JOSÉ GENARIO DE OLIVEIRA JÚNIOR

MODEL PREDICTIVE CONTROL APPLIED TO A 2-DOF HELICOPTER

São Paulo
2018

JOSÉ GENARIO DE OLIVEIRA JÚNIOR

MODEL PREDICTIVE CONTROL APPLIED TO A 2-DOF HELICOPTER

Dissertation submitted to Escola Politécnica of the University of São Paulo for the degree of Master of Science.

São Paulo
2018

JOSÉ GENARIO DE OLIVEIRA JÚNIOR

MODEL PREDICTIVE CONTROL APPLIED TO A 2-DOF HELICOPTER

Dissertation submitted to Escola Politécnica of the University of São Paulo for the degree of Master of Science.

Area of Concentration:

3139 - Systems Engineering

Advisor:

Prof. Claudio Garcia

São Paulo
2018

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catalogação-na-publicação

de Oliveira Júnior, José Genario
Model Predictive Control applied to a 2-DOF Helicopter / J. G. de Oliveira Júnior -- versão corr. -- São Paulo, 2018.
102 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Telecomunicações e Controle.

1.Controle Preditivo 2.Eletrônica Embarcada 3.Identificação de Sistemas I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Telecomunicações e Controle II.t.

To Carlinda, Maire and everyone that made this work possible.

ACKNOWLEDGMENTS

First, I would like to thank my supervisor, Professor Claudio Garcia, who was always eager to help, dedicated and inspiring. Second, to Professor Bruno Angélico, which is the head of the laboratory in which this work took place, for always providing interesting ideas and for all the help during my Master's studies. I am also grateful to Professor Darci Odloak, who sparked my interest in MPC, in the first place, and was always open to discussions about the many facets of an MPC implementation. Also, to Professor Jaime for the great insights on a wide array of topics and for the meticulous proof-reading of this work.

I would also like to thank everyone from LCA, which really made this work possible, especially Fernando and Giovanni for the immense help with the practical experiments and the others for the enlightening debates. Also, I would like to thank Dr. Christiam Alvarado for the initial help with system identification.

Finally, I would like to thank my mother Carlinda and Maire for being always understanding and supportive during this whole time.

RESUMO

Este trabalho apresenta uma aplicação de controle preditivo embarcado em um helicóptero de bancada com dois graus de liberdade.

A modelagem matemática é apresentada, junto com uma análise do modelo linear obtido. São obtidas duas representações de modelos de espaço de estados considerando a entrada incremental, que serão usadas posteriormente para a formulação do controlador.

Então, é definida a técnica de controle utilizada, juntamente com a inclusão das restrições físicas da planta na formulação do problema. Após isto, é feita uma discussão sobre qual *solver* para a programação quadrática utilizar, junto com algumas alternativas ao *solver* escolhido, bem como algumas considerações sobre a aplicação embarcada.

Finalmente, são apresentados os resultados da identificação de sistemas aplicadas ao protótipo, bem como os resultados experimentais obtidos.

Palavras-Chave – Controle Preditivo, Eletrônica embarcada, Identificação de Sistemas.

ABSTRACT

This work presents an embedded model predictive control application to a 2-DOF Helicopter Process.

The mathematical modeling of the plant is first presented along with an analysis of the linear model. Then, the incremental state-space representations used in the MPC formulation are derived.

The MPC technique is then defined, along with how to rewrite the physical constraints into the problem formulation. After that, a discussion on the utilized Quadratic Programming solver is presented along with possible alternatives to it, showing some considerations on which matrices to calculate beforehand for an embedded application.

Finally, system identification is performed and the experimental results are presented.

Keywords – Model Predictive Control, Embedded Electronics, System Identification.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

LIST OF ABBREVIATIONS

ABB	ASEA Brown Boveri
BLDC	Brushless DC Motor
CAD	Computer-Aided Design
DOF	Degrees of Freedom
ESC	Electronic Speed Controller
LCA	Applied Control Laboratory
LQG	Linear Quadratic Gaussian Regulator
LQR	Linear Quadratic Regulator
LTR	Loop Transfer Recovery
MPC	Model Predictive Control
MIMO	Multi-Input Multi-Output System
PID	Proportional-Integral-Derivative Controller
QP	Quadratic Programming
TWMS	Twin-Rotor MIMO System
Vestas	Vestas Wind Systems A/S

LIST OF SYMBOLS

$\boldsymbol{A, B, C}$	State, input and output matrices, respectively
$\bar{a}$	value of variable a at operating point
$\boldsymbol{Co}$	Controllability matrix
$e(t)$	Error between reference and system output
F_p	Force of the Pitch Propeller
F_g	Weight
F_y	Force of the Yaw Propeller
g	Gravity acceleration
I_n	n-dimensional Identity Matrix
$\vec{i}, \vec{j}, \vec{k}$	Unit vectors of fixed basis O'
J_x, J_y, J_z	Inertia around x, y and z-axis, respectively to the center of gravity
J_xy, J_yz, J_xz	Inertia Products
J_G	Inertia Matrix
K_c	Proportional gain in PID controller
K_{pp}, K_{py}	Pitch DC motor gains
K_{yp}, K_{yy}	Yaw DC motor gains
l_{cm}	Distance from the center of rotation to the center of gravity
$\boldsymbol{L}$	Observer Gain
m	Mass of the system
O	Coordinate system origin solidary to the center of gravity
O'	Absolute coordinate system origin
$\boldsymbol{Ob}$	Observability matrix
r	l_{cm}
r_p	Distance from the center of rotation and the pitch propeller
r_y	Distance from the center of rotation and the yaw propeller
$r(t)$	Reference signal

$\vec{r}_G$	Position vector of the center of gravity in O'
T	Kinetic Energy
T_D	Derivative time in PID controller
T_I	Integral time in PID controller
T_s	Sampling time
$\boldsymbol{u}(t)$	Control signal
$\Delta\boldsymbol{u}(t)$	Incremental Control signal
$\Delta u(k+j\|k)$	Incremental control signal on sample k+j calculated at sample k
U	Potential Energy in respect to the center of rotation
$\vec{v}_G$	velocity vector of the center of gravity in O'
V_{mp}, V_{my}	Motor Voltage percentages
w	j-angular speed vector
$\vec{w}_\psi$	Angular velocity originated by ψ
$\vec{w}_\theta$	Angular velocity
$\boldsymbol{x}(t)$	System states
$y(t)$	System output
ψ	Yaw Angle
τ_θ, τ_ψ	Generalized Torques for θ and ψ
θ	Pitch Angle originated by θ

1 INTRODUCTION

Model Predictive Control is currently the default advanced control technique used in industrial processes. Historically, it had limited usage in embedded applications due to the amount of time needed to solve quadratic problems in common microcontroller boards.

However, with the advances in the raw computational power of embedded controller hardware, many applications in other industries have arisen. In the automotive industry, some examples of MPC applications include: idle speed regulation (CAIRANO et al., 2008), powertrain control, chassis control (HROVAT et al., 2012), direct injection (GIORGETTI et al., 2006) and active steering for autonomous driving (BORRELLI; FALCONE; KEVICZKY, 2011). Some aerospace applications of embedded MPC include areas such as powered descent (PASCUCCI; BENNANI; BEMPORAD, 2015) and planetary rovers (BINET; KRENN; BEMPORAD, 2012). Other embedded applications include wind-turbine control (HOVGAARD et al., 2013) and compressor drive control (BESSELMANN et al., 2016).

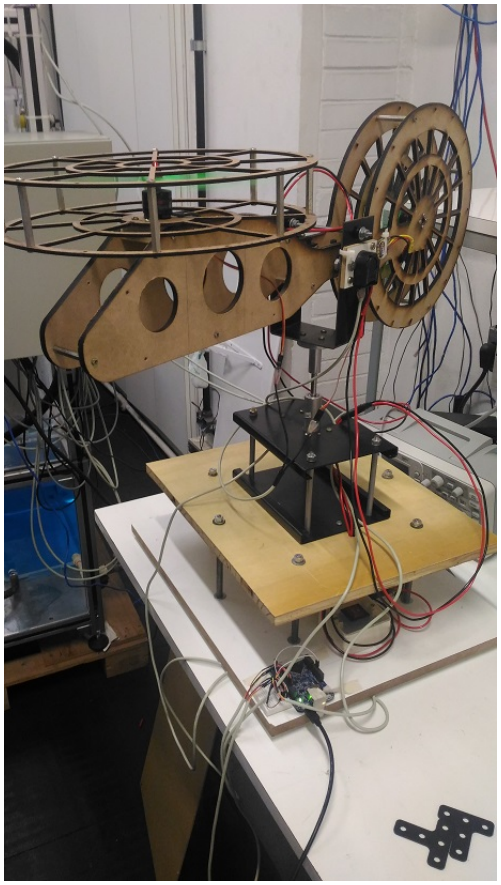
1.1 Objective

The objectives of this project are:

- Develop an embedded MPC controller for a 2-DOF Helicopter process.
- Perform a rigorous modeling of the physics involved in the helicopter motion.
- Improve the controller's response by using System Identification techniques.

The 2-DOF Helicopter is shown in 1 and is present at LCA-Escola Politécnica of the University of São Paulo - EPUSP.

Figure 1: Process at LCA-EPUSP



Source: Author

1.2 Methodology

This is attained by first using a rigorous modeling of the process to find a good initial guess for the dynamic model. Then, it is investigated how to embed the controller in a Freescale FRDM-K64F board shown in Figure 2 using C++ code.

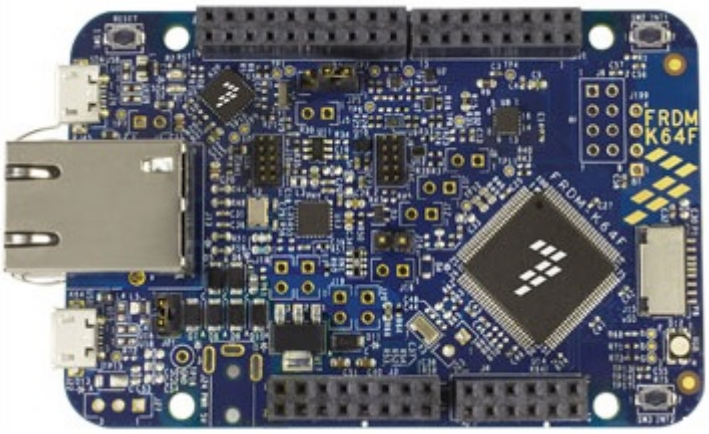
Finally, once a good solution of the embedded controller is found, system identification techniques are used in order to enhance the model for control.

Furthermore, the performance of the different models used in the MPC controller are compared against each other.

1.3 Justification

The topic of embedded MPC control is a recent one, with most impactful researches and applications being done in the last decade, such as the works already mentioned. This makes it an interesting research topic in a Master's Dissertation.

Figure 2: Freescale FRDM-K64F Developer Board



Source: http://www.nxp.com/

The real-time restrictions on embedded systems make the problem significantly harder from a computational point of view. This requires a great deal of thought into issues such as the type of model to use and how to select an appropriate Quadratic Programming (QP) solver because of the constraints involved.
The 2-DOF helicopter process is also a very interesting challenge from a control system perspective. A conceptual sketch of it is shown in Figure 3. Being a MIMO system with coupled outputs and hard constraints on inputs and outputs. The constraints on inputs/outputs can be directly rewritten as constraints in the MPC problem, making it a suitable controller choice from that perspective.
Also, since the 2-DOF helicopter is used at teaching, the embedded MPC controller is an unusual control design in fast mechanical processes when compared to more widespread techniques in the field, such as LQG/LTR or PID control.

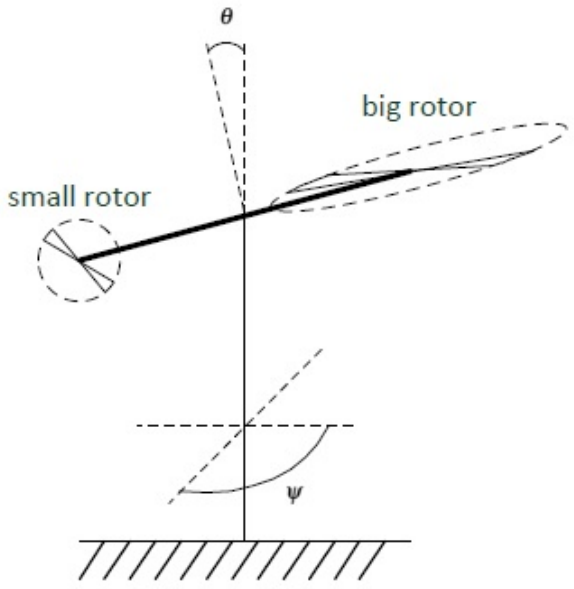
1.4 Literature review

There are plenty of works in the literature regarding 2-DOF Helicopters or Twin-Rotor MIMO systems (TWMS), with several different prototypes.

In (BARBOSA; NEVES; ANGÉLICO, 2016), the authors designed a LQG/LTR Controller for the same process used in this dissertation. The paper is interesting because it shows a practical example of how to choose the system barriers in the frequency domain for a MIMO system.

There are quite a few MPC control applications for 2DOF Helicopters. In (RAGHAVAN; THOMAS, 2017), the authors applied a similar state space MPC implementation

Figure 3: Sketch of the Helicopter model



Source: Author

to the one used in this work, verified how the tuning parameters change the output and compared the set point tracking performance against other controllers.

A version closer to the one presented here was done by (DUTESCU; RADAC; PRECUP, 2017), where the authors designed a state space MPC using the Hildreth's Quadractic Programming Procedure and an Extended Kalman filter for state estimation. A robust approach based on polytopic uncertainties was developed by (RAHIDEH; SHAHEED, 2009), based on the PhD Thesis of the same author (RAHIDEH, 2009) in which system identification is considered as well, along with a nonlinear MPC. In (SALAZAR; NEJJARI; SARRATE, 2014), it is described how to incorporate health monitoring of the actuators in the MPC problem for the same process.

Regarding applications to similar processes, (LIU; CHEN; ANDREWS, 2012), describes an interesting application of explicit MPC to a small scale RC helicopter. In (ALEXIS; NIKOLAKOPOULOS; TZES, 2011), MPC is used for attitude control of a quadrotor helicopter subject to atmospheric disturbances.

(WANG, 2009) describes the augmented state space model in order to eliminate steady state errors in the MPC formulation, along with a Hildreth Quadratic Programming solver, which is used for the embedded implementations.

(WANG; BOYD, 2010) presents an approximated interior-point QP solver that has

faster computational times than traditional ones and could be useful in an embedded application. The authors also mention an explicit solution for the MPC control law, turning the problem into a lookup table. This approach may be faster for embedded implementations if the system is simple enough, with low values for the prediction and control horizon.

In (BEMPORAD, 2016) it is shown how to write a convex QP problem in a non-negative least squares (LS) structure, making it possible to use robust and fast LS solvers for the MPC problem.

For the system identification section, the main references used are (LJUNG, 1999), which is a very well-known system identification book, (FORSELL, 1999), which is the basis for the closed-loop identification section and identification for control. Asymptotic identification (ZHU, 1994) is another approach that was used due to its formulation for MPC, along with results from (ROMANO; POTTS; GARCIA, 2012) as reference on MPC Relevant Identification.

2 SYSTEM MODELING

In this chapter, first the full non-linear differential equations of the system with the DC motors are developed, along with the physical parameters of the process. After that, the model is linearized and its continuous-time transfer matrix is presented. Furthermore, the discrete-time linearized transfer matrix is presented.

The discrete time state space in incremental form as in (WANG, 2009) is then developed, in order to eliminate the set point tracking error. Finally, it is shown a non-minimal state space representation called "Realignment Model" that eliminates the need of observer design, given its special choice of the system states.

2.1 Non-Linear Model

This section details how the non-linear differential equations are derived.

2.1.1 Lagrange Equation

The Lagrange equation is used to determine the dynamic model of the system (GARCIA, 2013). The Lagrange equation can be written as:

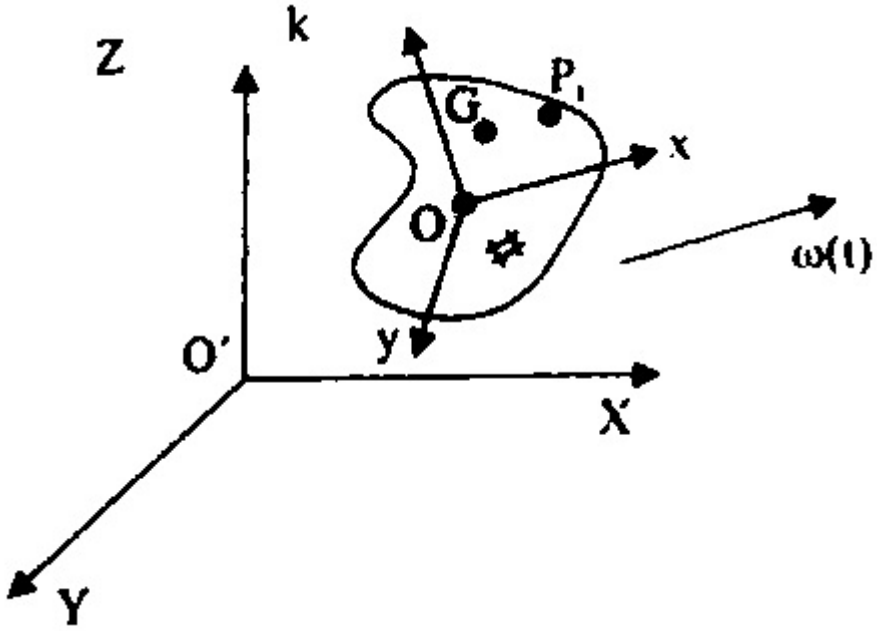
$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial T}{\partial \dot{q}_i} - \frac{\partial T}{\partial q_i} + \frac{\partial U}{\partial q_i} = Q_i, \tag{2.1}$$

where T is the total kinetic energy, U is the potential energy of the body, Q_i are the generalized forces that exist only in the non-conservative case and q_i are the system's independent coordinates.

2.1.1.1 Kinetic Energy

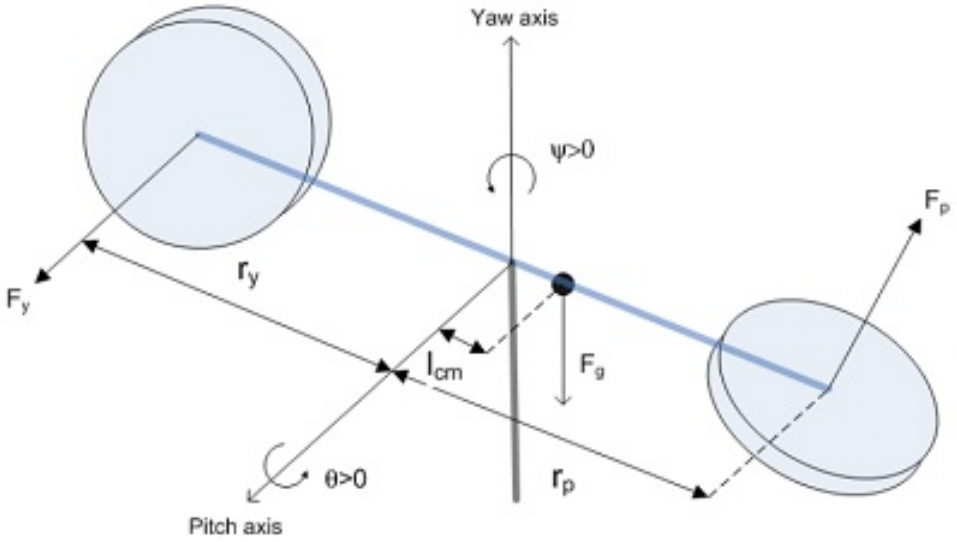
Consider a two coordinate systems O' and O as shown in Figure 4, with O' being coincident with the intersection between the yaw and pitch axis as shown in Figure 5 and

Figure 4: Coordinated Systems O and O'



Source: (KAMINSKI, 2000)

Figure 5: Free Body Diagram



Source: (BARBOSA; NEVES; ANGÉLICO, 2016)

O coincident with the system's center of gravity G.

The total kinetic energy is given by (KAMINSKI, 2000):

$$T = \frac{1}{2}m|\vec{v}_G|^2 + \frac{1}{2}\vec{w}^T J_G \vec{w}, \tag{2.2}$$

where m is the body's mass, $\vec{v}_G$ is the velocity of G in the absolute coordinate system O', w is the angular velocity of the body in the system O moving with the solid and J_G is the Inertia Matrix of the body related to O and can be written as:

$$J_G = \begin{bmatrix} J_x & -J_{xy} & -J_{xz} \\ -J_{xy} & J_y & -J_{yz} \\ -J_{xz} & -J_{yz} & J_z \end{bmatrix}, \tag{2.3}$$

where the terms J_i are the principal moments of inertia with respect to G and J_{ij} are the corresponding products of inertia. One important remark is that since the prototype has 2 planes of symmetry orthogonal to each other passing through the center of gravity, all of the products of inertia for this special case are zero.
Defining θ and Ψ as in Figure 5, one can write the position of G with respect to O' as:

$$\vec{r_G} = r\cos(\theta)\cos(\psi)\vec{i} + r\cos(\theta)\sin(\psi)\vec{j} + r\sin(\theta)\vec{k} \tag{2.4}$$

where $\vec{i}$, $\vec{j}$, $\vec{k}$ denotes the unit vectors of the basis O'. Note that there is an implicit assumption that the center of gravity is positioned along the axis perpendicular to the yaw and pitch axes.

Now one can calculate v_G by differentiating r_G with respect to time:

$$\begin{aligned}\vec{v_G} = \frac{\mathrm{d}\vec{r_G}}{\mathrm{d}t} =& -r\sin(\theta)\cos(\psi)\dot{\theta}\vec{i} - r\cos(\theta)\sin(\psi)\dot{\psi}\vec{i} + r\cos(\theta)\cos(\psi)\dot{\vec{i}} - r\sin(\theta)\sin(\psi)\dot{\theta}\vec{j} + \\ & r\cos(\theta)\cos(\psi)\dot{\psi}\vec{j} + r\cos(\theta)\sin(\psi)\dot{\vec{j}} + r\cos(\theta)\dot{\theta}\vec{k} + r\sin(\theta)\dot{\vec{k}}\end{aligned} \tag{2.5}$$

Since O' is fixed, $\dot{\vec{i}} = \dot{\vec{j}} = \dot{\vec{k}} = 0$, then simplifying the equation above:

$$\vec{v_G} = -(r\sin(\theta)\cos(\psi)\dot{\theta} + r\cos(\theta)\sin(\psi)\dot{\psi})\vec{i} - (r\sin(\theta)\sin(\psi)\dot{\theta} - r\cos(\theta)\cos(\psi)\dot{\psi})\vec{j} + r\cos(\theta)\dot{\theta}\vec{k} \tag{2.6}$$

The angular velocity $\vec{w}$ with respect to O can be written as:

$$\vec{w} = \vec{w_\theta} + \vec{w_\psi} = [0 \quad \dot{\theta} \quad \dot{\psi}] \tag{2.7}$$

where $\vec{w_\theta}$ and $\vec{w_\psi}$ are the angular velocities originated from θ and ψ.

The total Kinetic Energy T is then given by:

$$T = \frac{1}{2}mr^2(\theta^2 + \cos(\theta)^2\psi^2) + \frac{1}{2}(J_y\dot{\theta}^2 + J_z\dot{\psi}^2 - 2J_{yz}\dot{\theta}\dot{\psi}) \tag{2.8}$$

Appendix A contains a symbolic MATLAB® script that considers the case of J_{yz} non zero and an arbitrary position for the center of mass. In this particular case, by modeling the process in a CAD software, a previous work (NETO; BARBOSA; ANGÉLICO,) found that $J_{yz} \approx 0$. One can then simplify the equation above as:

$$T = \frac{1}{2}mr^2(\theta^2 + \cos(\theta)^2\psi^2) + \frac{1}{2}(J_y\dot{\theta}^2 + J_z\dot{\psi}^2) \tag{2.9}$$

2.1.1.2 Potential Energy

The potential energy with respect to the fixed coordinate system O' is:

$$U = mgr\sin(\theta) \tag{2.10}$$

2.1.1.3 Lagrange Equation Terms

By choosing the vector $q = [\theta \quad \psi]^T$, one gets:

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial T}{\partial \dot{\theta}} = (mr^2 + J_y)\ddot{\theta}, \tag{2.11}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial T}{\partial \dot{\psi}} = (mr^2\cos(\theta)^2 + J_z)\ddot{\psi} - 2mr^2\sin(\theta)\cos(\theta)\dot{\theta}\dot{\psi}, \tag{2.12}$$

$$\frac{\partial T}{\partial \theta} = -mr^2\sin(\theta)\cos(\theta)\dot{\psi}^2, \quad \frac{\partial T}{\partial \psi} = 0, \tag{2.13}$$

$$\frac{\partial U}{\partial \theta} = -mr^2\sin(\theta)\cos(\theta)\dot{\psi}^2, \quad \frac{\partial U}{\partial \psi} = 0 \tag{2.14}$$

2.1.2 Non-linear differential equations

Substituting the values found for the Lagrange terms in Equations (2.11) to (2.14) in the Lagrange equation 2.1, one has:

$$(mr^2 + J_p)\ddot{\theta} + mr^2\sin(\theta)\cos(\theta)\dot{\psi}^2 + mgr\cos(\theta) = \tau_\theta, \tag{2.15}$$

$$(mr^2\cos(\theta)^2 + J_z)\ddot{\psi} - 2mr^2\sin(\theta)\cos(\theta)\dot{\psi}\dot{\theta} = \tau_\psi \tag{2.16}$$

Since the dynamics of the brushless DC motors used to generate the torques in both axes have a time constant of approximately 8ms and are much faster than the process dynamics, they are approximated as constant gains. This approximation is used to avoid the need of mixing the motor with the process equations. The generalized torques can

then be written as:

$$\tau_\theta = K_{pp}V_{mp} + K_{py}V_{my} - B_p\dot{\theta}, \tag{2.17}$$

$$\tau_\psi = K_{yp}V_{mp} + K_{yy}V_{my} - B_y\dot{\psi} \tag{2.18}$$

where $K_{pp}, K_{py}, K_{yp}, K_{yy}$ are the constant gains of the motors, V_{mp}, V_{my} are the motors throttle percentages which range from 1-100%, and $B_p\dot{\theta}, B_y\dot{\psi}$ dissipating forces in the process.

