Per Anders Stadheim

Master's thesis

2020

Master's thesis

NTNU
Norwegian University of
Science and Technology
Faculty of Information Technology and Electrical
Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Per Anders Stadheim

Robustness Analysis of a Nonlinear Model Predictive Controller for Fixed-Wing UAVs

June 2020

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Robustness Analysis of a Nonlinear Model Predictive Controller for Fixed-Wing UAVs

Per Anders Stadheim

MIIK
Submission date: June 2020
Supervisor: Tor Arne Johansen
Co-supervisor:
Dirk Peter Reinhardt

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Summary

One of the primary concerns for an NMPC is the evaluation of the controller's performance in varying environments, more specifically the robustness of the controller. Robustness can be defined as a byproduct of controller-stability and the ability to handle unknown model disturbances experienced in real-time flights. NMPC is a nonlinear control method that systematically evaluates model constraints and a user defined cost function, ultimately finding an optimal input sequence that exploits the fast manoeuvrability of a UAV. The predictive capabilities of the controller, are a direct function of the model constraints and the references. If the plant-model is insufficient, the real-time performance will suffer. Some of the largest model disturbances affecting a typical UAV might be: 1) random turbulence, 2) wear and tear of plant, and 3) icing accretion. All these factors are random, not possible to implement in the controller plant-model. A full offset correction NMPC algorithm is proposed in this Thesis, showing promising theoretical robustness regarding ambient disturbances. Two attitude controllers are tested with this controller architecture, further strengthening the robustness analysis. Finally, the OCP simulation times of several control methods are evaluated, further investigating the possibility of a real-flight implementation.

Preface

At Norwegian University of Science and Technology (NTNU) in the Department of Engineering Cybernetics, it is mandatory to write a Master Thesis before completion of the MSc. Degree. Each student is allocated a supervisor for guidance throughout the project.

The Master Thesis was initially conducted at the campus of NTNU Trondheim Gløshaugen. Due to corona crisis, the rest and majority of the project was undertaken the student's dorm. This Master Thesis would not be possible without invaluable guidance from Professor Tor Arne Johansen (supervisor) and Dirk Peter Reinhardt (co-supervisor). Their contributions and enthusiasm have been the foundation for this Project.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Fixed-wing unmanned aerial vehicles have a broad range of industrial applications. Over the years, the use of UAV technology has increased rapidly (Insider (2020)), partly due to its broad spectre of manoeuvres and low cost applications. UAVs are most often used in target specific tasks that ease the use of manned operations such as surveillance and monitoring. This makes the UAV technology attractive in commerce. Its high manoeuvrability and light weight give it the unique position of accessing areas that otherwise would be difficult and risky for manned operations. Ultimately, it can mitigate cost and lower the risk in situations such as: search and rescue, offshore monitoring, geographical data collections, wild-life monitoring among others. The complex nonlinear dynamic structure of a fixed-wing UAV has attracted many researches around the world, giving rise to a set of complex and well designed controllers available today.

1.1 Motivation

A fixed-wing Unmanned Aerial Vehicle is commonly considered to be a complicated system due to its nonlinear coupled dynamics. Nonlinear control techniques hinge on mathematical models that prove to be difficult to model sufficiently due to the ambient disturbances and process noises. The literature review is motivated by the results obtained in the Project Thesis (Stadheim, 2019). Figure 1.1 is based on a longitudinal model simplification of the algorithm proposed in (Reinhardt and Johansen, 2019). The interval $t \in [5, 25]$ is affected by reduced aerodynamic coefficients $(C_D(\alpha), C_L(\alpha))$. By observing Figure 1.1, a clear offset of pitch and airspeed relative to the references can be seen. This offset motivates further investigation of the complete 6-DOF NMPC algorithm's robustness towards ambient disturbances.

Designing a controller that exploits the system's limitations while satisfying actuator- and physical constraints is a challenging task, for which a nonlinear model predictive controller is well-suited. An NMPC systematically solves nonlinear optimal control problems

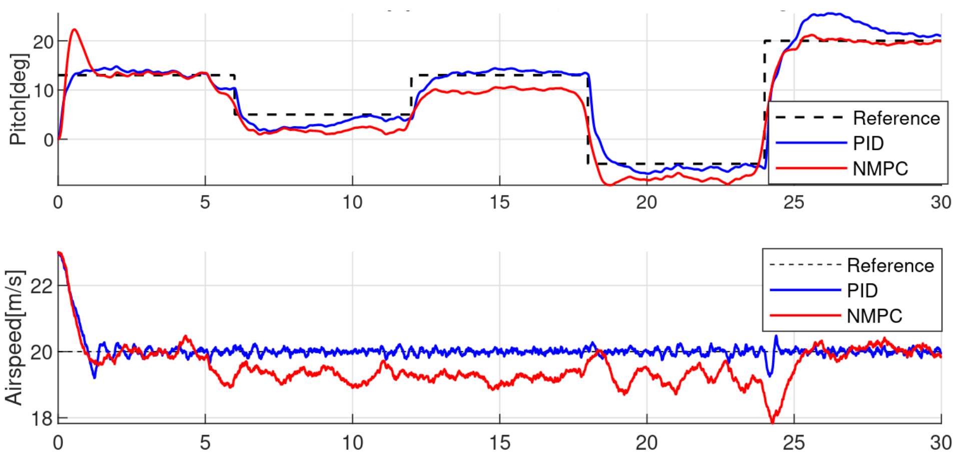


Figure 1.1: Simulation with reduced drag and lift coefficients from Project Thesis (Stadheim, 2019)

(OCP) at each time step, obtaining an optimal input sequence. Thereby, applying the first control input back to the plant, closing the loop. Nonlinear programming (NLP) problems are often nonconvex and can cause difficulties related to multiple local optima, making it difficult to find a global optimum. Therefore, nonlinear optimization problems tend to demand a larger number of computations relative to the linear programming problems. NMPC algorithms are therefore more common in multi-constrained systems with slower dynamics. The NMPC controller proposed in (Reinhardt and Johansen, 2019), is a well designed attitude NMPC that uses a mathematical model of a UAV to predict futuristic model patterns, allowing exploitation of the UAV's fast manoeuvrability. This sets demanding computer processing requirements to the microcontroller of the UAV, due to the fast dynamics. An NMPC's performance is greatly affected by the accuracy of the model predictions, and thus also the mathematical model itself.

This Master Thesis proposes a robust offset correction NMPC architecture based on (Morari and Maeder, 2012),(Maeder et al., 2009),(Borrelli and Morari, 2007) and (Ruscio, 2013). This controller is then tested on the wind formulated pitch-yaw attitude NMPC in (Reinhardt and Johansen, 2019), and a similar lower level roll-pitch controller in (Reinhardt et al., 2020). The ability of handling unknown random disturbances is thoroughly considered through a series of simulation cases. Performance is then compared to other control techniques, ensuring optimal tuning and controller set-up. Well known reference-offset mitigation methods such as integral action and offset-free control as in (Morari and Maeder, 2012) are implemented in the NMPC. Furthermore, we will test the controllers under varying wind conditions, center of mass offsets, altering of control weight matrices, icing on wings, and reduction of lift and drag coefficients. The objective of these simulations is to determine the robustness of the controller, more specifically, the controller's ability to withstand uncertainties in the event of a real-time implementation.

This Thesis is based on an ongoing project undertaken with the Supervisor and Cosupervisor. Furthermore, this Master Thesis is also a continuation of the project thesis. It is structured with a literature review in the beginning, considering controlling methods for mitigating plant-model mismatch and ambient disturbances. Chapter 3 gives an overview of the system equations combined with key terms and other relevant theories used for simulations and discussion in Chapter 5. Some of this theory is taken from the Project Thesis. A simulation time study is also presented, evaluating the NMPC's ability of real-time implementation. Finally, future work will give a framework for the current situation of the project, combined with the remaining work that is yet to be done.

Chapter 2

Literature Review

This section discusses integral action and offset-free control methods, and how they can be used to mitigate the effect of model uncertainties in Nonlinear Model Predictive Control Theory.

An NMPC algorithm uses a mathematical model of the plant to predict the future evolution of the system. The mathematical model tends to be a result of several simplifications and assumptions. Dynamical environment such as air-turbulence is not possible to model perfectly because of its random nature. The small mass of a UAV makes it more affected by small ambient disturbances. It is therefore necessary to accommodate these differences in the algorithm. Large plant-model uncertainties might lead to deviations and instabilities in the the physical system. Implementing offset-free control in the NMPC is an effective method mitigates this challenge. The goal is to achieve zero steady state tracking error: $y(t) - y_{ref}(t) \longrightarrow 0$ as $t \longrightarrow \infty$, with unknown process disturbance and noise inputs. The offset-free Model Predictive Controllers presented in (Morari and Maeder, 2012),(Maeder et al., 2009) and (Borrelli and Morari, 2007) obtain offset-free control by augmenting the states with a disturbance model. This model is used to estimate a potential mismatch between the measured output (y_m) and the model output (y), with asymptotic constant references. The plant-model constraint is then persistently changed towards a more accurate model, increasing model estimation accuracy. Finally, the improved estimations are used to initialise the MPC algorithm. We now consider the nonlinear discrete plant-model:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k) \tag{2.1a}$$

$$\boldsymbol{y}_{k,m} = \boldsymbol{g}(\boldsymbol{x}_k). \tag{2.1b}$$

Where $\boldsymbol{x}, \boldsymbol{u}$ and $\boldsymbol{y}$ are the state, input and output vector respectively. The dynamical model in 2.1 is nonlinear and is given by $\boldsymbol{f}$. Both (Maeder et al., 2009) and (Borrelli and Morari, 2007) consider linear plants, while (Morari and Maeder, 2012) considers a nonlinear plant. Even if this paper evaluates a nonlinear MPC, the linear cases still apply, due to the technique similarities. We can now consider the model augmented with the disturbance vector $\boldsymbol{d}$:

$$\boldsymbol{x}_{k+1} = \mathrm{f}_{aug}(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{d}_k) \tag{2.2a}$$
$$\boldsymbol{d}_{k+1} = \boldsymbol{d}_k \tag{2.2b}$$
$$\boldsymbol{y}_k = \mathrm{g}_{aug}(\boldsymbol{x}_k, \boldsymbol{d}_k). \tag{2.2c}$$

