the optimisation reduces from pN to pN_u, where $u(k)$ is the control signal vector of the size $p \times 1$.

Now to define the following variables which are based on the system that is to be controled:

- $d =$ the dead time of the system (due to delays or inverse response)

- n =number of poles of the original system
- $t_s = 5\%$ settling time

It must be noted that all the variables consider the original IO model and not the augmented IIO model, which was developed for the purpose of control.

The definition of variables for the model is quite straight forward. The system has no delays, and the system is of seventh order with seven poles. Subsequently, the following initial variables for the controller are found:

- $N_m = 1 + d = 1$
- $N_u = n = 7$
- $N = int(\alpha_N t_s) = 25$ for well damped systems

where $\alpha_N = [1.5, 2]$ and t_s is the settling time of the uncontrolled nominal system. The 5% settling time is 30 seconds, and as that gives a prediction horizon of $N = 3$, it was decided to use $N = 25$ initially.

Observer/Low pass filtering

The other parameters to be tuned are the weighting parameters. The first to be found is the polynomial $P(q) = 1 + p_1 q^{-1} + \cdots + p_{n_p} p^{-n_p}$. The trucking filter is to contain the desired poles of the close loop. Consequently, an filter gain must be chosen such as too obtain those closed loop poles. As the a too aggressive filter might render the system unstable, like a dead beat filter, it was necessary to reach for more optimal options. As information about sensor noise are known it was decided to make use of *Kalman* theory [60, 62–64].

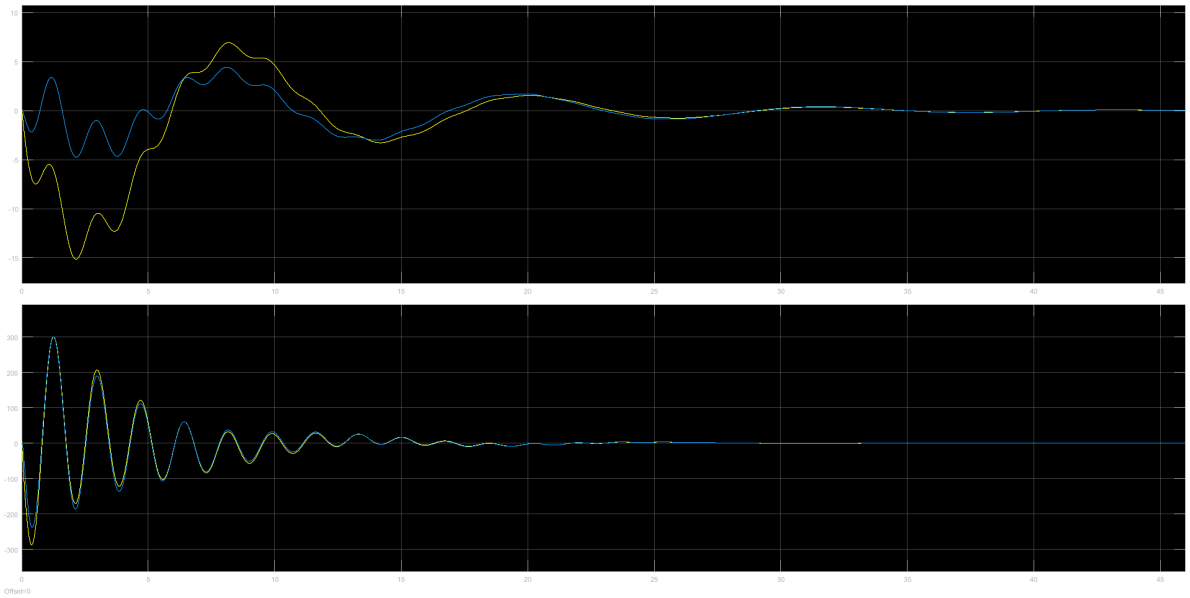


Figure 4.14: Kalman filter output estimation with initial untuned weights.

Going with the assumption, that both the measured states of the system and the system outputs are not perfect, some measurement errors might be introduced. Those errors appear as sensor noise. As follows, $v(k)$ (measurement noise) and $w(k)$ (process noise) are defined as:

$$x(k+1) = x(k) + w(k) \tag{4.63}$$

$$y(k) = x(k) + v(k) \tag{4.64}$$

then $v(k)$ and $w(k)$ are assumed to be zero-mean white-noise sequences with variance $E = R$ and Q respectably, and the two noise signals to be uncorolated, meaning a cross-covariance of zero, $S = 0$. Using equations (4.63-4.64), the problem of finding a minimum-error variance estimate of the quality $x(k)$ is a special case of the well-known *Kalman-observer* problem.

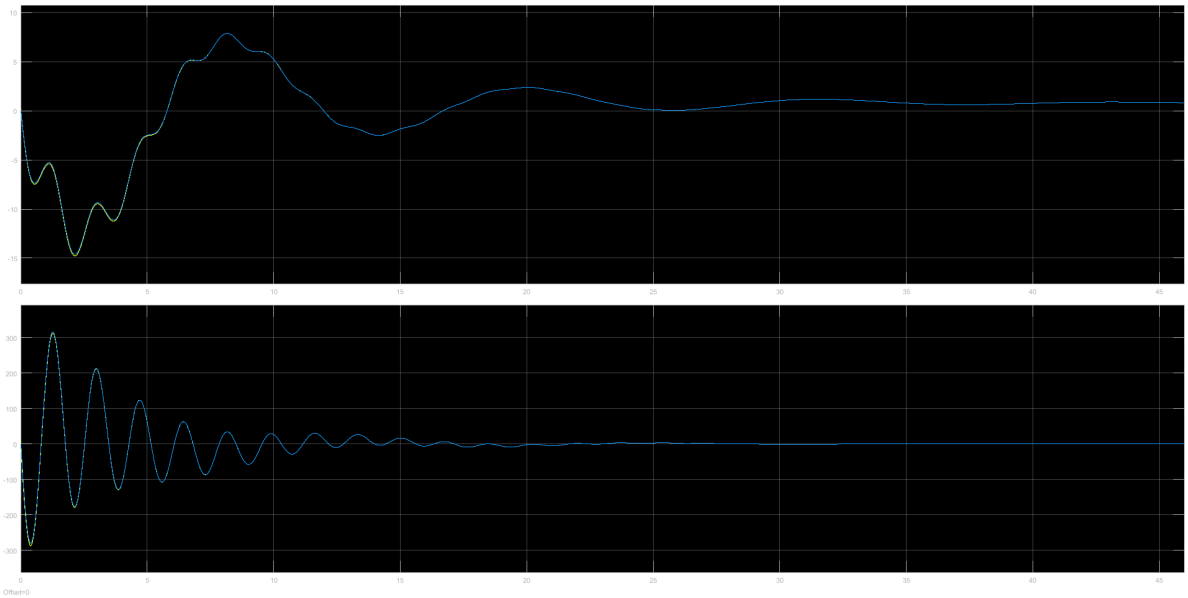


Figure 4.15: Kalman filter output estimation with tuned weights.

The observer approximates the state vector of a dynamical system from measurements of the input and output data. In order to work an observer needs a system model. In the case of a LTI system the model is identical to the model used so far:

$$x(k+1) = Ax(k) + Bu(k) \tag{4.65}$$

$$y(k) = Cx(k) \tag{4.66}$$

when an approximation is made of the state vector $x(k)$, equation (4.65) is of the form:

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

where $\hat{x}(k)$ is the approximation of $x(k)$. If at $t = 0, x(0) = \hat{x}(0)$, then $x(k) = x(\hat{k})$. If the approximation is not equal to the actual state initially, it will converge provided that A is asymptotically stable. If x_e is:

$$x_e(k) = \hat{x}(k) - x(k)$$

then $x_e(k+1) = Ax_e(k)$. Over time the estimation becomes equal to the real values. That leaves us with very little control over it though. As the control signal is sometimes not enough, the output of the system can be used to reconstruct the states. The reconstruction can be aided by introducing the difference between measured and estimated output $\hat{y}(k) = C\hat{x}(k)$. So the following holds true:

$$\hat{x}(k+1) = A\hat{x}(k) + Bu(k) + L\left(y(k) - C\hat{x}(k)\right), \tag{4.67}$$

where K is a gain matrix. The system defined by (4.67) is defined as an *observer*. As the error between measured and approximated states satisfies:

$$x_e(k+1) = (A - LC)x_e(k)$$

the matrix K must be chosen such, as to ensure that $A - KC$ is asymptotically stable. Additionally, it is desired for $x_e(k)$ to tend to zero as time tends to infinity, meaning that the following is wanted:

$$\lim_{k\to\infty} x_e(k) = \lim_{k\to\infty} (\hat{x}(k) - x(k)) = 0.$$

In order to be able to find the matrix gain K, the following lemma is given:

Lemma 4.3.2. ***Observability*** *(Kailath, 1980) Given matrices $A \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{l\times n}$, if the pair (A,C) is observable, then there exists a matrix $L \in \mathbb{R}^{n\times l}$ such that $A - LC$ is asymptotically stable.*

Subsequently, matrix K can be found, the rate at which $x_e(k)$ goes to zero will be determined by that matrix. System (4.67) becomes an *asymptotic observer*.

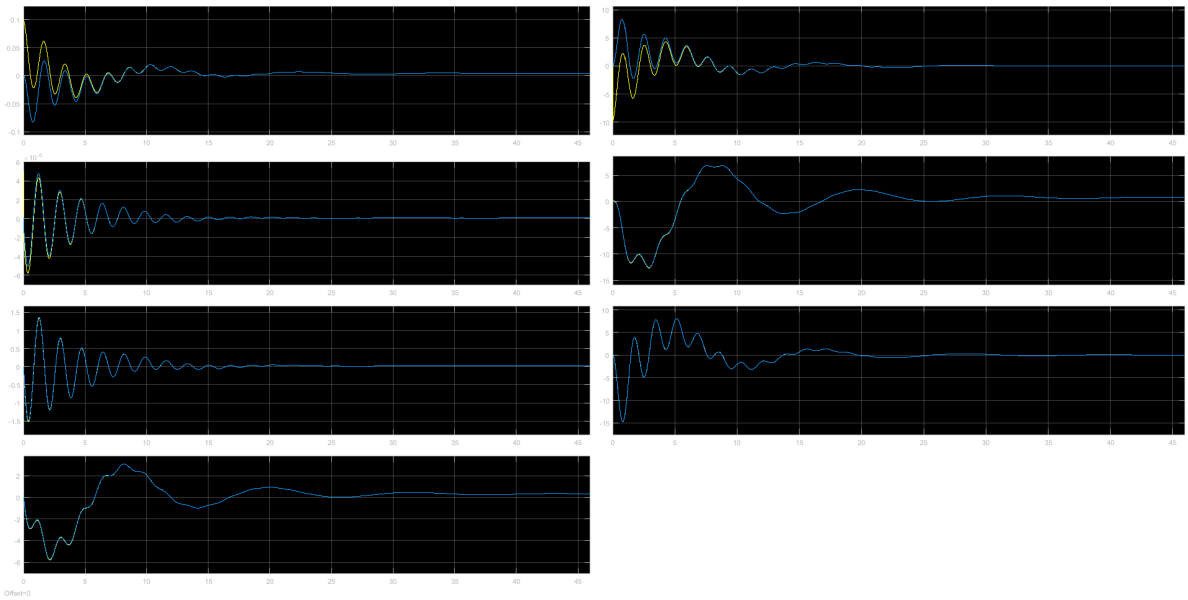


Figure 4.16: System states under active error rejection by a Kalman Observer

As the observability matrix of our controller is full rank, the pair (A_{GSSA}, C_{GSSA}) is observable and therefor a Kalman observer can be found. In order to find the matrix gain K two equations need to be solved. From [63], given prior information $\hat{x}(k|k-1)$ and $P(k|k-1)$, the update equation for the state-error covariance matrix $P(k+1|k)$ is given by the *Riccati difference* equation:

$$\begin{aligned} P(k+1|k) =& A(k)P(k|k-1)A(k)^T + Q(k) \\ &- \left(S(k) + A(k)P(k|k-1)C(k)^T\right) \\ &\times \left(R(k) + C(k)P(k|k-1)C(k)^T\right)^{-1} \\ &\times \left(S(k) + A(k)P(k|k-1)C(k)^T\right)^{-1}. \end{aligned} \tag{4.68}$$

The Kalman gain, denoted by $K(k)$, is found be solving:

$$L(k) = \left(S(k) + A(k)P(k|k-1)C(k)^T\right) \times \left(R(k) + C(k)P(k|k-1)C(k)^T\right)^{-1}. \tag{4.69}$$

The state-update equation is given by:

$$\hat{x}(k|k+1) = A\hat{x}(k|k-1) + Bu(k) + K(k)\left(y(k) - C\hat{x}(k|k-1)\right), \tag{4.70}$$

and the covariance matrix is given by:

$$E\left[\left(x(k+1) - \hat{x}(k+1|k)\right)\left(x(k+1) - \hat{x}(k+1|k)\right)^T\right] = P(k+1|k).$$

In the above equations Q is the covariance matrix of the process noise $Q = E[w(k)w(k)^T]$, R is the covariance matrix of the measurement noise $R = E[v(k)v(k)^T]$, and S is the cross-covariance between process and measurement noise $S = E[w(k)v(k)^T] = 0$, is assumed yo be zero.

Now taking from the theory in [62], a Kalman filter is designed. First from measurements the variance of the process noise is found, $v(k)$. Subsequently the covariance matrix is found to be

$Q_n = E[v(k)v(k)^T] = 0.105$. With the measurement noise unavailable, the covariance matrix $R_n = E[W(k)w(k)^T]$ becomes the design variable. It will be tuned to achieve the most optimal observer. Important to mention at this point the trade-off between Q_n and R_n. If the covariance of measurement noise is bigger than the process noise, the observer will have great noise suppression, but will be slow to minimise the error between estimated and measured output. On the other hand, a much bigger Q_n gives a filter that is aggressive, but has a poorer noise suppression.