Now one can derive the full approximated non-linear equations as:

$$\ddot{\theta} = \frac{K_{pp}V_{mp} + K_{py}V_{my} - B_p\dot{\theta} - mr^2\sin(\theta)\cos(\theta)\dot{\psi}^2 - mgr\cos(\theta)}{mr^2 + J_p}, \tag{2.19}$$

$$\ddot{\psi} = \frac{K_{yp}V_{mp} + K_{yy}V_{my} - B_y\dot{\psi} + 2mr^2\sin(\theta)\cos(\theta)\dot{\psi}\dot{\theta}}{mr^2\cos(\theta)^2 + J_z} \tag{2.20}$$

The values of the physical parameters of the process can be seen in Table 1.

Table 1: Helicopter Parameters

Parameter [Unit]	Value	Parameter [Unit]	Value
K_{pp} [N.m]	$9.013 \cdot 10^{-6}$	r_p [m]	0.25
K_{yy} [N.m]	$-5.668 \cdot 10^{-6}$	r_y [m]	0.20
K_{py} [N.m]	$-1.663 \cdot 10^{-7}$	l_{cm} [m]	0.038
K_{yp} [N.m]	$1.489 \cdot 10^{-7}$	m [kg]	1.317
B_p [N.m.s]	0.1	θ *angle range* [deg]	85
B_y [N.m.s]	0.1	ψ *angle range* [deg]	360
J_p [kg.m^2]	0.384	θ *encoder res.* [counts/rev]	1024
J_z [kg.m^2]	0.0432	ψ *encoder res.* [counts/rev]	1024

2.2 Linearized Model

In this section, a linearized model of the non-linear equations presented in the previous section is presented.

2.2.1 Operating Points

The operating points for the linearized model $\bar{V}_{mp}, \bar{V}_{my}$ are calculated for the desired output at the point $[\bar{\theta} \quad \bar{\psi}]^T = [0 \quad 0]^T$.

From (2.19) and (2.20) stationary condition, one has:

$$K_{pp}\bar{V}_{mp} + K_{py}\bar{V}_{my} = mgr \tag{2.21}$$

$$K_{yp}\bar{V}_{mp} + K_{yy}\bar{V}_{my} = 0 \tag{2.22}$$

Or in a matrix notation:

$$\begin{bmatrix} K_{pp} & K_{py} \\ K_{yp} & K_{yy} \end{bmatrix} \begin{bmatrix} \bar{V}_{mp} \\ \bar{V}_{my} \end{bmatrix} = \begin{bmatrix} mgr \\ 0 \end{bmatrix} \tag{2.23}$$

Solving the linear system above:

$$\begin{bmatrix} \bar{V}_{mp} \\ \bar{V}_{my} \end{bmatrix} = \frac{1}{K_{pp}K_{yy} - K_{yp}K_{py}} \begin{bmatrix} K_{yy}mgr \\ K_{yp}mgr \end{bmatrix} \tag{2.24}$$

Replacing the parameters of the previous expression by those in Table 1, the desired operating point becomes:

$$\begin{bmatrix} \bar{V}_{mp} & \bar{V}_{my} & \bar{\theta} & \bar{\dot{\theta}} & \bar{\psi} & \bar{\dot{\psi}} \end{bmatrix}^T = \begin{bmatrix} 27.60 & 8.95 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{2.25}$$

Note that any difference between the estimated and the true parameters of the motor gains will possibly lead to different values for the steady state control signals.

2.2.2 Linearization

The non-linear equations of the model (2.19) and (2.20) can be written in the form:

$$\ddot{\theta} = f_1(\theta, \dot{\theta}, \dot{\psi}, V_{mp}, V_{my}), \tag{2.26}$$

$$\ddot{\psi} = f_2(\theta, \dot{\theta}, \dot{\psi}, V_{mp}, V_{my}) \tag{2.27}$$

The linearization around the point $\bar{x} = \begin{bmatrix} \bar{V}_{mp} & \bar{V}_{my} & \bar{\theta} & \bar{\dot{\theta}} & \bar{\psi} & \bar{\dot{\psi}} \end{bmatrix}^T$ results in:

$$\begin{aligned} \ddot{\theta} = {} & \left.\frac{\partial f_1}{\partial \theta}\right|_{\bar{x}} (\theta - \bar{\theta}) + \left.\frac{\partial f_1}{\partial \dot{\theta}}\right|_{\bar{x}} (\dot{\theta} - \bar{\dot{\theta}}) + \left.\frac{\partial f_1}{\partial \dot{\psi}}\right|_{\bar{x}} (\dot{\psi} - \bar{\dot{\psi}}) + \\ & \left.\frac{\partial f_1}{\partial V_{mp}}\right|_{\bar{x}} (V_{mp} - \bar{V}_{mp}) + \left.\frac{\partial f_1}{\partial V_{my}}\right|_{\bar{x}} (V_{my} - \bar{V}_{my}), \end{aligned} \tag{2.28}$$

$$\begin{aligned} \ddot{\psi} = {} & \left.\frac{\partial f_2}{\partial \theta}\right|_{\bar{x}} (\theta - \bar{\theta}) + \left.\frac{\partial f_2}{\partial \dot{\theta}}\right|_{\bar{x}} (\dot{\theta} - \bar{\dot{\theta}}) + \left.\frac{\partial f_2}{\partial \dot{\psi}}\right|_{\bar{x}} (\dot{\psi} - \bar{\dot{\psi}}) + \\ & \left.\frac{\partial f_2}{\partial V_{mp}}\right|_{\bar{x}} (V_{mp} - \bar{V}_{mp}) + \left.\frac{\partial f_2}{\partial V_{my}}\right|_{\bar{x}} (V_{my} - \bar{V}_{my}), \end{aligned} \tag{2.29}$$

Now, considering the variables to be incremental $\hat{v} = v - \bar{v}$, one gets the following linear equations:

$$\ddot{\theta} = \frac{K_{pp}V_{mp} + K_{py}V_{my} - B_p\dot{\theta}}{mr^2 + J_p}, \tag{2.30}$$

$$\ddot{\psi} = \frac{K_{yp}V_{mp} + K_{yy}V_{my} - B_y\dot{\psi}}{mr^2 + J_z} \tag{2.31}$$

2.2.3 Continuous-time State Space

The state space model has the form:

$$\dot{\mathbf{x}}(t) = \mathbf{A}_c\mathbf{x}(t) + \mathbf{B}_c\mathbf{u}(t) \tag{2.32}$$

$$\mathbf{y}(t) = \mathbf{C}_c\mathbf{x}(t) \tag{2.33}$$

Fixing the state vector $x = \begin{bmatrix}\theta & \psi & \dot{\theta} & \dot{\psi}\end{bmatrix}^T$ and $u = \begin{bmatrix}V_{mp} & V_{my}\end{bmatrix}^T$, the state matrices become:

$$\mathbf{A}_c = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{-B_p}{J_p+mr^2} & 0 \\ 0 & 0 & 0 & \frac{-B_y}{J_z+mr^2} \end{bmatrix}$$

$$\mathbf{B}_c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{K_{pp}}{J_p+mr^2} & \frac{K_{py}}{J_p+mr^2} \\ \frac{K_{yp}}{J_z+mr^2} & \frac{K_{yy}}{J_z+mr^2} \end{bmatrix}$$

$$\mathbf{C}_c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

Using the values for the constants in Table 1, the state space matrices become:

$$\mathbf{A}_c = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -0.2591 & 0 \\ 0 & 0 & 0 & -2.217 \end{bmatrix}$$

$$\mathbf{B}_c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.04664 & -0.001645 \\ 0.02386 & -0.07356 \end{bmatrix}$$

2.2.4 Transfer Function Matrix

The model transfer matrix, given by $C_c(sI_4 - A_c)^{-1}B_c$ can be written as:

$$G_c = \begin{bmatrix} \frac{\frac{Kpp}{Bp}}{s(\tau_1 s+1)} & \frac{\frac{Kpy}{Bp}}{s(\tau_1 s+1)} \\ \\ \frac{\frac{Kyp}{By}}{s(\tau_2 s+1)} & \frac{\frac{Kyy}{By}}{s(\tau_2 s+1)} \end{bmatrix} \tag{2.34}$$

with $\tau_1 = \frac{mr^2+J_p}{Bp}$ and $\tau_2 = \frac{mr^2+J_z}{By}$. It is noted that the continuous-time transfer function is given by a first-order plus integrator system (FOI), with time constants τ_1 for the pitch angle and τ_2 for the yaw angle.
With the help of Table 1, this transfer function matrix is:

$$G_c = \begin{bmatrix} \frac{0.04664}{s(s+0.2591)} & \frac{-0.001645}{s(s+0.2591)} \\ \\ \frac{0.02386}{s(s+2.217)} & \frac{-0.07356}{s(s+2.217)} \end{bmatrix} \tag{2.35}$$

2.3 Discrete-time Model

For a sampling time Ts, the discrete-time state space system is represented by:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{A}_d\mathbf{x}(k) + \mathbf{B}_d\mathbf{u}(k) \\ \mathbf{y}(k) &= \mathbf{C}_d\mathbf{x}(k) \end{aligned} \tag{2.36}$$

The matrices A_d, B_d, C_d are calculated as:

$$\begin{aligned} A_d &= e^{A_c Ts} \\ B_d &= \int_0^{Ts} e^{A_c s} ds B_c \\ C_d &= C_c \end{aligned}$$

The sampling time Ts chosen for the project is 0.05s, equivalent to a sampling rate of 20Hz, the reason for this are the previous works such as (BARBOSA; NEVES; ANGÉLICO, 2016), which used similar sampling frequencies. In this case, 20Hz is currently the upper limit for the sampling frequency, due to the timing of the embedded QP solver, which, for the dimensions of the problem in this work, may take up to 10ms of processing time

for each sample, depending on the problem conditioning.
The state space matrices for fs=20Hz are then:

$$\begin{aligned}
\mathbf{A}_d &= \begin{bmatrix} 1 & 0 & 0.04968 & 0 \\ 0 & 1 & 0 & 0.04733 \\ 0 & 0 & 0.9871 & 0 \\ 0 & 0 & 0 & 0.8951 \end{bmatrix} \\
\mathbf{B}_d &= \begin{bmatrix} 5.804e-05 & -2.048e-06 \\ 2.876e-05 & -8.865e-05 \\ 0.002317 & -8.174e-05 \\ 0.001129 & -0.003482 \end{bmatrix} \\
\mathbf{C}_d &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}
\end{aligned}$$

2.3.1 Incremental State Space Model

The implementation of the MPC control used in this work based on the state space presented in (2.36) does not eliminate steady state errors. To solve that, the incremental formulation with an embedded integrator is proposed (WANG, 2009):

$$\begin{aligned}
\mathbf{x}(k+1) &= \mathbf{A}\mathbf{x}(k) + \mathbf{B}\Delta\mathbf{u}(k) \\
\mathbf{y}(k) &= \mathbf{C}\mathbf{x}(k)
\end{aligned} \tag{2.37}$$

Defining $\Delta x(k) = x(k) - x(k-1)$, $\Delta u(k) = u(k) - u(k-1)$ and using (2.36), one can write:

$$\Delta x(k+1) = \mathbf{A}_d\Delta x(k) + \mathbf{B}_d\Delta u(k) \tag{2.38}$$

$$y(k+1) = \mathbf{C}_d x(k+1) = \mathbf{C}_d\mathbf{A}_d\Delta x(k) + y(k) + \mathbf{C}_d\mathbf{B}_d\Delta u(k) \tag{2.39}$$

Choosing new states $\mathbf{x}(k) = \begin{bmatrix} \Delta x(k) & y(k) \end{bmatrix}^T$, the state space matrices for the augmented system (2.37) are calculated:

$$\begin{aligned}
\mathbf{A} &= \begin{bmatrix} \mathbf{A}_d & \mathbf{0_{nx,ny}} \\ \mathbf{C}_d\mathbf{A}_d & \mathbf{I_{ny}} \end{bmatrix} \\
\mathbf{B} &= \begin{bmatrix} \mathbf{B}_d \\ \mathbf{C}_d\mathbf{B}_d \end{bmatrix} \\
\mathbf{C} &= \begin{bmatrix} \mathbf{0_{nx,ny}} & \mathbf{I_{ny}} \end{bmatrix},
\end{aligned}$$

in which **ny**, **nx** represents, respectively, the number of system outputs and the number of states (dimension of $\mathbf{A}_d$).

2.4 Realignment Model

The realignment model (MACIEJOWSKI, 2002) is a non-minimal state space representation, in which the state is represented by the outputs and control signals delayed in time. This way, the states in a certain sample are all known once the output is measured, eliminating the need of tuning observer weights, such as in the Kalman Filter. This approach is very common when some system identification techniques, such as subspace methods, are used to obtain a model of the process. In that case, the user has no *apriori* information about the physical meaning of the states in the obtained model.

Given the following matrix difference equation:

$$y(k) + \sum_{i=1}^{na} A_i y(k-i) = \sum_{i=1}^{nb} B_i u(k-i) \tag{2.40}$$

where $y(k)$ is the vector of the outputs and $u(k)$ is the vector of the inputs.

One can define the following state vector:

$$x(k) = [y(k)^T \quad y(k-1)^T \; ... \; y(k-na)^T \quad u(k-1)^T \quad u(k-2)^T \; ... \; u(k-nb+1)^T]^T \tag{2.41}$$

The state space matrices then become:

$$\begin{bmatrix} y(k) \\ y(k-1) \\ y(k-2) \\ \dots \\ y(k-na+1) \\ u(k-1) \\ u(k-2) \\ \dots \\ u(k-nb+2) \\ u(k-nb+1) \end{bmatrix} = \begin{bmatrix} -A_1 & -A_2 & \dots & -A_{na-1} & -A_{na} & B_2 & \dots & B_{nb-2} & B_{nb-1} & B_{nb} \\ I_{ny} & 0_{ny} & \dots & 0_{ny} & 0_{ny} & 0_{nyxnu} & \dots & 0_{nyxnu} & 0_{nyxnu} & 0_{nyxnu} \\ 0_{ny} & I_{ny} & \dots & 0_{ny} & 0_{ny} & 0_{nyxnu} & \dots & 0_{nyxnu} & 0_{nyxnu} & 0_{nyxnu} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0_{ny} & 0_{ny} & \dots & I_{ny} & 0_{ny} & 0_{nyxnu} & \dots & 0_{nyxnu} & 0_{nyxnu} & 0_{nyxnu} \\ 0_{ny} & 0_{ny} & \dots & 0_{ny} & 0_{ny} & 0_{nyxnu} & \dots & 0_{nyxnu} & 0_{nyxnu} & 0_{nyxnu} \\ 0_{ny} & 0_{ny} & \dots & 0_{ny} & 0_{ny} & I_{nu} & \dots & 0_{nyxnu} & 0_{nyxnu} & 0_{nyxnu} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0_{ny} & 0_{ny} & \dots & 0_{ny} & 0_{ny} & 0_{nyxnu} & \dots & I_{nu} & 0_{nyxnu} & 0_{nyxnu} \\ 0_{ny} & 0_{ny} & \dots & 0_{ny} & 0_{ny} & 0_{nyxnu} & \dots & 0_{nyxnu} & I_{nu} & 0_{nyxnu} \end{bmatrix} \begin{bmatrix} y(k-1) \\ y(k-2) \\ y(k-3) \\ \dots \\ y(k-na) \\ u(k-2) \\ u(k-3) \\ \dots \\ u(k-nb+1) \\ u(k-nb) \end{bmatrix} + \begin{bmatrix} B_1 \\ 0_{nyxnu} \\ 0_{nyxnu} \\ \dots \\ I_{nu} \\ 0_{nu} \\ 0_{nu} \\ \dots \\ 0_{nu} \\ 0_{nu} \end{bmatrix} u(k-1) \tag{2.42}$$

where 0_{nyxnu} denotes a matrix of zeros with ny lines and nu columns, 0_{ny} denotes a square matrix of dimension ny and I_a denotes the identity of dimension a.

By writing (2.40) for samples k and $k-1$ and subtracting the second from the first

and redefining the state vector as:

$$x(k) = [y(k)^T \quad y(k-1)^T \; ... \; y(k-na)^T \quad \Delta u(k-1)^T \quad \Delta u(k-2)^T \; ... \; \Delta u(k-nb+1)^T]^T \tag{2.43}$$

The incremental state space matrix for the realignment model then appears in the form (2.37), with its matrices being (ODLOAK, 2014):

$$A_r = \begin{bmatrix} I_{ny}-A_1 & A_1-A_2 & A_2-A_3 & A_3-A_4 & \dots & A_{na-1}-A_{na} & -A_{na} & B_2 & \dots & B_{nb-2} & B_{nb-1} & B_{nb} \\ I_{ny} & 0_{ny} & 0_{ny} & 0_{ny} & \dots & 0_{ny} & 0_{ny} & 0_{nyxnu} & \dots & 0_{nyxnu} & 0_{nyxnu} & 0_{nyxnu} \\ 0_{ny} & I_{ny} & 0_{ny} & 0_{ny} & \dots & 0_{ny} & 0_{ny} & 0_{nyxnu} & \dots & 0_{nyxnu} & 0_{nyxnu} & 0_{nyxnu} \\ 0_{ny} & 0_{ny} & I_{ny} & 0_{ny} & \dots & 0_{ny} & 0_{ny} & 0_{nyxnu} & \dots & 0_{nyxnu} & 0_{nyxnu} & 0_{nyxnu} \\ 0_{ny} & 0_{ny} & 0_{ny} & I_{ny} & \dots & 0_{ny} & 0_{ny} & 0_{nyxnu} & \dots & 0_{nyxnu} & 0_{nyxnu} & 0_{nyxnu} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0_{ny} & 0_{ny} & 0_{ny} & 0_{ny} & \dots & I_{ny} & 0_{ny} & 0_{nyxnu} & \dots & 0_{nyxnu} & 0_{nyxnu} & 0_{nyxnu} \\ 0_{ny} & 0_{ny} & 0_{ny} & 0_{ny} & \dots & 0_{ny} & 0_{ny} & 0_{nyxnu} & \dots & 0_{nyxnu} & 0_{nyxnu} & 0_{nyxnu} \\ 0_{ny} & 0_{ny} & 0_{ny} & 0_{ny} & \dots & 0_{ny} & 0_{ny} & I_{nu} & \dots & 0_{nyxnu} & 0_{nyxnu} & 0_{nyxnu} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0_{ny} & 0_{ny} & 0_{ny} & 0_{ny} & \dots & 0_{ny} & 0_{ny} & 0_{nyxnu} & \dots & I_{nu} & 0_{nyxnu} & 0_{nyxnu} \\ 0_{ny} & 0_{ny} & 0_{ny} & 0_{ny} & \dots & 0_{ny} & 0_{ny} & 0_{nyxnu} & \dots & 0_{nyxnu} & I_{nu} & 0_{nyxnu} \end{bmatrix}$$

$$B_r = \begin{bmatrix} B_1 \\ 0_{nu} \\ 0_{nu} \\ 0_{nu} \\ ... \\ 0_{nu} \\ I_{nu} \\ 0_{nu} \\ ... \\ 0_{nu} \\ 0_{nu} \end{bmatrix}, \qquad C_r = \begin{bmatrix} I_{ny} & 0_{ny} & ... & 0_{ny} \end{bmatrix}$$

Considering an observer with gain matrix L, its dynamics is determined by the eigenvalues of $A(I-LC)$. Now, with $L = \begin{bmatrix} I_{ny} & 0_{ny} & ... & 0_{ny} \end{bmatrix}^T$ one has:

$$A(I-LC) = \begin{bmatrix} 0_{nu} & X \\ 0 & J \end{bmatrix} \tag{2.44}$$

where $X = \begin{bmatrix} -A_2 & -A_3 & \dots & B_r \end{bmatrix}$ and $J = \begin{bmatrix} 0_{ny} & 0_{ny} & \dots & 0_{ny} & 0_{ny} \\ I_{ny} & 0_{ny} & \dots & 0_{ny} & 0_{ny} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0_{ny} & 0_{ny} & \dots & I_{ny} & 0_{ny} \end{bmatrix}$.

Since the matrix defined by (2.44) is lower block-triangular, all of its eigenvalues are set to 0. Therefore, this choice of observer is always stable without assumptions on the original plant stability. Actually, this specific choice of gain L results in what is called the deadbeat observer of the process.

3 MODEL PREDICTIVE CONTROL

In Model Predictive Control, the current input $u(k)$ normally is obtained by solving an on-line QP problem. This is achieved by using the controller internal model of the process to predict the future outputs based on current measurements and inputs. This prediction is what allows the controller to check if, for a given set of inputs, the outputs will violate any constraints imposed on the problem.

3.1 Control and Prediction Horizon

At each sample, the finite horizon model predictive controller calculates the next m control moves that will optimize the criterion defined in the QP problem for the next p samples. Then, the controller implements the first control move and continue the iteration at the next sample, using the new measurements obtained.
In this case, p is called the prediction horizon, it is a measure of the samples in the future for which the controller will predict the outputs. m is called the control horizon, which can be seen as the degree of freedom of the controller to change the current control signal, in order to bring the output to the desired value. This behavior of looking at the next p samples each instant iteratively is known as receding horizon control.

3.2 Discrete Finite Horizon MPC

In this work, the controller considered is the Discrete Finite Horizon MPC. This means that it will use discrete-time models, as well as having the prediction horizon and control horizon to be finite. There is another popular formulation for stable plants called the Infinite Horizon MPC, in which the prediction horizon p is treated as infinity (JOHANSSON, 2013).

3.2.1 Problem Formulation

Given a discretized process described by an incremental form:

$$\begin{aligned} x(k+1) &= Ax(k) + B\Delta u(k) \\ y(k) &= Cx(k) \end{aligned} \tag{3.1}$$

find the control sequences $\Delta u(k|k), \Delta u(k+1|k), \ldots, \Delta u(k+m-1|k)$ that minimizes the cost function below, subject to restrictions on the inputs and outputs, implement the first result, then repeat this procedure at the next sampling instant:

$$J_k = \sum_{j=1}^{p} (y(k+j|k) - y^{sp})^T Q (y(k+j|k) - y^{sp}) + \sum_{j=0}^{m-1} \Delta u(k+j|k)^T R \Delta u(k+j|k), \tag{3.2}$$

$s.t.$

$$y_{min} < y(i) < y_{max}, \quad 0 \leqslant i \leqslant p$$

$$u_{min} < u(j) < u_{max}, \quad 0 \leqslant j \leqslant m-1$$

$$\Delta u_{min} < \Delta u(j) < \Delta u_{max}, \quad 0 \leqslant j \leqslant m-1$$

where $\mathbf{p}$ is the prediction horizon, $\mathbf{m}$ is the control horizon, $\mathbf{y^{sp}}$ is the desired output set point, $\mathbf{Q}$ is the semi-positive definite output weighting matrix and $\mathbf{R}$ is the positive-definite control signal weighting matrix.