From the disturbance term in 2.2b, it can be seen that the augmented model assumes the disturbance to be constant over the prediction horizon in the MPC. The goal is to find an approximately constant disturbance $\boldsymbol{d}$, based on the error in the state dynamic, such that the state model $\boldsymbol{f}$ can be corrected to a more accurate estimate. The placement of the disturbance terms in the nonlinear model is important, and may result in a poor controller if wrongfully placed. An observer can now be employed, and is obtained from (Morari and Maeder, 2012),

$$\hat{\boldsymbol{x}}_{k+1} = \mathrm{f}_{aug}(\hat{\boldsymbol{x}}_k, \boldsymbol{u}_k, \hat{\boldsymbol{d}}_k) + \boldsymbol{\ell}_{\boldsymbol{x}}(\boldsymbol{y}_{k,m} - \mathrm{g}_{aug}(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{d}}_k)) \tag{2.3a}$$
$$\hat{\boldsymbol{d}}_{k+1} = \hat{\boldsymbol{d}}_k + \boldsymbol{\ell}_{\boldsymbol{d}}(\boldsymbol{y}_{k,m} - \mathrm{g}_{aug}(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{d}}_k)) \tag{2.3b}$$
$$\boldsymbol{y}_{k,m} = \mathrm{g}_{aug}(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{d}}_k). \tag{2.3c}$$

For simplicity, we assume noise free output from the measurements. The estimator in Equation 2.3a, will converge towards the original model if the estimates are close to the real values. The two terms followed by the gains $\boldsymbol{\ell}_x, \boldsymbol{\ell}_d$, are the correction terms, correcting the estimates closer to the measured values. In (Morari and Maeder, 2012), the following NMPC is proposed:

$$\min_{\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \boldsymbol{u}_{0,1,..,N-1}} \boldsymbol{F}(\boldsymbol{x}_N - \bar{\boldsymbol{x}}) + \sum_{t=0}^{N-1} \boldsymbol{l}(\boldsymbol{x}_t - \bar{\boldsymbol{x}}, \boldsymbol{u}_t - \bar{\boldsymbol{u}}) \tag{2.4a}$$

s.t.

$$\boldsymbol{x}_0 = \hat{\boldsymbol{x}}_k \tag{2.4b}$$
$$\boldsymbol{d}_0 = \hat{\boldsymbol{d}}_k \tag{2.4c}$$
$$\bar{\boldsymbol{x}} = \mathrm{f}_{aug}(\bar{\boldsymbol{x}}, \boldsymbol{d}_0, \bar{\boldsymbol{u}}) \tag{2.4d}$$
$$\boldsymbol{r}_k = \mathrm{g}_{aug}(\bar{\boldsymbol{x}}, \boldsymbol{d}_0) \tag{2.4e}$$
$$\bar{\boldsymbol{x}} \in \mathbb{X}, \qquad \bar{\boldsymbol{u}} \in \mathbb{U} \tag{2.4f}$$
$$\boldsymbol{x}_{t+1} = \boldsymbol{f}_{aug}(\boldsymbol{x}_t, \boldsymbol{d}_0, \boldsymbol{u}_t) \, \forall \, t = 1, \ldots, N-1, \tag{2.4g}$$

where $\boldsymbol{l}(.)$ is a positive definite function that penalises model deviation from the target equilibrium $\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}$. Target equilibrium is the solution to $\bar{\boldsymbol{x}} = \boldsymbol{f}_{aug}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \hat{\boldsymbol{d}}_k)$ for any given reference and disturbance. The NMPC is initialised by the state and disturbance estimate models, solving the optimal control problem (OCP) with a constant disturbance over the prediction horizon N. At the next time step the disturbance is corrected by the dynamics given in 2.3b. Good estimations lead to more accurate solutions over the prediction horizon, and the NMPC's predictive abilities become stronger; ultimately, leading to a faster and more predictive controller. The optimal input sequence from the NMPC will therefore be based on the real dynamics of the system. This controller will strive towards a predefined constant reference, using model correction. Offset-free control with model correction is an effective way of coupling feedback from measurements to the actual model, thus achieving an overall more accurate model estimation. This can increase performance and robustness towards unmodelled dynamics for a real-time implementation. Offset-free control techniques will be tested with two control architectures in Chapter 5.

There are multiple methods of mitigating the extent of unknown disturbances in UAV applications. The most common is implementation of integral action. A controller with integral action integrate past control variables, obtaining values corresponding to the total deviation at the given timeframe. Usually, integral action is used in PI or PID controllers. A standard PI controller has the following Control Law:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau. \tag{2.5}$$

The PI control input is defined by the standard notation of a proportional gain term amplified by K_p, and the integrating term given by the gain K_i. If the error $(e(t))$ of the given state increases, the proportional term will increase in a proportional manner with respect to the error. The integrator term however, is assigned a value corresponding to the total historical error over the given time-frame. Controller input is therefore a function of present and historic state and reference error. Integral action is thus often used in control theory to account for unknown disturbances and eliminating of variable offsets. Compared to the model correction offset-free control method, that uses the error dynamic to account for plant-model offsets, integral action integrates over the horizon and assigns input obtained from historical state-error. A major concern of using integral action is the integral-windups. The integral value may reach values that become dominating in the control scheme, forcing the system to deviate from the reference in the other direction, changing the sign of $e(t)$ in order to cancel out the large values. This undesirable effect is mitigated by assigning limits for the integral variables. However, it may be difficult to tune the controller in a way that integral windup is eliminated. Integral action can be seen in MPC algorithms, most common on the control inputs; Typically, to mitigate total energy consumption of the plant. In (Ruscio, 2013) an MPC controller with integral action is proposed. The state model is defined as:

$$x_{k+1} = Ax_k + Bu_K + v \tag{2.6a}$$
$$y_k = Dx_k + w, \tag{2.6b}$$

where the integral action is obtained as a result of augmenting the regular model in 2.6 with Δx_k. The perturbation term is defined as: $\Delta x_k = x_k - x_{k-1}$, and v, w are model disturbance and measurement noise. The resulting augmented system can thus be described by:

$$\underbrace{\begin{bmatrix}\Delta x_{k+1} \\ y_k\end{bmatrix}}_{\tilde{x}_{k+1}} = \underbrace{\begin{bmatrix}A & \mathbf{0} \\ D & \mathbf{I}\end{bmatrix}}_{\tilde{A}} \underbrace{\begin{bmatrix}\Delta x_k \\ y_{k-1}\end{bmatrix}}_{\tilde{x}_k} + \underbrace{\begin{bmatrix}B \\ \mathbf{0}\end{bmatrix}}_{\tilde{B}} \Delta u_k \tag{2.7a}$$

$$y_k = \underbrace{\begin{bmatrix}D & \mathbf{I}\end{bmatrix}}_{\tilde{D}} \underbrace{\begin{bmatrix}\Delta x_k \\ y_{k-1}\end{bmatrix}}_{\tilde{x}_k}. \tag{2.7b}$$

Where $\mathbf{I}, \mathbf{0}$ are appropriate sized identity and zero vectors and matrices respectively. The augmented state vector $\tilde{x}_k$ is implemented in the quadratic MPC cost function, achieving integral action for the state vector in the cost function. Another way of implementing integral action is to augment the model with a set of new integral states. An example of this setup is presented in the linear quadratic regulator scheme given in (Malkapure and Chidambaram, 2014). The new augmented model can be considered:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{2.8a}$$

$$\boldsymbol{y}(t) = \boldsymbol{g}(\boldsymbol{x}(t)) \tag{2.8b}$$

$$\dot{\boldsymbol{\zeta}}(t) = \mathbf{D}_1\boldsymbol{y}(t) - \mathbf{D}_2\boldsymbol{r}(t), \tag{2.8c}$$

where $\boldsymbol{u}(t) \in \mathbb{R}^{n_u}, \boldsymbol{y}(t) \in \mathbb{R}^{n_y}, \boldsymbol{x}(t) \in \mathbb{R}^{n_x}, \boldsymbol{\zeta}(t) \in \mathbb{R}^{n_\zeta}$ are the continuous input, output, state and integral vectors respectively. The integral vector $\boldsymbol{\zeta}$ is assigned values representing the integrated total deviation of certain defined states. Purposely, it should be implemented integral action on states that the user finds important to reach convergence. For the controller in this Thesis, it can be the attitudes and airspeed respectively. We implement integral action to the nonlinear system via the ODE in 2.8c, augmenting the error-state vector to form a new vector $\tilde{\boldsymbol{x}}(t) = [\boldsymbol{x}(t) \quad \boldsymbol{\zeta}(t)]^\top$. The standard quadratic cost function formulation ensures integral action for the MPC algorithm:

$$\boldsymbol{J} = \min_{\boldsymbol{x},\boldsymbol{u}} \int_t^{T+t} \tilde{\boldsymbol{x}}(\tau)^\top \boldsymbol{Q} \tilde{\boldsymbol{x}}(\tau) d\tau. \tag{2.9}$$

The magnitude of integral action state will differ by deviation from reference. Also, the tuning of the integral action can be altered in $\boldsymbol{Q}$. This control method is implemented and tested in chapter 5.

Chapter 3

Preliminaries

This chapter outlines a fundamental view of the theory applied to the analysis and simulations in this thesis. Some of the theory is directly from the project thesis. It begins with an overview of the used system as well as the control layout. Secondly, coordinate frames and conventions are discussed. We also consider the attitude representation: *Euler Angle Representation*, followed by the vehicle model. Lastly, some important key theoretical terms are explained.

3.1 Skywalker X8

The Fixed Wing UAV considered for simulations in this paper is the *Skywalker X8* produced by the company Skywalker Technology Co. Ltd (Ltd, 2020). It is popular in research communities, due to its low price, modularity and large payload for its small size. (Ryan et al., 2014) and (Fortuna et al., 2013) both use customised *Skywalker X8*s to investigate glacier behaviour on Greenland, and shark activity in the Portuguese island Faial. As seen in Figure 3.1, the *X8* does not have a rudder, and all rudder values will therefore remain zero in this Thesis. Instead, to achieve a yaw angle-rate, it excites both roll and pitch. The *X8* is also easy to customise, for target specific missions, making it an ideal candidate to test the algorithms purposed in this thesis. All parameter values for the Skywalker X8 are presented in Appendix B, and are from experimental tests in (Gryte et al., 2018).

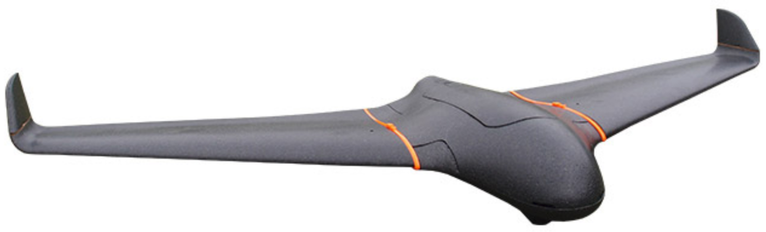


Figure 3.1: Skywalker X8, (Ltd, 2020)

3.1.1 Control layout

The UAV's control system consists of a path-following and an attitude controller. The path-following controller regulates the UAV's ability of staying on the trajectory, feeding a reference to the attitude controller. This reference consists originally of: $\boldsymbol{r}_0' = [V_{r,d}, \theta_d, \psi_d]$. Given these references, an optimal attitude can be calculated based on trim conditions of the UAV. Throughout this Project it becomes relevant to investigate a second lower level attitude controller. An attitude controller that focuses more on the attitude of the aircraft and not the heading. This controller uses $\boldsymbol{r}_1' = [V_{r,d}, \phi_d, \theta_d]$ as a reference. Given a desired roll-angle instead of desired yaw-angle makes the controller focusing purely on roll-pitch attitudes. The attitude controller then computes an optimal input sequence based on these references in order to satisfy the reference input according to the OCP. The first control input of the control input sequence is thereby applied to the plant. Measurements are then sent to an estimator evaluating the current state, feeding this approximation back to the high and low level controllers, thus closing the feedback loop. The two controllers are tested in Chapter 5.