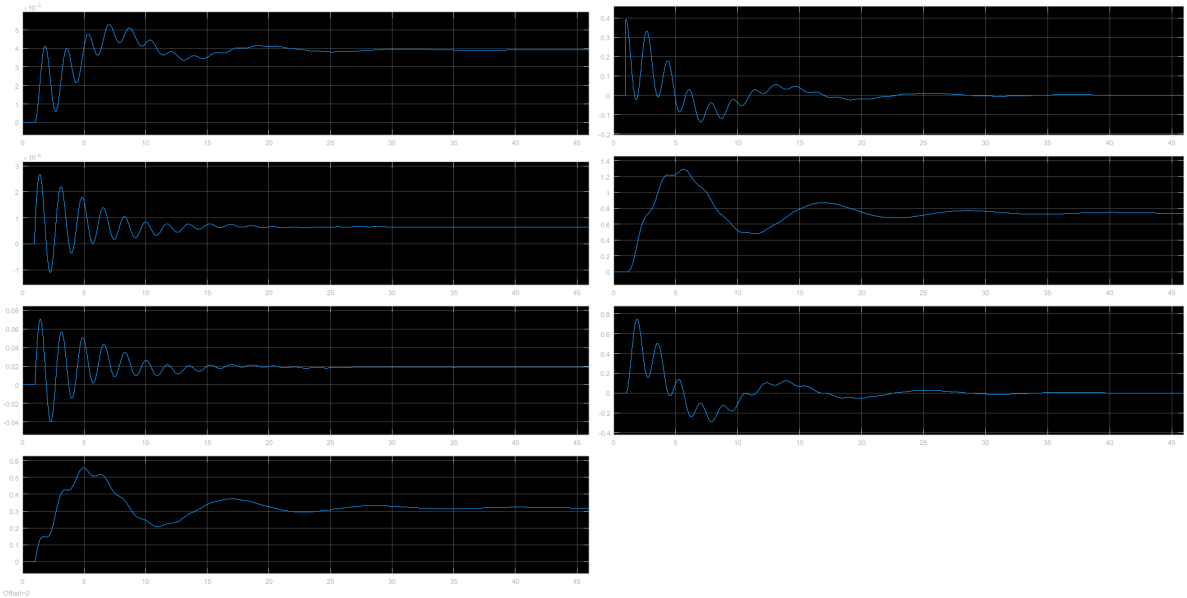


Figure 4.17: System states with no error between the Observer and the system.

With the covariance matrix of the process noise found from measurements $Q_n = 0.105$, the initial value for the measurement noise covariance matrix is picked as identity $R_n = I$, as it is the simplest and most intuitive. In Figure (4.14) the approximation of the system output can be seen compared to the real output of the system. The response presented is to a constant input of 0. The initial values of the states were altered, with the states of the Kalman observer being kept at 0. Subsequently it is presented how the system reacts to states different than the linearization points, and how the filter deals with the subsequent error in the output. The response is clearly not very satisfactory as the simulation was run for over 40 seconds and the observer only reduced the error between the outputs after 20 seconds.

As a result, the Kalman filter had to be tuned to give a satisfactory performance without destabilising the system. Tuning resulted in the measurement covariance matrix being:

$$R_n = \begin{bmatrix} 0.001 & 0 \\ 0 & 0.01 \end{bmatrix}.$$

The resultant response is presented in Figure(4.15). Clearly the observer reduces the error between the actual and estimated output almost instantly, as within second the estimation matches the simulation.

The matrix gain corresponding to the chosen covariance matrices was found using the MATLAB Control toolbox and the form:

$$L_{Kalman} = \begin{bmatrix} 0.000002603 & 0.00000061 \\ 0.000000029 & 0.00000014 \\ 0.000168264 & 0.00074470 \\ 0.000540354 & 0.00113072 \\ -0.00028179 & -0.00015831 \\ 0.000716700 & -0.00003937 \\ 0.000308884 & 0.00008484 \end{bmatrix}$$

the gain places the poles of the augmented state system, ie. $\Phi_{DZ} - L_{Kalman}C_{DZ}$, at different location to the original system:

$$\lambda[A_d z - L_{Kalman}C_d z] = \begin{bmatrix} 0.0000009 + 0.000005i \\ 0.0000009 - 0.000005i \\ 0.8567951 \\ 0.8718502 \\ 0.9996001 + 0.000429i \\ 0.9996001 - 0.000429i \\ 0.9993672 \end{bmatrix} \tag{4.71}$$

those are visibly faster then the poles of the original nominal system's.

The last point of interest it to make sure that the observer is not unstable in the presence of no error. Figure(4.16) shows how the system states are estimated and how the estimation converges to simulation. This is the same scenario as in previous experiments when input of 0 is introduced to the system and only the states are augmented. In Figure (4.17) on the other hand, the initial state error was reduced to zero, and the system was subjected to a unit step input. As seen the responses are identical if the no state error is present.

4.3.7. MPC Simulation and tuning

In this section the problem of implementing a tuning an initial MPC controller are to be handled. For this purpose the *MPC MATLAB Toolbox* is used. The MPC controller consists of a matrix gain, when no constraints are active, or an optimiser when constraints are active. An observer is needed when full state information is not available. A Kalman filter was designed for that purpose in previous section, with the uses of the statistical information about the noise acting on the system. Initial weights and horizons were suggested in previous in the previous section. Those are used as the starting point of the controller.

Subsequently, the initial penalising weight on the rate of change of the control signal is chosen as:

$$R = 1. \tag{4.72}$$

As the system has only one input, naturally only one of the two outputs can be controller at any one time. That results in one of the diagonal weights being 0 in the error weight matrix. The reference signal is for the position of the plant, and the loop is closed around the position feedback. Naturally then the error matrix becomes:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}. \tag{4.73}$$

as a unity weight is felt as the best starting point. The horizons are defined as previously stated. The prediction horizon is set to $N = 25$, with the control horizon being $N_u = 7$. As there are no input, internal or output delays, the controller can start optimising immediately.

Figure (4.18) illustrates the response of the system when subjected to a sinusoidal wave of frequency $25Hz$ and an amplitude of $8mm$. Such reference signal was chosen as it is in the middle of the operational range and as the plant has a limit placed on it of $8.5mm$. If the plant can operate in its upper limits, it is felt that without the loss of generality it can be said, that it can operate in lower amplitudes as well. In appendix (B) the reader can find other responses of the initial untuned plant. In Figure (4.18) it is easy to notice that the initial MPC controller gives an awful performance the plant output does not come even close to achieving the desired reference. It is quite clear, that the controller requires tuning.

As the system might have active constraints during operation the control horizon was increased $N_u = 10$. From the initial simulation it was quite clear that the controller is not aggressive enough and does not drive the plant output to zero output error. Subsequently both the penalising weight on the rate of change in control signal was lowered and the weight on the position output was increased to emphasise its importance. As a result the output error weighting matrices become as follows:

$$Q = \begin{bmatrix} 100 & 0 \\ 0 & 0 \end{bmatrix} \tag{4.74}$$

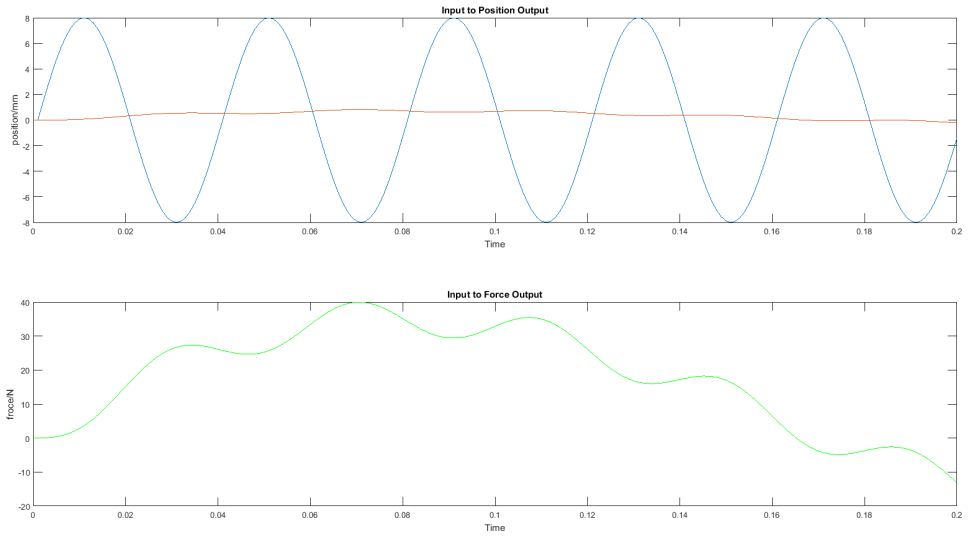


Figure 4.18: Initial response of the system with an untuned MPC controller to a 25 Hz sinusoidal wave reference.(blue line - reference, orange line - system response)

and the penalising weight matrix of rate of change in control signal becomes:

$$R = 0.001. \tag{4.75}$$

As a consequence, the performance was greatly improved. From Figure (4.19) it is obvious that the MPC controller, when no constraints are active, can achieve perfect trucking with zero steady state error. The signal send to the controller is again a sinusoidal wave with a frequency of $25Hz$ and an amplitude of $8mm$. As seen the plant output matches the reference perfectly and the error is zero. The reader is send to appendix (B) for system response, with a tuned MPC controller for the whole operational range. To analyse the stability and asses the behaviour of the system Figure (4.20) show the controlled system response to a unit step input. In comparison to the free, uncontrolled response of the plant in Figure (3.12) in Section (3.2), the performance is significantly increased. The settling time decreased from 30 seconds to 1 millisecond. Steady state error was fully eliminated with integral action and on top, no over shoot can be observed. The MPC controller gives a satisfactory, well behaved response which allows to infer stability.

To further prove stability the theory established in Section (4.3.5) is called upon. As stated there to have nominal stability both the controller and observer need to be stable. By principle of separation the controller can be separately form the observer. Subsequently first the stability of the controller needs to be tackled. As discussed in Section (4.3.5), in order to prove stability of the controlled SS system, the dynamics matrix needs to have all its eigenvalues strictly with in the unit circle:

$$\lambda[A - BK_{MPC}\Psi] \leq 1$$

where the gain matrix K_{MPC} can be found as stated already:

$$K_{MPC} = [1, 0, \dots, 0]\mathcal{H}^{-1}\Theta^T\mathcal{Q}$$

and as Section (4.3.3) shows matrix $\mathcal{H}$ can be found using Equation (4.46) as:

$$\mathcal{H} = \Theta^T\mathcal{Q}\Theta + \mathcal{R}.$$

Matrices Θ,$\mathcal{Q}$ and $\mathcal{R}$ are calculated as defined in Section (4.3.3). As a result the matrix gain K_{MPC} is calculated as:

$$K_{MPC} = [1, 0, \dots, 0](\Theta^T\mathcal{Q}\Theta + \mathcal{R})^{-1}\Theta^T\mathcal{Q} \tag{4.76}$$

where the vector $[1, 0, \dots, 0]$ has length equal to $H_u - 1$, and comes from the receding horizon strategy. As MPC is a predictive control strategy the optimal control signal is calculated for the whole of the

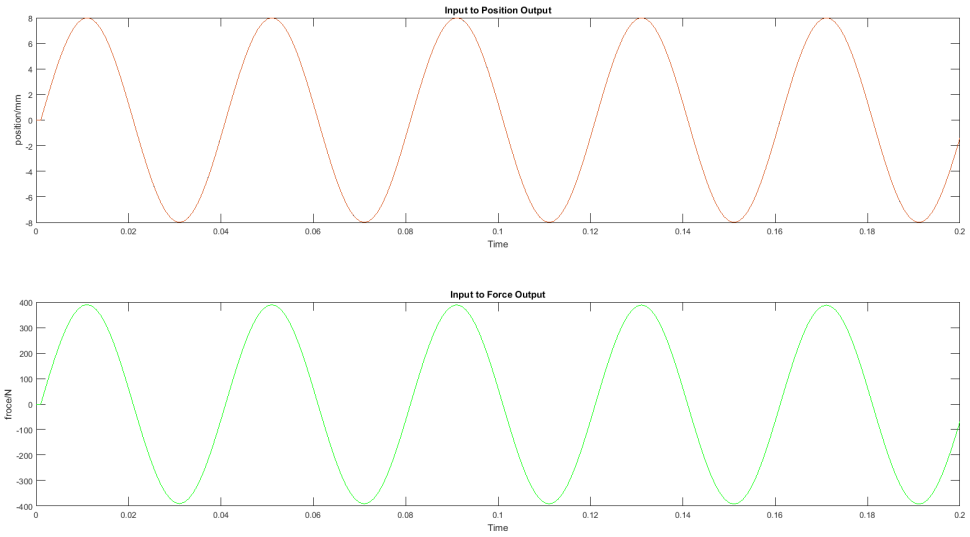


Figure 4.19: Response of the system with tuned MPC controller to a 25 Hz sinusoidal wave reference.

control horizon, but according tot he receding horizon strategy only the signal for the next time step is applied. Subsequently, although $(\Theta^T Q\Theta + \mathcal{R})^{-1}\Theta^T Q$ gives a vector as a solution (as there is only one control input), only the first position is applied.

When all the matrices are substituted into Equation (4.76), a static gain is found which acts as a dynamics controller. The K_{MPC} with the weighting matrices (4.74) and (4.75) come out to be:

$$K_{MPC} = [0.000999 \quad 0.049723]. \tag{4.77}$$

With the MPC gain found it can be substituted into the dynamic system matrix (4.58), at which point the controller dynamics must be:

$$A - BK_{MPC}\Psi(1,1) = A - B[0.000999 \quad 0.049723]CA. \tag{4.78}$$

Only the first position in the Φ vector as, the dynamics analysis of the system is the same at the begging of the prediction horizon as at the end. Subsequently equation (4.78) is solved an the poles of the controlled system are found to be:

$$\lambda[A - BK_{MPC}\Psi(1,1)] = \begin{bmatrix} 0.0000009 + 0.000005i \\ 0.0000009 - 0.000005i \\ 0.8718463 \\ 0.9985137 + 0.003498i \\ 0.9985137 - 0.003498i \\ 0.9999110 + 0.000525i \\ 0.9999110 - 0.000525i \end{bmatrix}. \tag{4.79}$$

Clearly all the poles are within the unit circle, rendering the controlled system stable. Those eigenvalues will not change through out the prediction horizon. With that said, the first condition set in Section(4.3.5) is achieved. Now, with the poles of the Kalman observer having been found previous as (4.71), the second condition for nominal stability is fulfilled. As a consequence it can be concluded that in the unconstrained case an MPC controller becomes discrete LTI SS nominally stable system.