3.2.1.1 Cost Function

By using the state update Equation (3.1), one can write the following relationships:

$$y(k+1|k) = CAx(k) + CB\Delta u(k|k) \tag{3.3}$$

$$y(k+2|k) = CA^2x(k) + CAB\Delta u(k|k) + CB\Delta u(k+1|k) \tag{3.4}$$

$$\vdots$$

$$y(k+j|k) = CA^jx(k) + CA^{j-1}B\Delta u(k|k) + CA^{j-2}B\Delta u(k+1|k) + \cdots + CB\Delta u(k+j|k). \tag{3.5}$$

Also, it is considered that the control signal is constant after the control horizon m.

$$\Delta u(k+m+j|k) = 0, \forall j \geq 0 \tag{3.6}$$

The output prediction vector can then be written as:

$$\begin{bmatrix} y(k+1|k) \\ y(k+2|k) \\ \vdots \\ y(k+m|k) \\ y(k+m+1|k) \\ \vdots \\ y(k+p|k) \end{bmatrix} = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^m \\ CA^{m+1} \\ \vdots \\ CA^p \end{bmatrix} x(k) + \begin{bmatrix} CB & 0 & \dots & 0 \\ CAB & CB & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{m-1}B & CA^{m-2}B & \dots & CB \\ CA^mB & CA^{m-1}B & \dots & CAB \\ \vdots & \vdots & \ddots & \vdots \\ CA^{p-1}B & CA^{p-2}B & \dots & CA^{p-m}B \end{bmatrix} \begin{bmatrix} \Delta u(k|k) \\ \Delta u(k+1|k) \\ \vdots \\ \Delta u(k+m-1|k) \end{bmatrix}. \tag{3.7}$$

Defining:

$$\Psi = \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^m \\ CA^{m+1} \\ \vdots \\ CA^p \end{bmatrix}, \quad \bar{\Theta} = \begin{bmatrix} CB & 0 & \dots & 0 \\ CAB & CB & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{m-1}B & CA^{m-2}B & \dots & CB \\ CA^mB & CA^{m-1}B & \dots & CAB \\ \vdots & \vdots & \ddots & \vdots \\ CA^{p-1}B & CA^{p-2}B & \dots & CA^{p-m}B \end{bmatrix} \tag{3.8}$$

One can write Equation (3.7) as:

$$\bar{y}(k) = \Psi x(k) + \bar{\Theta}\Delta\mathbf{u}(k), \tag{3.9}$$

defining the set point vector as:

$$\bar{y}^{sp} = \begin{bmatrix} y^{sp} & \dots & y^{sp} \end{bmatrix}^T \tag{3.10}$$

and weighting matrices:

$$\bar{Q} = diag\begin{bmatrix} Q & \dots & Q \end{bmatrix}^T \quad \text{and} \quad \bar{R} = diag\begin{bmatrix} R & \dots & R \end{bmatrix}^T. \tag{3.11}$$

Then, the cost function (3.2) can be rewritten as:

$$J_k = \Delta\mathbf{u}(k)^T H \Delta\mathbf{u}(k) + 2c_f^T \Delta\mathbf{u}(k) + \bar{c} \tag{3.12}$$

where,

$$H = \bar{\Theta}^T \bar{Q} \bar{\Theta} + \bar{R} \tag{3.13}$$

$$c_f^T = (\Psi x(k) - \bar{y}^{sp})^T \bar{Q} \bar{\Theta} \tag{3.14}$$

$$\bar{c} = (\Psi x(k) - \bar{y}^{sp})^T \bar{Q} (\Psi x(k) - \bar{y}^{sp}) \tag{3.15}$$

3.2.1.2 Problem Constraints

MPC offers a vast number of possible constraint choices for a given problem. These constraints can be complex and be the product of an optimization software run to, for example, define a forbidden region in the problem variables or maximize an operational profit. Though the possibilities of constraints in the problem are vast, here only simple constraints on inputs and outputs are considered, such as the ones presented in (3.2).
The constraints on the output $Y_{min} < \bar{y}(k) < Y_{max}$ can be transformed to constraints into the incremental control signal. From Equation (3.9) and considering that $\bar{\Theta}$ is non-square in the general case, one has:

$$\Delta \mathbf{u}(k) < (\bar{\Theta}^T \bar{\Theta})^{-1} \bar{\Theta}^T (Y_{max} - \Psi x(k)) \tag{3.16}$$

For the lower bound on the output, the process is the same, yielding:

$$\Delta \mathbf{u}(k) > (\bar{\Theta}^T \bar{\Theta})^{-1} \bar{\Theta}^T (Y_{min} - \Psi x(k)) \tag{3.17}$$

The constraint on the control signal $U_{min} < u(k) < U_{max}$ can be written as a constraint on the incremental control signal, considering the relationship:

$$u(k) = u(k-1) + \Delta u(k) \tag{3.18}$$

Since $u(k-1)$ is known at the instant k, one can calculate the restriction on the incremental control signal at an instant k as:

$$u_{min} - \mathbf{u}(k-1) < \Delta \mathbf{u}(k) < u_{max} - \mathbf{u}(k-1) \tag{3.19}$$

where u_{min} and u_{max} are hard constraints on the control signal.
Finally, the constraint on the incremental control signal is straightforward to implement as:

$$\Delta u_{min} < \Delta \mathbf{u}(k) < \Delta u_{max} \tag{3.20}$$

3.2.1.3 Standard QP Form

The control law of the Incremental Finite Horizon MPC is then given by the solution of the following Quadratic Programming Problem, where only the first result is implemented:

$$\begin{aligned} \underset{\Delta u_k}{\text{minimize}} \quad & \Delta u_k^T H \Delta u_k + 2c_f^T \Delta u_k \\ \text{subject to} \quad & u_{min} \leq u(k+j|k) \leq u_{max}, \quad j = 0, 1, \ldots, m-1, \\ & -\Delta u_{min} \leq \Delta u(k+j|k) \leq \Delta u_{max}, \quad j = 0, 1, \ldots, m-1, \\ & y_{min} < y(k+j|k) < y_{max}, \quad j = 1, 2, \ldots, p, \end{aligned} \tag{3.21}$$

3.2.1.4 Disturbance Model Inclusion

When conducting system identification experiments on real processes, often the methods considering structures such as Box-Jenkins, ARX and ARMAX generate two different and complimentary models for the process. One is a deterministic model and the other is the stochastic part, such that:

$$\begin{aligned} y(k) = G(q)u(k) + H(q)e(k), \\ v(k) = H(q)e(k) \end{aligned} \tag{3.22}$$

where $G(q)$ is a deterministic transfer function, $H(q)$ is the disturbance model and $e(k)$ is white-noise with variance σ^2.
Consider a state-space representation of the disturbance such that one has:

$$\begin{aligned} \mathbf{x}_d(k+1) &= \mathbf{T}\mathbf{x}_d(k) + \mathbf{Z}\mathbf{e}(k) \\ \mathbf{v}(k) &= \mathbf{F}\mathbf{x}_d \end{aligned} \tag{3.23}$$

From (3.22), one has:

$$y(k) = y_m(k) + v(k) \tag{3.24}$$

where $y_m(k)$ denotes the deterministic model output. Since the expected value of $e(k)$ is $\mathbf{E}e(k) = 0$, the one-step ahead predictor of $v(k)$ is obtained when $Z = 0$ in (3.23). Then, the prediction equation can be rewritten analogously to the prediction of $x(k)$ as:

$$\bar{y}(k) = \Psi x(k) + \bar{\Theta}\Delta\mathbf{u}(k) + \Psi_d x_d(k), \tag{3.25}$$

where the matrix Ψ_d is obtained analogously to Ψ, by rewriting equations (3.3) to (3.5) and it is given by:

$$\Psi_d = \begin{bmatrix} FT & FT^2 & \dots & FT^p \end{bmatrix}^T \tag{3.26}$$

The disturbance $v(k)$ can be obtained from (3.24) by noting that it can be calculated as:

$$v(k) = y(k) - CAx(k-1) - CB\Delta u(k-1) \tag{3.27}$$

If a suitable state-space structure for the disturbance model is chosen, such as the realignment model, once an estimate of $v(k)$ is calculated at the beginning of iteration k, the value of the state x_d is obtained. Also, note that since $\mathbf{E}e(k) = 0$, the dimension of the realignment model for the disturbances is small. This way of implementing the disturbance model in the MPC formulation has some resemblance to the feedforward control scheme.

The inequalities for constraints on the outputs need to be changed to:

$$\Delta\mathbf{u}(k) < (\bar{\Theta}^T\bar{\Theta})^{-1}\bar{\Theta}^T(Y_{max} - \Psi x(k) - \Psi_d x_d(k)) \tag{3.28}$$

$$\Delta\mathbf{u}(k) > (\bar{\Theta}^T\bar{\Theta})^{-1}\bar{\Theta}^T(Y_{min} - \Psi x(k) - \Psi_d x_d(k)) \tag{3.29}$$

Finally, the cost function matrices are updated to:

$$c_f^T = (\Psi x(k) + \Psi_d x_d(k) - \bar{y}^{sp})^T\bar{Q}\bar{\Theta} \tag{3.30}$$

$$\bar{c} = (\Psi x(k) + \Psi_d x_d(k) - \bar{y}^{sp})^T\bar{Q}(\Psi x(k) + \Psi_d x_d(k) - \bar{y}^{sp}) \tag{3.31}$$

3.3 Quadratic Problem Solver Methods

One very important step in an efficient implementation is to be sure that the QP solver chosen can handle the real-time requirement. This directly imposes an upper bound to the maximum allowed sampling rate. It has to be pointed out that the timing must be verified in the target hardware, since normally a domestic PC running compiled C++ code is much faster than the FRDM-K64F board.

In the literature, the two families of QP solvers most widely used are derived from active set methods and interior-point methods.

Let $V(x)$ be a quadratic cost function subject to constraints $Ax \leq b$.

$$\begin{aligned} V(x) &= \frac{1}{2}x^T Hx + c_f x \\ &s.t. \\ &Ax \leq b \end{aligned} \tag{3.32}$$

In active-set methods (MACIEJOWSKI, 2002) it is assumed that an initial feasible solution x is available. The algorithm then checks what constraints are active in the sense that, for a constraint of the type $g_i(x) \leq 0$, in which for different values of i one has $g_i(x) = 0$. This set of active constraints is called the active-set. The algorithm then calculates the solution for the next iteration without worrying about the constraints, such that the next solution $x_{new} = x + \Delta x$ decreases the cost function value. If the solution x_{new} is feasible, then the solver proceeds to the next iteration. If it is not, then a line search in the direction of Δx is made to find for which point the inactive constraint became active. This solution point is then the start value for the next iteration. To determine if the new solution is the global minimum, the algorithm check the Karush–Kuhn–Tucker (KKT) conditions for the problem (BOYD; VANDENBERGHE, 2009).

For a quadratic problem described by (3.32), the idea behind early interior-point methods is to consider the minimization of the following function:

$$f(x, \gamma) = \gamma V(x) - \sum_i \log\Big(b_i - a_i^T x\Big) \tag{3.33}$$

where γ is a scalar, $\sum_i \log\Big(b_i - a_i^T\Big)$ is denoted the barrier function, a_i^T is the i_{th} row of a and b_i is the i_{th} element of b. If a search algorithm starts from the feasible region, the barrier function prevents the search from leaving the feasible region, since its value near the boundary tends to infinity. However, simply minimizing (3.33) can be dangerous due to numerical errors that arise when one gets near the boundary. One way to prevent that is to consider primal-dual-methods. These methods find the solution of the primal and dual problems simultaneously.

One of the drawbacks of active-set methods is the need for a feasible initial solution, that will return an error if not found. However, since the solution of the MPC problem is repeated each sample, the algorithm may be "hot started" since the previous solution is normally a good initial guess. Another advantage of these methods is that if the code gets interrupted in the middle of its execution, the intermediate solution will be feasible.

For interior-point methods, one drawback is that its performance varies wildly, de-

pending on how well the implementation is done. One big advantage is that normally the speed of execution of interior-point algorithms varies in polynomial time with its parameters, while active-set algorithms can be exponential in these parameters in the worst case. Also, in the case of many constraints, which is usually the case in MPC problems, the active set takes a large computational load to be calculated.

3.3.1 Hildreth's Quadractic Programming Procedure

Since a real-time implementation that requires a fast sampling rate is proposed here, the choice of the QP solver is a primal-dual interior-point algorithm called the Hildreth's Quadractic Programming Procedure. One big advantage of it is that it is straightforward to port into C++ code.

Let J_k be a quadratic cost function of the type (3.32). Its dual problem can be written as (WANG, 2009)

$$\max_{\lambda \geq 0} \min_{x} \frac{1}{2} x^T H x + c_f^T x + \lambda^T (Ax - b) \tag{3.34}$$

where the vector λ denotes its x Lagrange multipliers. Since the minimization is unconstrained over x, one has:

$$x = -H^{-1}(c_f + A^T \lambda) \tag{3.35}$$

Substituting (3.35) into (3.34):

$$\max_{\lambda \geq 0} \; -\frac{1}{2} \lambda^T P \lambda - K^T \lambda - \frac{1}{2} c_f H^{-1} c_f^T \tag{3.36}$$

With $P = AH^{-1}A^T$, $K = b + AH^{-1}c_f$, this is equivalent to:

$$\min_{\lambda \geq 0} \frac{1}{2} \lambda^T P \lambda + K^T \lambda + \frac{1}{2} b^T H^{-1} b \tag{3.37}$$

This dual problem with simpler constraints is what the Hildreth's QP procedure will solve. In this algorithm, the direction vectors are chosen to be equal to the basis vectors of the Lagrange multipliers. In this, λ can be varied one component at a time. λ_i is adjusted to minimize the objective function, if that requires $\lambda < 0$, it is set to 0. One complete iteration m will calculate the vector $\lambda^m = [\lambda_1^m \dots \lambda_n^m]$ with n components from

its predecessor λ^{m-1} by the following rule:

$$\lambda_i^{m+1} = max(0, w_i^{m+1}) \tag{3.38}$$
$$w_i^{m+1} = -\frac{1}{p_{ii}}[k_i + \sum_{j=1}^{i-1} p_{ij}\lambda_j^{m+1} + \sum_{j=i+1}^{n} p_{ij}\lambda_j^m]$$

where p_{ij} is the element ij from the matrix P, and k_i is the i_{th} row from vector K. The algorithm has two termination conditions, one when the problem is close to infeasible, when the algorithm will run a maximum number of iterations m_{max} and stop. The other one is when the problem converges, in which case the vector λ^m is equal to λ^{m+1}. Once $\lambda*$ is computed, the solution of the constrained primal problem (3.35) is:

$$x^* = -H^{-1}(c_f + A^T\lambda^*) \tag{3.39}$$

One important thing to notice is that the constrained solution of x is given by the unconstrained solution $H^{-1}c_f$ added by a correcting factor related to the constraints. The algorithm also does not require any online matrix inversion, this means that in case of an infeasible problem, the algorithm will terminate at the maximum number of iterations and the output will be a near-optimal solution with constraints. This capability of returning from ill-conditioned problems or bad initial solutions is of vital importance for a real-time application algorithm. The convergence of the algorithm was demonstrated in (WISMER, 1978).

3.3.2 Other QP solver candidates

In (WANG; BOYD, 2010), the authors describe initially an interior-point primal barrier method, citing forms of improving the algorithm speed by some techniques such as: Warm starts, fixing the barrier parameter, improves the Newton step's performance by using the Schur's Complement and limiting the number of algorithm iterations. While the results seem promising, the complexity of the C++ program is far greater than the Hildreth's QP solver and rely on many Linear Algebra package libraries, that are not available in the FRDM-K64F board online compiler. Due to that, the testings with this solver were abandoned for this application.

(BEMPORAD, 2016) proposes an active set method that exploits the convex structure of the QP problem normally present in MPC applications. This is done by re-writing the QP problem as a Least Distance Problem, that is solved via a Nonnegative Least Squares solver. Since the Least Squares algorithms are usually faster than QP algorithms and

they are usually fast and straightforward to code, this method is a good candidate to be tested in future works.

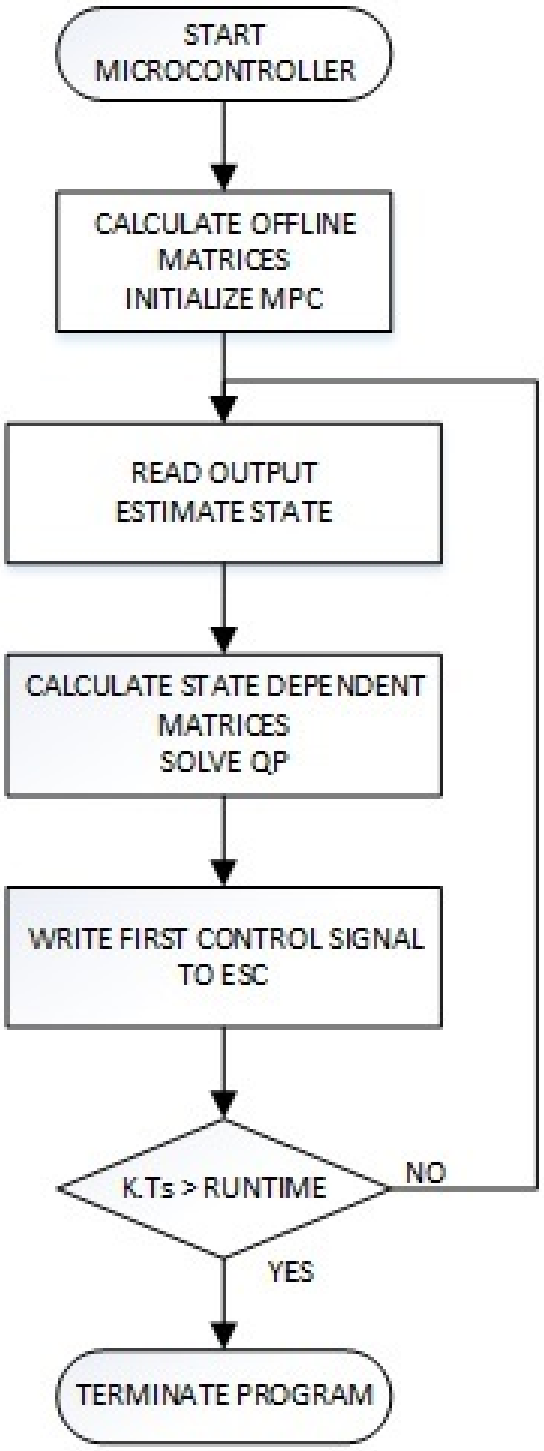
3.4 Embedded Implementation Considerations

In order to achieve efficient implementation of the controller described above, some considerations regarding numerical issues had to be made. One of them is to calculate all the matrices not related to a measure of control signal or state off-line, such as: H, $\bar{Q}\bar{\Theta}$, $(\bar{\Theta}^T\bar{\Theta})^{-1}\bar{\Theta}^T$, Ψ, P, AH^{-1}. Another one is to make sure that the dimensions and tuning parameters of the controller do not make the solver take more than the sampling time to process. The dimension of the Quadratic Problem is $nu * m$.

3.4.1 Controller Algorithm Flowchart

The controller algorithm is displayed in Figure 6. At the start, it loads from memory all precomputed matrices and start the controller object instance. Then, for each sample calculates the matrices depending on the current state and control signal, solves the QP Problem via Hildreth's Procedure, then implements the first result of dimension nu.

Figure 6: Controller Flowchart



Source: Author

4 SIMULATION RESULTS

This chapter presents the simulation results for the MPC Controller using Simulink®.

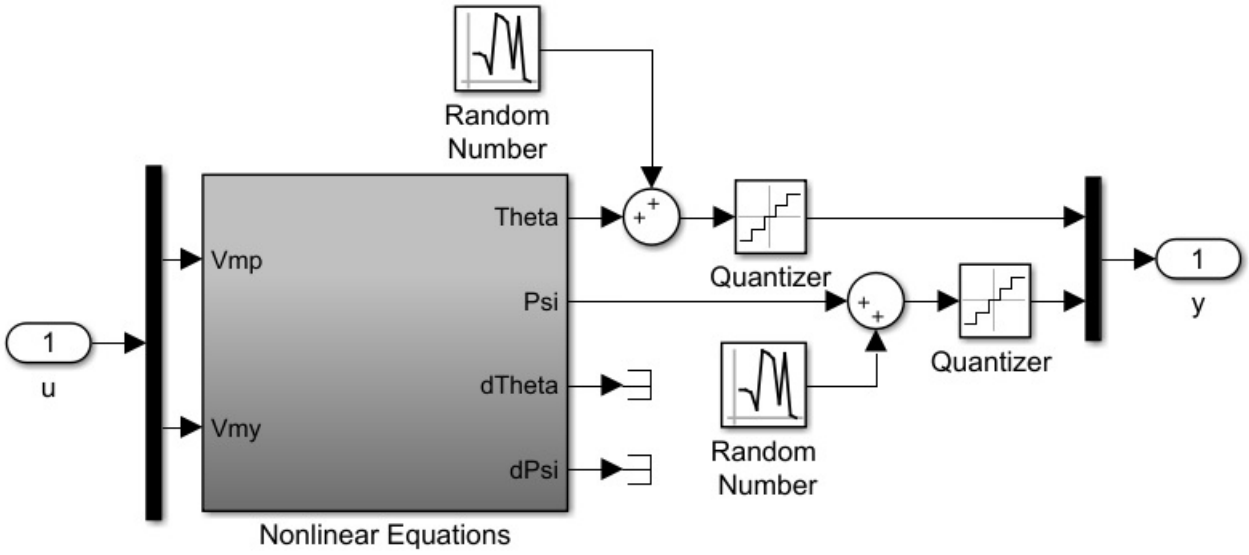
4.1 Simulink® Model Presentation

In this section the Simulink® scheme used for the simulations is shown.

4.1.1 Nonlinear Model

Using the nonlinear Equations (2.19) and (2.20), the nonlinear simulation model is described by Figure 7. The quantizer block values are both 0.003142, which is the sensor quantization of the real process. The "Random Number Block" is a gaussian noise with zero mean and variance 1e-6, which is the estimated variance of the process based on previous works (NETO; BARBOSA; ANGÉLICO,). The quantizer block makes it

Figure 7: Non-Linear Simulink Model



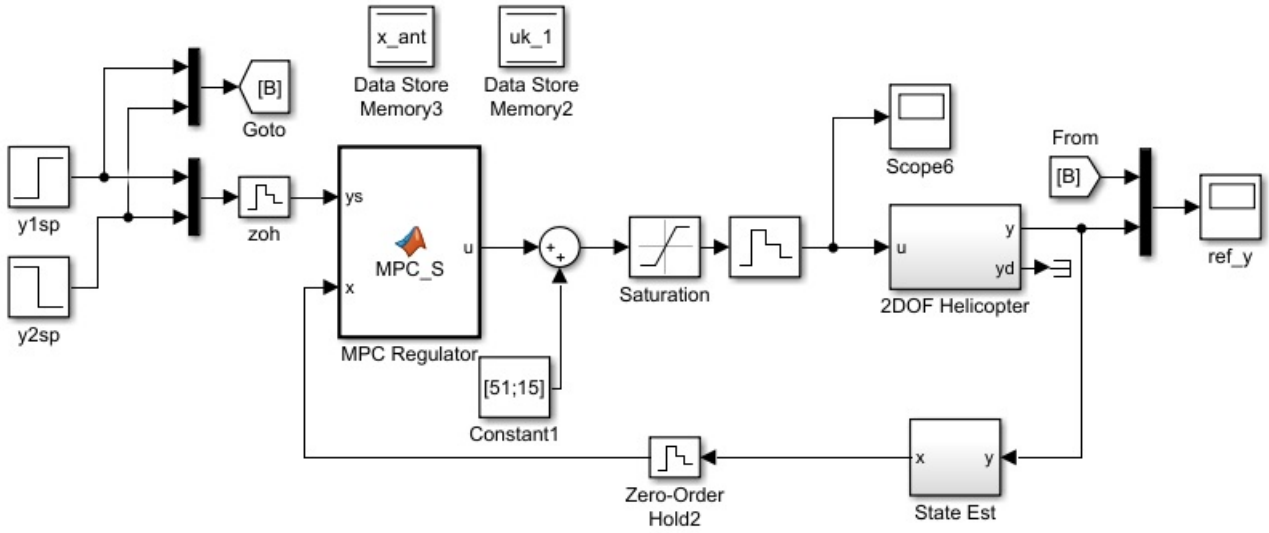
Source: Author

significantly harder to obtain a suitable derivative estimate. In the simulations, there is

no direct access to the values of $\dot{\theta}$ and $\dot{\psi}$, since there is no way of obtaining them in the real process, so they must be estimated.

The basic simulation model for the controller and the plant is represented in Figure 8, a detailed view in the "Non-linear Equations" block is shown in Appendix C. The block that estimates the derivatives is represented as "State Est" in the same figure. The "MPC Regulator" block implements the flowchart represented in Figure 6 and a detailed view is presented in Appendix B.

Figure 8: MPC Control Schematic



Source: Author

4.2 Incremental State Space Simulation

The internal controller model considered in this section for simulation is the Incremental State Space represented in Equation (2.37). The initial tuning parameters for the controller in the simulations were found heuristically and can be seen in Table 2. The limits on the control signals are naturally 0 and 100, since they are voltage percentages. The closed-loop response targets are set as a rise time of around 4 seconds and a maximum overshoot of 20%. Given that the estimation of the states under a quantizer at 50Hz, the initial choice sampling frequency, were extremely noisy and considering the desired rise time, the sampling frequency was reduced to 20Hz.

Table 2: MPC Design Parameters

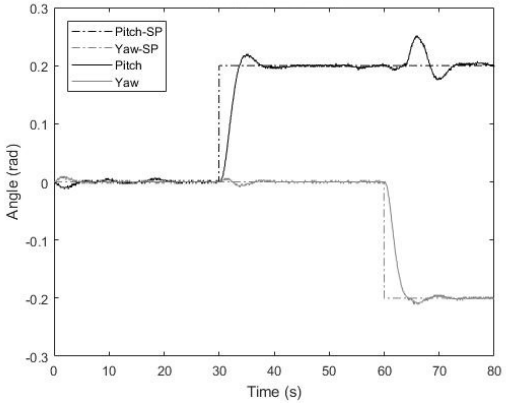
Parameter	Value
p	50
m	3
Q	I_2
R	I_2
Δu_{max}	10%

4.2.1 No Derivative Filter

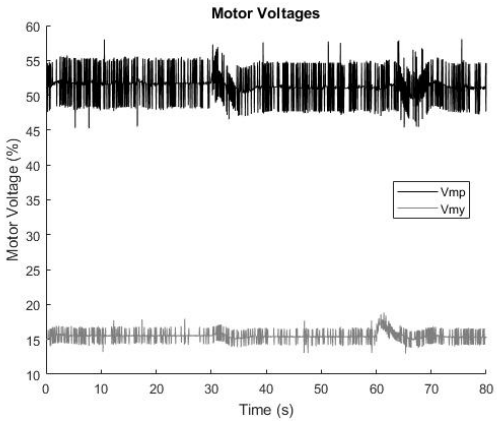
Figure 9a shows the set point tracking for a step input in each input channel. Even considering "slow" sampling times for this process, specially when compared to previous works such as (BARBOSA; NEVES; ANGÉLICO, 2016), which used a sampling time of 100Hz, if the derivative is estimated by a simple backward approximation, the control signal presents chattering. This behaviour can be verified in Figure 9b. A close-up in this figure also shows that the deviation in the control signal is bounded by the maximum allowed slew-rate of the MPC formulation, Δu_{max}. One can also notice that despite the control signal having a high variance, which is highly undesirable, since that will quickly result in the actuators no longer working properly, the set point tracking response shown in Figure 9a sufficiently met the project specifications, with a rise time of 3.7 seconds and a maximum overshoot of 10%. Also, it was verified in simulations that this difficulty to estimate the derivatives is linked to the fact that the output is quantized. This behaviour increased with the sampling frequency, and since filtering introduces a delay in the system, this is a hard constraint on how fast the system can be controlled and on the output accuracy.

Figure 9: No derivative filter - Tracking and control signal

(a) SetPoint Tracking



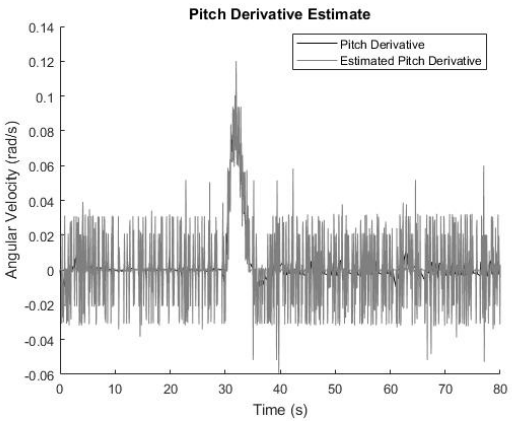
(b) Control Signal



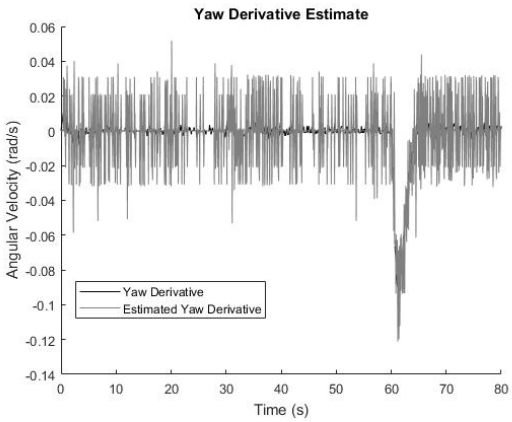
Source: Author

Figure 10: Derivative Estimates - No filter

(a) Pitch Derivative Estimate



(b) Yaw Derivative Estimate



Source: Author

Figures 10a and 10b show the estimated derivatives of the angles when no filter is used. The value shown for the true derivatives is the output of the non-linear equations.