3.2 UAV Coordinate Frames

This section provides a quick overview of the conventions and definitions of the different frames used in this thesis. The conventions are adopted by (Beard and McLain, 2012). Frames are described using the convention $\mathcal{F}^z$, where $\mathcal{F}$ tells the reader which frame is in question, and z describes which frame in question.

3.2.1 The Inertial Frame ($\mathcal{F}^i$)

Because a UAV has a limitation of range and operational-height, it is sufficient to assume that the inertial frame and NED frame are aligned. In this Thesis we therefore assume the inertial coordinate system to be the earth-fixed coordinate system. It will therefore be denoted as the north-east-down (NED) reference frame $\mathcal{F}^n$. North is aligned with $\boldsymbol{i}^n$, east with $\boldsymbol{j}^n$, and down with $\boldsymbol{k}^n$. The origin is fixed at the user defined reference point, often the control point.

3.2.2 The Vehicle Frame ($\mathcal{F}^v$)

The origin of the vehicle frame is fixed in the center of mass of the UAV, while the frames $\boldsymbol{i}^v, \boldsymbol{j}^v, \boldsymbol{k}^v$ are aligned with the inertial frame.

3.2.3 The Vehicle-1 and Vehicle-2 Frame ($\mathcal{F}^{v1}$, $\mathcal{F}^{v2}$)

Both the vehicle-1 and vehicle-2 frame have identical origins as the vehicle frame. $\mathcal{F}^{v1}$ is rotated in a positive right hand rule direction about $\boldsymbol{k}^v$. This rotation is defined as the *Yaw Angle* (ψ). The vehicle-2 frame is rotated in a similar fashion, only it is rotated about $\boldsymbol{j}^{v1}$. This angle of rotation is defined as the *Pitch Angle* (θ).

3.2.4 The Body Frame ($\mathcal{F}^b$)

The body frame is obtained rotating the the vehicle-2 frame about $\boldsymbol{i}^{v2}$ in a right hand rule direction. This angle is defines as the *Roll Angle* (ϕ).

3.2.5 The Stability and Wind Frame ($\mathcal{F}^s$, $\mathcal{F}^w$)

The lift force of the UAV is a function of the angle of attack (AoA). The AoA is defined as the angle between $\boldsymbol{i}^b$ and $\boldsymbol{i}^s$, and is denoted as $\alpha \in \mathbb{R}$. Rotating the body frame about $\boldsymbol{j}^b$ with a magnitude of α, forms the stability frame. The wind frame is defined by rotating the stability frame by a right-handed rotation about $\boldsymbol{k}^s$. This angle of rotation is defined as the *Side-Slip Angle* (SSA), and denoted as $\beta \in \mathbb{R}$. It can thus be concluded that the $\boldsymbol{i}^w$ axis is always aligned with the airspeed vector ($\boldsymbol{v}_r \in \mathbb{R}^3$).

3.3 Euler Angle Representation

The Euler Angle Representation is commonly known and used in applications such as aircraft, marine vessels and robotics. The convention of Roll, Pitch and Yaw is denoted as $\boldsymbol{\Theta} = [\phi,\ \theta,\ \psi]^\top \in \mathbb{R}^3$. Figure 3.2 from (Beard and McLain, 2012) displays the body frame ($\mathcal{F}^b$) of the UAV, and the roll-, pitch- and yaw rates measured in $\mathcal{F}^b$. These rates are denoted as p, q, r, while u, v, w are the frame velocities measured in $\mathcal{F}^b$ respectively.

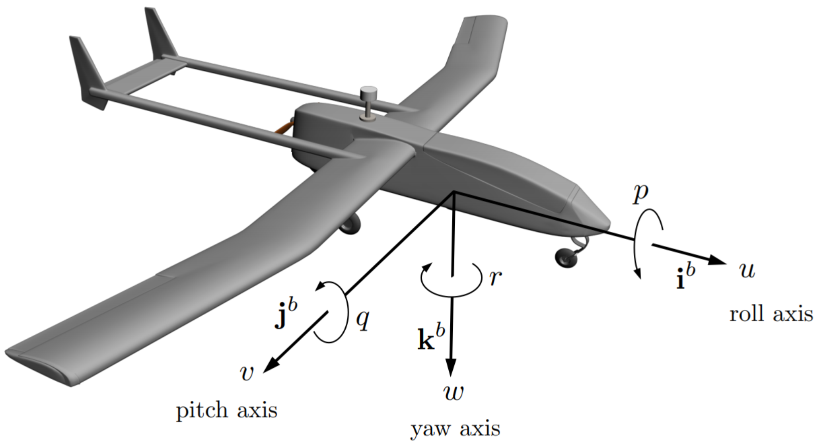


Figure 3.2: Axis of motion on a small UAV. (Beard and McLain, 2012)

3.3.1 Rotaion Matrices

Rotation matrices of order three are defined by the Special orthogonal group (SO(3)), and are used to rotate one frame to a new orientation.

$$SO(3) = \{\mathbf{R} \in \mathbb{R}^{3\times3} |\ \mathbf{R}^\top \mathbf{R} = \mathbf{R}\mathbf{R}^\top = \mathbf{I}^{3\times3},\ |\mathbf{R}| = 1\} \tag{3.1}$$

The attitude transformation matrices with the roll, pitch yaw convention from $\mathcal{F}^v$ to $\mathcal{F}^b$ are given by:

$$\mathbf{R}_v^{v1}(\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2a}$$

$$\mathbf{R}_{v1}^{v2}(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{3.2b}$$

$$\mathbf{R}_{v2}^{b}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \tag{3.2c}$$

$$\mathbf{R}_v^b(\mathbf{\Theta}) = \mathbf{R}_{v2}^b(\phi)\mathbf{R}_{v1}^{v2}(\theta)\mathbf{R}_v^{v1}(\psi). \tag{3.2d}$$

Similarly, The attitude transformation matrices from $\mathcal{F}^b$ to $\mathcal{F}^w$ are given by:

$$\mathbf{R}_b^s(\alpha) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} \tag{3.3a}$$

$$\mathbf{R}_s^w(\beta) = \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3b}$$

$$\mathbf{R}_b^w(\alpha, \beta) = \mathbf{R}_s^w(\beta)\mathbf{R}_b^s(\alpha). \tag{3.3c}$$

3.4 Vehicle Model

The vehicle model is formulated in the body frame, with exception of the position vector that is given in $\mathcal{F}^n$. The state and input vectors are thus given as:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}_{nb}^{n\top} & \boldsymbol{v}_{nb}^{b\top} & \boldsymbol{\Theta}^\top & \boldsymbol{\omega}_{nb}^{b\top} & \boldsymbol{\delta}^\top \end{bmatrix}^\top \tag{3.4a}$$

$$\boldsymbol{u} = \begin{bmatrix} \dot{\delta}_t & \dot{\delta}_a & \dot{\delta}_e & \dot{\delta}_r \end{bmatrix}^\top = \dot{\boldsymbol{\delta}}, \tag{3.4b}$$

where $\boldsymbol{p}_{nb}^n, \boldsymbol{v}_{nb}^b \in \mathbb{R}^3$ are the position and velocity vectors of $\mathcal{F}^b$ relative to $\mathcal{F}^n$ evaluated in $\mathcal{F}^n$ and $\mathcal{F}^b$ respectively. The angular velocity $\boldsymbol{\omega}_{nb}^b \in \mathbb{R}^3$ is given in $\mathcal{F}^b$ relative to $\mathcal{F}^n$ evaluated in $\mathcal{F}^b$, and is denoted as $\boldsymbol{\omega}_{nb}^b = [p,\ q,\ r]^\top$. The input vector $\boldsymbol{u} = [\dot{\delta}_t, \dot{\delta}_a, \dot{\delta}_e, \dot{\delta}_r]^\top \in \mathcal{U}$ consists of the inputs which correspond to aileron, elevator, rudder and throttle rates respectively. The state deflections and input limits are given as $\boldsymbol{\delta} \in \mathcal{D}$. Input spaces $\mathcal{U}$ and $\mathcal{D}$ are given in 3.5a. Corresponding values are given in Table 4.1.

$$\mathcal{U} = \{\boldsymbol{u} \in \mathbb{R}^4 | \dot{\delta}_{t,a,e,r}^{min} \leq \dot{\delta}_{t,a,e,r} \leq \dot{\delta}_{t,a,e,r}^{max}\} \tag{3.5a}$$

$$\mathcal{D} = \{\boldsymbol{\delta} \in \mathbb{R}^4 | \delta_{t,a,e,r}^{min} \leq \delta_{t,a,e,r} \leq \delta_{t,a,e,r}^{max}\} \tag{3.5b}$$

Now consider the arbitrarily vector $\mathbf{w} = [w_0, w_1, w_2]^\top$. The skew symmetric matrix is then defined as:

$$\mathbf{S}(\mathbf{w}) = \begin{bmatrix} 0 & -w_2 & w_1 \\ w_2 & 0 & -w_0 \\ -w_1 & w_0 & 0 \end{bmatrix}. \tag{3.6}$$

From Newton's second law, the forces acting on the body are defined by:

$$\boldsymbol{F}^b = m\frac{d}{dt_i}\boldsymbol{v}_{nb}^b, \tag{3.7}$$

where $\boldsymbol{F}^b \in \mathbb{R}^3$ is the sum of all acting forces on the body of the UAV and $\frac{d}{dt_i}$ is the time derivative in $\mathcal{F}^i$. The rule of differentiation between frames yields the following expression:

$$\frac{d}{dt_i}\boldsymbol{v}_{nb}^b = \frac{d}{dt_b}\boldsymbol{v}_{nb}^b + \mathbf{S}(\boldsymbol{\omega}_{nb}^b)\boldsymbol{v}_{nb}^b. \tag{3.8}$$

Inserting equation 3.8 in Newtons expression in 3.7 yields the following total force term:

$$\frac{d}{dt_b}\boldsymbol{v}^b_{nb} = \frac{1}{m}\boldsymbol{F}^b - \mathbf{S}(\boldsymbol{\omega}^b_{nb})\boldsymbol{v}^b_{nb}. \tag{3.9}$$