Lastly the case of active constraints needs to be addressed. The main advantage of MPC is that it handles constrains naturally, as it was design mainly for that purpose. Subsequently, the given controller is not satisfactory if it cannot handle active constraints. In Figure (4.21) the response of the plant is presented when controlled by a MPC with active output constraints. Again the reference signal

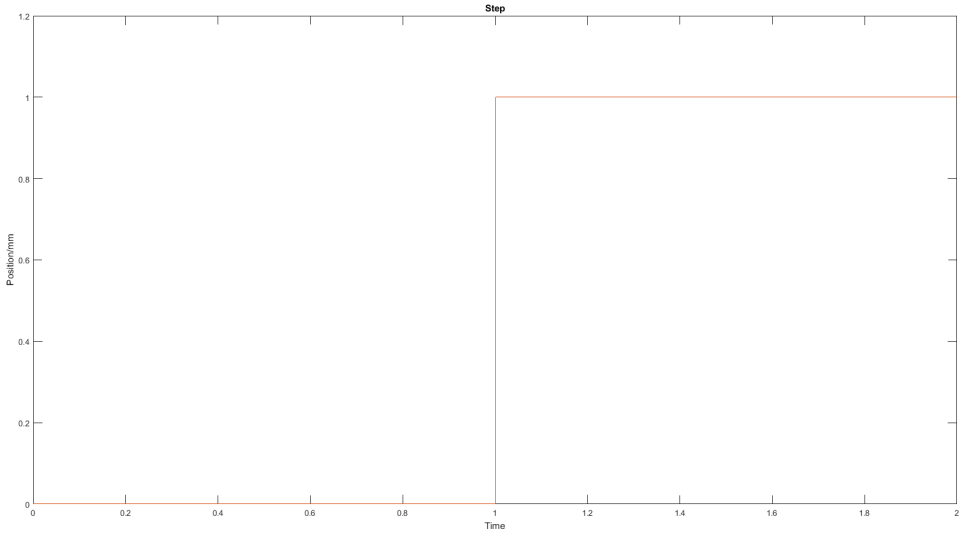


Figure 4.20: Response of the system with tuned MPC controller to unit step reference signal.

is a sinusoidal $25Hz$ wave, of amplitude $8mm$. It is easy to conclude from the Figure that the MPC handles constraints flawlessly. The controller does exactly what is expected of it. When the force limits activate, the achieved position is limited as to stay within the force limits.

Just as in the case of the unconstrained MPC, stability needs to be guaranteed. Falling back on Section (4.3.5), the principle of infinity horizon is to be applied. Subsequently, the prediction horizon was increased until further increase made no difference to the control performance. As a result, the prediction horizon was set to $N = 50$. With that done stability is inferred by extension. Infinite horizon is approximated by a finite horizon, as no difference is observed after a certain point.

4.3.8. MPC without Cascade Control

Most of the literature on MPC for PMSM operates on the assumption that MPC is the only control used on the electric motor, as referenced at the very begging of this section on MPC. That said the literature does not, to the authors knowledge, approach the problem of PMSM being a driving force for another mechanical system. IT is felt that at this junction I is necessary to compare the performance of MPC controller on the GSSA, but with the cascade velocity-current PI control omitted. Leaving thus only the system dynamics to be catered for.

To be able to make the comparison, first the PI control free system has to be modelled. To do so the same dynamic equations (3.11) in section (3.1.2) are used, to describe the dynamics of the system:

$$\begin{aligned}
\dot{i}_d &= \frac{V_d}{L_d} - \frac{R}{L_d} i_d - \frac{L_q}{L_d} \frac{p}{2} \dot{\theta} i_q \\
\dot{i}_q &= \frac{V_q}{L_q} - \frac{R}{L_q} i_q + \frac{L_d}{L_q} \frac{p}{2} \dot{\theta} i_d - \frac{K_e}{L_q} \dot{\theta} \\
\ddot{\theta} &= \frac{3p}{2} \frac{[(L_d - L_q) i_q i_q]}{J_a} + \frac{K_t}{J_a} i_q - \frac{b_a}{J_a} \dot{\theta} - \frac{b_a + b_l}{J_m} f(\dot{\theta}) + \frac{b_l}{J_m} \dot{x} - \frac{k_b}{J_m} f(\theta) + \frac{k_b}{J_m} x \\
\ddot{x} &= \frac{b_l}{M} f(\dot{\theta}) - \frac{b_l}{M} \dot{x} + \frac{k_b}{M} f(\theta) - \frac{k_b + k_s}{M} x
\end{aligned} \tag{4.80}$$

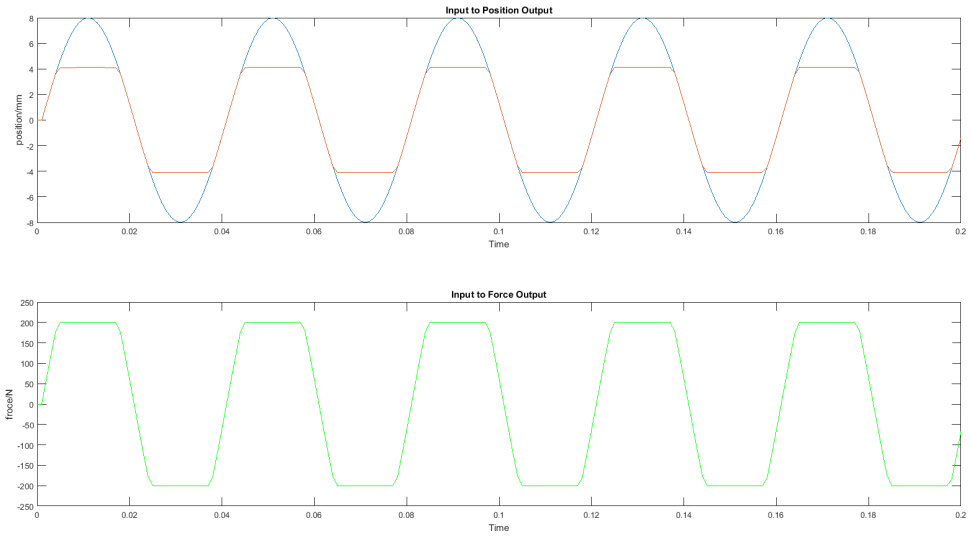


Figure 4.21: System response for an input reference sinusoid of $25Hz$ frequency and an amplitude of $8mm$, with a constrained MPC controller.

and so the nonlinear equations describing the plant are summarised to be as follows:

$$f = \begin{bmatrix} \dot{i_d} \\ \dot{i_q} \\ \ddot{\theta} \\ \ddot{x} \end{bmatrix}. \tag{4.81}$$

The output equations also do not change from prior, assuming that the sensors a placed in the same fashion as previously. Equations (3.13) for linear position and (3.18) force acting on the load are used to find the output from the plant as:

$$\begin{aligned} x_a &= l\sqrt{1-\left(\frac{r\sin(\theta)-q}{l}\right)^2} - r\cos(\theta) \\ F &= M\ddot{x} = b_l\dot{x}_a - b_l M\dot{x} + k_b M x_a - k_b + k_s M x \end{aligned} \tag{4.82}$$

and so the eqautions describing the output of the plant to be controlled are:

$$g = \begin{bmatrix} x_a \\ F \end{bmatrix}. \tag{4.83}$$

lastly the state vector is found as:

$$x = \begin{bmatrix} i_d & i_q & \theta & \dot{\theta} & x & \dot{x}. \end{bmatrix} \tag{4.84}$$

It understandably differs from the previous case, as the error in $\dot{\theta}$ and i_q do not come into play due to the lack of PI control.

The same as previously, the nonlinear equations have to be linearised around an equilibrium point. Again the origin is chosen as the best pick for the linearization. To find the dynamics linear equations, the nonlinear dynamic equations have to be partially differentiated with respect to the state and input. The change from the PI controlled plant is, that the input to the plant the same as the input to the PMSM, voltage. Additionally, inverse kinematic is not used at the input to the plant. Subsequently, the

dynamic matrices A is found as:

$$\frac{\partial f}{\partial x}_{||0} = A = \begin{bmatrix} -\frac{R}{L_d} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{R}{L_q} & 0 & -\frac{K_e}{L_q} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{K_t}{J_m} & -\frac{k_b qr}{J_m l\sqrt{1-\frac{q^2}{l^2}}} & -\frac{b}{J_m} - \frac{qr(b+b_l)}{J_m l\sqrt{1-\frac{q^2}{l^2}}} & \frac{k_b}{J_m} & \frac{b_l}{J_m} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{k_b qr}{Ml\sqrt{1-\frac{q^2}{l^2}}} & \frac{b_l qr}{Ml\sqrt{1-\frac{q^2}{l^2}}} & -\frac{k_b+k_s}{M} & -\frac{b_l}{M} \end{bmatrix} \tag{4.85}$$

and the input matrix B is:

$$\frac{\partial f}{\partial u}_{||0} = B = \begin{bmatrix} 0 \\ \frac{1}{L_q} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{4.86}$$

Carrying on the output matrix C is found by calculating the partially differential equations of the output set g. Again the Jacobin matrix is linearised around the origin. The outputs are found then to be:

$$\frac{\partial g}{\partial x}_{||0} = C = \begin{bmatrix} 0 & 0 & (qr)/(l(1-q^2/l^2)^(1/2)) & 0 & 0 & 0 \\ 0 & 0 & \frac{k_b qr}{l\sqrt{1-\frac{q^2}{l^2}}} & \frac{b_l qr}{l\sqrt{1-\frac{q^2}{l^2}}} & -k_b - k_s & -b_l \end{bmatrix} \tag{4.87}$$

and the feedforward term, following the same practise as before, becomes:

$$\frac{\partial g}{\partial u}_{||0} = D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{4.88}$$

Using equations (4.85-4.88) the continuous SS LTI system can be constructed. Comparing it to the plant controller by the PI controller several differences can be noticed. In Figure (4.22) the step of the new plant is illustrated. Clearly the system shows a more oscillatory behaviour, and has a settling time slower by about 20 seconds $t_s = 50$. This is reflected in the slower poles of the system:

$$\lambda(A) = \begin{bmatrix} -47.677 + 9.6432i \\ -47.677 - 9.6432i \\ -23.529 \\ -0.0821 + 0.0070i \\ -0.0821 - 0.0070i \end{bmatrix}.$$

The new plant has to naturally be discretized as MPC uses discrete time plants. To fins such plant the same approach is used as previously. The plant is sample with the Zoh technique, subsequently giving a discrete SS LTI system. The same sample time of $1KHz$ was used as the MPC controller operates with sampling intervals of $\Delta k = 1ms$. Figure (4.22) shows the step response of the discretized system. Clearly a sampling time of $1KHz$ is enough to capture all the dynamics of the new system. The translated poles of the discrete system are found to be:

$$\lambda(\Phi) = \begin{bmatrix} 0.5434 + 0.7834i \\ 0.5434 - 0.7834i \\ 0.9767 \\ 0.9999 + 0.0007i \\ 0.9999 - 0.0007i \end{bmatrix}$$

which are again clear different with out PI control influencing the dynamics of the i_q loop and θ loop.

For a new plant a new optimal filter needed to be designed. As the noise characteristics do not change with the change of the plant, assuming that the same sensors are used. Subsequently, the

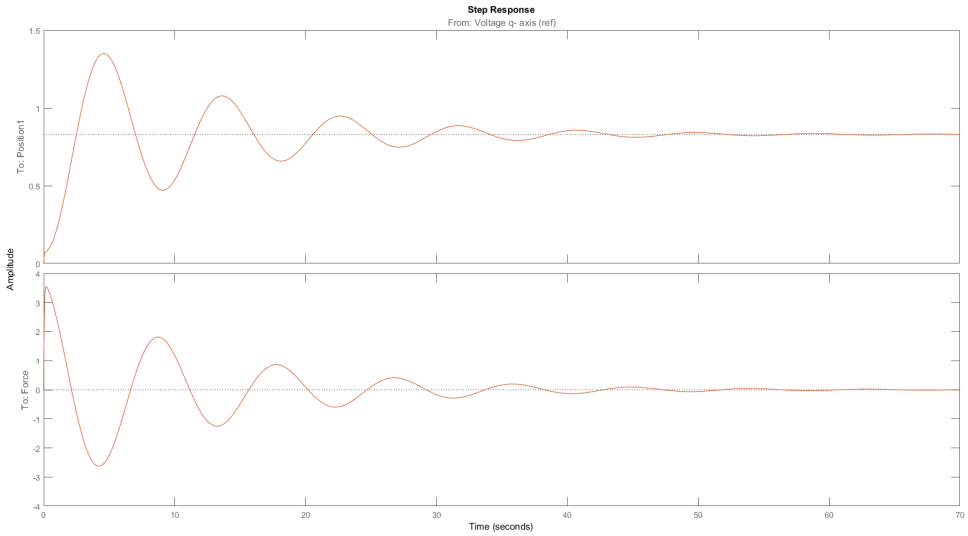


Figure 4.22: Response of the plant to a unit step input fro both the continuous-time and discretized system. Plant without cascade control. (blue line - continuous, orange line - discrete)

process noise covariance matrix Q remains unchanged. Due to slightly different plant dynamics the measurement noise R matrix, that is unknown, is slight different giving the covariance matrices as:

$$Q = 0.1005 \quad R = \begin{bmatrix} 0.01 & 0 \\ 0.1 & 0.1 \end{bmatrix} \tag{4.89}$$

where R is by a factor ten bigger then previously, which gives a less aggressive observer and greater robustness against noise. The new plant and different covariance matrices give a Kalman gain of:

$$L_{Kalman} = \begin{bmatrix} -0.000239 & -0.00931 \\ 0.00038 & 0.00063 \\ -0.03452 & -0.16271 \\ 0.00058 & -0.00008 \\ 0.00023 & 0.00004 \end{bmatrix}$$

which positions the poles of the observer in the new following locations:

$$\lambda(\Phi - L_{Kalman}C) = \begin{bmatrix} 0.5328 + 0.7728i \\ 0.5328 - 0.7728i \\ 0.9140 \\ 0.9996 + 0.0004i \\ 0.9996 - 0.0004i \end{bmatrix}.$$

From the pole placement it is clear to tell that the observer is stable as all poles are with in the unit circle.

Now lastly the MPC controller has to be designed and simulated. As main point is to compare if MPC does better aided by cascade PI control or not, the previous MPC design was used as the initial controller. Regrettably, simulations have shown that the controller does not perform satisfactory and there for needs to be retuned. Subsequently, suggesting the that aid of PI loops might be beneficial.