4.2.2 Derivative Filter

Approximating the step responses obtained in the previous section as second order systems, the parameters of these systems (OGATA, 2002) are shown in Table 3. For these parameters, the worst case bandwidth of the system is 2.9rad/s. A cutoff bandwidth for the derivative filter is then designed to be approximately the worst case bandwidth. Considering the sampling time of 0.05s, this yields the filter:

$$G_f(z) = \frac{0.1331}{z - 0.8669} \tag{4.1}$$

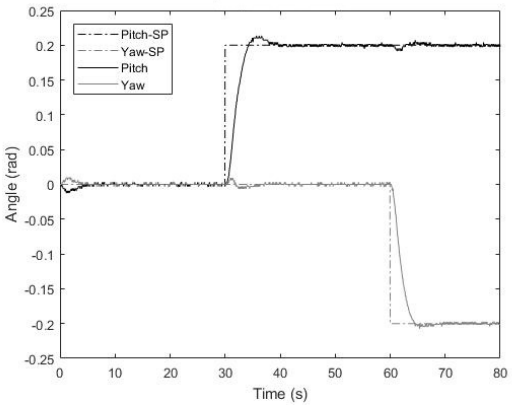
Implementing the filter described above, the new step responses and control signal plots are represented in Figures 11a and 11b. The first aspect noticed from the responses is that while the closed loop rise times stayed around 4s and the maximum overshoot close to 10%, the chattering in the control signal lowered considerably, when compared to the previous responses. The derivative estimations can be seen in Figures 12a and 12b.

Table 3: Second Order Response Approximation

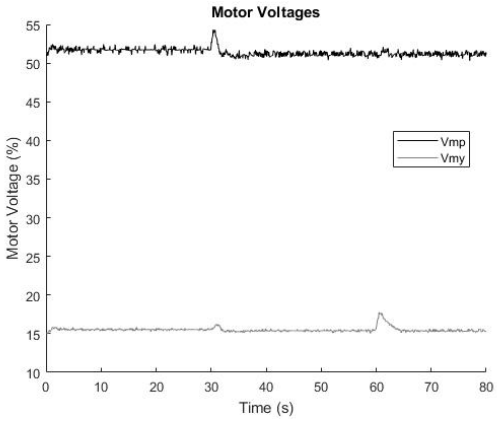
Parameter	Pitch	Yaw
w_n	0.2516 rad/s	4.0389 rad/s
w_d	1.5915 rad/s	1.4324 rad/s
ξ	0.917	0.935

Figure 11: Derivative Filter - Tracking and control signal

(a) Set Point Tracking



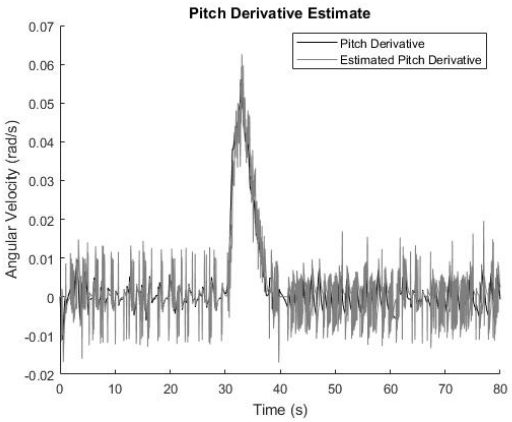
(b) Control Signal



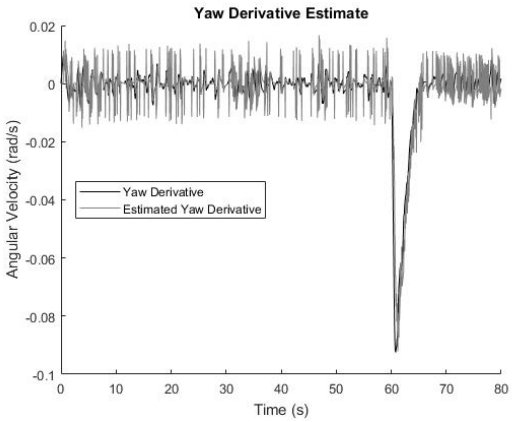
Source: Author

Figure 12: Derivative Estimates - Derivative filter

(a) Pitch Derivative Estimate



(b) Yaw Derivative Estimate



Source: Author

4.2.3 Output Filtering

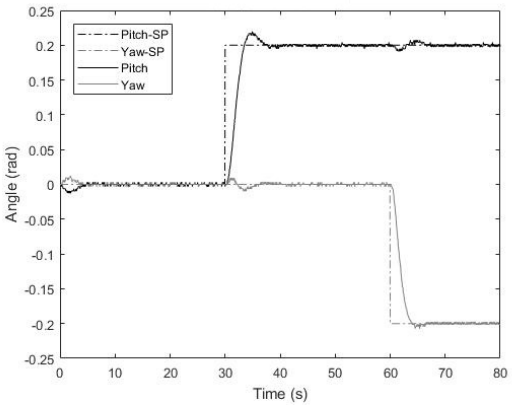
Another approach to deal with noisy data is to filter the output with a filter described by (ÅSTRÖM; HÄGGLUND, 2006):

$$Gf(s) = \frac{1}{1 + sTf + (sTf)^2/2} \tag{4.2}$$

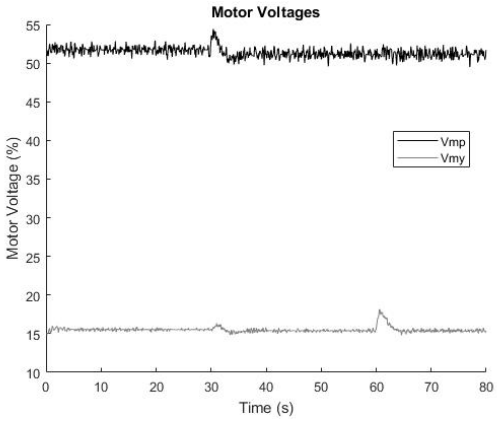
The reason for this is that the controller gain for high-frequencies decreases, making the process more robust.This is called high-frequency roll-off. One advantage of this approach over the previous one is that it can be directly implemented to realigned models, which will be used further due to its ease of implementation with identified models. On the simulations, Tf was chosen as 0.1s, following a similar logic of the derivative filter.
The step responses and control efforts for this approach can be verified in Figures 13a and 13b. Once again, the step responses have a rise time close to 4s and a maximum overshoot of 10%, while the control signal variance lowered severely. The derivative estimates using this approach can be seen in Figures 14a and 14b.

Figure 13: Output Filter - Tracking and control signal

(a) Set Point Tracking



(b) Control Signal



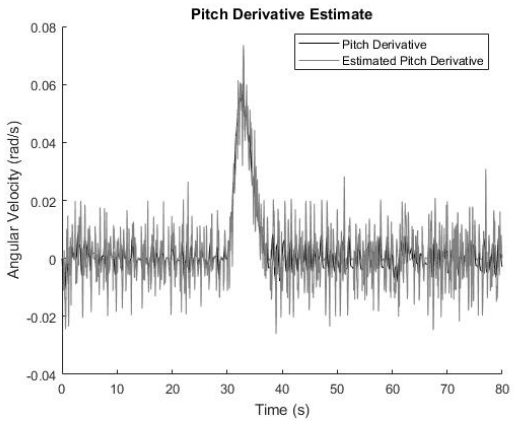
Source: Author

4.2.4 The Kalman Filter

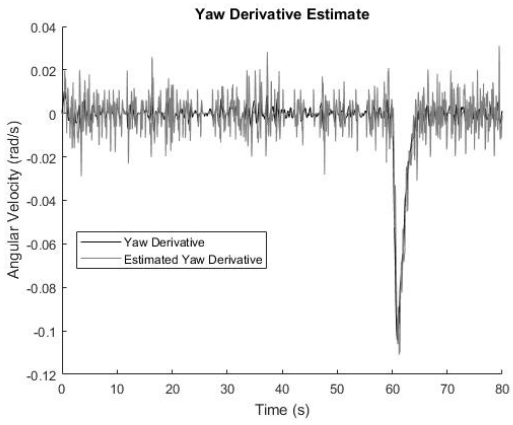
For the Kalman filter, described in more details in appendix D, once again the time requirements of the closed-loop system were met as seen in Figure 15a, but this time the derivative estimation as well as the control signal were much smoother, as verified in Figures 15b, 16a and 16b. One can notice that there is an offset in the derivative estimates, but since the state space model is incremental, that does not matter.

Figure 14: Derivative Estimates - Output filter

(a) Pitch Derivative Estimate



(b) Yaw Derivative Estimate



Source: Author

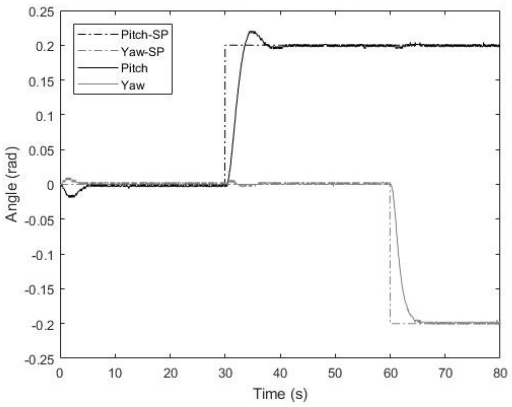
The R matrix of the Kalman filter was estimated by knowing that since the sensor quantization for both channels is $quant = 0.003142$ and the process does not change measurements for a fixed point, this yields a maximum theoretical standard deviation of the measurement noise of around $quant/2$. Knowing the standard deviation, the measurement noise variance is then $\sigma = STD^2$, this yields a noise variance $\sigma = 2.46e-6$. However, when this noise variance is simulated, the results gets far worse than verified by experimentation. This suggests that the true noise variance is lower. Considering this, a variance of $1e-6$, for both channels, was used instead. The matrix Q was estimated heuristically and the values found here correspond to a state noise variance of 0.1. Note that these two matrices are a weighting between how good are the sensors versus how much the model approximates the real system.

Despite the Kalman Filter result being the best one in simulation, there are two main problems to address. First is that it is difficult to find suitable guesses for the noise covariances in the real process, specially when the states have no physical meaning, which is often the case in System Identification. Secondly is that the true state space knowledge is very good, perfect in this simulations, which is not the case in real processes. Figure 17 shows what happens to the process outputs when the state noise variance is decreased by 10, note that now there is a bias between the Kalman angle estimates and the real outputs, resulting in a positive steady-state error.

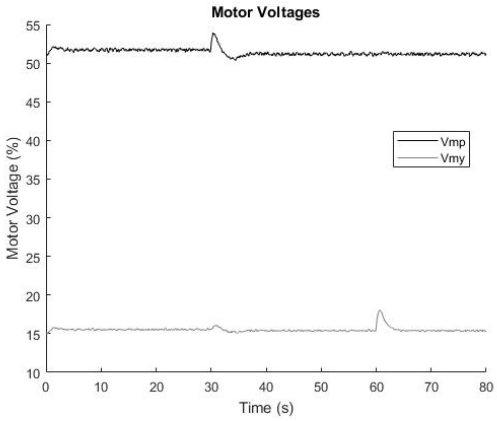
This is why the combined control strategies of output filtering and realignment model of the next section were chosen for tests on the real process.

Figure 15: Kalman Filter - Tracking and control signal

(a) Set Point Tracking



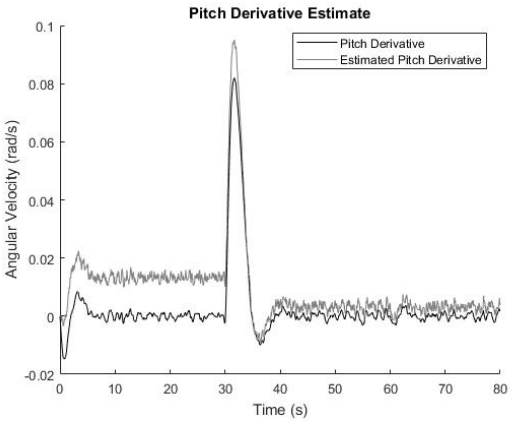
(b) Control Signal



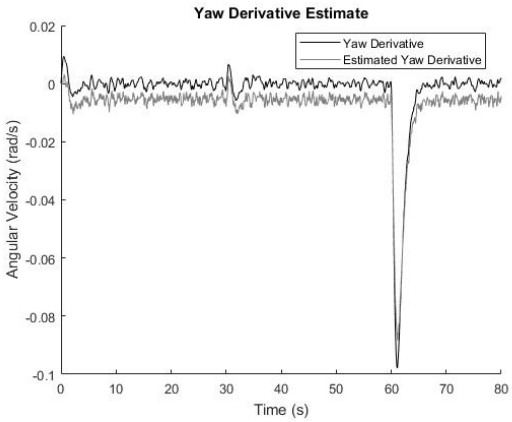
Source: Author

Figure 16: Derivative Estimates - Kalman filter

(a) Pitch Derivative Estimate

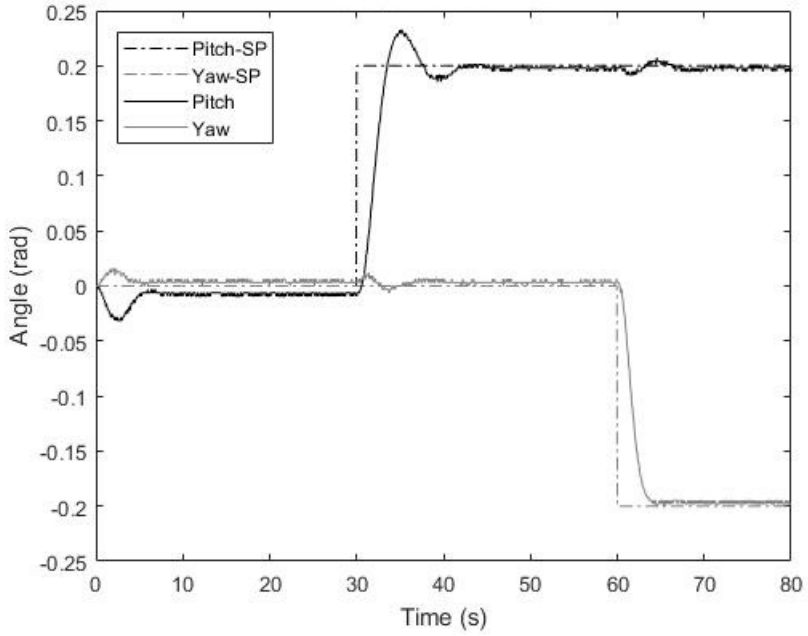


(b) Yaw Derivative Estimate



Source: Author

Figure 17: Set Point Tracking Bad Filter Tuning



Source: Author

4.2.5 Changes in the Tuning Parameters

Fixing the control horizon to 3, the prediction horizon to 50 and the control signal weight R to the identity, in the case of no derivative filter, the output weight Q of the form $K * I_2$ loses stability for $K = 1.5$. This behavior changes with output filtering, making the closed loop marginally stable for K up to 100. The control horizon for $p > 60$ makes virtually no difference in the step response performance while increasing the control signal variance. Increasing or decreasing the control horizon while fixing the parameters of Table 2 also does not change the responses significantly.

4.2.6 Optimal Weight Tuning

Another way to properly tune the controller parameters is by using optimization tools, in order to find the set of parameters that minimize a certain criterion. Three criteria are considered, the ITSE criterion, the ISE criterion and one denoted as Loop Target (LT), in which the plant output is compared to the desired closed loop output. The solver used to minimize the different criteria is a Genetic Algorithm solver due to the problem being inherently highly non-linear. Another advantage of using the Genetic Algorithm to solve the problem over another popular solvers such as MATLAB® *fmincon* is the fact that many MPC tuning parameters are, or can be made, integer. This alone allows the

optimization parameter space to be severely reduced.

4.2.6.1 ITSE Criterion

A common approach when tuning PID controllers is to use the ITSE (Integrated Time Square Error) criterion (ÅSTRÖM; HÄGGLUND, 2006) as an optimization cost function:

$$ITSE = \int_0^{T_{end}} te(t)dt \tag{4.3}$$

where T_{end} is the simulation time, t is the time in the same unit and $e(t)$ denotes the tracking error. Since a MIMO system is being considered, $e(t)$ is calculated as the 2-norm of the error:

$$\|e(t)\| = \sqrt{e_p(t)^2 + e_y(t)^2} \tag{4.4}$$

where $e_p(t)$ and $e_y(t)$ denotes the pitch and yaw tracking error.

4.2.6.2 ISE Criterion

The ISE (Integrated Square Error) cost function is given by:

$$ISE = \int_0^{T_{end}} e(t)dt \tag{4.5}$$

Since the time is not explicitly multiplied by the error, this criterion tends to result in tunings with lower gains than the previous one. The same considerations regarding the error norm are used.

4.2.6.3 Loop Target (LT)

This is a modified ISE criterion in which the error is given by:

$$e_p(t) = y_p(t) - ym_p(t) \tag{4.6}$$

$$e_y(t) = y_y(t) - ym_y(t) \tag{4.7}$$

where $y_p(t)$ is the process pitch output, $y_y(t)$ the process yaw output, $ym_p(t)$ is the desired pitch output response and $ym_y(t)$ the desired yaw output response.
Since the project specifications are that the rise time is close to 4s and a maximum overshoot of 10%, the following transfer function is used as closed loop target for both

outputs:

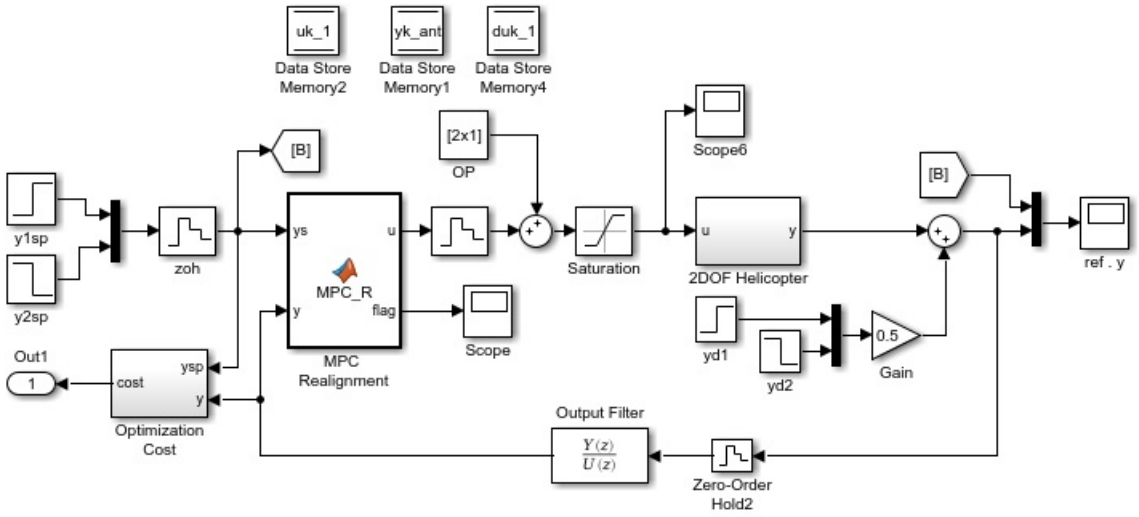
$$G_t(s) = \frac{w_n^2}{s^2 + \xi w_n s + w_n^2} \tag{4.8}$$

with $w_n = 0.7$ and $\xi = 1.2$.

4.3 Realignment Model Simulation

The simulation scheme for the realignment model presented in Section 2.4 is presented in Figure 18. As one would expect, there is no need for an observer or to estimate

Figure 18: Realignment MPC Model



Source: Author

derivatives because of the realignment model special state choices as delayed outputs and inputs. In Figure 18, if an optimal tuning scheme is selected, the block "Optimization Cost" sends to the workspace the evaluation of the selected optimization criterion. The output filter is the discrete version of the transfer function presented in Equation (4.2). In this section and further on, the MPC parameters will be selected as the results of one of three optimization problems using the criteria described above as the cost function, this is because it was found out that they yield good initial guesses in the real process, provided the controller are not too aggressive. The different tuning parameters for the Realigned MPC are presented in Table 4. The maximum control signal slew rate was also considered as a design parameter since that, in practice, it obliges the controller to be more careful with sudden variations in the control signal. The current limitations of the actuator were not considered because they are much higher than from 0-100 % in one sampling instant at 20Hz frequency. The optimization costs presented here are some among many possible

Table 4: Optimization Parameters

Parameter	Heuristic	ITSE	ISE	Loop Target
p	50	100	86	56
m	3	20	19	9
q	2.5 I_2	19.72 I_2	19.47 I_2	16.27 I_2
r	1 I_2	0.53 I_2	0.5 I_2	12.86 I_2
Δu_{max}	$[10\ 10]^T$	$[1\ 1]^T$	$[1.5\ 1.5]^T$	$[1.5\ 1.5]^T$

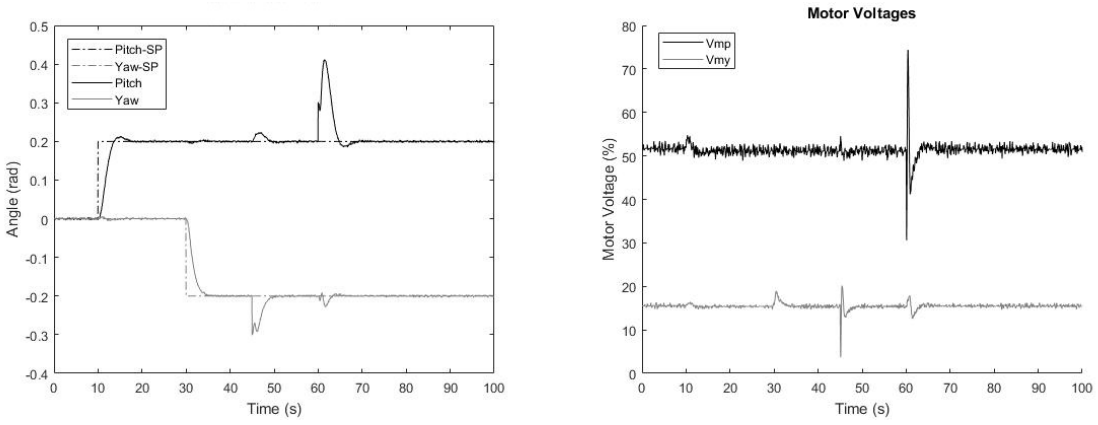
choices of optimization-based parameter tuning. Another common approach is to design a linear controller with desired closed loop characteristics and compare its output with the unconstrained MPC solution, which is linear. More information on this approach can be seen in (HARTLEY; MACIEJOWSKI, 2013), (CAIRANO; BEMPORAD, 2010).

4.3.1 Heuristic Tuning Results

The reference tracking and step disturbance rejection for the first column of parameters in Table 4 are shown in Figure 19a. While the set point tracking met the closed loop criteria with a worst case of 3.6s rising time and 11.3% maximum overshoot, the disturbance rejection was not good in the sense that it first amplified the disturbance before rejecting it.

Figure 19: Realignment Heuristic Tuning - Tracking and control signal

(a) Set Point Tracking (b) Control Signal



Source: Author

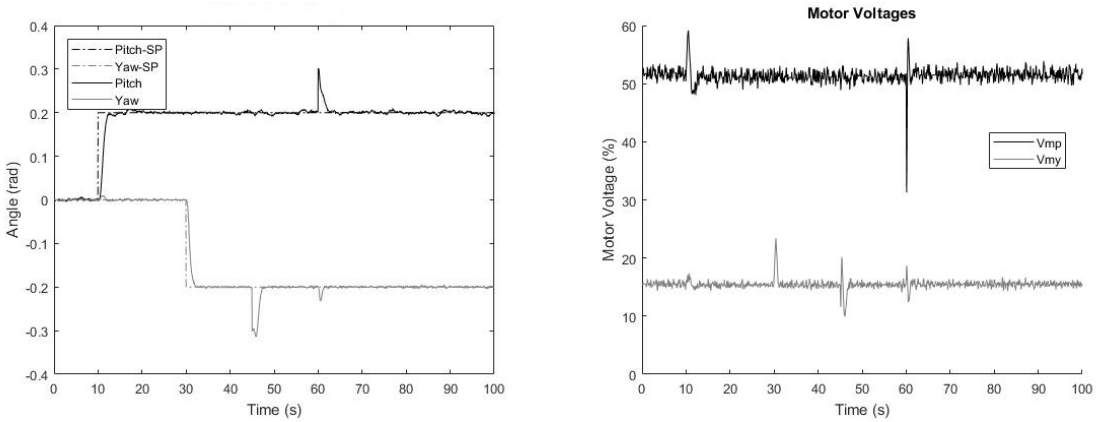
4.3.2 ITSE Criterion Tuning Results

For the ITSE parameters, the closed loop presented in Figure 20a has a worst case of 1.3s rising time and no overshoot. Despite the ITSE criterion normally generating an

overly aggressive control signal, the one displayed in Figure 20b has acceptable variance and rejects the step disturbance quickly, in around 3 seconds.