The body-frame angular rates can be defined by roll, pitch, yaw and their corresponding dynamics. The relationship between the angular-rates and the Euler angles is given by the following equation:

$$\boldsymbol{\omega}^b_{nb} = \dot{\phi}\boldsymbol{i}^b + \dot{\theta}\mathbf{R}^b_{v2}(\phi)\boldsymbol{j}^{v2} + \dot{\psi}\mathbf{R}^b_{v2}(\phi)\mathbf{R}^{v2}_{v1}(\theta)\boldsymbol{k}^{v1}. \tag{3.10}$$

Rearranging equation 3.10 yields the equation:

$$\dot{\boldsymbol{\Theta}} = \mathbf{T}_\Theta(\boldsymbol{\Theta})\boldsymbol{\omega}^b_{nb}, \tag{3.11}$$

where $\mathbf{T}_\Theta$ is given as:

$$\mathbf{T}_\Theta(\boldsymbol{\Theta}) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}. \tag{3.12}$$

Considering Newton's second law of rotation, the following expression is obtained assuming a rigid body:

$$\mathbf{M}^b = \frac{d}{dt_i}\boldsymbol{\tau}^b, \tag{3.13}$$

where $\boldsymbol{\tau}^b \in \mathbb{R}^3$ is the angular momentum vector. Using the same differentiation rule as in 3.8 yields the following moment vector in $\mathcal{F}^b$:

$$\mathbf{M}^b = \mathbf{J}^b\dot{\boldsymbol{\omega}}^b_{nb} + \mathbf{S}(\boldsymbol{\omega}^b_{nb})\mathbf{J}^b\boldsymbol{\omega}^b_{nb} \tag{3.14}$$

$$\dot{\boldsymbol{\omega}}^b_{nb} = (\mathbf{J}^b)^{-1}(\mathbf{M}^b - \mathbf{S}(\boldsymbol{\omega}^b_{nb})\mathbf{J}^b\boldsymbol{\omega}^b_{nb}). \tag{3.15}$$

Assuming NED to be the inertial frame gives the following kinematic and dynamic equations:

$$\dot{\mathbf{p}}^n_{nb} = \mathbf{R}^b_v(\boldsymbol{\Theta})^\top\mathbf{v}^b_{nb} \tag{3.16a}$$

$$\dot{\mathbf{v}}^b_{nb} = \frac{1}{m}\boldsymbol{F}^b - \mathbf{S}(\boldsymbol{\omega}^b_{nb})\boldsymbol{v}^b_{nb} \tag{3.16b}$$

$$\dot{\boldsymbol{\Theta}} = \mathbf{R}_\Theta(\boldsymbol{\Theta})\boldsymbol{\omega}^s_{nb} \tag{3.16c}$$

$$\dot{\boldsymbol{\omega}}^b_{nb} = (\mathbf{J}^b)^{-1}(\mathbf{M}^b - \mathbf{S}(\boldsymbol{\omega}^b_{nb})\mathbf{J}^b\boldsymbol{\omega}^b_{nb}) \tag{3.16d}$$

$$\dot{\boldsymbol{\delta}} = \boldsymbol{u}, \tag{3.16e}$$

where $m \in \mathbb{R}$ is the mass of the UAV and the moment of inertia in $\mathcal{F}^b$ is given as $\mathbf{J}_b \in \mathbb{R}^{3\times3}$. The mathematical model assumes the UAV to be symmetrical about the $\boldsymbol{i}^b\boldsymbol{k}^b$- plane. The forces acting on the body are stated as $\boldsymbol{F}^b = \boldsymbol{F}^b_A + \boldsymbol{F}^b_T + \boldsymbol{F}^b_g$, and represent the force contribution from aerodynamic, thrust and gravity forces. Let the gravity acceleration vector $\boldsymbol{g}^n \in \mathbb{R}^3$ be defined as $\boldsymbol{g}^n = [0,\ 0,\ g]^\top$, and the trust be obtained from the Fitzpatrick model of the X8, in (Gryte et al., 2018). The force components are then given as:

$$\boldsymbol{F}_A^b = \frac{1}{2}\rho V_r^2 S\, \mathbf{R}_b^w(\alpha,\beta)^\top \begin{bmatrix} C_x(V_r,\alpha,\beta,p,\delta_e) \\ C_y(V_r,\beta,p,r,\delta_a,\delta_r) \\ C_z(V_r,\alpha,p,\delta_e) \end{bmatrix} \tag{3.17a}$$

$$\boldsymbol{F}_T^b = \begin{bmatrix} T(V_r,\delta_t) & 0 & 0 \end{bmatrix}^\top \tag{3.17b}$$

$$\boldsymbol{F}_g^b = m\mathbf{R}_v^b(\boldsymbol{\Theta})\boldsymbol{g}^n, \tag{3.17c}$$

where $\rho, S \in \mathbb{R}$ are the air density and the wing surface area. The aerodynamic coefficients are obtained by a first-order Taylor expansion and obtained as:

$$C_x(V_r,\alpha,\beta,p,\delta_e) = \underbrace{C_{D,0} + C_{D,\alpha}\alpha}_{C_D(\alpha)} + C_{D_q}\frac{c}{2V_r}q + C_{D\delta_e}\delta_e \tag{3.18a}$$

$$C_y(V_r,\beta,p,r,\delta_a,\delta_r) = C_{Y,0} + C_{Y,\beta}\beta + C_{Y,\delta_a}\delta_a + C_{Y,\delta_r}\delta_r + \frac{b(C_{Y,p}p + C_{Y,r}r)}{2V_r} \tag{3.18b}$$

$$C_z(V_r,\alpha,p,\delta_e) = \underbrace{C_{L,0} + C_{L,\alpha}\alpha}_{C_L(\alpha)} + C_{L_q}\frac{c}{2V_r}q + C_{L\delta_e}\delta_e. \tag{3.18c}$$

Parameters $c, b \in \mathbb{R}$ are the span and chord of the UAV. The aerodynamic coefficients are highly influenced by the state and condition of the MAV, simultaneously as being volatile to the flow condition over the wings. Due to the nonlinear and random dynamics of the airflow combined with rarely exact parameter values for small UAVs, these functions are common to linearise with a first-order Taylor approximation. The lift coefficient has an approximately linear behaviour at AoAs under stall condition, and the approximation tends to be good at these values. Taylor approximation parameter values in Equation 3.18 are found in (Gryte et al., 2018) by experimental tests of the X8, and given in the Appendix B. The Fitzpatrick model in (Fitzpatrick, 2013) creates the following thrust force component:

$$T(V_r,\delta_t) = \frac{1}{2}\rho S_p C_p V_{dis}(V_{dis} - V_r) \tag{3.19a}$$

$$V_{dis} = V_r + \delta_t(k_{motor} - V_r). \tag{3.19b}$$

The total wind vector $\boldsymbol{w}^n = \boldsymbol{w}_s^n + \boldsymbol{w}_t^n \in \mathbb{R}^3$ is the wind from the static wind component and the turbulence contribution in $\mathcal{F}^n$. This is further elaborated in Chapter 3.8. The modeled relative velocity $V_r \in \mathbb{R}$ and the wind-defined angles AoA and SSA, are defined in equation 3.21 - 3.23.

$$\mathbf{v}_r^b = \begin{bmatrix} u_r & v_r & w_r \end{bmatrix}^\top = \mathbf{v}_{nb}^b - \mathbf{R}_v^b(\boldsymbol{\Theta})\boldsymbol{w}_s^n \tag{3.20}$$

$$V_r = \|\mathbf{v}_r\| = \sqrt{u_r^2 + v_r^2 + w_r^2} \tag{3.21}$$

$$\alpha = \arctan\left(\frac{w_r}{u_r}\right) \tag{3.22}$$

$$\beta = \arcsin\left(\frac{v_r}{V_r}\right) \tag{3.23}$$

Let $\mathbf{M}^b : (\boldsymbol{\delta}, \boldsymbol{\omega}_{nb}^b, V_r, \beta, \alpha) \to \mathbb{R}^3$ in Equation 3.16d be the total moment vector in $\mathcal{F}^b$. We assume the thrust-force $\boldsymbol{F}_T^b$ to be aligned with the center of mass, and the small lateral moment caused from the propeller torque to be so small it can be neglected. This assumption is described in Appendix A. The total moment vector is then defined by the wind-moment contributions $\mathbf{M}^b = \mathbf{M}_A^b$. This gives the following expression for $\boldsymbol{M}^b$:

$$\mathbf{M}^b = \frac{1}{2}\rho V_r^2 S \begin{bmatrix} b\left(C_{l,0} + C_{l,\beta}\beta + C_{l,\delta_a}\delta_a + C_{l,\delta_r}\delta_r + \frac{b(C_{l,p}p + C_{l,r}r)}{2V_r}\right) \\ c\left(C_{m,0} + C_{m,\alpha}\alpha + C_{m,\delta_e}\delta_e + C_{m,q}\frac{c}{2V_r}q\right) \\ b\left(C_{n,0} + C_{n,\beta}\beta + C_{n,\delta_a}\delta_a + C_{n,\delta_r}\delta_r + \frac{b(C_{n,p}p + C_{n,r}r)}{2V_r}\right) \end{bmatrix}. \tag{3.24}$$

The moment-components in Equation 3.24 are linearised with a first-order Taylor approximation. All values of parameters are given in Appendix B.

3.4.1 Vehicle Model in $\mathcal{F}^s$ and $\mathcal{F}^w$

When forming the OCP, it is not necessary to have the position in the state vector since it is not needed in the attitude or speed controller. We therefore, represent the velocity with airspeed, SSA and AoA dynamics. The new state and input vector $\boldsymbol{x}_s, \in \mathbb{R}^{13}, \boldsymbol{u} \in \mathbb{R}^4$ are now expressed in the stability and wind frame:

$$\boldsymbol{x}_s = \begin{bmatrix} V_r & \beta & \alpha & \boldsymbol{\Theta}^\top & {\boldsymbol{\omega}_{nb}^s}^\top & \boldsymbol{\delta}^\top \end{bmatrix}^\top \tag{3.25}$$

$$\boldsymbol{u} = \dot{\boldsymbol{\delta}}, \tag{3.26}$$

where $\boldsymbol{\omega}_{nb}^s \in \mathbb{R}^3$ is the angular velocity of $\mathcal{F}^b$ with respect to $\mathcal{F}^n$ expressed in $\mathcal{F}^s$. The angular velocity in the stability frame is obtained by the following rotation: $\boldsymbol{\omega}_{nb}^s = \mathbf{R}_b^s(\alpha)\boldsymbol{\omega}_{nb}^b = [p_s,\ q_s,\ r_s]^\top$. In order to derive the dynamical expressions for V_r, β, α, we consider the dynamical expression for linear velocity in Equation 3.16b and relative velocity in Equation 3.20. From (Stevens et al., 2016), the dynamical relation of airspeed, SSA and AoA is obtained in Equation C.1 respectively. The state equality constraint is then obtained as:

$$\dot{V}_r = \frac{T(V_r, \delta_t)\cos\beta\cos\alpha - D + mg_1}{m} \tag{3.27a}$$

$$\dot{\beta} = \frac{T(V_r, \delta_t)\sin\beta\cos\alpha + Y + mg_2}{mV_r} - r_s \tag{3.27b}$$

$$\dot{\alpha} = \frac{-T(V_r, \delta_t)\sin\alpha - L + mg_3}{m\cos\beta V_r} - p_s\tan\beta + q_s \tag{3.27c}$$

$$\dot{\mathbf{\Theta}} = \mathbf{T}_\Theta(\mathbf{\Theta})\mathbf{R}_b^s(\alpha)^\top \boldsymbol{\omega}_{nb}^s \tag{3.27d}$$

$$\dot{\boldsymbol{\omega}}_{nb}^s = (\mathbf{J}^s)^{-1}\left(\mathbf{R}_b^s(\alpha)\mathbf{M}^b - S(\boldsymbol{\omega}_{nb}^s)\mathbf{J}^s\boldsymbol{\omega}_{nb}^s\right) - S(\boldsymbol{\omega}_{bs}^s)\boldsymbol{\omega}_{nb}^s. \tag{3.27e}$$