The tuning resulted in weight matrices twice as big as for the previous controller and both control and prediction horizons twice as long. The tuning has stopped when the control horizon was set to $n_u = 20$ and the prediction horizon was set to $N = 100$. The output weight matrix was left at:

$$Q = \begin{bmatrix} 200 & 0 \\ 0 & 0 \end{bmatrix}$$

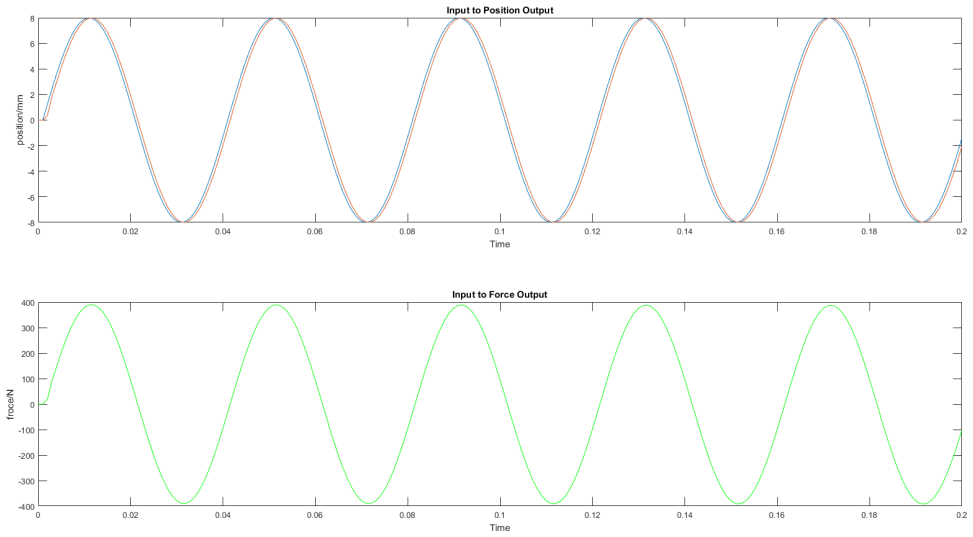


Figure 4.23: Response of the plant to a sinusoidal wave of $25Hz$ and $8mm$ amplitude, whilst being controller by MPC. (blue line - reference, orange line - system output)

and the control rate was left at:

$$R = 0.0002$$

The results of that tuning can be sin in Figure (4.23). Quite obviously the trucking is still not perfect and would require further tuning. At this point the tuning was concluded as the objective was reached. A clear comparison between an MPC aided by cascade can be made with an MPC without the help of PI loops.

This concludes the chapter on control and marks the end of the project. All three control approaches were testes and compared. The following chapter discusses the results and concludes on the project.

5

Conclusions

5.1. Discussion

In general research conducted in this thesis was a success. With that said several problems were encountered on the way. In chapter (3) an accurate description of the GSSA was found in a state space form. This in turn led to design of accurate controllers. However, although the system description is accurate it is not perfect as Section (3.1.4) indicates. The problem is with the system parameters found, as in general initial parameter values are not far from parameters obtained from system identification. This is not the case for several parameters of the GSSA, as indicated by Table (3.1). This can be attributed to two main factors: ill choice of identification initial points, and/or the controller parameters of inner PI controller.

A defiantly ill initialised parameter, was the value of inertia for the PM synchronous motor. As the pm synchronous motor has only one moving part, which is the crank, the moment of inertia of PMSM is equivalent to the moment of inertia of the crank. With the initial vale for crank inertia given as $11.42kgm^2$, assuming that the crank is approximately one centimetre in radius (based on the design and size of the GSSA) and remembering that the moment of inertia is found as $I = mass \times radius^2$, the crank would have a mass of approximately 114 tonnes. Such mass of the crank, based on the size of the GSSA, is highly unlikely. As a result the initial value of the PMSM inertia is far from the actual value.

An additional problem was the inner loop of the cascade controller. Due to limitations of the PLC (Moog's MSD), sampling above $8.5kHz$ was impossible. Documentation suggests that the PI controller has a bandwidth approximately as high as $w_b = 2kHz$. Remembering that as a rule of thumb for signal sampling, the sampling frequency must be at least ten times the bandwidth of the system sampled, finding the dynamics of the PI current controller was not possible. Subsequently, when identifying the current loop the PI controller was treated as unknown and the circuit values as known. This is on contrary to the velocity loop, where the parameters of the PI velocity controller were found by sampling I/O pairs. With the exact structure of the PI controller unknown and only a simple lag filter approximation, a local minimum of the parameters was found. This in turn gives rather implausible values for the motor constants K_t and K_e.

Three different control strategies were explored in Chapter (4). The chapter started with Section (4.2.2) talking about Switched Systems. As the GSSA switches between two PI control loop, it is possible to describe it as a Switched System, where switching occurs between two dynamic control loops. Initially, the assumption was that the nature of the switching between the subsystem causes the instability and subsequent switch off of the GSSA. This assumption came from both subsystems (force loop, position loop) being asymptotically stable systems, as proved in Section (4.2.4). The nature of the switching problem was assumed to be speed. With low end part of the reference signals range being rather slow (1-5Hz), attention was shifted to the event of switching itself. The theory being that after a switching event, the small oscillations that take place, cause further switching between subsystems

and lead to instability. To test that theory, the already found Lyapunov functions of the systems (one function per subsystem) were used to find a minimum dwell-time that has to pass between switches. As discussed in Section (4.2.4), the found dwell-time ensures stability of the switched system. Simulations with the state space model found previously, disprove the oscillation theory, showing that the oscillations are too small to affect the system in any significant way.

In Section (4.2.6) of Chapter (4), the true problem was identified as integrator wind up, causing actuator saturation. One of the main reasons for integrator wind up and subsequently actuator saturation, are big changes in set point. Such changes occur during a switch between the position and force loop. Depending on how big the change in set point is, the integrator saturates the input either over time or instantaneously. For both cases the well-known back-calculation anti-wind up technique was implemented. Simulations, using the SS LTI model, have proven that the right reset time can be found for a given set of force limits, $\|f_{lim}\|$. Problems arise when the limits are changed. One tuned, integrator reset time T_s, has a very limited range of effect. This results in several rests times needed to cover the whole operating range (it was not researched what exact number might be needed). A set of reset times, needs a scheduling technique like fuzzy logic. Alternatively, different anti-wind up strategies can be explored like Conditional Integration or Incremental algorithm. If a SS LTI system description is kept, an anti-wind up use of state observer can be potentially designed to prevent the saturation.

Lastly, the most probable solution was research at the end of Chapter (4) in Section(4.3.2) was explored. Model Predictive Control is the most advance control strategy tested in this thesis. Model predictive control, requires a very accurate state space model to be effective. Which was the reason for great lengths taken to achieve such model for the GSSA. Simulations on the model proved that the integrator wind up problem was overcome. The controller can make the system follow a reference signal equally well with limits active and inactive, which was expected as the main advantage of optimal control is that it can deal with limits in a natural way. The use of MPC also makes the use of two controllers, and subsequently switching between them, redundant. The GSSA is a SIMO (single-input-multi-output) system. For MPC the extension to MIMO or SIMO systems is natural (in SIMO case emphasis is placed on only one output). Additionally, as it is very rare that full state knowledge is available to the controller, MPC makes use of an observe. As from measurement data the statistical properties of the measurement and process noise were known, a Kalman filter could be designed. The use of the Kalman filter helps to deal with the sensor noise present. The only worry with MPC is the speed of operation. MPCs are in general considered to be quite slow. The simulations run on a Intel i5-4690 processor, suggest that speed of calculating the optimal control signal should not be an issue.

5.2. Conclusions

To start, the research question identified at the begging of this thesis is recapped to be the following research problem:

- Develop a controller that ensures stability and required performance of the Generic Short Stroke Actuator, under all operational conditions

To answer the research question, first an accurate state space linear time invariant model was found to enable the analysis of the GSSA system and design of high level accurate controllers. The problem of instability of the GSSA was found to be actuator saturation due to integrator wind up. The integrator wind up was caused by big changes in reference set point during switching between a PI position and PI force controller. Three main approaches were tested to overcome the problem: Dwell-time for switched systems, back calculation anti-wind up, and model predictive control. The latter two solutions were successful, with integrator anti-wind up coming short in comparison to model predictive control, which due to its structure naturally deals with limits.

Compering the three control strategies dwell-time failed completely, anti-wind up was partially successful and MPC fulfilled all expectations. The stability problem for the GSSA was initially suspected to be caused by the nature of switching between the PI position and PI force controller. It was theorised

that oscillations occur around switching instances and case fast switching which destabilised the plant. This was disproved with implementation of dwell-time. Simulations indicated that the use of dwell-time fails as it does not address the true problem. The two remaining techniques were both successful, as the address the true reason of the plant instability. Back calculation anti-wind up method can solve the problem of actuator saturation, but is limited on its own. In comparison to MPC it comes short. Even if back calculation would be combined with a gain scheduling technique like fuzzy logic it still would be worse than MPC. As all optimal control strategies MPC can deal with any limits placed on a system naturally as it was the main reason for developing this control technique. Added benefits of MPC come in the use of an observer, those days is usually a Kalman filter, which is a great tool to deal with noise present in the system. Disturbances can also be incorporated into the model used for prediction by MPC. Over all this leaves MPC superior to Back Calculation, even when combined with gain scheduling like fuzzy logic. Lastly, MPC is the only approach which makes the use of two different controllers redundant and substitutes one in their place.

5.3. Future Work

In this thesis the following techniques were used to overcome actuator saturation: dwell-time, back calculation anti-wind up, and model predictive control. The latter two can were sacksful. Further research should be directed into two areas.

In the case of anti-wind up it can be couple with some gain scheduling technique. Fuzzy-logic is suggested as a well-known and established field, with a vast pool of possibilities to choose from.

As Model predictive control was very successful, further research should focus on implementing the controller in C++ code on the Moog's MITS controller. In the case of the MPC operating to slow explicit MPC techniques ca be explored, where rather than updating every time step MPC only updates every few time steps. On the other hand alternative cost functions can be used, which are specifically tailored to periodic input signals.

Bibliography

[1] J. L. Kirtley and H. W. Beaty, *ELECTRIC MOTOR Handbook* (McGraw-Hill Handbooks, 1998).

[2] T. Wilson and P. Trickey, *D.c. machine. with solid state commutation,* AIEE pape (1962).

[3] M. Gopa, *Control systems: Principles and Design* (Tata McGraw-Hil, 2002).

[4] A. A. J.-R. P. J. M. S. A. J. van der Schaft, M. Kanat Camlibel, *Mathematical control theory I : Nonlinear and Hybrid control systems* (Springer, 2015).

[5] M. R. G. Claire J. Tomlin, *Hybrid systems : computation and control : 5th International Workshop, HSCC 2002, Stanford, CA, USA, March 25-27, 2002 : proceedings* (HSCC (Conference) Stanford, Calif.), 2002).

[6] X. L.-S. A. L. F. Morel, J.-M. R tif, *Fixed switching frequency hybrid control for a permanent magnet synchronous machine,* (2004).

[7] A. M. L.-B. A. J.-M. R. Florent Morel, Xuefang Lin-Shi, *Implementation of hybrid control for motor drives,* (2007).

[8] X. L.-S.-C. V. Florent Morel, Jean-Marie Rétif, *Permanent magnet synchronous machine hybrid torque control,* (2008).

[9] A. C. R. J. P. Karunadasa, *Design and implementation of microprocessor based sliding mode controller for brushless servomotor,* (1991).

[10] F. P. Marwa Ezzat, Alain Glumineau, *Sensorless high order sliding mode control of permanent magnet synchronous motor,* (2010).

[11] O. U. Cetin Elmas, *A hybrid controller for the speed control of a permanent magnet synchronous motor drive,* Control Engineering Practice 16, 260–270 (2008).

[12] E. B. Jr., *Optimal control - 1950-1985,* IEEE Control Systems (1996).

[13] F. L. Lewis, *Optimal control (linear systems)* (School of Electrical Engineering, Georgia Institute of Technology, Atlanta, Georgia, 2014).

[14] R. Vinter, *Optimal Control* (Birkhauser, 2000).

[15] M. Z. Silverio Bolognani, Luca Tubiana, *High dynamic pmsm current control by optimal saturation management of current regulators.* Industrial Electronics Society, 2003. IECON '03. The 29th Annual Conference of the IEEE (2003).

[16] M. Z. Nicola Bianchi, Silverio Bolognani, *Time optimal current control for pmsm drives,* IECON 02 [Industrial Electronics Society, IEEE 2002 28th Annual Conference of the] (2002).

[17] U. K. P.M. Petczewski, W. Oberschelp, *Optimal model-following control of a positioning drive system with a permanent-magnet synchronous motor,* IEE Proceedings D - Control Theory and Applications (1991).

[18] J. D. Joris Lemmens, Piet Vanassche, *Pmsm drive current and voltage limiting as a constraint optimal control problem,* IEEE Journal of Emerging and Selected Topics in Power Electronics (2015).

[19] S. E. Lyshevski, *Electromechanical Systems, Electric Machines, and Applied Mechatronics* (CRC press, 2002).

[20] K. Astrom and R. Murray, *Feedback Systems: An introduction for scientists and engineers* (Princeton University Press, 2008).

[21] G. Ellis, *Control system design guide : a practical guide* (Elsevier Academic Press, 2004).

[22] F. El-Sousy, *Hybrid recurrent cerebellar model articulation controller-based supervisory h_∞ motion control system for permanent-magnet synchronous motor servo drive,* IET Electric Power Applications (2011).

[23] Y.-S. L. F.-J. Lin and S.-L.Chiu, *Slider-crank mechanism control using adaptive computed torque technique,* IEE Proceedings (1998).

[24] L. Prokop and P. Grasblum, *3-Phase PM Synchronous Motor Vector Control Using a 56F80x, 56F8100, or 56F8300 Device* (Freescale Semiconductor, 2005).

[25] Z. W. Zhang YAOU and K. XIAOMING, *Control of the permanent magnet synchronous motor using model reference dynamic inversion,* in *WSEAS TRANSACTIONS on SYSTEMS and CONTROL* (2010).

[26] B. K. Bose, *Modern Power Electronics and AC Drives* (Prentice Hall PTR, 2002).

[27] R. V. Dukkipati, *Engineering System Dynamics* (Alpha Science international, Harrow UK, 2005).