Figure 20: Realignment ITSE Tuning - Tracking and control signal

(a) Set Point Tracking (b) Control Signal



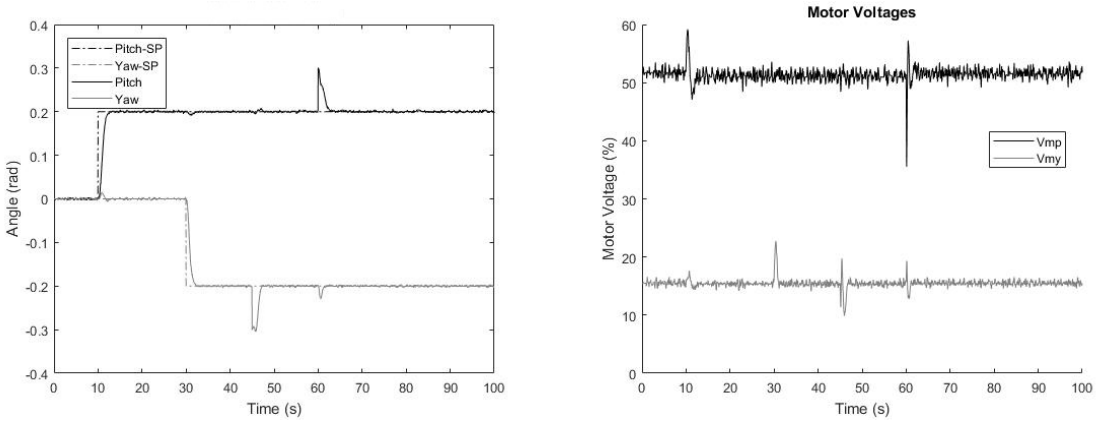
Source: Author

4.3.3 ISE Criterion Tuning Results

Since in the ISE criterion the time is not explicitly part of the cost, the tuning parameters tend to be less aggressive than with the ITSE criterion. The closed loop presented in Figure 21a has a worst case of 1.4s rising time and no overshoot. The control signal displayed in Figure 21b has an acceptable variance and rejects the step disturbance quickly, in around 3 seconds as well. It's important to notice that the ISE results were

Figure 21: Realignment ISE Tuning - Tracking and control signal

(a) Set Point Tracking (b) Control Signal



Source: Author

better than the ITSE ones, which turned out to exhibit oscillations in the pitch variable,

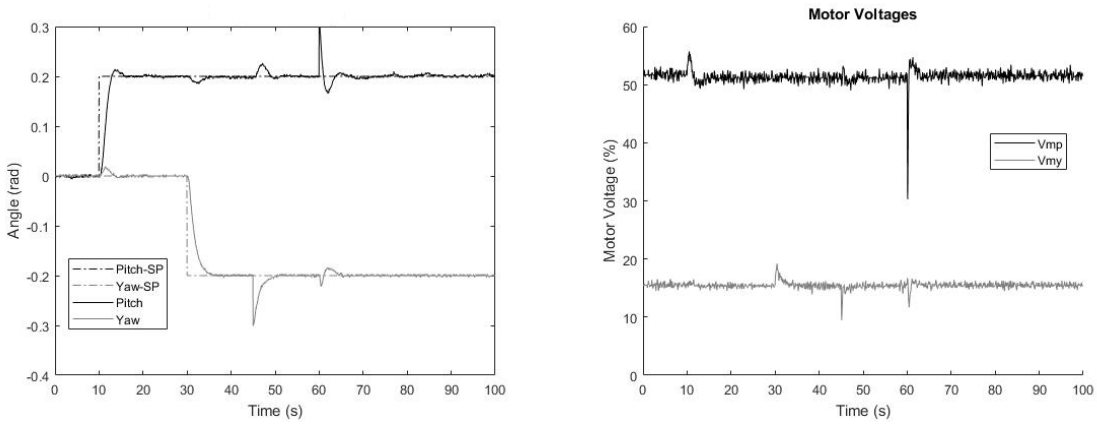
as well as having a control signal with higher variance.

4.3.4 LT Criterion Tuning Results

The Loop Target (LT) criterion set point responses and load disturbance rejection can be seen in Figure 22a. Here it is possible to notice the worst case rise time of 3.6 seconds and maximum overshoot of 10%. The control signal shown in 22b is close to the values found in the previous sections. The graph for the step response of both angles compared to the second order system defined in Equation (4.8) can be seen in Figures 23a and 23b.

Figure 22: Realignment LT Tuning - Tracking and control signal

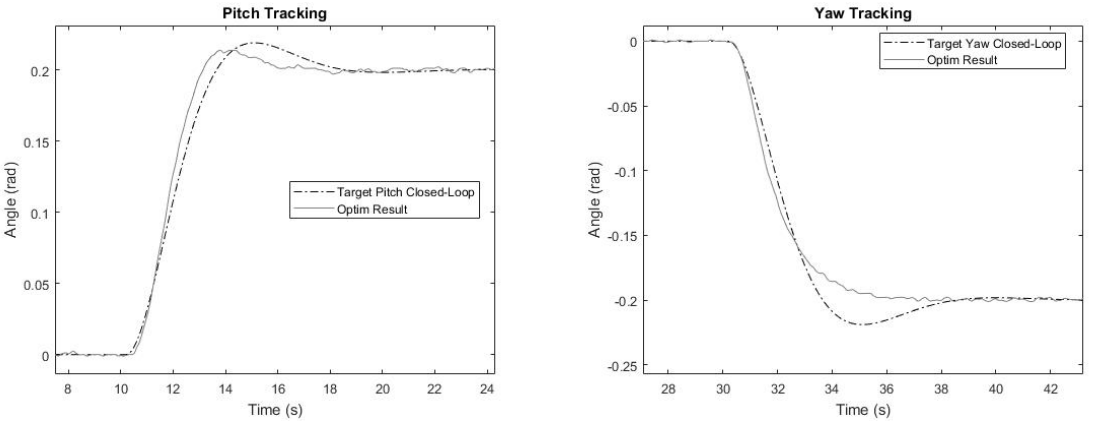
(a) Set Point Tracking (b) Control Signal



Source: Author

Figure 23: Step Response Comparison - Second Order System

(a) Set Point Tracking (b) Control Signal



Source: Author

5 SYSTEM IDENTIFICATION

In order to improve the results obtained with tests on the real process, system identification is needed to improve the model used in the MPC, resulting in better closed-loop performance.

Topics on closed-loop identification, identification for control and the considered system identification methods are presented here. It is assumed a basic familiarity of the reader with topics such as multi-step ahead predictors and model structures. More information on these topics can be obtained in (LJUNG, 1999).

5.1 The Identification Procedure

In order to successfully obtain a model for a given process, one usually has to solve the problem in four steps:

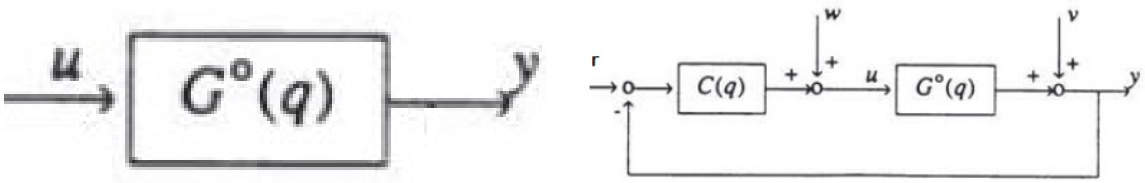
- Experiment Design
- Model structure selection
- Choice of identification method
- Model Validation

5.1.1 Experiment Design

The optimal experiment design is closely tied to the chosen identification method. The sampling time of the experiment must be low enough, so that all important dynamics of the process are captured and sufficiently high in order to avoid numerical errors. Another decision that is closely related to the sampling time choice is the experiment duration, which basically limits the amount of samples in a dataset.
It is important to perform, when possible, simple preliminary experiments such as tests for

Figure 24: Open-Loop vs Closed-Loop Identification Structures

(a) Open-Loop Structure (b) Closed-Loop Structure



Source: (ZHU,1994)

time-invariance, linearity, presence of integrator, rough DC gain, dead time and dominant time constant estimates. This was partially done in Section 5.5.

The structure of the experiment is mainly divided into closed and open loop identification. Figures 24a and 24b shows the difference between open-loop and closed-loop structures for system identification. In the first one, the identification input is superposed directly to the operating point u_{ss} whereas in the second one, the identification input is introduced in the system as the signal w. This specific structure for the closed-loop identification was chosen because it was the one used in the main references of this work. There are other structures that might be more commonly used. However, since the MPC controller is linear when there is no active restrictions, this structure did not caused any problems during the identification procedures. Regarding the input choice, there are mainly three factors that are important:(LJUNG, 1999):

- The asymptotic properties of the estimate (bias and variance) depend only on the input spectrum.
- The input must be bounded $\underline{u} \geq u(k) \leq \bar{u}$.
- Periodic signals may present certain advantages.

5.1.2 Model Structure Selection

The choice of a suitable structure that describes the process and the noise characteristics is paramount to achieve a good result. Here is where physical knowledge of the process dictates if a Black-Box or a Grey-Box approach is more suitable.

The orders of the model can also be estimated by previous knowledge of the process, or by trial and error tied with an estimator, such as the Akaike Information Criterion, which is basically a trade-off between the value of the fit index described below and the model

complexity.

$$FIT(\%) = 100(1 - \frac{\left\|Y - \hat{Y}\right\|^2}{\left\|Y - \mathbf{E}(Y)\right\|^2}) \tag{5.1}$$

where Y denotes the measured output, $\hat{Y}$ the model estimated output and $\mathbf{E}(Y)$ the expected value of the measured output.

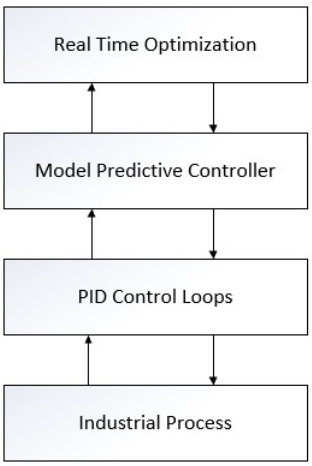
5.1.3 Choice of the Identification Method

There are several aspects that influences the choice of different methods for system identification. The main ones are:

- Familiarity with the method or available software
- Is the process running in closed-loop or in open-loop?
- What is the model going to be used for?

It is important to emphasize that although most processes operate in closed-loop in a more rigorous control theory definition, normally in some industries such as petrochemical, where MPC is mostly applied, open-loop means the regulatory control loop is in automatic and MPC in manual. This can be seen in Figure 25. The naming convention used here is the standard control theory definition.

Figure 25: Common Industrial Control Loop Hierarchy



Source: Author

5.1.4 Model Validation

When performing the system identification experiment, it is usual to split the dataset in two. The first one will be used in the identification method, in order to estimate a model. The last part will be used to validate the model obtained in the first step. It is important to notice that once the experiment is performed, the remaining three steps, structure selection, identification method and model validation are inherently iterative. It is possible as well to combine multiple experiments in order to find a better model estimate.

5.2 Closed-Loop Identification

The identification of a good model for the 2-DOF Helicopter is slightly more difficult than a standard system identification problem. First, the process is multivariable, requiring the identification methods to be extended to the multivariable case. Second, the process is integrative on both outputs, thus unstable, meaning that, in most cases, standard open-loop identification techniques cannot be used directly.
This happens because many methods such as instrumental variables, standard subspace approaches or spectral analysis methods assume that the input of the process u is uncorrelated with the output disturbance v, as shown in Figure 24a.
The family of methods that will be chosen is the prediction error framework. These methods work fairly well with closed-loop data as long as the experiments are informative and the chosen structure flexible enough (FORSELL, 1999).
There are three main approaches in closed-loop identification:

- Direct Approach
- Indirect Approach
- Joint Input-Output Approach

They differ from each other in which assumptions need to be made in order to find a good estimate for the open loop model.

5.2.1 Direct Approach

The direct approach is basically the application of the standard open-loop prediction error method directly to input-output data, ignoring any eventual feedback present in the

data. The following model is considered (FORSELL, 1999):

$$y(t) = G(q, \theta)u(t) + H(q, \theta)e(t) \tag{5.2}$$

One of the drawbacks of the direct approach is that if the true noise model is not equal to the estimated one, the predictions will display bias. This can be sometimes ignored if the noise models are close enough of the real disturbances.

5.2.2 Indirect Approach

The idea behind the indirect approach is to estimate the closed-loop transfer function between the input w and the output y, as seen in Figure 24b and then use this result to estimate the open-loop transfer function. This of course can only be done if the regulator $C(q)$ is linear and well-known and it can be carried out by applying the following procedure:

- Identify the closed-loop transfer function $G^c(q)$ from w to y.
- Compute the open loop transfer function by solving the following equation:

$$\hat{G}^c(q) = (I + \hat{G}^o(q)C(q))^{-1}\hat{G}^o(q) \tag{5.3}$$

where $\hat{G}^c(q)$ is the estimate of the closed-loop transfer function and $\hat{G}^o(q)$ is the estimate of the open-loop transfer function.
This approach makes the assumption that the regulator $C(q)$ is time-invariant and that can pose a problem if there is not a well-defined synchronization between measurement and control (JOHANSSON, 2012). In the indirect approach, as long as the real process model $G^o(q)$ is contained in the subset of considered model structures for $\hat{G}^o(q)$, there will be no bias. The variance of the first two cases are the same (LJUNG, 1999).

5.2.3 Joint Input-Output Approach

The Joint Input-Output Approach consists of considering the control signal u and output y as outputs driven by the extra input w. The idea here is to recover knowledge of the system and the regulator. Note that in this case, the regulator does not need to be known *a priori*. Consider a process described by the following model in closed-loop:

$$y(t) = S^o(q)G^o(q)w(t) + S^o(q)v(t) \tag{5.4}$$

$$u(t) = S_i^o(q)G^o(q)w(t) - S_i^o(q)C(q)v(t) \tag{5.5}$$

where

$$S^o(q) = (I + G^o(q)C(q))^{-1} \tag{5.6}$$

$$S_i^o(q) = (I + C(q)G^o(q))^{-1} \tag{5.7}$$

From the equations above, the transfer functions for the process can be estimated for the inputs $[w(t)v(t)]^T$ and outputs $[y(t)u(y)]^T$.

Since in the 2-DOF Helicopter case the loop is closed with the MPC controller, as long as the process does not violate the imposed restrictions, the MPC controller is a linear regulator, so this method is a good candidate for our problem.

5.2.4 Identification for Control

When the purpose of the identified model is to be used in a model-based control scheme, one needs to notice that normally, in system identification, the objective is to find a model as similar as possible to the process. This does not directly implies that this model is ideal to be used for control. The first reason is that normally good models used to formulate control strategies are historically simple. The second is that the model used for control does not need to be accurate in all the frequency domain, but mostly around the cross-over frequency (FORSELL, 1999). If the controller has integrative behavior, for example, the low-frequency modeling error does not need to be low, since the integrator will eliminate steady-state error anyway. So, it is important to emphasize the importance of pre-filtering using the right window.

5.3 Two-step Projection Method

The Two-step Projection Method consists of decoupling the closed-loop identification problem into two open-loop problems. It consists of the following steps (FORSSELL; LJUNG, 2000):

1. Estimate the parameters s_k in the noncausal FIR model

$$u(t) = S(q)w(t) + e(t) = \sum_{k=-M1}^{M2} s_k w(t-k)e(t) \tag{5.8}$$

where $M1$ and $M2$ are chosen large enough, so that any correlation between $w(t)$ and $u(t)$ can be ignored, then simulate $\hat{u} = \hat{S}(q)w(t)$.

2. Identify the open-loop system using the model structure:

$$y(t) = \hat{G}^o(q)\hat{u}(t) + H_*(q)e(t) \tag{5.9}$$

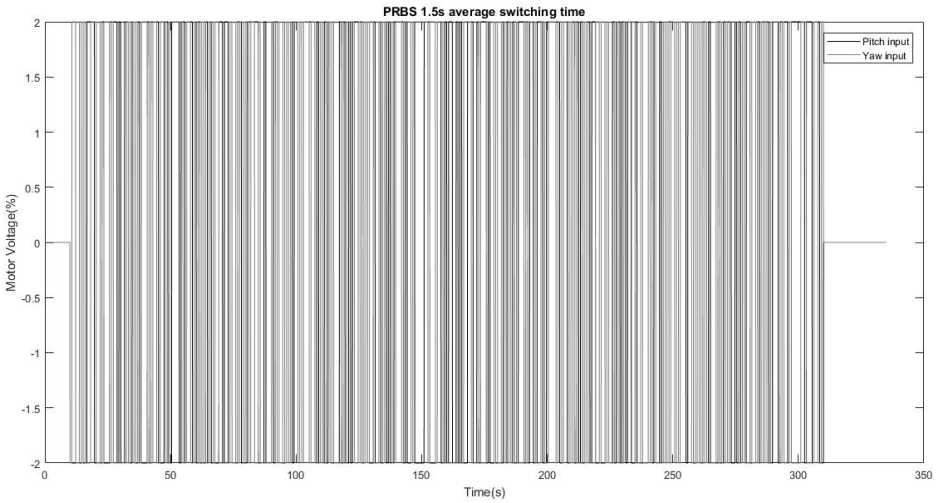
where H_* is chosen adequately, in order to get an arbitrary frequency weighting as in the open-loop case.

The first step of the algorithm can be seen as a way to asymptotically achieve the least squares estimate $\hat{u}$ and the residual $\tilde{u} = u - \hat{u}$ is orthogonal to $\hat{u}$. One advantage of the method over other approaches is that it can be applied to arbitrary feedback.

5.3.1 Input Signal

Due to ease of implementation and available guidelines in the literature, the selected input type for process excitation is a PRBS (Pseudo-Random Binary Sequence) signal. In (ZHU, 1998) the author suggests that the PRBS average switching time be around a third of the longest response time of the process. In the 2-DOF Helicopter this results in an average switching time of roughly 1.5 s. The amplitude of the PRBS was chosen to yield the maximum SNR possible, without deviating the process too much from the linear region. The resulting 2-channel PRBS signal is shown in Figure 26

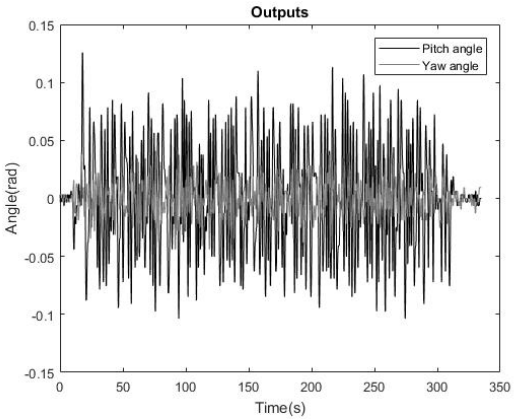
Figure 26: Identification Input



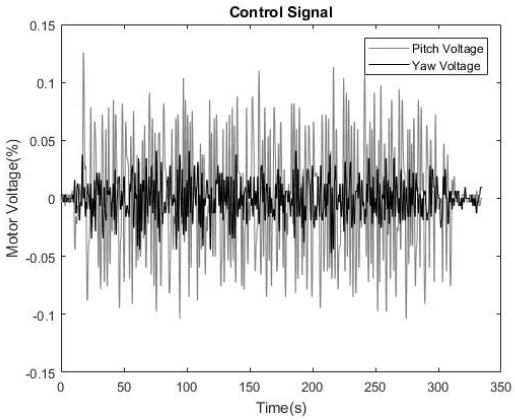
Source: Author

Figure 27: Output Angles and Control Signal

(a) Process Output



(b) Control Signal



Source: Author

5.3.2 Identification Results

The input signal from the previous section was applied to the process in the presence of an MPC controller, using the theoretical model described previously in Chapter 2. The process response along with the control signal can be seen in Figures 27a and 27b.

For the first step, the FIT indexes for the Sensitivity functions were 82.75% and 59%, for the pitch and yaw angles, respectively, with simulation focus, as seen in Figure 28. The FIR method was able to enforce the orthogonality of $\hat{u}$ and $\tilde{u}$, as expected. Its correlation coefficients are shown in Table 5. Note that initially no assumption about the presence of an integrator in the process was made.

Table 5: Correlation between $\hat{u}$ and $\tilde{u}$

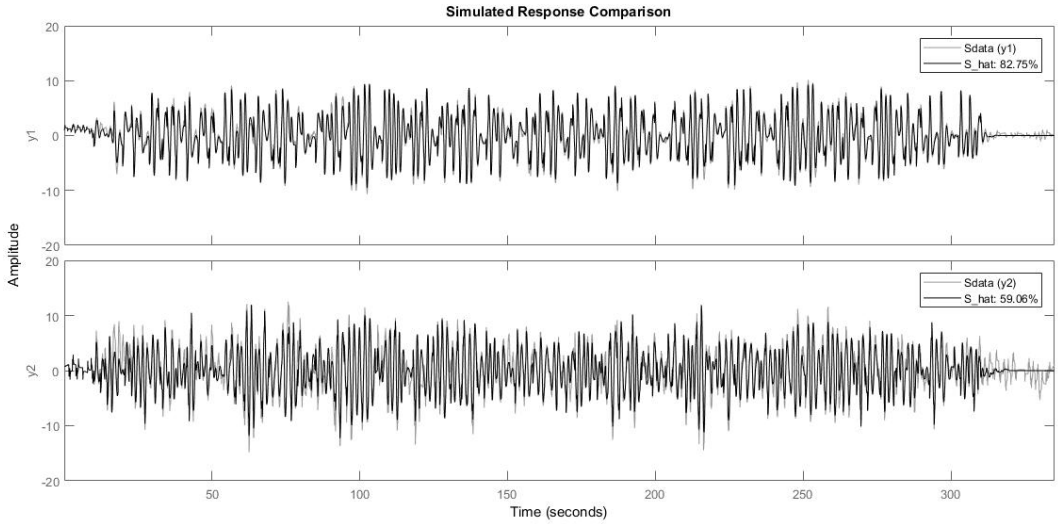
	$\hat{u}$	$\tilde{u}$
$\hat{u}$	1	0.001
$\tilde{u}$	0.001	1

In the second step of the method, concatenated MISO structures were considered for the identification, so there is no effect from one output to the other. A Box-Jenkins structure, as seen below, was chosen for its flexibility:

$$y(t) = \frac{B(q^{-1})}{F(q^{-1})}u(t - nk) + \frac{C(q^{-1})}{D(q^{-1})}e(t) \tag{5.10}$$

For both outputs, the order of the $B(q^{-1})$ and $F(q^{-1})$ polynomials is two for both inputs, the transport delay nk is one for both inputs and the order of $C(q^{-1})$ and $D(q^{-1})$ is two. These orders were selected to get a good compromise between complexity and simplicity.

Figure 28: FIR estimate $\hat{S}$



Source: Author

It is fully possible that the order is too low to describe the true process, however that was not verified to be true in this work. The FIT indexes for this final step were 96.74% and 91.3%, for the pitch and yaw angles, respectively, with prediction focus. The identified transfer function matrix in this case is:

$$G_{2step}(z) = \begin{bmatrix} \frac{-0.0035652(1-1.066z^{-1})z^{-1}}{(1-z^{-1})(1-0.7756z^{-1})} & \frac{0.00029424(1-0.9803z^{-1})z^{-1}}{(1-0.9781z^{-1})(1-0.8165z^{-1})} \\ \frac{-7.4662e-05(1-1.067z^{-1})z^{-1}}{(1-0.9846z^{-1})(1-0.8487z^{-1})} & \frac{0.0011024(1-1.026z^{-1})z^{-1}}{(1-0.984z^{-1})(1-0.8018z^{-1})} \end{bmatrix} \tag{5.11}$$

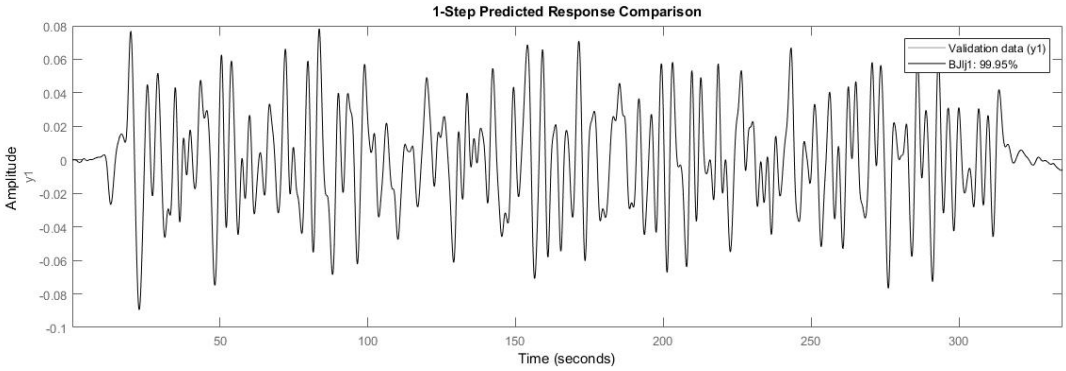
The infinite step-ahead prediction of the system when auto-validated with the identification data does not produce good results. However, as stated before, the model only needs to be good in the frequency range that is relevant for control.

For the pitch axis, the estimated cut-off frequency based on the controller step response for this case is 1.22 rad/s. For the yaw axis, the same procedure was done and the cut-off frequency was found to be around 0.57 rad/s.

By using a band-pass filter on the validation data with a conservative window estimate from 0.08 rad/s to 2 rad/s, the one-step ahead predictor FIT improves to 99.95% and 99.6%, for the pitch and yaw angles, respectively. This can be seen in Figures 29 and 30.

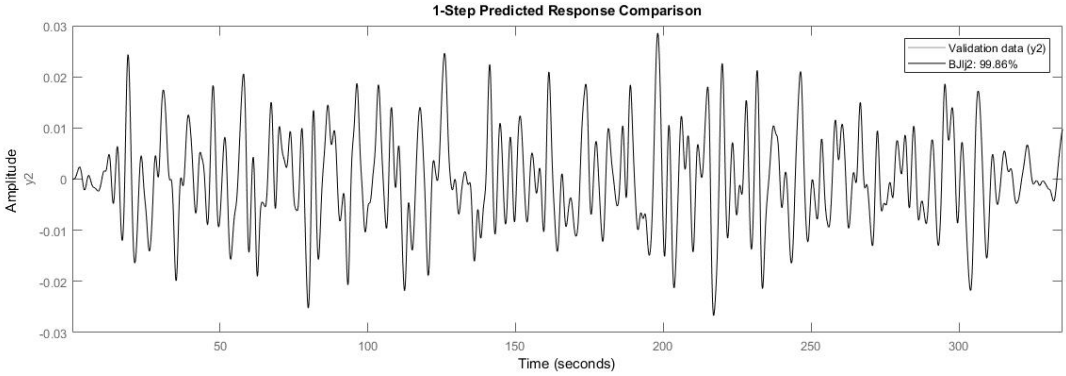
The 20-step ahead predictions are shown in Figures 31 and 32.