We formulate the system in 3.27 as $\dot{\boldsymbol{x}}_s = \boldsymbol{f}(\boldsymbol{x}_s(t), \boldsymbol{u}(t)) \in \mathbb{R}^{13}$. Considering that $\boldsymbol{g}^n = [0,\ 0,\ g]$ is the gravity vector in the wind frame, and is obtained by the rotation:

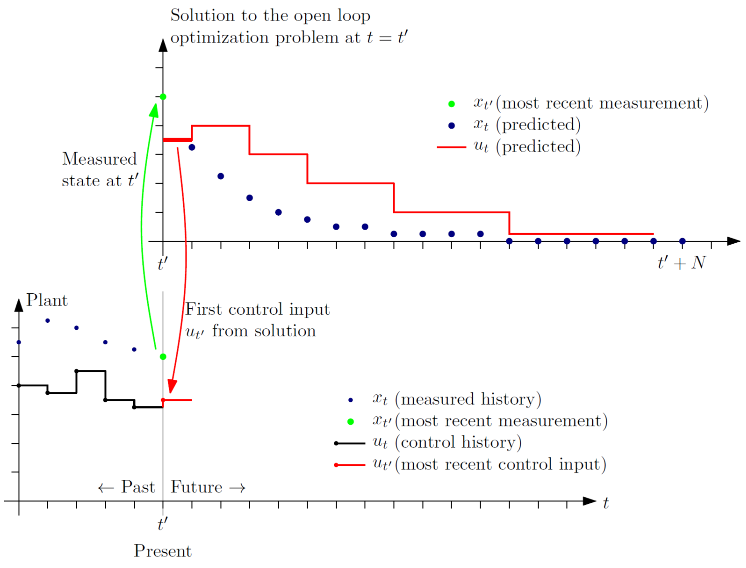
$$\boldsymbol{g}^w = \mathbf{R}_b^w(\alpha, \beta)\mathbf{R}_v^b(\mathbf{\Theta})\boldsymbol{g}^n = \begin{bmatrix} g_1 & g_2 & g_3 \end{bmatrix}^\top. \tag{3.28}$$

The inertia matrix in the stability frame is denoted as: $\mathbf{J}^s = \mathbf{R}_b^s(\alpha)\mathbf{J}^{b\top}\mathbf{R}_s^b(\alpha)$. If we consider the UAV to be a rigid body and having symmetry about $\boldsymbol{i}^b\boldsymbol{k}^b$-plane, the body inertia matrix has the form presented in 3.29. The impact of this assumption is elaborated in the appendix section A.3. Numerical values in $\mathbf{J}$, are obtained in Appendix B.

$$\mathbf{J}^b = \begin{bmatrix} J_{xx} & 0 & J_{xz} \\ 0 & J_{yy} & 0 \\ J_{xz} & 0 & J_{zz} \end{bmatrix} \tag{3.29}$$

3.5 Nonlinear Model Predictive Control

The model predictive control (MPC) method is widely used in process technology. A big reason being that it is an effective way of handling multi-variable constrained control problems (Bemporad and Morari, 2007). (Foss and Heirung, 2016) is an article describing the MPC scheme using the illustration in Figure 3.3. MPC uses a receding horizon that changes from one time step to the next. The optimal trajectory is calculated at each time step over the horizon N, acquiring an optimal input sequence based on the model constraints and the cost function. The first input of the optimal input sequence is then applied to the plant.



Solution to the open loop optimization problem at $t = t'$
$x_{t'}$ (most recent measurement)
x_t (predicted)
u_t (predicted)
Measured state at t'
t'
$t' + N$
Plant
First control input $u_{t'}$ from solution
x_t (measured history)
$x_{t'}$ (most recent measurement)
u_t (control history)
$u_{t'}$ (most recent control input)
← Past
Future →
t
t'
Present

Figure 3.3: Illustration of the MPC architecture. (Foss and Heirung, 2016)

The nonlinear MPC (NMPC) works very similar to a the linear MPC scheme with the main difference being that the model constraints are nonlinear instead of linear. There are some numerical methods described in (Nocedal and Wright, 2006) specifying how to solve these types of optimization problems. The most relevant is the sequential quadratic programming (SQP) method and Interior Point Methods for Nonlinear Programming. This project uses the CASADI framework (Andersson et al., 2019) to formulate the OCP and solve the nonlinear optimization problems at each step k using the ACADOS optimization toolbox (Verschueren et al., 2019)(Verschueren et al., 2018). The OCP is non-convex and can therefore be computationally heavy to solve. This often makes it difficult to guarantee whether the solution is the global minimum or a local minimum. These challenges make the implementation of a nonlinear model predictive controller difficult in a fast dynamical system.

3.6 Nonlinear Optimisation

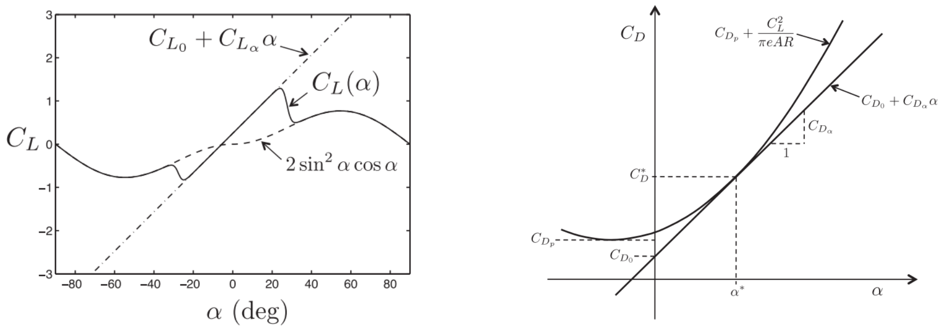
Numerical Optimisation is the method of locating the minimum of an objective function with a range of constraints. The complexity of the optimisation problem tends to increase with an increase in number of decision variables. There are several ways of solving NLPs. Typically, the most common methods are active set, interior-point and gradient projection methods. Interior point methods are currently considered to be an effective optimising method when evaluating large scaled nonlinear problems. Originally, this Thesis uses the interior point optimisation algorithm *IPOPT* (Wächter and Biegler, 2006), with the linear solver ma97 from the HSL package (HSL). The robust setup of the *IPOPT* algorithm makes the simulations robust regarding varying simulation cases. Due to the shorter computation time, the project changes to use the faster ACADOS framework. We use a sequential quadratic programming solver in the ACADOS toolbox. Further information of solver and ACADOS settings are given in Table 4.2.

3.6.1 CasADI

CasADI is an open-source tool for nonlinear optimization and algorithm differentiation (Andersson et al., 2019). It is a general-purpose tool for gradient-based numerical optimization with focus on optimal control. A CasADI based syntax in a Python based algorithm is used to find the optimal solutions of the OCP.

3.7 Stall

The lift force of a Fixed Wing Aerial Vehicle is generated by the pressure difference caused by the relative flow-velocity around the foil. Stall is a phenomenon the system undergoes above stall angle values for the AoA. The stall condition occurs differently as UAVs tend to have different designs. If a MAV undergoes stall condition, the airflow over the wing is separated from the foil, changing the separation point, resulting in a sudden drop of lift. This is graphically displayed in 3.4a after approximately 20°.



(a) Lift coefficient as a function of α

(b) Drag coefficient as a function of α. Both the quadratic and linear case are presented.

Figure 3.4: Displayed trend of lift- and drag coefficient from (Beard and McLain, 2012)[p.47-48]

The lift coefficient is an approximate linear function of the AoA as long as the magnitude of AoA is kept below stall values. A linear approximation of the lift coefficient may therefore be sufficient, due to the constrained values of AoA (see $\alpha_{max}, \alpha_{min}$, Table 4.1). A linear approximation of the drag coefficient may be less accurate, due to its real quadratic growth with respect to the AoA (Figure 3.4b).

3.8 Turbulence

Disturbances such as turbulence, rain, pressure drops and other ambient factors may influence the stable flight conditions of the UAV. The low mass makes it more susceptible to ambient disturbances than larger aviation vessels. Turbulence is considered one of the most influential factors because it can change the AoA to a critical state value. Turbulence is random and thereby difficult to simulate accurately. Most often, the UAV will fly in stable and low weather conditions, but in some applications it is important to use the UAV regardless of weather conditions (search and rescue, arctic exploration etc). The turbulence profile in the simulations will therefore also represent a worst case scenario for reasonable flight conditions.

Turbulence tends to vary with altitude. It is often more intense and volatile at low altitudes. Because UAVs operate in low altitudes, they often experience turbulence even with low wind conditions. Several methods of creating turbulence profiles for simulations are developed. Some of these can be seen in (Róbert, 2009a) and (Róbert, 2009b). The most known stochastic turbulence methods are the *Dryden Method* and the *Kármán Method.* We use the *Dryden Method* approximation model in our simulations. This model hinges on the concept of filtering white noise, creating a turbulence component which is added to the static wind components. The transfer functions used are from (Beard and McLain, 2012) and given as:

$$H_u(s) = \sigma_u \sqrt{\frac{2V_a}{L_u}} \frac{1}{s + \frac{V_a}{L_u}} \tag{3.30}$$

$$H_v(s) = \sigma_v \sqrt{\frac{3V_a}{L_v}} \frac{(s + \frac{V_a}{\sqrt{3}L_v})}{(s + \frac{V_a}{L_v})^2} \tag{3.31}$$

$$H_v(s) = \sigma_w \sqrt{\frac{3V_a}{L_w}} \frac{(s + \frac{V_a}{\sqrt{3}L_w})}{(s + \frac{V_a}{L_w})^2}. \tag{3.32}$$

Parameters $L_u, L_v, L_w \in \mathbb{R}$ are defined in the Parameters in the Appendix B. The values are obtained from (Beard and McLain, 2012)[56]. These values vary with height and class of turbulence. The parameters: $\sigma_u, \sigma_v, \sigma_w \in \mathbb{R}$ are the intensities of the turbulence. If we consider the transfer-function vector to be $\boldsymbol{H}(s) = [H_u(s)\ H_w(s)\ H_v(s)]^\top$ and the white noise as $\sigma \sim \mathcal{N}(0,1)$, the resulting wind turbulence component is defined as $\boldsymbol{H}(s) : \sigma \to \boldsymbol{w}_t^b$. The total wind-gust profile used in this thesis is simulated for 30 seconds and is given in Figure 3.5.