[28] M. M. Peter and V.Ján, *Pi-controllers determination for vector control motion,* University of Ţilina (2009).

[29] J. C. Balda and P. Pillay, *Speed controller design for a vector-controlled permanent magnet synchronous motor drive with parameter variations,* University of Newcastle,University of Arkansas (1989).

[30] T. Keviczky, *Control theory lecture notes,* (2015).

[31] A. R.-J. v. E. Robert M. Schmidt, Georg Schitter, *The Design of High Performance Mechatronics* (Delft University Press, 2014).

[32] K. Astrom and T. Hagglund, *PID Controllers: Theory, Design, and Tuning* (ISA - The Instrumentation, Systems, and Automation Society, 1995).

[33] H. Lin and P. J. Antsaklis, *Stability and stabilizability of switched linear systems: A survey of recent results,* in *IEEE Transactions on Automatic Control* (2009).

[34] K. R. Santarelli and M. A. Dahleh, *Comparison of a switching controller to two lti controllers for a class of lti plants,* in *American Control Conferance* (2008).

[35] J. P. Hespanha and A. S. Morse, *Switching between stabilizing controllers,* Automatic (2002).

[36] B. D. Schutter and W. Heemels, *Modeling and Control of Hybrid Systems* (TU Delft, Netherlands, 2012).

[37] A. Filippov, *Differential Equations with Discontinous Righthand Sides* (Mathematics and it's applications, 1988).

[38] V. Utkin, *Variable structure systems with slidng modes,* in *IEEE Transactions on Automatic Control* (1977).

[39] D.Liberzon, *Switching in System and Control* (Birkhauser Boston, 2003).

[40] E. F. Stephen Boyd, Laurent El Ghaoui and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory* (Society for Industrial and Applied Mathematics, 1994).

[41] J. T. J. Richalet, A. Rault and J. Papon, *Algorithmic control of industrial processes,* in *In Proceedings of the 4th IFAC Symposium on Identification and System Parameter Estimation* (1976).

[42] J. T. J. Richalet, A. Rault and J. Papon, *Model predictive heuristic control: Applications to industrial processes,* Automatic (1978).

[43] C. Cutler and B. Ramaker, *Dynamic matrix control - a computer control algorithm,* in *In Proceeding Joint American Control Conference, San Francisco, CA, USA* (1980).

[44] D. Clarke and R. Scattolini, *Constrained receding-horizon predictive control,* IEE PROCEEDINGS-D, Vol. 138, No. 4, (1991).

[45] L. P. Saverio Bolognani, Silverio Bolognani and M. Zigliotto, *Combined speed and current model predictive control with inherent field-weakening features for pmsm drives,* 978-1-4244-1633-2/08/.00 IEEE (2008).

[46] L. P. Saverio Bolognani, Silverio Bolognani and M. Zigliotto, *Design and implementation of model predictive control for electrical motor drives,* (2009).

[47] C. M. Michael A. Stephens and M. C. Good, *Model predictive control for reference tracking on an industrial machine tool servo drive,* (2013).

[48] M. Preindl and S. Bolognani, *Model predictive direct speed control with finite control set of pmsm drive systems,* IEEE TRANSACTIONS ON POWER ELECTRONICS, VOL. 28, NO. 2 (2013).

[49] C. M. D.W. Clarke and P. Tuffs, *Generalized predictive control - part 1. the basic algorithm,* (Automatica, 23(2):137–148, 1987).

[50] C. M. D.W. Clarke and P. Tuffs, *Generalized predictive control - part 2. the basic algorithm,* (Automatica, 23(2):149–160, 1987).

[51] T. J. van den Boom, *Model predictive control,* (2013).

[52] R. R.Fletcher, *Practical Methods of Optimisation* (Wiley, 2nd edition, 1987).

[53] W. M. P.E.Gill and M. Wright, *Practical Optimisation* (Academic Press,London, 1981).

[54] S. Boyd and C. Barrat, *Linear Controller Design, Limits of Performance* (Prentice Hall, 1991).

[55] Y. Nesterov and A. Nemirovskij, *Interior-point polynomial algorithms in convex programming* (SIAM, Philadelphia, 1994).

[56] J. P. V. Cornelis Roos, Tamás. Terlaky, *Interior point methods for linear optimisation* (Springer, 2006).

[57] J. Rawlings and R. Muske, *The stability of constrained receding horizon control.* (1993).

[58] M. Vidyasagar, *Nonlinear Systems Analysis* (SIAM Classics in Applied Mathematics,SIAM Press, 2002).

[59] M. G. R.R. Bitmead and V. Wertz, *Adaptive optimal control. the thinking man's gpc,* (1990).

[60] K. J. Astrom and B. Wittenmark, *Computer Controlled Systems: Theory and Design* (Dover Publication, New York, 1997/2011).

[61] D. Clarke and C. Mohtadi, *Properties of generalized predictive control,* (Automatica, 25(6):859–875, 1989).

[62] M. S. Grewal and A. P. Andrews, *Kalman Filtering: Theory and Practice Using MATLAB* (A John Wiley and Sons, Inc., Publication, 2008).

[63] M. Verhaegen and V. Verdult, *Filtering and System Identification: A Least Squares Approach* (Cambrige University Press, 2007).

[64] B. Friedland, *Control System Design: An Introduction to State-Space Methods* (Dover Publications, Inc Mineola, New York, 1986/2005).

A

Additional Theory

A.1. Stability

This appendix explores Lyapunov stability theory. Attention is restricted to time invariant systems as the system discussed in this thesis was a LIT. So the appendix focuses on:

$$\dot{x} = f(x), \quad x \in \mathcal{R}^n \tag{A.1}$$

like systems. The mapping $f : \mathcal{R}^n \to \mathcal{R}^n$ is locally a Lipschitz function. For simplicity, the origin is as always assumed to be a equilibrium point.

A.1.1. Definition of stability

As the system described by (A.1) is assumed to be time-invariant, $t_0 = 0$ is the simplest initial time. The origin is an equilibrium of the system (A.1), in the sens of Lyapunov, if fer every $\epsilon > 0$ there exists a $\delta > 0$ were:

$$|x(0)| \leq \delta \Rightarrow |x(t)| \leq \epsilon \quad \forall t \geq 0$$

In this case the system can only be described as being simply *stable*. A stronger statement is to call as system *asymptotically stable*. For a system to be considered asymptotic stability, it has to be stable and δ can be chosen as:

$$|x(0)| \leq \delta \Rightarrow x(t) \to 0 \text{ as } t \to \infty.$$

A initial conditions for which a dynamic system converges to the origin, will be called *attractive*. The region in which all initial conditions tend to the origin is the *region of attraction*. If that region spans the whole of the state space a system is *globally* stable. Other wise the system is only *locally* stable.

The highest level of stability is if a dynamic system can be called *exponentially stable*. For exponential stability there must exists a positive constant δ, c and λ such that all solutions of a system with $|x(0)| \geq \delta$ satisfy the inequality:

$$|x(t)| \leq c|x(0)|e^{-\lambda t}. \tag{A.2}$$

If exponential decay holds for all δ the system is said to be *globally exponentially stable*. Constant λ in (A.2) is referred to as a *stability margin*.

A.1.2. Lyapunov function

Let $\mathcal{U} \subset \mathbb{R}^n$ be an open set such that $G(\mathcal{U}) \subset \mathcal{U}$ and let $\mathcal{A} \subset \mathcal{U}$ be compact. A function $V : \mathcal{U} \to \mathbb{R}\_\geq 0$ is a Lyapunov function for $\mathcal{A}$ on $\mathcal{U}$ if it is continuously differentiable at every $x \in \mathcal{C} \cap \mathcal{U}$ and there exists a proper indicator ω of $\mathcal{A}$ on $\mathcal{U}$ and $\underline{\alpha}, \overline{\alpha} \in \mathcal{K}\_\infty$ such that:

$$\underline{\alpha}(\omega(x)) \leq V(x) \leq \bar{\alpha}(\omega(x)) \quad \forall x \in (C \bigcup D \bigcup G(D)) \bigcap \mathcal{U};$$

$$\langle \nabla V(x), f \rangle \leq -V(x) \quad \forall x \in C \bigcap \mathcal{U}, f \in F(x);$$

$$V(g) \leq V(x)/e \quad \forall x \in D \bigcap \mathcal{U}, g \in G(x).$$

A smooth Lyapunov function for $\mathcal{A}$ on $\mathcal{U}$ is a Lyapunov function for $\mathcal{A}$ on $\mathcal{U}$ which is continuously differentiable on the whole set $\mathcal{U}$.

A.1.3. Lyapunov's direct method

A continuously differentiable function ($\mathcal{C}^1$) $V : \mathcal{R}^n \to \mathcal{R}$, is considered *positive definite* if $V(0) = 0$ and $V(x) > 0, \forall x \neq 0$. If V can be said to be both positive definite and radially unbounded, the there exist two class $\mathcal{K}_\infty$ functions α_1, α_2 such that V satisfies

$$\alpha_1(|x|) \leq V(x) \leq \alpha_2(|x|). \tag{A.3}$$

The function $\dot{V}$ is the derivative of V along solutions of the system (A.1), i.e.:

$$\dot{V} = \frac{\partial V}{\partial x} f(x).$$

The main result of Lyapunov's stability theory is expressed by the following theorem.

Theorem A.1.1. *(Lyapunov) Suppose that there exists a positive definite $\mathcal{C}^1$ function $V : \mathcal{R}^n \to \mathcal{R}$ whose derivative along solution of the system (A.1) satifies*

$$\dot{V} \leq 0 \quad \forall x. \tag{A.4}$$

Then the system (A.1) is stable. if the derivative of V satisfies

$$\dot{V} \leq 0 \quad \forall x \neq 0 \tag{A.5}$$

then system (A.1) is asymptotically stable. If in the latter case V is also radially unbounded, then (A.1) is globally asymptotically stable.

A positive definite $\mathcal{C}^1$ function V is referred to as a *weak Lyapunov function* if it satisfies inequality (A.4) and a *Lyapunov function* if it satisfies the inequality (A.5). The conclusion of the theorem remains valid even if V is only continuous, without being $\mathcal{C}^1$, provided that the both inequalities in the theorem are replaced by the conditions that V is non-increasing and strictly decreasing along nonzero solutions, respectively.

A.1.4. LaSalle's invariance principle

With some additional knowledge about the behaviour of solutions, it is possible to prove asymptotic stability using a weak Lyapunov function, which satisfies the non-strict inequality (A.4). This is facilitated by the so called *LaSalle's invariance principle*. A set Q is called *invariant* with respect to the given system if all solutions starting in Q remain in that set, for all time.

Theorem A.1.2. *(LaSalle) Suppose that there exists a positive definite $\mathcal{C}^1$ function $V : \mathcal{R} \to \mathcal{R}$ whose derivative along solution of system (A.1 satisfies inequality (A.4). Let Q be the largest invariant set contained in the set $x : \dot{V}(x) = 0$. Then the system (A.1)is stable and every solution that remains bounded for $t \geq 0$ approaches Q as $t \to \infty$. In particular, if all solutions remain bounded and $Q = 0$, then the system (A.1) is globally asymptotically stable.*

To deduce global asymptotic stability with the help of this result, one needs to check two conditions. First, all solutions of the system must be bounded. This follows naturally from inequality (A.4). Second, $\dot{V}$ is not identically zero along any nonzero solution. To note is that, if only asymptotic convergences of bounded solution to zero is to be proved, with out worrying about the Lyapunov stability of the origin, then positive definiteness of V is not needed.

A.1.5. Lyapunov's indirect method

Lyapunov's indirect method allows to reduce stability properties of a nonlinear system (the system function must be $\mathcal{C}^1$) from stability properties of its *linearization*, which for a $\dot{x} = Ax$ system is as follows:

$$A := \frac{\partial f}{\partial x}(0). \tag{A.6}$$

By the mean value theorem, it can be written:

$$f(x) = AX + g(x)x$$

where g is given component wise by $g_i(x) := \frac{\partial f}{\partial x}(z_i) - \frac{\partial f}{\partial x}(0)$ for some point z_i on the line segment connecting x to the origin, $i = 1, \dots, n$. Since $\frac{\partial f}{\partial x}$ is continuous, it is true that $g(x) \to 0$ as $x \to 0$. From this it follows that if the matrix A is Hurwitz (all eigenvalues strictly negative), then a quadratic Lyapunov function for the linearization serves (locally) as a Lypunov function for the original non-linear system. Additionally, the rate of decay int he neighbourhood of the origin can be bounded from below by a quadratic function, which implies exponential stability.

Theorem A.1.3. *If f is $\mathcal{C}^1$ and the matrix (A.6) is hurwitz, then the system (A.1) is locally exponentially stable.*

It is also known that if the matrix A has at least one pole with a positive real part (or part bigger then 1 for discrete systems), the original non-linear system is not stable. If A has poles on the imaginary axis, but no poles in the open right half-plan (or part bigger then 1 for discrete systems), the linearisation test is inconclusive. On the other hand, in this critical case system (A.1) cannot be exponentially stable, since exponential stability of the liniearization is not only a sufficient, but also necessary condition for a system to be *locally exponentially stable.*

A.2. Hybrid theory

Two topics in hybrid systems are very important and need explicit explanation for complete understanding of the topic of hybrid theory.

A.2.1. Sliding modes

Consider a switching system with state-dependent switching, described by single surface $\mathcal{S}$ and two subsystems $\dot{x} = f\_i(x), i = 1, 2$ one on each side of the surface. We so far assumed that when the continuous trajectory hits $\mathcal{S}$, it crosses over. Such behaviour can only be observed if both system vectors point in the same direction relative to $\mathcal{S}$. Problems arise when the said vectors both point towards the switching surface $\mathcal{S}$. According to Filippov's definition, one enriches the set of admissible velocities for points $x \in \mathcal{S}$ by including all convex combinations of the vectors $f\_1(x)$ and $f\_2(x)$. Thus an absolutely continuous function $x(\cdot)$ is a solution of the switched system in the sense of Filippov if it satisfies the *differential inclusion*:

$$\dot{x} \in F(x) \tag{A.7}$$

where F is a multi-valued function.

Following Filippov solution, it is easy to deduce that the moment we hit the switching surface we never leave it "slid" along it. Thus we obtained so called *sliding mode*. Sliding mode can be interpreted as infinity fast switching, which we call *chattering*. This phenomenon is undesirable in real life as it leads to very high actuator wear.