Figure 29: One-step ahead predictions - Pitch Axis



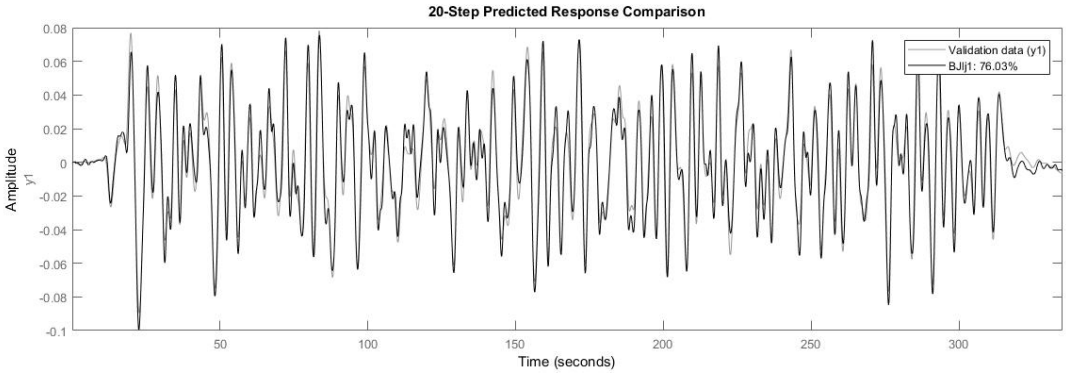
Source: Author

Figure 30: One-step ahead predictions - Yaw Axis



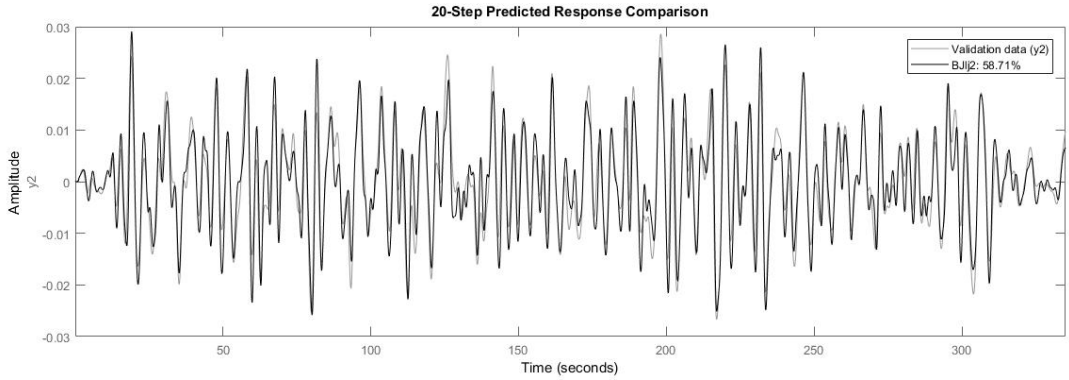
Source: Author

Figure 31: Twenty-step ahead predictions - Pitch Axis



Source: Author

Figure 32: Twenty-step ahead predictions - Yaw Axis



Source: Author

5.4 Asymptotic Method

The asymptotic method (ZHU, 1998) is an Identification for MPC approach that is based on the asymptotic theory presented in (LJUNG, 1999).
The algorithm consists of four steps:

1. Identification Test
   It was used here the same rules as in the previous method as described in (ZHU, 1998).

2. Estimate a high order ARX model, such that it is a good representation of the process. The method author suggests at least 20.

$$\hat{A}^n(z^{-1})y(t) = \hat{B}^n(z^{-1})u(t) + e(t) \tag{5.12}$$

   denoting $\hat{G}^n(z^{-1})$ as the high order deterministic model and $\hat{H}^n(z^{-1})$ as the high order disturbance model.

3. Perform a frequency weighted model reduction
   This is done by minimizing the asymptotic negative log-likelihood function for a fixed order (WAHLBERG, 1989):

$$V = \sum_{i=1}^{p}\sum_{j=1}^{m}\int_{w_1}^{w_2}\left|\left|\hat{G}_{ij}^n(w) - \hat{G}_{ij}(w)\right|^2 \Phi_{u_j}\Phi_{v_j}^{-1}\right| dw \tag{5.13}$$

   where Φ_{u_j} is the j-th input spectrum matrix and $\Phi_{v_j}^{-1}$ is the inverse disturbance input spectrum matrix. w_1 and w_2 are the band frequencies that are important for the MPC application.

4. Utilize the asymptotic criterion (ASYC) for order selection (ZHU, 1994)
   The ASYC is given by:

$$ASYC = \sum_{i=1}^{p}\sum_{j=1}^{m}\int_{w_1}^{w_2}\left|\left|\hat{G}_{ij}^{n}(w) - \hat{G}_{ij}(w)\right|^2 - \frac{n}{N}\Phi_{u_j}^{-1}\Phi_{v_j}\right|dw \tag{5.14}$$

   The basic idea of the criterion is to equalize the bias and variance of the model parameters in the range that is important for control. The selected order is then the one that yields lower ASYC. However, in this work, since the models are limited on purpose to a low order, the criterion was not used strictly.

5. Model validation using error bound matrix
   It is shown in (ZHU, 1998) that there is an upper bound for the model reduction that can be written as:

$$\left|\hat{G}_{ij}^{o}(e^{iw}) - \hat{G}_{ij}^{n}(e^{iw})\right| \leq 3\sqrt{\frac{n}{N}\Phi_{u_j}^{-1}\Phi_{v_j}}\ \textit{with prob. } 99.9\% \tag{5.15}$$

   The author suggests an engineering solution, in which the error bounds are compared to the model gain in lower to middle frequencies. Then it is suggested to grade the model as A (error bound at maximum 30% of the model gain for every considered frequency), B (same as A, but 60%), C (same as A, but 90%) and D (poor, or no model exists, same as A, but greater than 90%).

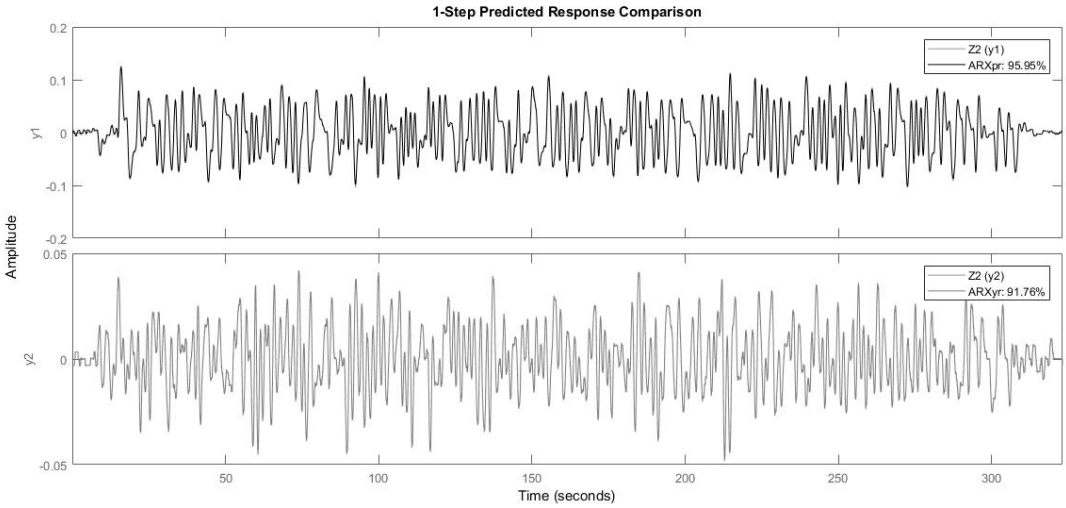
It is important to notice here that there is no guarantee that a high order ARX model can capture the structure of an arbitrary system. From the practical tests in the Two-DOF Helicopter it was found that, in many times, this is not the case.

5.4.1 Asymptotic Method Results

Here the results of the one-step ahead and the twenty-step ahead predictors without using any filtering in the data is shown in Figures 33 and 34. However, since the frequency band important for control was estimated, a band-pass filter is used, as in the last method, and the new high order ARX model result for the filtered signal is shown in Figures 35 and 36.

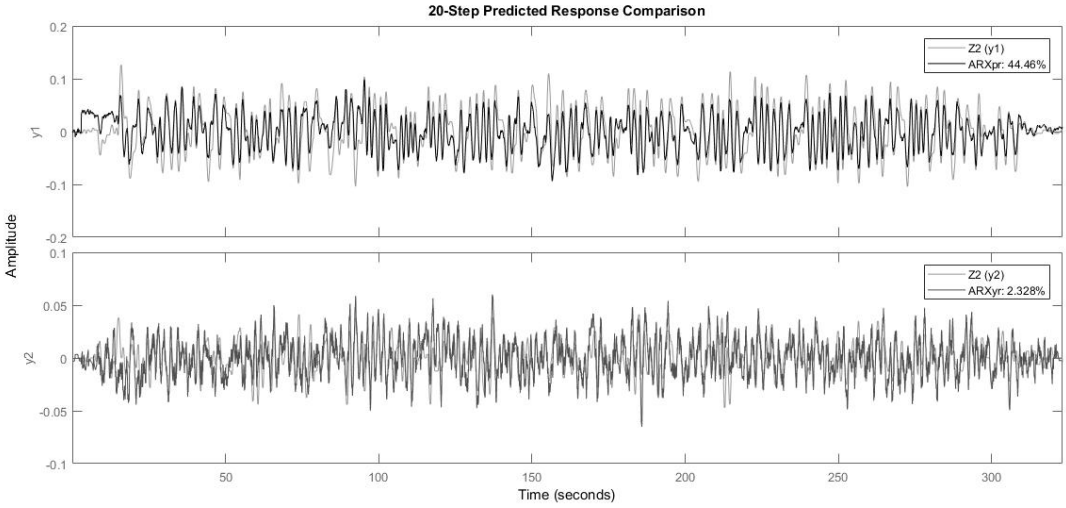
The maximum order for the reduced system was set to two, in order to preserve simplicity. The error bounds plotted with the model gains for the pitch output can be seen in Figure 37a. Notice that in the frequency band emphasized in the identification, the error bounds are lower in both cases. For the yaw output, the same graphs are shown

Figure 33: One-step ahead prediction ARX - No filter



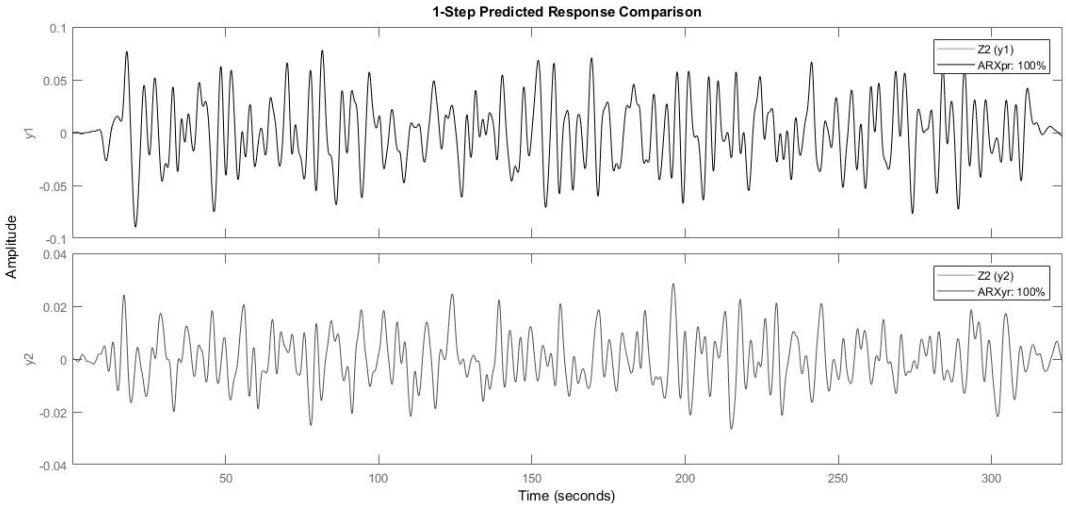
Source: Author

Figure 34: Twenty-step ahead prediction ARX - No filter



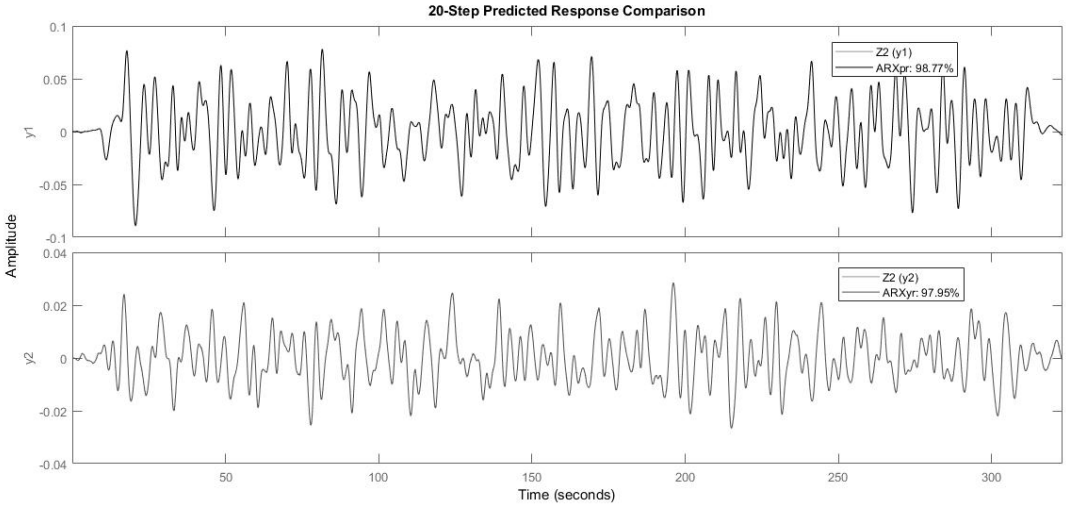
Source: Author

Figure 35: One-step ahead prediction ARX - Filtered



Source: Author

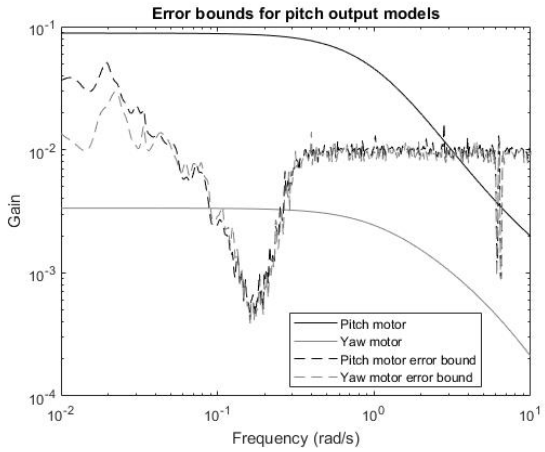
Figure 36: Twenty-step ahead prediction ARX - Filtered



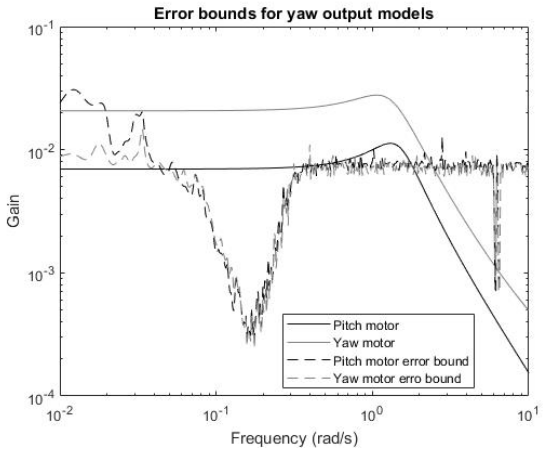
Source: Author

Figure 37: Error bounds for the high order model

(a) Pitch output

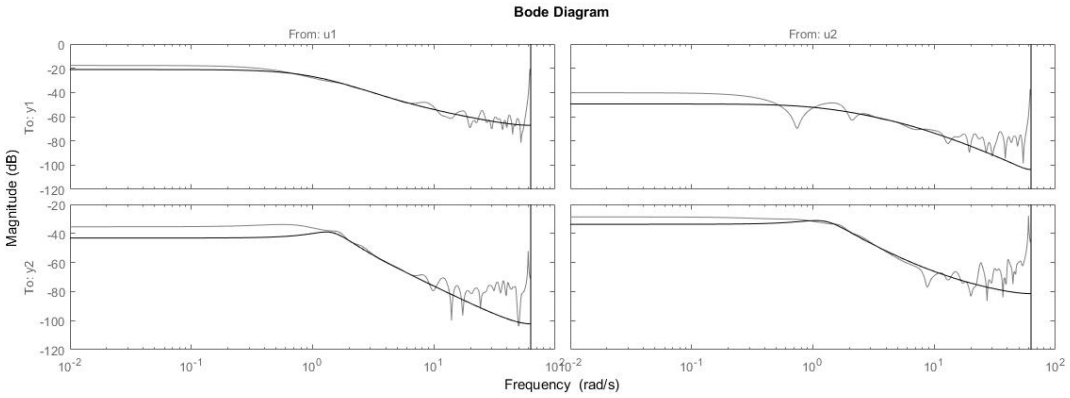


(b) Yaw output



Source: Author

Figure 38: Bode graphs of the high and reduced order model



Source: Author

in Figure 37b. In 38 it is shown the bode plots of the high order ARX models and the reduced order Box-Jenkins models are similar due to the maximum likelihood principle.

The four grades for the identified models are shown in Table 6. These results indicate

Table 6: Model Grades for the Asymptotic Method

	Pitch Angle	Yaw Angle
Pitch Motor	B	D
Yaw Motor	D	B

that either the couplings are small in the real process or that the data is not informative enough in order to capture them adequately. In fact, it was verified experimentally that the effect of the yaw motor in the pitch angle is negligible, despite the existence of the

effect of the pitch motor in the yaw angle. The resulting lower order model is:

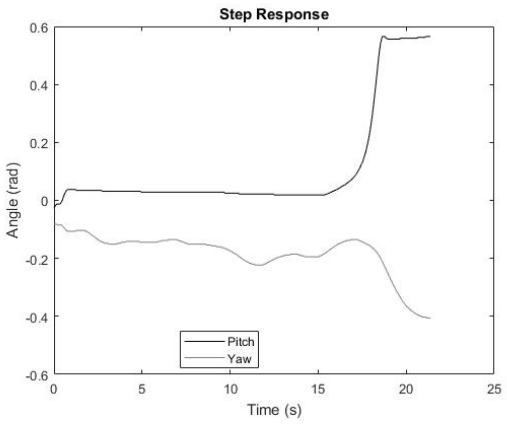
$$G_{asym}(z) = \begin{bmatrix} \frac{-7.2334e{-}04(1{-}1.306z^{-1})z^{-1}}{(1{-}0.9578z^{-1})(1{-}0.9408z^{-1})} & \frac{-1.6108e{-}05(1{+}2.336z^{-1})z^{-1}}{(1{-}0.9476z^{-1})(1{-}0.6942z^{-1})} \\ \frac{3.3361e{-}05(1{-}0.0972z^{-1})z^{-1}}{(1{-}1.948z^{-1}{+}0.9532z^{-2})} & \frac{1.2227e{-}04(1{-}1.699z^{-1})z^{-1}}{(1{-}1.944z^{-1}{+}0.9479z^{-2})} \end{bmatrix} \tag{5.16}$$

5.5 Open Loop Step Test

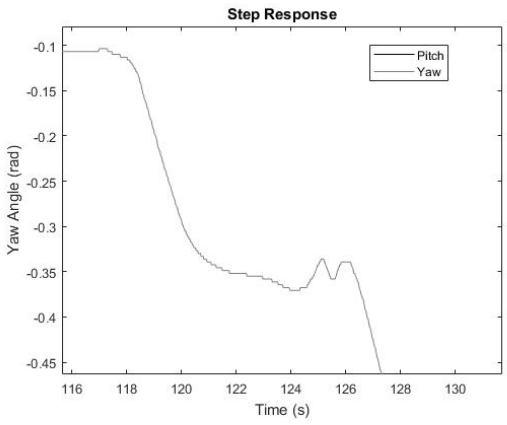
Since the step responses obtained in the previous sections could be improved in the author's opinion, a simple open loop step test was performed for each motor input. The operating point found for the process is approximately 56% for the Pitch Motor and 10% for the Yaw Motor. One consideration that has to be made is that the operating points depend on the ESC calibrations. In the Pitch case, the weight of the process forces it to go down, so the motor has to compensate for gravity in the operating point. Since there is no load disturbance in the Yaw Axis, the operating point for the motor is much lower. The step response for the pitch motor for an amplitude of 1% is shown in Figure 39a. The yaw response could not be estimated in the same experiment, it was repeated and its result is shown in Figure 39b. The test was interrupted before the process reached the saturation, in order to get a first order plus integrator plus deadtime approximation (FOIDT).

Figure 39: Open Loop Step Responses 1% Yaw Motor Voltage

(a) Pitch Step Response Estimate



(b) Yaw Step Response Estimate



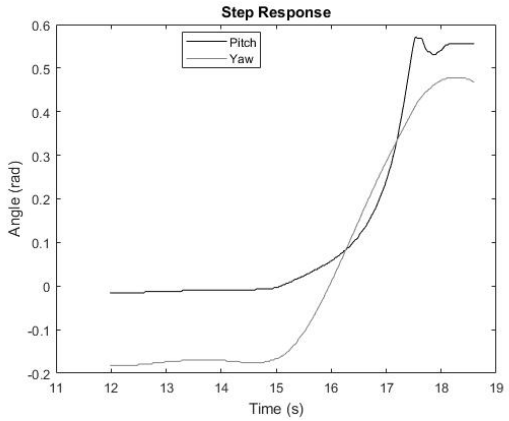
Source: Author

This structure was chosen because it is the same as the linear model. The step response with an amplitude of 1% for the Yaw motor can be seen in Figure 40. Both responses are not exactly represented by a FOIDT structure, so they were approximated, extrapolating the slew rate of the process around 0.1 rad to infinity. Despite what is

shown in Figure 40, since any minor disturbance leads the plant to saturation, the results for the influence of the yaw motor on the pitch were disconsidered, as As one could infer from the previous closed loop results that it's effect is very small.
The estimated continuous time transfer function matrix is:

$$G_c = \begin{bmatrix} \frac{0.261}{s(s+0.6303)} & 0 \\ \frac{0.018}{s(s+0.9554)} & \frac{-0.0311}{s(s+0.9554)} \end{bmatrix} \tag{5.17}$$

Figure 40: Open Loop Step Responses 1% Pitch Motor Voltage



Source: Author

5.6 Grey-Box Subspace Estimation

The idea of this section is that since the discrete state space presented in Equation (2.36) structure is known, it could be used to estimate a grey-box state model for the process. Assuming that the structure of the state matrices can represent the real process, the matrix A_d has only four unknown parameters, while B_d has eight and C_d has zero. The operational data for step responses along 4 minutes were detrended and plugged in MATLAB routine IDSS, using a model parametrized as described above, having 10 deterministic estimation parameters and 4 stochastic model estimation parameters. The IDSS routine is based on Subspace methods as presented in (OVERSCHEE; MOOR, 1996). The basic concept is that instead of identifying an input-output relation, the method attempts the route of input-state-output. Normally, its not very well conditioned for closed-loop data, however there are methodologies that yield unbiased estimates when applied to closed-loop data, such as the SSARX method (JANSSON, 2003).The resulting

deterministic model was obtained for 20Hz sampling frequency:

$$\mathbf{A}_d = \begin{bmatrix} 1 & 0 & 2.004 & 0 \\ 0 & 1 & 0 & 1.754 \\ 0 & 0 & 0.9173 & 0 \\ 0 & 0 & 0 & 0.9566 \end{bmatrix}$$

$$\mathbf{B}_d = \begin{bmatrix} -0.0002831 & -5.368e-05 \\ 2.183e-05 & 6.705e-05 \\ 2.311e-05 & 3.323e-06 \\ 2.749e-06 & -1.711e-05 \end{bmatrix}$$

$$\mathbf{C}_d = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

The results of the usage of these models for control is presented in chapter 6.

6 REAL PROCESS RESULTS

Here are presented initial results of the MPC implementation in the real process.

6.1 C++ Code Export Preliminaries

The initial results were obtained by exporting the controller as C++ code using MATLAB® Embedded Coder. The embedded coder options that worked was: "Device Vendor" as "ARM Compatible", "Device Type" as "ARM 7" and "Optimize for execution". The exported code was fairly simple, since ARM7 was released from 1993 to 2001, and was found to work with the Freescale FRDM-K64F. The list of tuning parameters that were used in the following sections are shown in Table 7. All of these parameters were found heuristically. Note that the initial state is the process resting position with $\theta = 0.78$ rad and $\psi = 0$rad. Since the system is non-linear, it is up to the controller to successfully stabilize the process around the operating point. The measured steady state inputs were $V_{mp} = 60.1\%$ and $V_{my} = 18.15\%$. This alone suggests, at least, a wrong estimation of the torque gains of the mathematical model.

6.1.1 C++ code development

After initial results with the exported code from MATLAB, standalone C++ code was developed in order to ease the tuning process, since the embedded coder needed to be exported then compiled every time the tuning parameters changed. The main challenges of this development was to handle memory allocation in the microcontroller, along with the non-existent support of linear algebra from the mbed platform. This required the development of every matrix operation needed in the MPC.