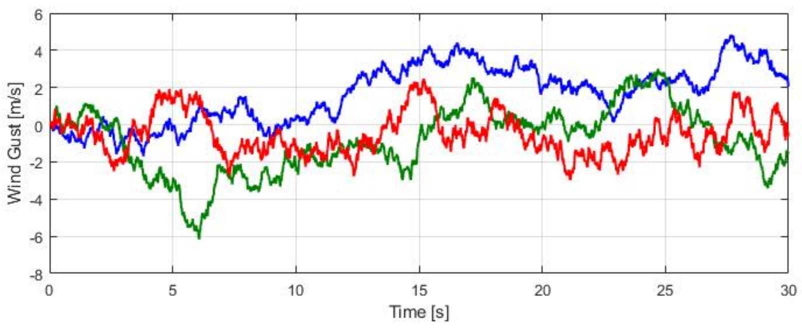


Figure 3.5: Wind gust profile with the three gust components: longitudinal (blue), lateral (green) and z-axis (red).

Chapter 4

Simulation Setup

In this chapter, we formulate the OCP based on the wind-formulation in Equations 3.27a-3.27e. We utilise two controllers: a yaw-pitch-controller from (Reinhardt and Johansen, 2019) and a lower level roll-pitch controller based on (Reinhardt et al., 2020). For comparisons in Chapter 5, a PID controller from (Reinhardt and Johansen, 2019) is also given.

4.1 The OCP Formulation

The cost function is one of the main parts of the NLP. It is the expression to be minimised with respect to the constraints. It is therefore the part of the NLP that defines the desired behaviour of the system. When formulating a cost function, it is important to evaluate which variable to penalise. The cost function needs to be formulated in a way that allows fast convergence, concurrently providing good in-air flight dynamics.

To descritise the model, a *Direct Multiple Shooting Method* is used. The time horizon T_f is divided into N control intervals. The result is a uniform timegrid $t \in \{t_0, t_1, \dots, t_N\}$ with piecewise constant control inputs $u \in \{u_0, u_1, \dots, u_{N-1}\}$. Furthermore, we define an optimization variable $\boldsymbol{\chi} \in \mathbb{R}^{n_\chi}$ as:

$$\boldsymbol{\chi} = \begin{bmatrix} \boldsymbol{x}_0^\top & \boldsymbol{u}_0^\top & \dots & \boldsymbol{x}_{N-1}^\top & \boldsymbol{u}_{N-1}^\top & \boldsymbol{x}_N^\top \end{bmatrix}^\top, \tag{4.1}$$

where $n_\chi = N(n_x + n_u) + n_x$.

4.1.1 Reference

The pitch-yaw attitude NMPC receives a reference from the path-following controller with desired relative velocity, pitch and yaw. These three states are enough to define an attitude that complies with the path. For the roll-pitch attitude controller this reference will be the airspeed and the Reduced Attitude Vector $\mathbf{\Gamma}(\phi, \theta) \in \mathbb{R}^3$. The reduced attitude vector is a function of roll and pitch angles respectively. It is therefore effectively using only airspeed, roll and pitch as references. The reference vector is then based on the initial conditions and the desired states from the path-following controller. Assuming stable trim conditions, the full reference vector for the two controllers has the form:

$$\boldsymbol{x}_d^{\psi} = \begin{bmatrix} V_{r,d} & \beta_d & \alpha_d & \mathbf{\Theta}_d^\top & {\boldsymbol{\omega}_{s,d}^{s}}^\top & \boldsymbol{\delta}_d^\top \end{bmatrix}^\top \tag{4.2}$$

$$\boldsymbol{x}_d^{\phi} = \begin{bmatrix} V_{r,d} & \mathbf{\Gamma}_d^\top \end{bmatrix}^\top . \tag{4.3}$$

4.1.2 Pitch-Yaw Attitude Controller

An error state vector is commonly used giving a magnitude of state deviation to be penalised relative to the reference vector $\boldsymbol{x}_d$ in order to define a cost function. For the pitch-yaw controller, this vector is denoted as $\tilde{\boldsymbol{x}}_\psi \in \mathbb{R}^{13}$.

$$\tilde{\boldsymbol{x}}_k^{\psi} = \boldsymbol{x}_{s,k} - \boldsymbol{x}_{d,k} \tag{4.4}$$

The cost function for the pitch-yaw attitude controller is then obtained from (Reinhardt and Johansen, 2019):

$$\mathbf{\Omega}_\psi(\boldsymbol{\chi}) = \sum_{k=0}^{N-1} \left({\tilde{\boldsymbol{x}}_k^{\psi}}^\top \boldsymbol{Q}_x^{\psi} \tilde{\boldsymbol{x}}_k^{\psi} + \boldsymbol{u}_k^\top \boldsymbol{Q}_u^{\psi} \boldsymbol{u}_k \right) + {\tilde{\boldsymbol{x}}_N^{\psi}}^\top \boldsymbol{Q}_N^{\psi} \tilde{\boldsymbol{x}}_N^{\psi}, \tag{4.5}$$

where the weighting matrices $\boldsymbol{Q}_x^{\psi} = \boldsymbol{Q}_N^{\psi} \in \mathbb{R}^{13\times 13}, \boldsymbol{Q}_u^{\psi} \in \mathbb{R}^{4\times 4}$ are used to weight perturbation of states and inputs respectively.

4.1.3 Roll-Pitch Attitude Controller

We formulate the controller based on a lower level objective formulation. Purposely, this controller will only consider roll, pitch angles as well as airspeed. We obtain the new state-error vector in Equation 4.6.

$$\tilde{\boldsymbol{x}}^{\phi} = \begin{bmatrix} V_r - V_{r,d} \\ \mathbf{\Gamma} - \mathbf{\Gamma}_d \end{bmatrix} \tag{4.6}$$

Where $\mathbf{\Gamma} \in \mathcal{S}^2$ is the reduced attitude representation given by the n-sphere-space $\mathcal{S}^n$.

$$\mathbf{\Gamma} = \mathbf{R}_v^b(\phi, \theta, \psi)\boldsymbol{k}^n \tag{4.7}$$

$$\mathcal{S}^n = \{\boldsymbol{x} \in \mathbb{R}^{n+1} | \sqrt{\boldsymbol{x}^\top \boldsymbol{x}} = 1\}. \tag{4.8}$$

The $\boldsymbol{k}^n$-axis defined in NED-frame is constant and given as $\boldsymbol{k}^n = [0, 0, 1]^\top$. If we consider $\phi \in [-\pi, \pi]$ and $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$, the reduced attitude representation is parameterised to be:

$$\mathbf{\Gamma} = \begin{bmatrix} -\sin\theta & \cos\theta \sin\phi & \cos\theta\cos\phi \end{bmatrix}^\top \tag{4.9}$$

The cost function can now be obtained as:

$$\mathbf{\Omega}_\phi(\boldsymbol{\chi}) = \sum_{k=0}^{N-1} \left({\tilde{\boldsymbol{x}}_k^{\phi}}^\top \boldsymbol{Q}_x^{\phi} \tilde{\boldsymbol{x}}_k^{\phi} + \boldsymbol{u}_k^\top \boldsymbol{Q}_u^{\phi} \boldsymbol{u}_k \right) + {\tilde{\boldsymbol{x}}_N^{\phi}}^\top \boldsymbol{Q}_N^{\phi} \tilde{\boldsymbol{x}}_N^{\phi}. \tag{4.10}$$

4.1.4 NLP Formulation

Considering the pitch-yaw NMPC's cost function in Equation 4.5, we obtain the following NLP.

$$
\begin{aligned}
\min_{\boldsymbol{\chi}} \quad & \boldsymbol{\Omega}_{\psi,\phi}(\boldsymbol{\chi}) && \text{(4.11a)}\\
\text{s.t.} \quad & && \text{(4.11b)}\\
& \boldsymbol{x}_0 = \boldsymbol{y}_m(t_0) && \text{(4.11c)}\\
& \boldsymbol{u}_0 = \boldsymbol{u}(t_0) && \text{(4.11d)}\\
& \boldsymbol{x}_{k+1} = \boldsymbol{f}\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) && \text{(4.11e)}\\
& \boldsymbol{x}_{min} \le \boldsymbol{x}_k \le \boldsymbol{x}_{max} && \text{(4.11f)}\\
& \boldsymbol{u}_{min} \le \boldsymbol{u}_k \le \boldsymbol{u}_{max} && \text{(4.11g)}
\end{aligned}
$$

Constraint 4.11e is the plant-model constraint which defines the system behaviour. The constraint does not include ambient disturbances such as wind, varying payloads, wear or icing. NMPC is a method that relies on the plant-model and is therefore susceptible to poor performance if the plant-model deviates from the physical system. The quadratic form of the cost function ensures that the cost function value is positive and strictly increasing.

The diagonal matrices $\boldsymbol{Q}_x^{\psi} = \boldsymbol{Q}_N^{\psi} \in \mathbb{R}^{13\times13}$, $\boldsymbol{Q}_x^{\phi} = \boldsymbol{Q}_N^{\phi} \in \mathbb{R}^{4\times4}$ and $\boldsymbol{Q}_u^{\psi}, \boldsymbol{Q}_u^{\phi} \in \mathbb{R}^{4\times4}$ are actively used to penalise the deviation of simulation state error and input deviation respectively. High values in the weighting matrices correspond to a large increase in the cost function, dependent on the magnitude of a variable increase. These matrices are thus used actively to tune the NMPC to an overall desirable response. If the system is in a position that violates the constraints, the OCP solver returns an infeasible solution making the algorithm crash. This effect can be eliminated using slack variables to soften up the constraints. Constraints 4.11c - 4.11d are the initial conditions based on the observer model and previous prediction horizon optimization. Finally, the inequality constraints in 4.11f - 4.11g are the state and input limits. They are constant through all simulations and are given in Table 4.1.

4.1.5 Integrator and Optimizer setup

Constraint 4.11e is integrated using an explicit fourth order Runge-Kutta method. The NLP is implemented using a multiple shooting algorithm in *Python 3.6.8* (Foundation (2018)). ACADOS is used as the optimisation tool solving the NLP with the SQP-solver *SQP_RTI*. The underlying optimisation and integrating options are presented in Table 4.2.

4.2 PID Controller

For comparisons, we implement a PID controller to the simulations. The PID controller is based on the pitch-yaw configuration and therefore creates a desired roll behaviour to form inputs controlling pitch and yaw. A PI controller is therefore created making the desired

Table 4.1: State and input simulation constraint values.