A.2.2. Zeno behaviour

A well-known example of so called *zeno behaviour* is a bouncing ball. A bouncing ball can have a physical interpretation that it is at rest within a finite time span, but after infinitely many bounces. This is a specific example of zeno behaviour, i.e. an infinite number of events in a finite length of time interval. In the case of the ball we have an infinite number of state reinitialisation and set of event times $\mathcal{E}$ for the bouncing ball contains a so-called *right-accumulation point*. In this example, each time the ball bounces it loses energy, making the subsequent jumps (impacts with the ground) closer and closer together in time. Detecting possible Zeno behaviour and extending them beyond their accumulation point for complicated hybrid systems, is far from trivial.

B

Simulation results of the Controllers

Int his appendix the results of the control simulations are shown that did not make it in to the main body of the thesis.

Figure (B.1) illustrates the behaviour of the system when a sinusoid of 1Hz is applied. The system clearly does not behave in a desired way. Serving as an additional conformation that the nature of the switching is not the issue.

Figure (B.2) illustrates the behaviour of the system when a sinusoid of 50Hz is applied. The system clearly does not behave in a desired way. That said, a clear improvement can be sin over the 1Hz case. That is attributed tot he nature of the reference signal, rather than any influence of the dwell time.

Figure (B.3) illustrates the behaviour of the system when a sinusoid of 1Hz is applied. The system clearly does not behave in a desired way. The system becomes unstable due to the limitations placed on it. The use of anti-wind up, back calculation is clearly visible, but rather then help it drives the output further away from the reference.

Figure (B.4) illustrates the behaviour of the system when a sinusoid of 50Hz is applied. The system clearly does almost behave in a desired way. The behaviour observed is even better then for the 25Hz case, in the main body of the thesis as the same reset time was used. This further strengthens the notion that anti-wind up is a possible solution which would need further research.

Figure (B.5) illustrates the behaviour of the system when a sinusoid of 1Hz is applied, with no constraints are applied. Clearly the MPC controller achieves perfect trucking.

Figure (B.6) illustrates the behaviour of the system when a sinusoid of 1Hz is applied, with constraints active. Clearly the MPC controller can deal with the force limits, by restricting the motion of the load when the limits are breached. This solves the problem which could not be over come by dwell-time and the simple implementation of anti-wind up.

Figure (B.7) illustrates the behaviour of the system when a sinusoid of 50Hz is applied, with no constraints are applied. Clearly the MPC controller achieves perfect trucking.

Figure (B.8) illustrates the behaviour of the system when a sinusoid of 50Hz is applied, with constraints active. Clearly the MPC controller can deal with the force limits, by restricting the motion of the load when the limits are breached. The trucking is not perfect, but it is an improvement over the other techniques applied. Further samll tuning would result in perfect trucking.

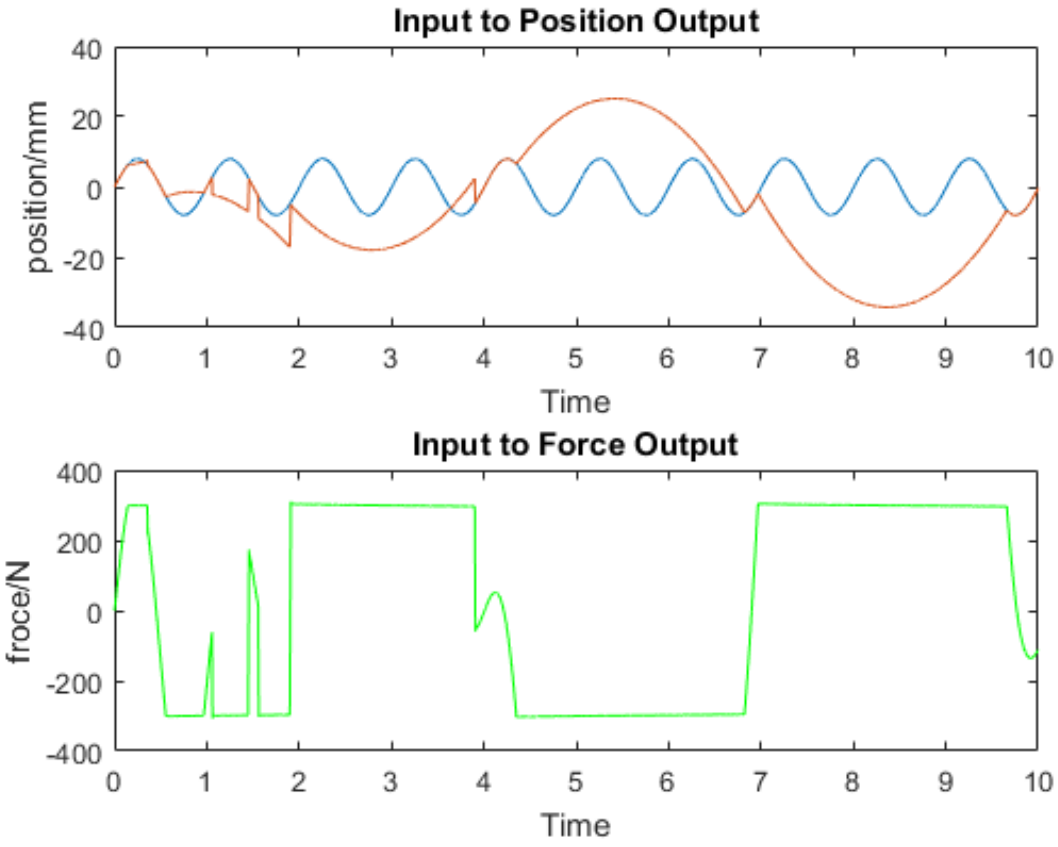


Figure B.1: Simulation of the system with a sinusoidal input of 1Hz, constrained with active dwell time.

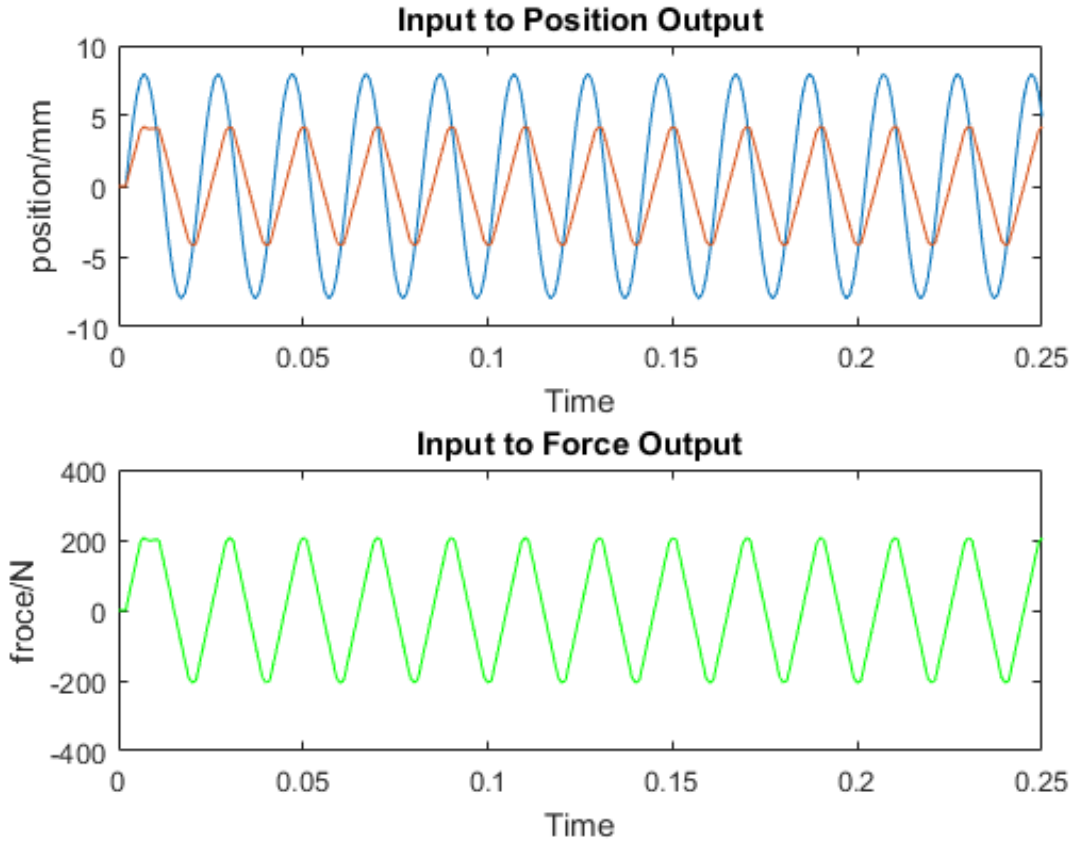


Figure B.2: Simulation of the system with a sinusoidal input of 50Hz, constrained with active dwell time.

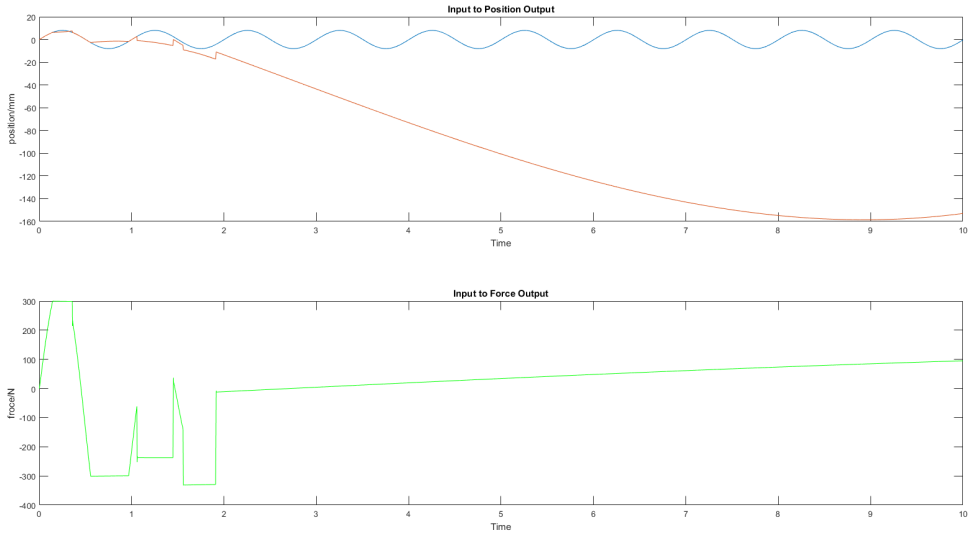


Figure B.3: Simulation of the system with a sinusoidal input of 1Hz, constrained with anti-wind up implemented

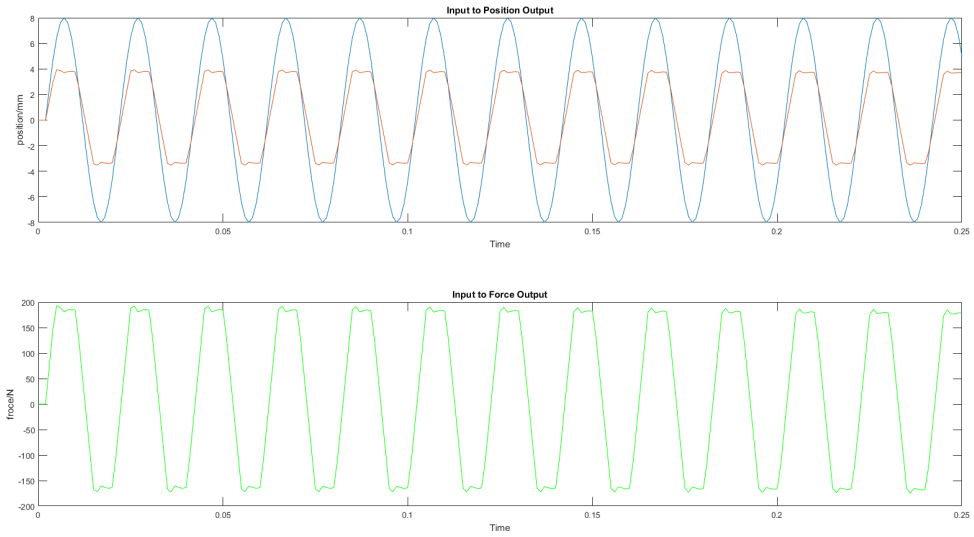


Figure B.4: Simulation of the system with a sinusoidal input of 1Hz, constrained with anti-wind up implemented

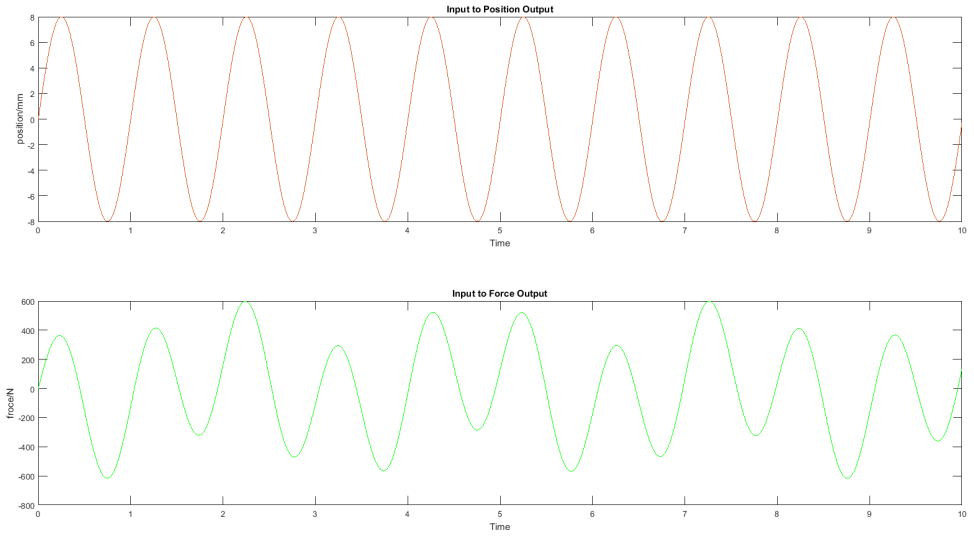


Figure B.5: System controlled by MPC, acting on a sinusoidal input of 1Hz. No constraints.