Table 7: MPC Tuning Parameters - Real Process

Parameter	Inc.SS	Realig	StepTest	GreyBox	Two-step	Asym
p	50	50	50	50	50	50
m	5	5	1	3/1	5/3	5/3
q	1.2 I_2	1.2 I_2	5 I_2	20 I_2	10 I_2	5 I_2
r	I_2	I_2	0.5 I_2	I_2	0.5 I_2	0.5 I_2
Δu_{max}	$[5\ 5]^T$	$[5\ 5]^T$	$[5\ 5]^T$	$[5\ 5]^T$	$[5\ 5]^T$	$[5\ 5]^T$

6.2 Incremental State Space

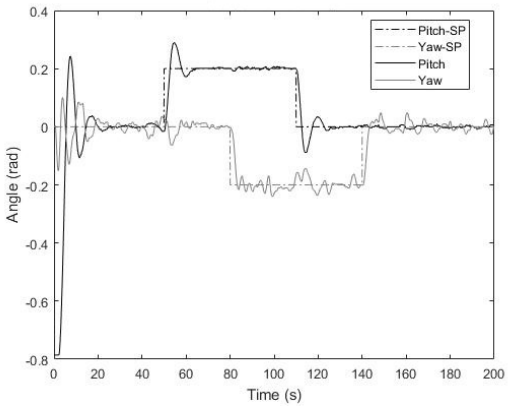
The model used by the controller in this section is the linear model used in the simulations of the previous chapter, with a frequency of 20Hz. The derivatives were estimated using a first order filter, as described in the simulations. The step responses along with the control signals can be seen in Figures 41a and 41b. The obtained control signal oscillations can be seen as the controller trying to compensate the model mismatches, or as the need for a better filter to estimate the derivatives. Figures 42a and 41b show the step responses and control signals when the following prefilter $F(s)$ is used:

$$F(s) = \frac{1}{s+1} \tag{6.1}$$

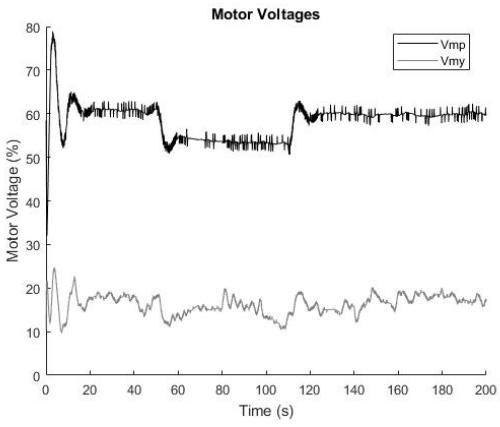
From Figure 41a, the rise time of the process was around 2.75s and 5s for the pitch and

Figure 41: Step Response Incremental State Space with Derivative Filter

(a) Set Point Tracking



(b) Control Signal

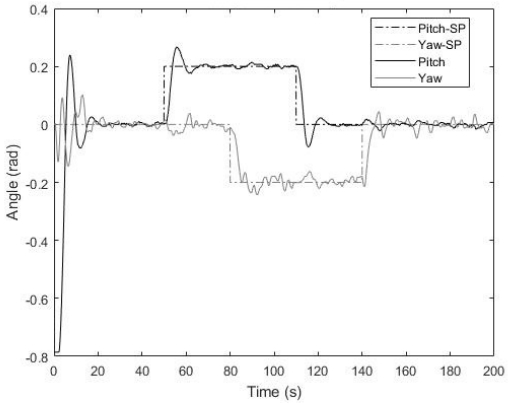


Source: Author

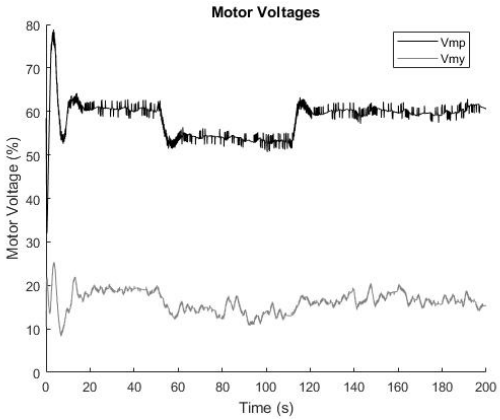
yaw angles, with an overshoot of 45% and 20%. For the prefiltered response shown in Figure 42a, the pitch angle rising time was 3.85s with an overshoot of 33% and a rising time of 5.5s with an overshoot of 22% for the yaw angle. The motivation behind the prefilter usage is that from Figure 41a, it can be inferred that there is a strong coupling

Figure 42: Step Response Incremental State Space with Prefilter and Derivative Filter

(a) Set Point Tracking



(b) Control Signal



Source: Author

between the Pitch Voltage and the Yaw Angle, the prefilter reduces this behavior and slightly decreases the maximum overshoot while increases the rise time. Even with the prefilter, both step responses failed to follow the closed loop specifications, 4s for the rise time and 20% maximum overshoot. The oscillation in the yaw axis is due to model mismatch between the controller and the real process.

6.3 Realignment Model

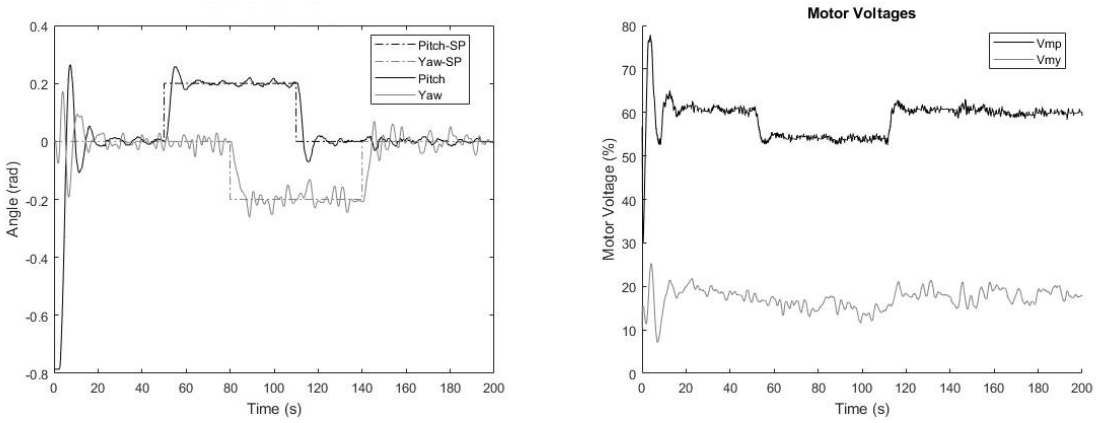
For the Incremental State Space MPC, the step responses using the prefilter along with the control signals can be seen in Figures 43a and 43b. The difference between the results from the previous section can be explained by the difference in the observers used, since this model has an embedded deadbeat observer and the incremental state space model uses a backward filtered derivative in conjunction with the sensor measurements to estimate the state. While the pitch angle oscillates more than in the previous case due to the output filter, the chattering in the control signal due to the derivative estimates almost disappeared when compared to the previous section. The rise time and overshoot for the pitch and yaw angles, in this case, were 3.5s, 28% and 5.2s, 23%.

6.4 First Order Deadtime plus Integrator Model

The MPC controller results with the prefilter, using this very rudimentary model, is shown in Figures 44a and 44b. Just the fact that the closed loop was stable using this model suggests that the MPC controller formulation used is indeed very robust. To the

Figure 43: Step Response Realigned Incremental State Space with Prefilter and Output Filter

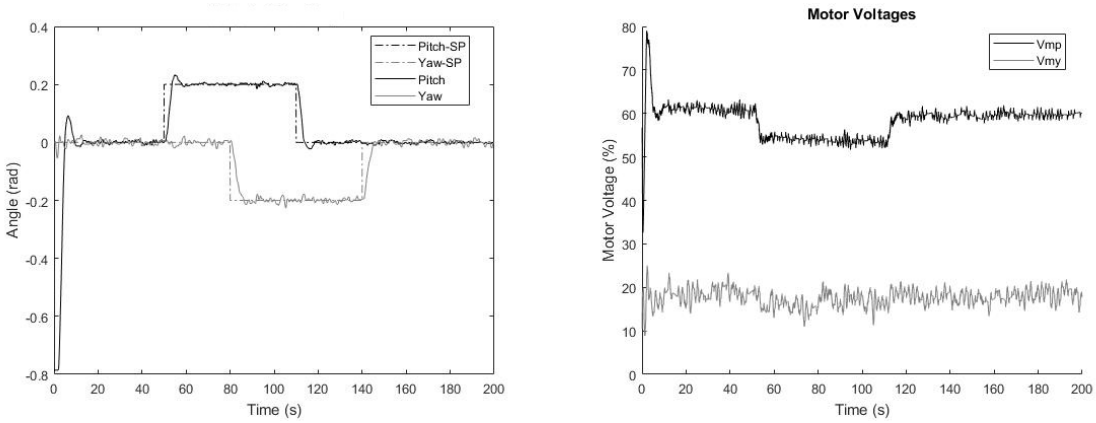
(a) Set Point Tracking (b) Control Signal



Source: Author

Figure 44: Step Response Estimated Realigned Model with Prefilter and Output Filter

(a) Set Point Tracking (b) Control Signal



Source: Author

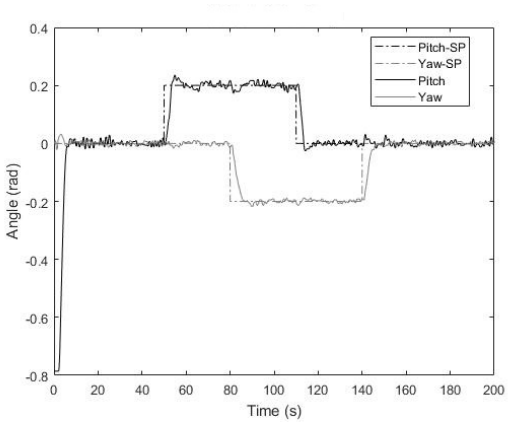
author's surprise, the response obtained with careful step tests on the open loop process were much better than in the previous cases, where the linear model was used. Notice from Figure 44a that the oscillation in the yaw angle was severely reduced, having effects as well on the pitch angle. The closed loop system showed 3.6s and 4.7s rise times and maximum overshoots of 15% and 0% for the pitch and yaw angles, respectively. Note that a simple tuning in the yaw prefilter makes the closed loop system meet the control design specifications, trading overshoot for a faster response. Also, one can note that the previous coupling between the angles were almost eliminated by the controller.

6.5 Grey-Box State Space Model

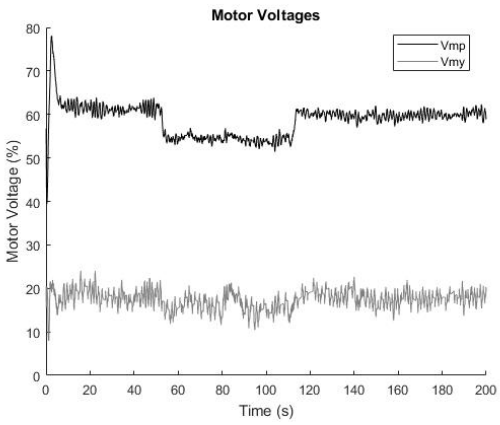
Using the grey-box model in the realigned form, the obtained results, for step responses and control signals, are shown in Figures 45a and 45b. The closed loop system showed 3.3s and 4.2s rise times and maximum overshoots of 17% and 0% for the pitch and yaw angles. Again, a simple adjustment in the yaw prefilter makes the closed loop system meet the control design specifications. It is also very important to point that this is the only case in which there is no overshoot when starting the process, strongly suggesting that this is the best model obtained until the present moment. The step responses can be improved by noting that in the last section, the control horizon m was set to 1. Doing that, one gets for the Grey-Box model the responses shown in Figures 46a and 46b. For the pitch angle, it was obtained an overshoot of 19% with a rise time of 3.1s and, for the yaw angle, 0% overshoot with a rise time of 4s. Although the variance in the pitch angle has decreased, the system is still a bit oscillatory.

Figure 45: Step Response Realigned Grey-Box with Prefilter and Output Filter with m=3

(a) Set Point Tracking



(b) Control Signal



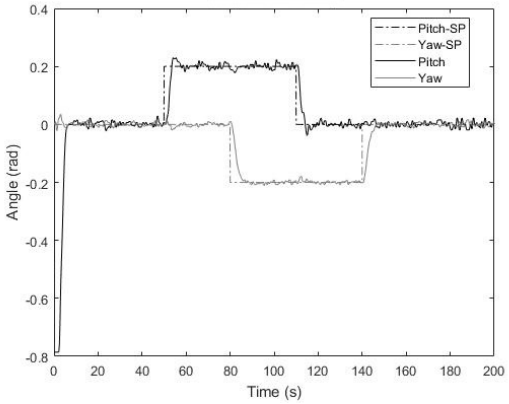
Source: Author

6.6 Two-step projection method results

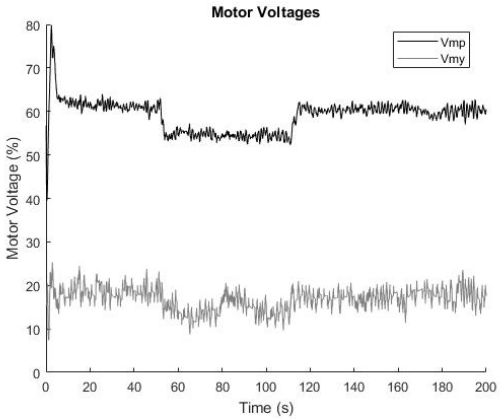
The initial settling of the process along with step responses in both angles using the identified model from using the two-step projection method, with control horizon $m = 3$, are shown in Figure 47a. The control signals are shown in Figure 47b. These results are by far the best obtained until now, concerning reference tracking. There is basically no overshoot, one quantization in the pitch axis, 1.8 seconds rise time in the pitch output and 3.5 seconds rise time on the yaw output. The results with $m = 3$ are shown in Figures

Figure 46: Step Response Realigned Grey-Box with Prefilter and Output Filter with m=1

(a) Set Point Tracking



(b) Control Signal

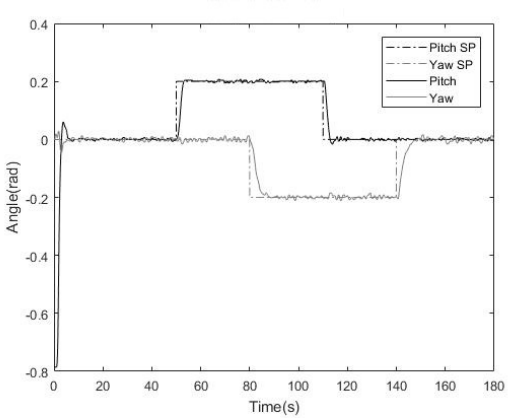


Source: Author

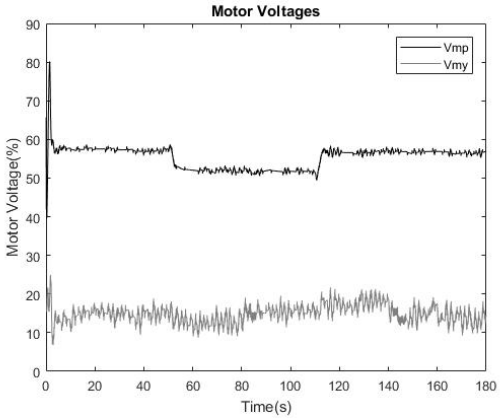
48a and 48b, the rise times here are 2.2 and 3 seconds, with an overshoot of 17% in the pitch angle and no overshoot for the yaw angle. It is important to emphasize here that the model used in the controller has no cross-coupling terms. The reason for that is that the SNR for the coupling identification was small, so instead of using a poor model to describe the coupling, it is better to let the controller treat it as a disturbance. Also, there is practically no effect from the yaw motor in the pitch angle.

Figure 47: Step Response Realigned Two-step Model with Prefilter and Output Filter with m=5

(a) Set Point Tracking



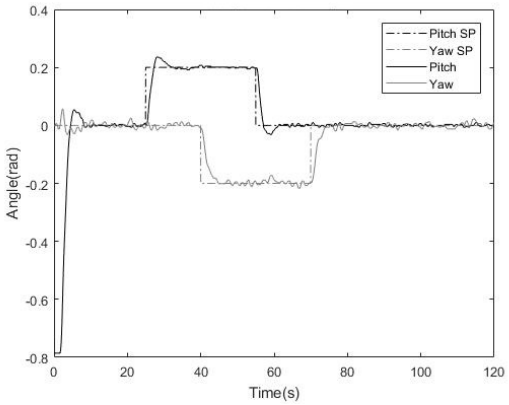
(b) Control Signal



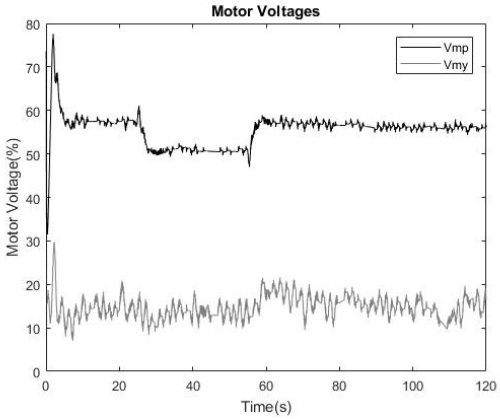
Source: Author

Figure 48: Step Response Realigned Two-step Model with Prefilter and Output Filter with m=3

(a) Set Point Tracking



(b) Control Signal



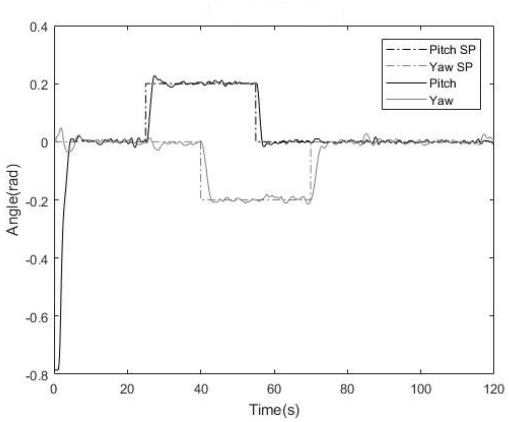
Source: Author

6.7 Asymptotic method results

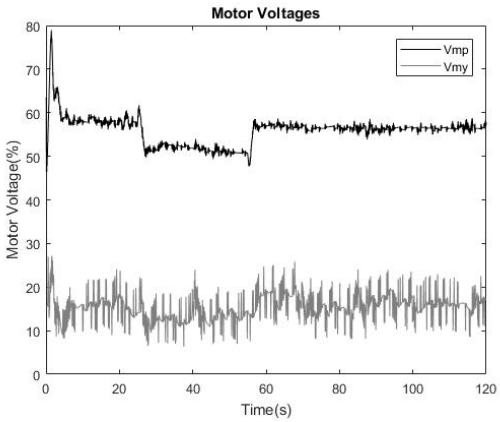
In general, this method results were a bit more oscillatory than the ones obtained with the two-step projection method or even the estimated first order with dead time plus integrator model, however, it still presented an improvement over the theoretical model. The step response along with the control signals for this method can be seen in Figures 49a and 49b. The rise times were around 1.8 and 2.8 seconds, in the pitch and yaw angles, respectively, being the fastest responses at the cost of a slight overshoot of 13% and undershoot of 5% in the pitch angle.

Figure 49: Step Response Realigned Asymptotic Model with Pre-Filter and Output Filter with m=3

(a) Set Point Tracking



(b) Control Signal



Source: Author

6.8 Comparison with PID controller using the Identified Model

Finally, the identified two-step model was used to generate an optimal tuning of a PID controller. This was done using a Genetic Algorithm in MATLAB with the optimization cost being the ISE criterion, when load disturbances were applied to the model. A first order Prefilter was then tuned heuristically in the real process in order to improve he reference tracking. The considered 2-DOF PID controller equations along with the Prefilter are described by:

$$C(z) = K_p + K_i T_s \frac{1}{z-1} + K_d \frac{N}{1 + N T_s \frac{z}{z-1}} \tag{6.2}$$

$$F(z) = \frac{K_{pf}}{z - p_f} \tag{6.3}$$

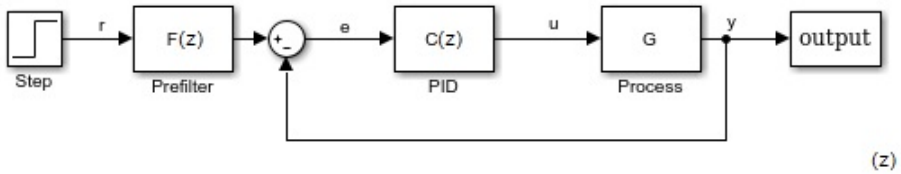
The PID and the prefilter parameters can be seen in Table 8. The structure of the 2-DOF

Table 8: Resulting optimization PID parameters

Parameters	Kp	Ki	Kd	N	K_{pf}	p_f
Pitch Controller	27.7	26.22	93.915	20	0.01534	0.9846
Yaw Controller	93.91	3.66	37.78	20	0.0198	0.9802

PID can be seen in Figure 50.

Figure 50: 2-DOF PID Structure

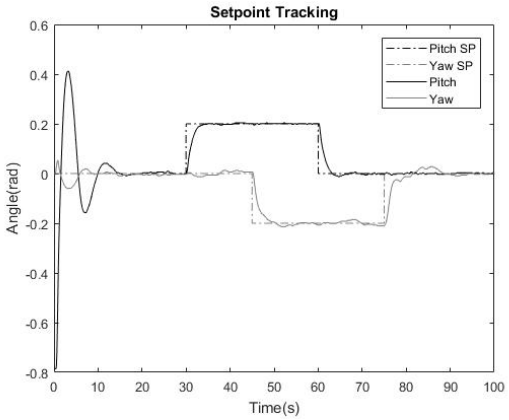


Source: Author

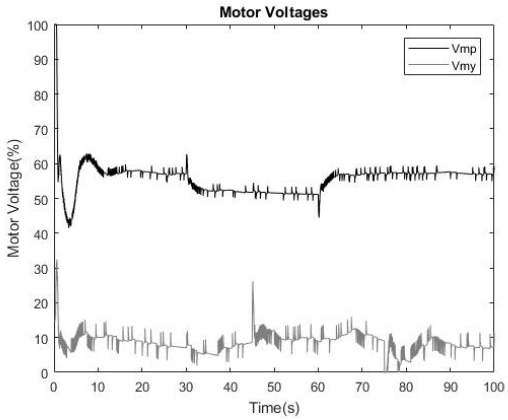
The reference tracking results along with the control signals can be seen in Figures 51a and 51b. The rise times are 2.2 and 5 seconds, for the pitch and yaw angles, with an no overshoot on the pitch angle and around 5% overshoot for the yaw angle. This results were slower than the MPC controller results with the identified model for the yaw angle and on par for the pitch angle. The PID controller, contrary to even the worst case of the MPC tuning, was barely able to bring the process from the starting point of -0.8 rad to the horizontal position of 0 radians in a satisfactory way, this is due to the process model change between the operating points, which makes a good tuning around one operating point sometimes oscillatory around another. This also suggests that the

Figure 51: Step Response using a 2-DOF PID Controller

(a) Set Point Tracking



(b) Control Signal



Source: Author

obtained MPCs are far more robust to these model changes. Since the PID parameters were found in a simulation with the identified two-step model, they yielded good results and given the excellent results of the MPC using that model as seen in Figure 47a, these results strongly suggest that the used two-step model is good for control. One important aspect of the PID performance vs MPC, in general, is that since the PID controller does not account for the saturation in its control strategy, its performance often drops when the saturation is active. This is showcased in Figure 51b, since the control signal cannot go below 0, when that happens, around 80s, the yaw controller performance severely drops. However, it must be pointed out that this tuning was the only one that was found satisfactory and relies on the results from system identification. Other tuning methods such as Ziegler-Nichols or Relay Tuning were not even close to the performance of the PID shown here.

6.9 Control Performance - Model Comparison

Given that the true process model is not known, it's hard to explicitly tell which identified model is closer to the true model. Since the objective of the identification experiments were ultimately to improve closed-loop performance, a more natural approach would be to compare the MPC performance when different identified models are used. The models will be compared in respect to different control performance indexes when used with the best tuning results.

Here it are considered the Maximum Overshoot (%), the Settling time (5%) in seconds and the IAE (Integrated Absolute Error) as an absolute number. The experiments had

Table 9: Control Performance Indexes - Pitch Axis

Parameters	$Inc.SS$	$Realig$	$StepTest$	$GreyBox$	$Two-step$	$Asym$	PID
Overshoot (%)	33	31.9	15	19	0	13	0
Settling Time(s)	13.7	11.3	7	16	2.8	9	3.2
Rise Time (s)	3.85	3.5	3.6	3.1	2.2	1.8	2.2
IAE	131.9	133.9	92.77	106.8	54.35	60.8	172.3

Table 10: Control Performance Indexes - Yaw Axis

Parameters	$Inc.SS$	$Realig$	$StepTest$	$GreyBox$	$Two-step$	$Asym$	PID
Overshoot (%)	20	30.4	0	0	0	0	7
Settling Time(s)	-	-	11.5	10.5	7.2	2.6	9
Rise Time (s)	5.5	5.2	4.7	4	3	2.8	5
IAE	85.8	106.8	48.3	40.4	37.8	38.2	91.9

the same duration of 200s. From Tables 9 and 10, one can see that while the two-step projection method was the winner in almost all categories, it must be emphasized that the results did not differ much from the asymptotic method. What is clear, is that these two identification techniques outperformed the rest, emphasizing the vital part system identification often plays in model based control design. While the results of the StepTest model were not on par with the other two, it took a very minimal effort to come up with the model, presenting the best trade-off between effort vs performance. Finally, the results of the Inc.SS and Realig models emphasized the need for system identification in this problem, often being outperformed by the proposed 2-DOF PID controller.

7 CONCLUSION

The results obtained are convincing that the strategy followed in this work in relation to the implementation of the controller is correct. Despite deceptively simple, the implementation on the true process presented a wide array of challenges, such as: C++ code implementation for MPC, choice of a suitable sampling time, presence of a non-negligible quantization, making the velocities estimation significantly harder.

The realignment model was used to bypass the estimation problem. However, that resulted in a controller initially too aggressive. To circumvent this problem, the output filter was implemented. Despite being designed for PID controllers, it presented good results when applied to MPC.