Parameter	Value	[Si-unit]
$V_{r,min}, V_{r,max}$	{10, 30}	[m/s]
β_{min}, β_{max}	{-90, 90}	[deg]
$\alpha_{min}, \alpha_{max}$	{0, 27}	[deg]
$\delta_{t,min}, \delta_{t,max}$	{0, 1}	[]
$\dot{\delta}_{t,min}, \dot{\delta}_{t,max}$	{-100, 100}	[1/s]
$\boldsymbol{\Theta}_{min}^{\top}$	-[180, 90, 180]	[deg]
$\boldsymbol{\Theta}_{max}^{\top}$	[180, 90, 180]	[deg]
${\boldsymbol{\omega}_{nb,min}^{s}}^{\top}$	-[180, 180, 180]	[deg/s]
${\boldsymbol{\omega}_{nb,max}^{s}}^{\top}$	[180, 180, 180]	[deg/s]
$\delta_{a,e,r,min}$	{-35, -35, -35}	[deg]
$\delta_{a,e,r,max}$	{35, 35, 35}	[deg]
$\boldsymbol{u}_{min}^{\top}$	-[100(%/s), 100, 100, 100]	[deg/s]
$\boldsymbol{u}_{max}^{\top}$	[100(%/s), 100, 100, 100]	[deg/s]

Table 4.2: ACADOS solver settings.

QP-solver	PARTIAL_CONDENSING_HPIPM
NLP-solver	SQP_RTI
Hessian approximation method	GAUSS_NEWTON
Integrator type	ERK4

roll angle which seeks to reduce the error in the yaw angle. The PID controller deflection and throttle scheme is given as:

$$\delta_t = k_{pV}(V_{r,d} - V_r) + k_{iV}\int_0^t (V_{r,d} - V_r)d\tau \tag{4.12a}$$

$$\delta_a = k_{p\phi}(\phi_d - \phi) + k_{i\phi}\int_0^t (\phi_d - \phi)d\tau + k_{d\phi}p \tag{4.12b}$$

$$\phi_d = k_{p\psi}(\psi_d - \psi) + k_{i\psi}\int_0^t (\psi_d - \psi)d\tau \tag{4.12c}$$

$$\delta_e = k_{p\theta}(\theta_d - \theta) + k_{i\theta}\int_0^t (\theta_d - \theta)d\tau + k_{d\theta}q. \tag{4.12d}$$

All gain parameters are given in Table 4.3. See (Reinhardt and Johansen, 2019) for more information. A simple anti-windup strategy is used, limiting the integral values when the values defy the interval given by $[\mathbf{I}_{min}, \mathbf{I}_{max}]$. The update rate of the PID controller is 100 Hertz. Actuator constraints in the PID controller are not implemented, meaning the input rate does not have a limit. However, the input is filtered through a low-pass filter.

Table 4.3: PID gain parameters.

Gain	Value
$k_{pV}, k_{p\phi}, k_{p\psi}, k_{p\theta}$	$\{0.69, 0.78, 1.08, -0.78\}$
$k_{iV}, k_{i\phi}, k_{i\psi}, k_{i\theta}$	$\{0.01, 0.01, 0.01, -0.30\}$
$k_{dV}, k_{d\phi}, k_{d\psi}, k_{d\theta}$	$\{0.1, -0.11, 0.1, -0.16\}$
$\mathbf{I}_{max} = -\mathbf{I}_{min}$	$[1.00, 0.09, 0.09, 0.09]^\top$

We therefore expect this controller to have fast inputs.

$$\boldsymbol{u}'_k = \boldsymbol{\delta}_{k-1} + \Delta t_{sim}(\boldsymbol{u}_{k-1}) \tag{4.13}$$

$$\boldsymbol{u}_k = \boldsymbol{u}_{k-1} + \frac{\Delta t_{sim}}{RC + \Delta t_{sim}} \left(\boldsymbol{u}'_k - \boldsymbol{u}_{k-1}\right) \tag{4.14}$$

The simulation parameters $RC, \Delta t_{sim} \in \mathbb{R}$ are the time filter constant and simulation time-step respectively.

Chapter 5

Simulations and Discussion

In this chapter we will present and discuss a series of simulation cases representing real-time flight situations for the UAV. The pitch-yaw and roll-pitch controllers are tested towards model uncertainties and ambient disturbances, in order to evaluate controller algorithm robustness. Typical model disturbance methods such as integral action and offset free control are implemented and compared towards the original NMPC controllers in (Reinhardt and Johansen, 2019) and (Reinhardt et al., 2020). Furthermore, the performance of the pitch-yaw controller is compared to a PID controller. The simulations are implicating situations where stable flight can be challenging to maintain, thus highlighting the controller's ability to exploit fast manoeuvrability and being robust towards disturbances. The objective of this chapter is to determine the proposed NMPCs abilities of real-time performance in severe conditions.

The general definition of a "robust controller" can be difficult to quantify, using a numerical approach. Since there are many characteristics underlying this definition, an error parameter that helps implicate deviation relative to the reference is made. Much of the controller's performance can be outlined as the ability of following the reference under the influence of numerous ambient disturbances. As a way of indicating total deviation, as well as easier comparisons foundation, the experimental error parameter (EP) (Δ^x) is developed:

$$\Delta^x = \sum_{t=t_0}^{T_{sim}/\Delta t_{sim}} \boldsymbol{x}_\Delta^\top \boldsymbol{I}^{6\times 6} \boldsymbol{x}_\Delta. \tag{5.1}$$

Where $T_{sim} = 30s, \Delta t_{sim} = 0.01s$ are the total simulated time and simulation time step size. The simplified deviation vector $\boldsymbol{x}_\Delta \in \mathbb{R}^6$ is defined in Equation 5.2, perpetrating most important state deviations respectively.

$$\boldsymbol{x}_\Delta = \begin{bmatrix} (\boldsymbol{\Theta} - \boldsymbol{\Theta}_d)^\top & \frac{1}{100}(V_r - V_{r,d}) & \beta - \beta_d & \alpha - \alpha_d \end{bmatrix}^\top \tag{5.2}$$

Table 5.1: NMPC and PID simulation parameters.

Parameter	Value	Description
N	30	Number of discretization steps in the NMPC.
N_{sim}	3000	Number of discretization steps in the integrator.
T_f	3[s]	Prediction horizon.
T_{sim}	30[s]	Total simulation time.
f_{nmpc}	20[Hz]	Update rate of the NMPC.
f_{sim}	100[Hz]	Update rate of the integrator, integrating Equation 5.6.
f_{PID}	100[Hz]	Update rate of PID controller.
Δt_{mpc}	0.1[s]	NMPC time-step. Given by $\Delta t_{mpc} = T_f / N$.
Δt_{sim}	0.01[s]	Integrator time-step. Given by $\Delta t_{sim} = T_{sim} / N_{sim}$.

The error parameter $\Delta^x \in \mathbb{R}$ will then be a representation of some "important" state deviations for each simulation. In Equation 5.2, we reduce the impact of the airspeed due to the high values relative to the other states in the deviation vector. In order to survey the input dynamics, a similar parameter is made, only representing total energy and wear on actuators:

$$\Delta^u = \sum_{t=t_0}^{T_{sim}/\Delta t_{sim}} \boldsymbol{u}_\Delta^\top \boldsymbol{I}^{5\times5} \boldsymbol{u}_\Delta, \tag{5.3}$$

where $\boldsymbol{u}_\Delta \in \mathbb{R}^5$ is the energy-usage vector. This vector includes both throttle input and the total input-rate.

$$\boldsymbol{u}_\Delta = \begin{bmatrix} \delta_t & (\boldsymbol{u}_t - \boldsymbol{u}_{t-1})^\top \end{bmatrix}^\top \tag{5.4}$$

The NMPC controller parameters are given in Table 5.1.

5.1 Sensors and Measurements

The ability for a UAV to maintain stable flight under rough conditions is highly influenced by the amount and quality of data from the on-board sensors. For simplicity of filtering noise, measurement-output is assumed to be noise-free in this Thesis. We also assume the UAV to have a set of sensors, feeding the controller with real-time measurements:

- An IMU, containing a gyroscope and an accelerometer, feeding the controller with $\boldsymbol{\omega}^b, \dot{\boldsymbol{v}}^b$.
- Global Positioning System (GPS) ensuring position of UAV in NED-frame. (not crucial for the attitude controller).
- Pitot-static tubes (differential pressure sensor), obtaining the relative velocity, as well as estimations of AoA and SSA.

The differential pressure sensor tends to be costly (Wenz et al., 2016), and is thus less common in low-cost UAVs. It is assumed that the UAV tested in this paper has a set of pressure sensors, ensuring the output signal transformed to the wind-frame to be fully defined. Finally, we also assumed that all actuators on the UAV are equipped with positioning-sensors fully defining $\boldsymbol{\delta}$. The measured output is then given as:

$$\boldsymbol{y}_m = \boldsymbol{x}_m. \tag{5.5}$$

In order to simulate scenarios that may come close to the real behaviour of the UAV, we have designed a model for the measured states $\boldsymbol{x}_m$ with the following dynamical equation:

$$\dot{\boldsymbol{x}}_m = \boldsymbol{f}_m(\boldsymbol{x}_m, \boldsymbol{u}), \tag{5.6}$$

where $\boldsymbol{f}_m : (\boldsymbol{x}_m, \boldsymbol{u}) \rightarrow \mathbb{R}^{13}$ is a dynamical system based on the system equations in 3.27, but with implemented turbulence and other disturbances. This will purposely design a simulation case towards a real-life test scenario. Varying disturbances are tested in the simulation cases later discussed, and alterations made will be emphasised. The NMPC schemes are tuned in Section 5.3.1, where the offset correction algorithms (model correction NMPC, input correction NMPC and NMPC with integral action) are tuned with a less severely altered measurement model. This makes them less tuned for the more severe case tested later.

5.1.1 Measurement Model

The controller proposed in this Thesis has to be highly robust towards varying weather conditions, as well as being able to handle different UAV platforms. Working conditions of the UAV might differ due to the varying use-purposes it may experience. Our incentive is to broaden the use of our algorithm; we consider the harshest environment this UAV might encounter. In this manner, We will then be able to conclude whether the NMPC algorithm is robust. For scientists, a popular use-case for UAVs, are glacier and wildlife monitoring in arctic conditions. From an industrial point of view, UAVs are also known to be used mapping glacier dynamics for the oil industry. To ensure robustness, we examine some features this might entail, implementing this in the measurement model in Equation 5.6.