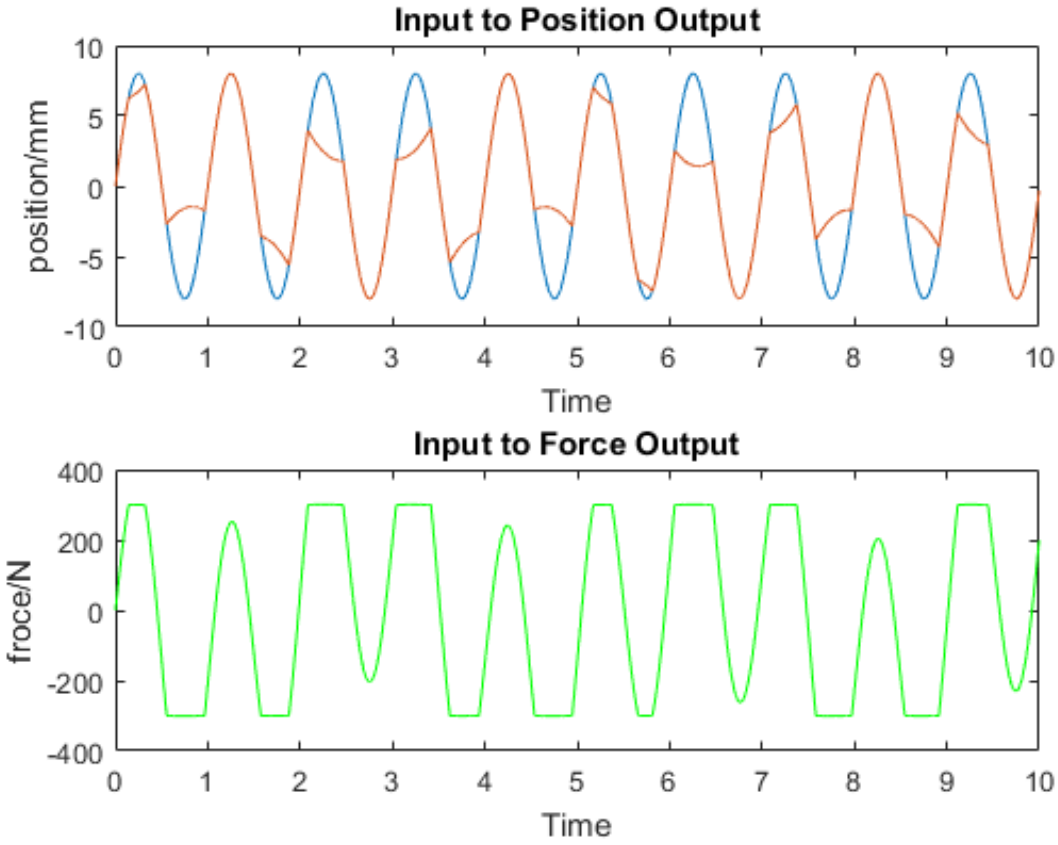


Figure B.6: System controlled by MPC, acting on a sinusoidal input of 50Hz. Constraints active.

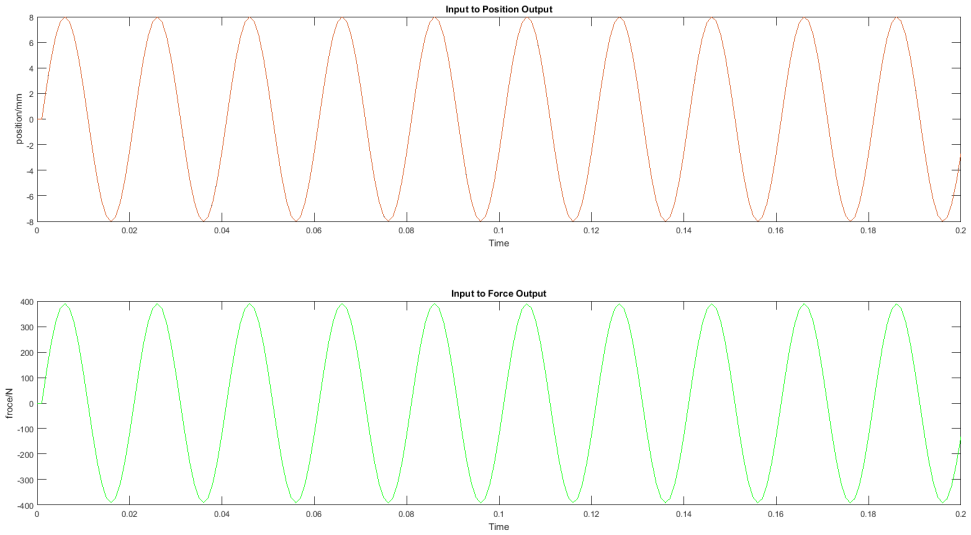


Figure B.7: System controlled by MPC, acting on a sinusoidal input of 1Hz. No constraints.

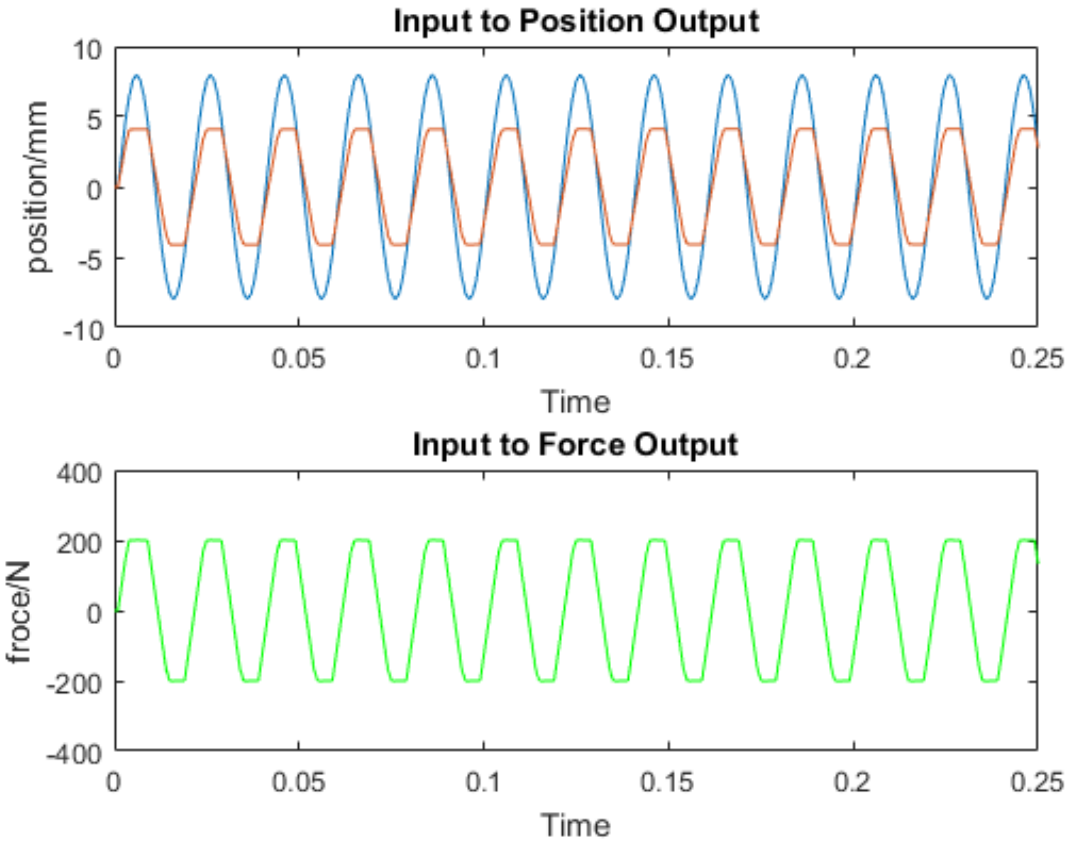


Figure B.8: System controlled by MPC, acting on a sinusoidal input of 50Hz.Constraints active.

C

MatLab code

This appendix presents the MatLab code used to achieve system equations for the GSSA. Three different parts can be distinguished within the code: Calculating the SS matrices from linearised system equations and connecting them, System Identification, and different loop equations.

C.1. Finding model

The following is the code used to find the A,B,C,D matrices, the PI controllers, and the series interconnection between them:

```
%% Parameter values
% Electric circuit
p = 6; % Number of poles of the ac motor
R = 378e-3; % Stator Resistance - Ohm
Ld = 3.427e-3; % Inductance in the other H
Lq = 3.334e-3; % Inductance in stator H
Ke = 1.13; % back emf

% Mechanical crank shaft
Jt = 57e-4; % Total inertia
Kt = 1.39; % Torque constant
ba = 0.1; % viscous damping
Tl = 1e-4; % Torque load

% Kinematics
l = 80; % length of the rod
r = 22.5; % radius of crank
q = 9; % offset

% Double mass
M = 10; % Mass of the load
Jm = 6.375e-04; % Inertia of the motor
ks = 48.75; % spring stiffens
kb = 4.875; % joint stiffens
bl = 0.1; % damping in the spring

% PI velocity
c1_v = 7.813;
c2_v = 4.867;
c3_v = 1;
c4_v = 100.4;
```

```
% PI current
c1_i = 30;
c2_i = 1.05e+04;
c3_i = 1;
c4_i = 510;
%% State-Space of PI velocity
syms c1_v c2_v c3_v c4_v
Api_v = -c4_v/c3_v;
Bpi_v = 1;
Cpi_v = (c3_v*c2_v - c1_v*c4_v)/(c3_v^2);
Dpi_v = c1_v/c3_v;

%% State-Space of PI current
syms c1_i c2_i c3_i c4_i
Api_i = -c4_i/c3_i;
Bpi_i = 1;
Cpi_i = (c3_i*c2_i - c1_i*c4_i)/(c3_i^2);
Dpi_i = c1_i/c3_i;

%% State-Space
syms Lq Ld R p lambda Ke Vd Vq
syms Jt b k Tr Tl Kt

syms id iq theta d_theta

%current----------------------------------------
% nonlinear current equations
d_id = (1/Ld)*Vd - (R/Ld)*id + (Lq/Ld)*p*d_theta*iq;
d_iq = (1/Lq)*Vq - (R/Lq)*iq - (Ld/Lq)*p*d_theta*id...
- ((lambda*p)/Lq)*d_theta;

f = [d_id d_iq];

%linearisation
A_i = jacobian(f,[id iq]);
B_i = jacobian(f,[Vd Vq]);
C_i = eye(2);
D_i = 0;

%substituting values
A_i = subs(A_i,[id iq d_theta],[0 0 0]);
A_i = subs(A_i,[R p],[0.378 6]);
A_i = subs(A_i,[Ld Lq],[3.334e-3 3.334e-3]);
B_i = subs(B_i,[Ld Lq],[3.334e-3 3.334e-3]);
A_i = double(A_i);
B_i = double(B_i);

%velocity---------------------------------------
% nonlinear velocity equations
dd_theta = (1.5*p*((Ld-Lq)*iq*id))/Jt + (Kt/Jt)*iq - (b/Jt)*d_theta;

f = [d_theta dd_theta];

%linearisation
A_vel = jacobian(f,[theta d_theta]);
B_vel = jacobian(f, [id iq]);
```

```
C_vel = eye(2):
D_vel = 0;


    %substituting values
A_vel = subs(A_vel,[id iq theta d_theta],[0 0 0 0]);
B_vel = subs(B_vel,[id iq theta d_theta],[0 0 0 0]);
A_vel = subs(A_vel,[Ld Lq],[3.427e-3 3.334e-3]);
B_vel = subs(B_vel,[Ld Lq],[3.427e-3 3.334e-3]);
A_vel = subs(A_vel,[Jt b],[0.0057 3.0073e-04]);
B_vel = subs(B_vel,Kt,191.2949);

%% Interconnection between subsystems
% Series interconection PI with current system
Aser_i = [[Api_i, 0;0, Api_i], zeros(size(A_i));
B_i*Cpi_i,A_i];
Bser_i = [[Bpi_i 0;0 Bpi_i];
B_i*Dpi_i];
Cser_i = [[D_i*Cpi_i 0;0 D_i*Cpi_i], C_i];
Dser_i = D_i*Dpi_i;

%current loop
Aloop_i = (Aser_i - Bser_i*eye(2)*Cser_i);
Bloop_i = Bser_i;
Cloop_i = Cser_i;
Dloop_i = Dser_i;

Aloop_i = subs(Aloop_i,[R c3_i],[0.378 1]);
Bloop_i = subs(Bloop_i,c3_i,1);
Aloop_i = subs(Aloop_i,[Lq Ld],[3.334e-3 3.427e-3]);
Bloop_i = subs(Bloop_i,[Lq Ld],[3.334e-3 3.427e-3]);
Aloop_i = subs(Aloop_i,[c1_i c2_i c4_i],[80.3333 1.1017e+04 263.0465]);
Bloop_i = subs(Bloop_i,[c1_i c2_i c4_i],[80.3333 1.1017e+04 263.0465]);

% Velocity loop
% Series interconnection PI with velocity system and current loop
Atemp_v = [Aloop_i, zeros(4,2);
B_vel*Cloop_i, A_vel];
Atemp_v(4,6) = -Ke/Lq;
Btemp_v = [Bloop_i;
B_vel*Dloop_i];
Ctemp_v = [D_vel*Cloop_i, C_vel];
Dtemp_v = Dloop_i*D_vel;

Aser_v = [Api_v, zeros(1,6);
Btemp_v(:,2)*Cpi_v, Atemp_v];
Bser_v = [Bpi_v;
Btemp_v(:,2)*Dpi_v];
Cser_v = [[0; Dtemp_v*Cpi_v], Ctemp_v];
Dser_v = Dpi_v*Dtemp_v;

%velocity loop
Aloop_v = (Aser_v - Bser_v*[0 1]*Cser_v);
Bloop_v = Bser_v;
Cloop_v = Cser_v(2,:);
Dloop_v = Dser_v;
```

```
Aloop_v = subs(Aloop_v,[Lq p c3_v],[3.334e-3 6 1]);
Bloop_v = subs(Bloop_v,c3_v,1);
Aloop_v = subs(Aloop_v,[c1_v c2_v c4_v],[7.813 4.8671 100.4]);
Bloop_v = subs(Bloop_v,[c1_v c2_v c4_v],[7.813 4.8671 100.4]);
Aloop_v = subs(Aloop_v,Ke,0.0169);

Aloop_v = double(Aloop_v);
Bloop_v = double(Bloop_v);
Cloop_v = double(Cloop_v);
Dloop_v = double(Dloop_v);

% Disturbance = Torque load
B_d = [0; 0; 0; 0; 0; 0; -(1/Jt)];
B_d = subs(B_d,Jt,0.0057);
B_d = double(B_d);

%% Kinematics and Dynamics of a two mass system
syms xl d_xl
syms r l q kb ks bl M Jm

% Angle of shaft to linear position
x = l*(1- (r/l*sin(theta)-q/l)^2)^(1/2)-r*cos(theta);

% Angular velocity of shaft to linear velocity
d_x = d_theta*(r*sin(theta) + (r*cos(theta)*(q/l - (r*sin(theta))/l))/...
(1 - (q/l - (r*sin(theta))/l)^2)^(1/2));

% Angluar acceleration of shaft to linear acceleration
dd_theta = (1.5*p*((Ld-Lq)*iq*id))/Jm + (Kt/Jm)*iq...
- (b/Jm)*d_theta...
- ((b+bl)/Jm)*d_x + (bl/Jm)*d_xl...
- (kb/Jm)*x + (kb/Jm)*xl;

%equations of moments, of the two mass system
dd_xl = (bl/M)*d_x - (bl/M)*d_xl + (kb/M)*x - ((kb+ks)/M)*xl;
F2 = M*dd_xl;

% Output equations
g = [x; F2];

%The output matrix of the full system
C = jacobian(g,[id iq theta d_theta xl d_xl]);
C = subs(C,[id iq theta d_theta xl d_xl],[0 0 0 0 0 0]);
C = subs(C, [Jm b p Kt],[6.375e-04 3.0073e-04 6 191.2949]);
C = subs(C, [r l q M kb ks bl],[22.5 80 9 10 48.75 4.875 0.001]);
C = double(C);
C_GSSA = [zeros(2,3) C];

%The state matrix A of the full system
block1 = [zeros(6,2);0 0];
block2 = [zeros(2,5),[0 0;0 0]];
block3 = [0 1;0 0];
Awhole_pf = [Aloop_v block1;block2 block3];

f2 = [d_theta dd_theta d_xl dd_xl];
```

```
block = jacobian(f2,[id iq theta d_theta xl d_xl]);
block = subs(block,[id iq theta d_theta xl d_xl],[0 0 0 0 0 0]);
Awhole_pf(6:end,4:end) = block;
A_GSSA = subs(Awhole_pf,[Jm b p Kt],[6.375e-04 3.0073e-04 6 191.2949]);
A_GSSA = subs(A_GSSA,[r l q M kb ks bl],[22.5 80 9 10 48.75 4.875 0.001]);
A_GSSA = double(A_GSSA);

%The input matrix B of the full system
B_GSSA = [Bloop_v*((l*(1 - q^2/l^2)^(1/2))/(q*r));0;0];
B_GSSA = subs(B_GSSA,[r l q],[22.5 80 9]);
B_GSSA = double(B_GSSA);

%Feedforward term
D_GSSA = zeros(2,1);

%creating the full SS system
sys_moj_open = ss(A_GSSA,B_GSSA,C_GSSA,D_GSSA);

%finding minimal realisation
sys_moj_open = minreal(sys_moj_open);
sys_moj_open.StateName = 'e_v','e_iq','iq','theta','d_theta','x','d_x';
sys_moj_open.InputName = 'v_ref';
sys_moj_open.OutputName = 'Position','Force';

%discretisation of the system
Ts = 1e-3;
sys_moj_d = c2d(sys_moj_open,Ts,'zoh');
(A_dz,B_dz,C_dz,D_dz) = ssdata(sys_moj_d);

%% Current loop equations
syms x1 x2 x3 x4 u1 u2

dot_x_i = Aloop_i*[x1;x2;x3;x4] + Bloop_i*[u1;u2];
y_i = Cloop_i*[x1;x2;x3;x4];

%% Velocity loop equations
syms x1 x2 x3 x4 x5 x6 x7 u

dot_x_v = Aloop_v*[x1; x2; x3; x4; x5; x6; x7] + Bloop_v*u + B_d;
y_v = Cloop_v*[x1; x2; x3; x4; x5; x6; x7];

%% Whole system equations
syms x1 x2 x3 x4 x5 x6 x7 x8 x9 u

dot_x_w = A_GSSA*[x1; x2; x3; x4; x5; x6; x7; x8; x9] + B_GSSA*u;
y_w = C_GSSA*[x1; x2; x3; x4; x5; x6; x7; x8; x9];
```