Although the theoretical model could not be validated with the real process data, that could be solved in two ways: improve the mathematical model, possibly including the motor dynamics or performing system identification experiments on the process. The second option is the one which was considered here. This is due to it being more common in MPC applications in the industry.

It is important to notice that despite the mathematical model described here not being a rigorous approximation of the process, it is still very useful for control. The results from the previous section confirm that it is still able to stabilize the process and follow step references. Of the two system identification techniques considered, the two-step projection method yielded better results. However, this may not be due to the method *per se*, but to a non-optimal choice of the frequency window in the asymptotic method, which was made by trial and error. The choice of this frequency window is vital for closed-loop performance and its closely tied to which reference one desired to track and which disturbances appear in the process. For instance, it is inherently linked to the pre-filter choice for step references.

Surprisingly, knowing the process structure, even though unstable in open loop, simple experiments, performed before saturation, allowed a very rough estimation of the system parameters and resulted in good control performance with minimum effort. One important point to be made is that the best controller results were achieved by a model in which

there was no prior consideration about the process structure, this reinforces the fact that system identification is indeed very useful and closely tied to the controller performance. This, in turn, makes analysing a "best model" for control a daunting task, since the performance of the closed-loop is deeply tied with the model quality and the controller tuning. Here, another problem that arises is that since the process is unstable, there is no straightforward way to validate data with the model without the presence of the controller. However, it is patent that the controller of choice is robust, achieving satisfactory control performance for a wide array of internal models.

Finally, the obtained two-step model was used to generate 2-DOF PID parameters through optimisation using GA in MATLAB, and the sole fact that the simulated parameters resulted in an adequate control performance, in practice, implies that the model is close to the process. It was, however, clear that the best MPC outperformed the PID controller, this did not happen if the model used on the MPC was a bad one. A possible suggestion for future works, would be to implement the Explicit MPC. If done right, this could greatly increase the controller speed, since the model complexity is simple enough.

REFERENCES

ALEXIS, K.; NIKOLAKOPOULOS, G.; TZES, A. Switching model predictive attitude control for a quadrotor helicopter subject to atmospheric disturbances. *Control Engineering Practice*, v. 19, n. 10, p. 1195 – 1207, 2011. ISSN 0967-0661. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0967066111001262⟩.

ÅSTRÖM, K. J.; HÄGGLUND, T. *Advanced PID Control*. [S.l.]: ISA-The Instrumentation, Systems, and Automation Society, 2006.

ÅSTRÖM, K. J.; WITTENMARK, B. *Computer-Controlled Systems*. [S.l.]: Prentice Hall, 2006.

BARBOSA, F. S.; NEVES, G. P.; ANGÉLICO, B. A. Discrete LQG/LTR control augmented by integrators applied to a 2-DOF helicopter. In: *2016 IEEE Conference on Control Applications (CCA)*. [S.l.: s.n.], 2016. p. 1238–1243.

BEMPORAD, A. A quadratic programming algorithm based on nonnegative least squares with applications to embedded model predictive control. *IEEE Transactions on Automatic Control*, v. 61, n. 4, p. 1111–1116, April 2016. ISSN 0018-9286.

BESSELMANN; JORG; KNUTSEN; LUNDE; STAVA; VAN DE MOORTEL Partial torque ride through with model predictive control. *2016 Petroleum and Chemical Industry Conference Europe (PCIC Europe)*, p. 1–8, June 2016.

BINET, G.; KRENN, R.; BEMPORAD, A. Model predictive control applications for planetary rovers. *11th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Montreal, Canada, Jan 2012.

BORRELLI, F.; FALCONE, P.; KEVICZKY, T. MPC-based approach to active steering for autonomous vehicle systems. *International Journal of Vehicle Autonomous Systems*, v. 3, n. 2, p. 265–291, 2011.

BOYD, S.; VANDENBERGHE, L. *Convex Optimization*. [S.l.]: Cambridge University Press, 2009.

CAIRANO; YANAKIEV; BEMPORAD; KOLMANOVSKY; HROVAT An MPC design flow for automotive control and applications to idle speed regulation. *Proceedings of the 47th IEEE Conference on Decision and Control*, Cancun, Mexico, p. 5686–5691, Dec 2008.

CAIRANO, S. D.; BEMPORAD, A. Model predictive control tuning by controller matching. *IEEE Transactions on Automatic Control*, v. 55, n. 1, p. 185–190, Jan 2010. ISSN 0018-9286.

DUTESCU, D. A.; RADAC, M. B.; PRECUP, R. E. Model predictive control of a nonlinear laboratory twin rotor aero-dynamical system. In: *2017 IEEE 15th International*

Symposium on Applied Machine Intelligence and Informatics (SAMI). [S.l.: s.n.], 2017. p. 000037–000042.

FORSELL, U. Closed-loop identification: Methods, theory, and applications. Doctoral Thesis, Linköping University. 1999.

FORSSELL, U.; LJUNG, L. A projection method for closed-loop identification. *IEEE Transactions on Automatic Control*, v. 45, n. 11, p. 2101–2106, Nov 2000. ISSN 0018-9286.

FRANKLIN, G. F.; POWELL, J. D.; WORKMAN, M. L. *Digital Control of Dynamic Systems*. [S.l.]: Addison-Wesley, 1997.

GARCIA, C. *Modelagem e Simulação*. [S.l.]: EDUSP, 2013.

GIORGETTI; RIPACCIOLI; BEMPORAD; KOLMANOVSKY; HROVAT Hybrid model predictive control of direct injection stratified charge engines. *IEEE/ASME Transactions on Mechatronics*, v. 11, n. 5, p. 499–506, Oct 2006. ISSN 1083-4435.

HARTLEY, E. N.; MACIEJOWSKI, J. M. Designing output-feedback predictive controllers by reverse-engineering existing LTI controllers. *IEEE Transactions on Automatic Control*, v. 58, n. 11, p. 2934–2939, Nov 2013. ISSN 0018-9286.

HOVGAARD; LARSEN; JØRGENSEN; BOYD MPC for wind power gradients; utilizing forecasts, rotor inertia, and central energy storage. *2013 European Control Conference (ECC)*, Zurich, Switzerland, p. 4071–4076, Jul 2013.

HROVAT; DI CAIRANO; TSENG; KOLMANOVSKY The development of model predictive control in automotive industry: A survey. *2012 IEEE International Conference on Control Applications*, Dubrovnik, Croatia, p. 295–302, Oct 2012.

JANSSON, M. Subspace identification and ARX modeling. *IFAC Proceedings Volumes*, v. 36, n. 16, p. 1585 – 1590, 2003. 13th IFAC Symposium on System Identification (SYSID 2003), Rotterdam, The Netherlands, 27-29 August, 2003.

JOHANSSON, R. *System Identification*. [S.l.]: KFS AB, 2012. Lecture Notes FRT041 System Identification LTH.

JOHANSSON, R. *Predictive and Adaptive Control*. [S.l.]: KFS AB, 2013. Lecture Notes FRTN15 Predictive Control LTH.

KAMINSKI, P. C. *Mecânica Geral para Engenheiros*. [S.l.]: Editora Edgard Blücher LTDA, 2000.

LIU, C.; CHEN, W.-H.; ANDREWS, J. Tracking control of small-scale helicopters using explicit nonlinear mpc augmented with disturbance observers. *Control Engineering Practice*, v. 20, n. 3, p. 258 – 268, 2012. ISSN 0967-0661. Disponível em: ⟨http://www.sciencedirect.com/science/article/pii/S0967066111002358⟩.

LJUNG, L. *System Identification: Theory for the User*. 2. ed. [S.l.]: Prentice Hall, 1999.

MACIEJOWSKI, J. *Predictive Control with Constraints*. [S.l.]: Prentice Hall, 2002.

NETO, G. G.; BARBOSA, F.; ANGÉLICO, B. A. 2-DOF helicopter controlling by pole-placements. *12th IEEE International Conference on Industry Applications (INDUSCON)*, Curitiba, Brazil.

ODLOAK, D. Chemical processes control lecture notes. PQI 5780 Poli-USP Graduate Program Class. 2014.

OGATA, K. *Modern control engineering*. 4. ed. [S.l.]: Prentice Hall, 2002.

OVERSCHEE, P. V.; MOOR, B. D. *Subspace Identification for Linear Systems*. [S.l.]: KLUWER Academic Publishers, 1996.

PASCUCCI, C. A.; BENNANI, S.; BEMPORAD, A. Model predictive control for powered descent guidance and control. *European Control Conference*, Linz, Austria, p. 1389–1393, Jul 2015.

RAGHAVAN, R.; THOMAS, S. Practically implementable model predictive controller for a twin rotor multi-input multi-output system. *Journal of Control, Automation and Electrical Systems*, v. 28, n. 3, p. 358–370, June 2017.

RAHIDEH, A. *Model Identification and Robust Nonlinear Model Predictive Control of a Twin Rotor MIMO System*. Tese (Doutorado) — Queen Mary University of London, 2009.

RAHIDEH, A.; SHAHEED, M. H. Robust model predictive control of a twin rotor mimo system. In: *2009 IEEE International Conference on Mechatronics*. [S.l.: s.n.], 2009. p. 1–6.

ROMANO, R. A.; POTTS, A. S.; GARCIA, C. Frontiers in advanced control systems. In: INTECH Open Access Publisher, 2012. Chapter 11, Model Predictive Control Relevant Identification.

SALAZAR, J. C.; NEJJARI, F.; SARRATE, R. Reliable control of a twin rotor mimo system using actuator health monitoring. In: *22nd Mediterranean Conference on Control and Automation*. [S.l.: s.n.], 2014. p. 481–486.

SIMON, D. *Optimal State Estimation: Kalman, H Infinity and Nonlinear Approaches*. [S.l.]: John Wiley & Sons, 2006.

WAHLBERG, B. Model reductions of high-order estimated models: the asymptotic ML approach. *International Journal of Control*, v. 49, n. 1, p. 169–192, 1989.

WANG, L. *Model Predictive Control System Design and Implementation Using MATLAB*. [S.l.]: Springer Science & Business Media, 2009.

WANG, Y.; BOYD, S. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, v. 18, n. 2, p. 267–278, March 2010. ISSN 1063-6536.

WISMER, R. C. D. A. *Introduction to Nonlinear Optimization: A Problem Solving Approach*. [S.l.]: Elsevier Science Ltd, 1978.

ZHU, Y. A frequency domain criterion for mimo model order/structure selection. *IFAC Proceedings Volumes*, v. 27, n. 8, p. 805 – 810, 1994. IFAC Symposium on System Identification (SYSID'94), Copenhagen, Denmark, 4-6 July.

ZHU, Y. Multivariable process identification for MPC: the asymptotic method and its applications. *Journal of Process Control*, v. 8, n. 2, p. 101 – 115, 1998. ISSN 0959-1524.

APPENDIX A

Here is presented the symbolic MATLAB® code used to derive the equations of motion in chapter 2

```
syms t theta(t) psi(t) xg yg zg;
syms dtheta(t) dpsi(t);
syms ddtheta(t) ddpsi(t);
syms Jx Jy Jz Jxy Jxz Jyz;
syms m r g;
syms Kpp Kpy Kyp Kyy Bp By;
syms Vmp Vmy;
assume([m r g Jx Jy Jz Bp By],'positive');
assume([t Jxy Jxz Jyz Kpp Kpy Kyy Vmp Vmy xg yg zg],'real');


J = [Jx -Jxy -Jxz;-Jxy Jy -Jyz;-Jxz -Jyz Jz];
Rtheta = [cos(theta) 0 -sin(theta);0 1 0;sin(theta) 0 cos(theta)
    ];
Rpsi = [cos(psi) -sin(psi) 0;sin(psi) cos(psi) 0;0 0 1];

pg2 = [xg;yg;zg];
GO=Rpsi*Rtheta*pg2;

vG = subs(diff(GO,t),[diff(theta(t), t) diff(psi(t), t)],[dtheta
    (t) dpsi(t)]);

w = [0;dtheta;dpsi];
wT = [0 dtheta dpsi];
```

```
23
24 T = simplify(1/2*m*expand(sum(vG.*vG))+1/2*wT*J*w);
25 U = m*g*r*sin(theta);
26
27 L = T-U; %Lagrangian
28
29 %% Lagrange Terms
30 lg11 = simplify(subs(diff(functionalDerivative(L,dtheta),t),[
       diff(theta(t), t) diff(psi(t), t) diff(dtheta(t), t) diff(
       dpsi(t), t)],[dtheta(t) dpsi(t) ddtheta ddpsi]));
31 lg12 = simplify(subs(functionalDerivative(L,theta),[diff(theta(t
       ), t) diff(psi(t), t) diff(dtheta(t), t) diff(dpsi(t), t)],[
       dtheta(t) dpsi(t) ddtheta ddpsi]));
32 lg22 = simplify(subs(diff(functionalDerivative(L,dpsi),t),[diff(
       theta(t), t) diff(psi(t), t) diff(dtheta(t), t) diff(dpsi(t),
        t)],[dtheta(t) dpsi(t) ddtheta ddpsi]));
33 lg21 = simplify(subs(functionalDerivative(L,psi),[diff(theta(t),
        t) diff(psi(t), t) diff(dtheta(t), t) diff(dpsi(t), t)],[
       dtheta(t) dpsi(t) ddtheta ddpsi]));
34
35 lg1 = lg11+lg12;
36 lg2 = lg22+lg21;
37 %% Matriz diferencial
38 Md=[functionalDerivative(lg1,ddtheta(t)),functionalDerivative(
       lg1,ddpsi(t));functionalDerivative(lg2,ddtheta(t)),
       functionalDerivative(lg2,ddpsi(t))];
39 %lado direito da equacaoo sem ddtheta
40 rhs1p = Kpp*Vmp+Kpy*Vmy-Bp*dtheta-simplify((lg1-
       functionalDerivative(lg1,ddtheta(t))*ddtheta)-
       functionalDerivative(lg1,ddpsi(t))*ddpsi);
41 rhs1y = Kyp*Vmp+Kyy*Vmy-By*dpsi-simplify((lg2-
       functionalDerivative(lg2,ddtheta(t))*ddtheta)-
       functionalDerivative(lg2,ddpsi(t))*ddpsi);
42
43 eqdiff = simplify(inv(Md)*[rhs1p;rhs1y],'
       IgnoreAnalyticConstraints',true);
```

```
44
45 eqddtheta = sum(eqdiff.*[1;0]);
46 eqddpsi = sum(eqdiff.*[0;1]);
47
48 op = [0 0 0 0]; % op = [psi theta dpsi dtheta]
49
50 dftheta1 = simplify(subs(subs(subs(subs(functionalDerivative(
       eqddtheta,theta),psi,op(1)),theta,op(2)),dpsi,op(3)),dtheta,
       op(4)));
51 dfpsi1 = simplify(subs(subs(subs(subs(functionalDerivative(
       eqddtheta,psi),psi,op(1)),theta,op(2)),dpsi,op(3)),dtheta,op
       (4)));
52 dfdpsi1 = simplify(subs(subs(subs(subs(functionalDerivative(
       eqddtheta,dpsi),psi,op(1)),theta,op(2)),dpsi,op(3)),dtheta,op
       (4)));
53 dfdtheta1= simplify(subs(subs(subs(subs(functionalDerivative(
       eqddtheta,dtheta),psi,op(1)),theta,op(2)),dpsi,op(3)),dtheta,
       op(4)));
54 dfvmp1= simplify(subs(subs(subs(subs(diff(eqddtheta,Vmp),psi,op
       (1)),theta,op(2)),dpsi,op(3)),dtheta,op(4)));
55 dfvmy1= simplify(subs(subs(subs(subs(diff(eqddtheta,Vmy),psi,op
       (1)),theta,op(2)),dpsi,op(3)),dtheta,op(4)));
56
57 dftheta2 = simplify(subs(subs(subs(subs(functionalDerivative(
       eqddpsi,theta),psi,op(1)),theta,op(2)),dpsi,op(3)),dtheta,op
       (4)));
58 dfpsi2 = simplify(subs(subs(subs(subs(functionalDerivative(
       eqddpsi,psi),psi,op(1)),theta,op(2)),dpsi,op(3)),dtheta,op(4)
       ));
59 dfdpsi2 = simplify(subs(subs(subs(subs(functionalDerivative(
       eqddpsi,dpsi),psi,op(1)),theta,op(2)),dpsi,op(3)),dtheta,op
       (4)));
60 dfdtheta2= simplify(subs(subs(subs(subs(functionalDerivative(
       eqddpsi,dtheta),psi,op(1)),theta,op(2)),dpsi,op(3)),dtheta,op
       (4)));
```

```
61 dfvmp2= simplify(subs(subs(subs(subs(diff(eqddpsi,Vmp),psi,op(1)
       ),theta,op(2)),dpsi,op(3)),dtheta,op(4)));
62 dfvmy2= simplify(subs(subs(subs(subs(diff(eqddpsi,Vmy),psi,op(1)
       ),theta,op(2)),dpsi,op(3)),dtheta,op(4)));
63
64 eqlintheta = simplify(dftheta1*theta+dfpsi1*psi+dfdtheta1*dtheta
      +dfdpsi1*dpsi+dfvmp1*Vmp+dfvmy1*Vmy);
65 eqlinpsi = simplify(dftheta2*theta+dfpsi2*psi+dfdtheta2*dtheta+
      dfdpsi2*dpsi+dfvmp2*Vmp+dfvmy2*Vmy);
66
67 eqsimptheta = simplify(subs(eqlintheta,[Jyz xg yg zg],[0 r 0 0])
      );
68 eqsimppsi = simplify(subs(eqlinpsi,[Jyz xg yg zg],[0 r 0 0]));
69
70 eqsimpthetanl = simplify(subs(eqddtheta,[Jxy Jxz Jyz xg yg zg
      ],[0 0 0 r 0 0]));
71 eqsimppsinl = simplify(subs(eqddpsi,[Jxy Jxz Jyz xg yg zg],[0 0
      0 r 0 0]));
```

APPENDIX B

Here is presented the MATLAB® code for the Incremental State-Space MPC controller function described in chapter 4

```
function u  = MPC_S(ys,x,Psi, Qbar, Thetabar, Itil, Dumax, Umin,
    Umax, H, nu, Mtil, m, p)
%#codegen

global uk_1; global x_ant;
dx = x-x_ant;

%Incremental states
xmk = [dx;x(1);x(2)];

%Setting up Ysp vector
ys = [ys(1);ys(2)];
ysp = zeros(2*p,1);

for i=1:p
    ysp(2*i-1) = ys(1);
    ysp(2*i) = ys(2);
end

% Setting Up restriction matrices and cT
el=ysp-Psi*xmk;
ct=-el'*Qbar*Thetabar;


```

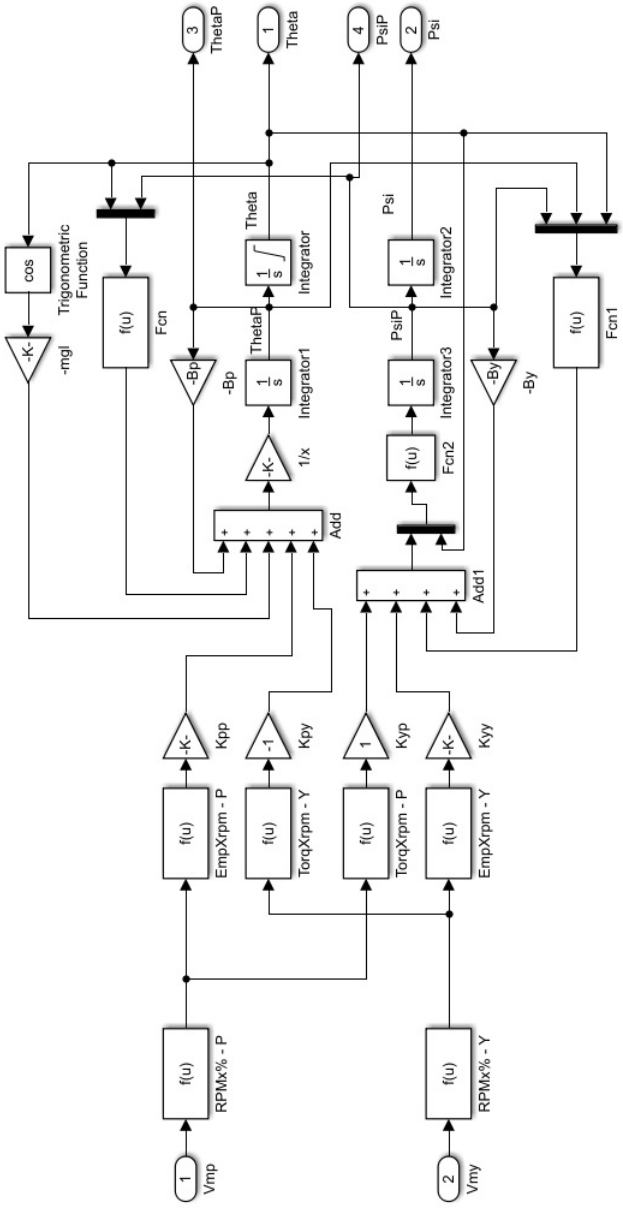
```
24 Ac=[Mtil;-Mtil;eye(m*nu);-eye(m*nu)];
25 bc=[Umax-Itil*uk_1;-Umin+Itil*uk_1;Dumax;Dumax];
26
27 %% QP Solver
28 f = ct';
29 A_cons = Ac;
30 b = bc;
31
32 [n1,m1]=size(A_cons);
33 eta=-H\f;
34 kk=0;
35
36 for i=1:n1
37     if (A_cons(i,:)*eta>b(i))
38         kk=kk+1;
39     else
40         kk=kk+0;
41     end
42 end
43
44 if (kk==0)
45
46 else
47     P=A_cons*(H\A_cons');
48     d=(A_cons*(H\f)+b);
49     [n,ms]=size(d);
50     x_ini=zeros(n,ms);
51     lambda=x_ini;
52     al=10;
53
54     max_it = 100;
55     km = 0;
56     while km < max_it
57         %find the elements in the solution vector one by one
58         % km could be larger if the Lagranger multiplier has a
               slow
```

```
59          % convergence rate.
60          lambda_p=lambda;
61          km = km+1;
62          for i=1:n
63              w= P(i,:)*lambda-P(i,i)*lambda(i,1);
64              w=w+d(i,1);
65              la=-w/P(i,i);
66              lambda(i,1)=max(0,la);
67          end
68          al=(lambda-lambda_p)'*(lambda-lambda_p);
69
70  %           if km == max_it
71  %               sqpflag = 1;
72  %           end
73
74          if (al<10e-8)
75              km = max_it;
76          end
77      end
78
79      eta=-H\f -H\A_cons'*lambda;
80  end
81
82  ukk = eta;
83
84  duk=ukk(1:nu);
85
86  %Write Control Signal and update measurements for next iteration
87  u = duk+uk_1;
88  uk_1 = u;
89  x_ant = x;
```

APPENDIX C

Implementation of the non-linear equations of motion in Simulink®.

Figure 52: Non-linear Equations



Source: Author

APPENDIX D

For a system described by (FRANKLIN; POWELL; WORKMAN, 1997):

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) + w(k) \\ y(k) &= Cx(k) + v(k) \end{aligned} \tag{D.1}$$

where $w(k)$ is the process noise and $v(k)$ is the process measurement noise, both zero mean white gaussian sequences with:

$$\begin{aligned} &\mathbf{E}w(k) = 0 \quad \mathbf{E}v(k) = 0 \\ &\mathbf{E}w(k)^T w(k) = Q \quad \mathbf{E}v(k)^T v(k) = R \\ &\mathbf{E}w(i)^T w(j) = 0 \quad \mathbf{E}v(i)^T v(j) = 0 \quad i \neq j \end{aligned} \tag{D.2}$$

The Kalman filter is the optimal linear observer that provides the minimum estimation errors, given *a priori* knowledge of the noise covariances Q and R, the state covariance P_0^+ and state x_0^+ initial condition. The filter recursive equations for each sample $k \geq 1$ are (SIMON, 2006):

$$P_k^- = AP_{k-1}^+ A^T + Q \tag{D.3}$$

$$K_k = P_k^- C^T (CP_k^- C^T + R)^{-1} = P_k^+ C^T R^{-1} \tag{D.4}$$

$$x_k^- = Ax_{k-1}^+ + Bu_k \tag{D.5}$$

$$x_k^+ = x_k^- + K_k(y_k - Cx_k^-) \tag{D.6}$$

$$P_k^+ = (I - K_k C)P_k^- = [(P_k^-)^{-1} + C^T R^{-1} C]^{-1} \tag{D.7}$$

where P_k^-, P_k^+ are *a priori* and *a posteriori* covariance estimates, K_k is the Kalman filter gain and x_k^-, x_k^+ are a priori and a posteriori state estimates.
Equation (D.7) can be rewritten by using the matrix inversion lemma (ÅSTRÖM; WITTENMARK, 2006) as:

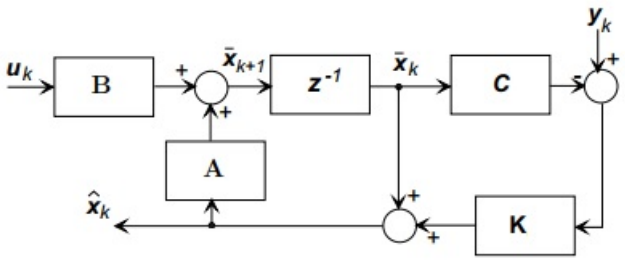
$$P_k^+ = P_k^- - P_k^- C^T (CP_k^- C^T + R)^{-1} CP_k^- \tag{D.8}$$

For a long enough running time, the covariance matrix P_k converges to the stationary solution P, that is given by the solution of the following Discrete Algebraic Ricatti Equation (DARE):

$$P = P - PC^T(CPC^T + R)^{-1}CP \tag{D.9}$$

with the stationary gain $K = PC^T R^{-1}$.
The Kalman filter can be implemented in order to compute a posteriori state estimates using the block diagram shown in Figure 53.

Figure 53: A posteriori Observer



Source: http://web.mit.edu/2.151/www/Handouts/Kalman.pdf