C.2. System Identification

The following code shows the manner in which the system parameters were optimised and validated:

```
%% Parameter values
% Electric circuit
p = 6; % Number of poles of the ac motor
R = 378e-3; % Stator Resistance - Ohm
Ld = 3.427e-3; % Inductance in the other H
Lq = 3.334e-3; % Inductance in stator H
```

```
Ke = 1.13; % back emf

% Mechanical crank shaft
Jt = 57e-4; % Total inertia
Kt = 1.39; % Torque constant
ba = 0.1; % viscous damping
Tl = 1e-4; % Torque load

% Kinematics
l = 80; % length of the rod
r = 22.5; % radius of crank
q = 9; % offset

% Double mass
M = 10; % Mass of the load
Jm = 6.375e-04; % Inertia of the motor
ks = 48.75; % spring stiffens
kb = 4.875; % joint stiffens
bl = 0.1; % damping in the spring

% PI velocity
c1_v = 7.813;
c2_v = 4.867;
c3_v = 1;
c4_v = 100.4;

% PI current
c1_i = 30;
c2_i = 1.05e+04;
c3_i = 1;
c4_i = 510;

%% Establishing data set
prompt = 'Current or velocity? (Current = 1/Velocity = 2/Position = 3)';
kvob = input(prompt);

(y,u,t_sample) = input_output(kvob);
z = iddata(y,u,t_sample,'Name','system_data');
%% Optimisation
% Three options are available
if kvob == 1 % Current
z.InputName = 'Curren_d','Currnet_q';
z.InputUnit = 'A','A';
z.OutputName = 'Current_d','Currnet_q';
z.OutputUnit = 'A','A';

FileName = 'equations_system_i';

Order = [2 2 4];

Parameters = [c1_i; c2_i; c4_i; R; Ld; Lq];

InitialStates = [0; 0; 0; 0];

Ts = 0;
```

```
nlgr = idnlgrey(FileName, Order, Parameters, InitialStates, Ts,...
'Name', 'current_model');

set(nlgr, 'InputName', 'Current_d','Currnet_q', 'InputUnit',...
'A','A','OutputName', 'Current_d','Currnet_q',...
'OutputUnit', 'A','A','TimeUnit', 's');

nlgr = setpar(nlgr, 'Fixed', 0,0,0,1,1);
nlgr.Parameter(1).Minimum = 0;
nlgr.Parameter(2).Minimum = 0;
nlgr.Parameter(3).Minimum = 0;
nlgr = setinit(nlgr, 'Fixed', false false false false);
disp('We are optimising');

opt = nlgreyestOptions('Display', 'on');
opt.SearchOption.MaxIter = 10;
nlgr_opt = nlgreyest(z, nlgr, opt);

elseif kvob == 2 % Velocity
z.InputName = 'Velocity ref';
z.InputUnit = 'rad/s';
z.OutputName = 'Velocity feedback';
z.OutputUnit = 'rad/s';
z.Tstart = t_sample;
z.TimeUnit = 's';

FileName = 'equations_system_v';

Order = [1 1 7]:

Parameters = [c1_v; c2_v; c3_v; c4_v; Kt; ba; Jt];

InitialStates = [0; 0; 0; 0; 0; 0; 0];

Ts = 0;

nlgr = idnlgrey(FileName, Order, Parameters, InitialStates, Ts,...
'Name', 'velocity_model');

set(nlgr, 'InputName', 'Velocity ref', 'InputUnit', 'rad/s',...
'OutputName', 'Velocity feedback', 'OutputUnit',...
'rad/s', 'TimeUnit', 's');

nlgr = setpar(nlgr, 'Fixed', 0,0,0,0,0,0,0);
nlgr.Parameter(1).Minimum = 0;
nlgr.Parameter(4).Minimum = 0;
nlgr.Parameter(5).Minimum = 0;
nlgr.Parameter(6).Minimum = 0;
nlgr.Parameter(7).Minimum = 0;
nlgr = setinit(nlgr, 'Fixed', false false false false false...
false false);
disp('We are optimising');

opt = nlgreyestOptions('Display', 'on');
opt.SearchOption.MaxIter = 10;
nlgr_opt = nlgreyest(z, nlgr, opt);
```

```
else % position
z.InputName = 'Position_ref';
z.InputUnit = 'mm';
z.OutputName = 'Position_feed';
z.OutputUnit = 'mm';
z.Tstart = t_sample;
z.TimeUnit = 's';

FileName = 'GSSApos';

Order = [1 1 9];

InitialStates = [0; 0; 0; 0; 0; 0; 0];

Parameters = [r; l; q; M; kb; ks; bl];

Ts = 0;

nlgr = idnlgrey(FileName, Order, Parameters, InitialStates, Ts,...
'Name', 'postion_validation');

set(nlgr, 'InputName', 'Position_ref', 'InputUnit', 'mm',...
'OutputName', 'Position_feed', 'OutputUnit', 'mm',..
. 'TimeUnit', 's');

nlgr = setpar(nlgr, 'Fixed', 0,0,0,0,0,0,0);
nlgr = setinit(nlgr, 'Fixed', false false false false false...
false false);
nlgr.Parameter(3).Minimum = 0;
nlgr.Parameter(4).Minimum = 0;
nlgr.Parameter(5).Minimum = 0;
disp('We are optimising');

opt = nlgreyestOptions('Display', 'on');
opt.SearchOption.MaxIter = 10;
nlgr_opt = nlgreyest(z, nlgr, opt);
end

getpar(nlgr_opt)

%% Validation
disp('We are compering');
compare(z, nlgr);
```

C.3. Current equations

The following code gives the equations of the current loop:

```
function [d_x, y ] = equations_system_i(t, x, u, c1_i, c2_i, c4_i,R, Ld, Lq, varargin)

% Output equations
y = [x(3); % current d-axis
x(4)]; % current q-axis

    % State equations
d_x = [
```

```
u(1) - x(3) - (c4_i*x(1))/c3_i;
u(2) - x(4) - (c4_i*x(2))/c3_i;
(c1_i*u(1))/(Ld*c3_i) - x(3)*(R/Ld + c1_i/(Ld*c3_i)) - (x(1)*(c1_i*c4_i - c2_i*c3_i))/(Ld*c3_i^2);
(c1_i*u(2))/(Lq*c3_i) - x(4)*(R/Lq + c1_i/(Lq*c3_i)) - (x(2)*(c1_i*c4_i - c2_i*c3_i))/(Lq*c3_i^2)];
end
```

C.4. Velocity equations

The following code gives the equations describing the mechanical part in series with a PI controller and the current loop:

```
function [d_x, y] = equations_system_v(t, x, u, c1_v, c2_v, c3_v, c4_v,...
Kt, ba, Jt, varargin)

% Output equations
y = x(7); % angular velocity

% state equations
d_x =[
u - x(7) - (c4_v*x(1))/c3_v;
- (4627562966332473*x(2))/17592186044416 - x(4);
(c1_v*u)/c3_v - x(5) - (x(1)*(c1_v*c4_v - c2_v*c3_v))/c3_v^2 - ...
(4627562966332473*x(3))/17592186044416 - (c1_v*x(7))/c3_v;
- (125210125610249800605676743282 47*x(2))/42424205146309981089300 48 - ...
(7754482786231216799744*x(4))/329255166357305375;
(23154497276508303360*c1_v*u)/(960960074089807*c3_v) -...
(2907931044836706299904*x(5))/120120009261225875 - ...
x(7)*((23154497276508303360*c1_v)/(960960074089807*c3_v)...
+ 169/(10000*Lq)) - (23154497276508303360*x(1)*(c1_v*c4_v...
- c2_v*c3_v))/(960960074089807*c3_v^2) - ...
(125210125610249800605676743282 47*x(3))/41272920909774580319518 72;
x(7);
(Kt*x(5))/Jt - 1/Jt - (ba*x(7))/Jt];
end
```

C.5. Full system equations

The following is the code which establishes the equations of the whole SS system representing the GSSA: function [d_x, y] = equations_system_whole(t, x, u, r, l, q, M, kb, ks, bl, varargin)

```
% Output equations
y =[(q*r*x(6))/(l*(1 - q^2/l^2)^(1/2))
(bl*q*r*x(7))/(l*(1 - q^2/l^2)^(1/2)) - bl*x(9) - x(8)*(kb + ks) +
(kb*q*r*x(6))/(l*(1 - q^2/l^2)^(1/2))];

% state equations
d_x = [
(l*u*(1 - q^2/l^2)^(1/2))/(q*r) - x(7) - (502*x(1))/5;
- (4627562966332473*x(2))/17592186044416 - x(4);
(7813*l*u*(1 - q^2/l^2)^(1/2))/(1000*q*r) -...
(4627562966332473*x(3))/17592186044416 - x(5) - (7813*x(7))/1000 ...
- (7795581*x(1))/10000;
- (125210125610249800605676743282 47*x(2))/42424205146309981089300 48 ...
- (7754482786231216799744*x(4))/329255166357305375;
(452265218053398435379 2*l*u*(1 - q^2/l^2)^(1/2))/ ...
(24024001852245175*q*r) - (125210125610249800605676743282 47*x(3))...
/41272920909774580319518 72 - (2907931044836706299904*x(5)) ...
```

```
/120120009261225875 - (113069348946697711300096*x(7))/600600046306129375 ...
- (22562844879162484501931 52*x(1))/120120009261225875;
x(7);
(4206619337684943125*x(5))/14018773254144 - x(7)*((80000*q*r*(bl...
+ 5547489345286673/18446744073709551616))/(51*l*(1 - q^2/l^2)^(1/2))...
+ 3467180840804170625/7349874591868649472) + (80000*bl*x(9))/51 +...
(80000*kb*x(8))/51 - (80000*kb*q*r*x(6))/(51*l*(1 - q^2/l^2)^(1/2));
x(9);
(bl*q*r*x(7))/(M*l*(1 - q^2/l^2)^(1/2)) - (bl*x(9))/M...
- (x(8)*(kb + ks))/M + (kb*q*r*x(6))/(M*l*(1 - q^2/l^2)^(1/2))];
end
```