Initially, we want to investigate the scenario of a changed center of mass. This is mathematically done by defining an offset-vector $\boldsymbol{r}^b_{b/cm} \in \mathbb{R}^3$, stating the distance from the body-frame origin to the real physical center of mass. The new moment of inertia matrix is then given from the Parallel Axis Theorem:

$$\mathbf{J}^b_m = \mathbf{J}^b - m\mathbf{S}(\boldsymbol{r}^b_{b/cm})\mathbf{S}(\boldsymbol{r}^b_{b/cm}). \tag{5.7}$$

The fact that *X8*s tend to be used by researchers, implicate that we can assume it to have mounted varying research devices on the UAV. Cameras, sensors, and strong, heavy on-board micro-controllers are common components that all have the potential of increasing the mass, as well as shifting the mass center. The measurement model will as a result add a mass component to the actual mass of the UAV, $m_m = m_{uav} + m_c = 3.36kg + 0.84kg = 4.2kg$. This is listed as the maximum take-off weight for the *Skywalker X8* in (Airelectronics). The total wind component in $\mathcal{F}^b$ is now implemented with the Dryden-gust profile given in Figure 3.5.

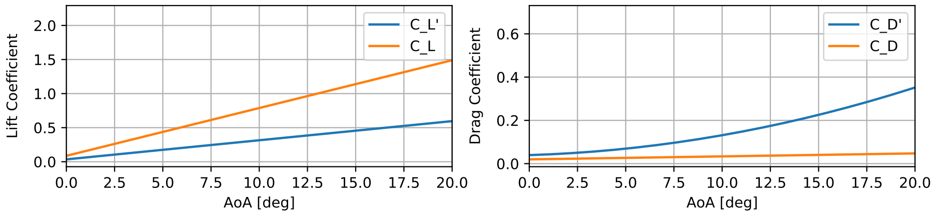


Figure 5.1: Effect of Icing given in the aerodynamic coefficients. The reduced coefficients are denoted with $'$ (blue).

Icing on Wings

Remote arctic conditions increase the cost and risk of retrieving-operations if the UAV fails. We therefore consider a worst-case scenario with atmospheric icing on air-foils. This is implemented in the measurement model by the following equations:

$$C'_D(\alpha) = \left(C_{D,0} + \alpha C_{D,\alpha 1} + \alpha^2 C_{D,\alpha 2}\right) S_D \tag{5.8}$$

$$C'_L(\alpha) = (C_{L,0} S_{L1}) + (C_{L,\alpha} S_{L2})\alpha. \tag{5.9}$$

The effect of iced airfoils for UAVs are found in (Hann et al., 2017),(Dalmau, 2018) and (Winter, 2019). They all conclude with a decrease in the lift coefficient, and an increase in the drag coefficient, with iced airfoils. To replicate an icing scenario, the drag coefficient is increased by a factor $S_D \in \mathbb{R}$ and is now a quadratic function of AoA. The lift coefficient still has close to linear tendencies even with an icing case. We therefore implement a small stagnation of growth given by $S_{L2} \in \mathbb{R}$, and an offset defined by $S_{L1} \in \mathbb{R}$. The aerodynamic coefficients are illustrated in Figure 5.1. The parameters S_D, S_{L1}, S_{L2} are assigned the values $(2, 0.4, 0.6)$. These are based on a visual inspection of the aerodynamic coefficients in (Winter, 2019). Because of the unpredictable icing accretions and placement of research components, the mass center is changed 7 cm in $\boldsymbol{i}^b$-direction. Throughout the simulations presented in this thesis, we use $\boldsymbol{r}^b_{b/c} = [0.07, 0.02, 0.02]^\top$. It will then represent a UAV with research components added (camera, sensors, etc.) on the front, slightly lower than the $\boldsymbol{i}^b$-axis. Finally, we also have an asymmetric offset along $\boldsymbol{j}^b$. This offset can be explained due to the wear and tear of the UAV and not perfectly balanced mounted research components.

5.2 Pitch-Yaw NMPC Modified For Extreme Conditions

There are many methods of mitigating random unforeseen disturbances in Nonlinear Control Theory. Typically, integral action and model correction offset-free control are most common. We design two modifications of Offset-free Nonlinear Model Predictive Controllers and an NMPC with integral action to investigate a potential performance enhancement under severe plant-model mismatches. This will potentially contribute to making a final NMPC algorithm robust against unforeseen cases, ultimately creating a robust controller for a wide range of application areas within the UAV community. The controllers are based on the controller schemes given in the Literature review in Chapter 2.

5.2.1 Model Correction NMPC

Offset-free control can be implemented in various ways. If we consider the observer equations in 2.3, the disturbance term can be implemented where we expect errors from the plant-model. We will form two different offset-free controller formulations. The first modification uses measurements from the IMU ($\boldsymbol{\omega}_m$) and pressure sensors to change the moment vector $\boldsymbol{M}^b$ and the relative velocity to more accurate estimations. Ultimately, this will lead to a more accurate plant-model. We consider the same system equations in 3.27a - 3.27e, the only difference being an added disturbance term to the moment vector and relative velocity in Equation 3.24 and 3.21 respectively. The new discretised moment and relative airspeed are thus given by the Equations in 5.10. If we consider the concatenated disturbance vector $\boldsymbol{d}_k = [\boldsymbol{d}_{M,k}^\top \ d_{v,k}]^\top \in \mathbb{R}^4$, we formulate the following model corrections:

$$\breve{\boldsymbol{M}}_k^b = \boldsymbol{M}_k^b + \boldsymbol{d}_{M,k} \tag{5.10a}$$

$$\breve{V}_r = \sqrt{u_r^2 + v_r^2 + w_r^2} + d_{v,k} \tag{5.10b}$$

$$\boldsymbol{e}_k = \left[(\boldsymbol{\omega}_{m,k} - \boldsymbol{\omega}_k)^\top \quad (V_{r,m,k} - V_{r,k})\right]^\top . \tag{5.10c}$$

Where the corrected moment vector $\breve{\boldsymbol{M}}_k^b : (\boldsymbol{M}_k^b, \boldsymbol{d}_{M,k}) \to \mathbb{R}^3$, and the relative airspeed $\breve{V}_{r,k} : (V_{r,k}, d_{v,k}) \in \mathbb{R}$ are used in the OCP to form more accurate model estimations respectively. The measured angular-rate vector and relative velocity $\boldsymbol{\omega}_{m,k}, V_{r,m,k}$ are obtained from the simulation model with implemented ambient disturbances in Section 5.1.1. To form a discretised model, we consider the uniformly distributed NMPC-time-grid $z \in \{0,\ 1,\ \ldots \frac{f_{nmpc}}{f_{sim}} N_{sim}\}$. The grid is representing the number of total performed OCP solvings. We finally state the new disturbance corrected plant-model as follows:

$$\boldsymbol{d}_{z+1} = \boldsymbol{f}_{d,z}(\boldsymbol{d}_{z:z-p}, \boldsymbol{x}_z, \boldsymbol{y}_{m,z}) = \begin{cases} \frac{1}{p+1}\left(\boldsymbol{d}_z + \boldsymbol{\ell}_d \boldsymbol{e}_z + \sum_{i=z-(p-1)}^{p} \boldsymbol{d}_i\right) & \forall p \le z \\ \boldsymbol{d}_z + \boldsymbol{\ell}_d \boldsymbol{e}_z & \forall 0 < z < p \\ \boldsymbol{0}^{4\times 1} & \forall z = 0s \end{cases} \tag{5.11}$$

where p is the low pass filter parameter. The diagonal gain matrix $\boldsymbol{\ell}_d \in \mathbb{R}^{4\times 4}$ is used to tune the model correction input $\boldsymbol{d}_{k+1}$. Tuned values for this matrix can be found in Appendix 5.3.2. From equation 5.11, it can be seen that good estimations from the NMPC leads to small values of the disturbance change $\boldsymbol{e}_k \to 0 : \boldsymbol{d}_{k+1} - \boldsymbol{d}_k \to 0$, ultimately making the disturbance converging to a constant: $\boldsymbol{d}_{k+1} = \boldsymbol{d}_k$. Because of the proportional modeling of the disturbance, it can be expected to be rather large oscillations in disturbance values. To reduce this, we use a moving average technique, taking the average of the last $p-1$ disturbances to model the next. This is purposely working as a simple low-pass filter.

In this Thesis it is used $p = 4$. We now obtain the given NLP:

$$\min_{\boldsymbol{x}} \quad \sum_{k=0}^{N-1} \left(\tilde{\boldsymbol{x}}_k^{\psi\top} \boldsymbol{Q}_x^{\psi} \tilde{\boldsymbol{x}}_k^{\phi} + \boldsymbol{u}_k^{\top} \boldsymbol{Q}_u^{\psi} \boldsymbol{u}_k \right) + \tilde{\boldsymbol{x}}_N^{\psi\top} \boldsymbol{Q}_N^{\psi} \tilde{\boldsymbol{x}}_N^{\psi} \tag{5.12a}$$

$$\text{s.t.} \tag{5.12b}$$

$$\boldsymbol{x}_0 = \boldsymbol{y}_m(t_0) \tag{5.12c}$$

$$\boldsymbol{u}_0 = \boldsymbol{u}(t_0) \tag{5.12d}$$

$$\boldsymbol{d}_0 = \boldsymbol{f}_d(\boldsymbol{d}_{z:z-3}, \boldsymbol{x}_z, \boldsymbol{y}_{m,z}) \tag{5.12e}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}\left(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{d}_k\right) \tag{5.12f}$$

$$\boldsymbol{d}_{k+1} = \boldsymbol{d}_k \tag{5.12g}$$

$$\boldsymbol{x}_{min} \leq \boldsymbol{x}_k \leq \boldsymbol{x}_{max} \tag{5.12h}$$

$$\boldsymbol{u}_{min} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{max}. \tag{5.12i}$$

5.2.2 Input Correction NMPC

Implementing offset-free control in the NMPC is also presented in this thesis by directly changing the input vector applied to the plant. This method can resemble a standard P-controlling scheme on the output of the NMPC. The previous method of implementing disturbance directly in the state equations is dependent of knowing where a possible bad estimation is to acquire good results. For a UAV, we can assume some simplifications of input meaning. Elevator can be directly linked to the pitch dynamic of the UAV. Likewise, we assume aileron, rudder and throttle to be linked to roll, yaw and airspeed respectively. The input law is given in Equation 5.13, and depicted by a block diagram in Figure 5.2.

$$\breve{\boldsymbol{u}}_k = \boldsymbol{u}_k + \Delta \boldsymbol{u}_k \tag{5.13a}$$

$$\Delta \boldsymbol{u}_k = \boldsymbol{\ell}_u \mathbf{C}(\boldsymbol{x}_{d,k} - \boldsymbol{y}_{m,k}) \tag{5.13b}$$

Where the vector $\Delta \boldsymbol{u}_k \in \mathbb{R}^4$ is the input correction matrix, and $\boldsymbol{C} \in \mathbb{R}^{4\times 13}$ is defined as:

$$\boldsymbol{C} = \begin{bmatrix} 1 & \mathbf{0}^{1\times 2} & \mathbf{0}^{1\times 3} & \mathbf{0}^{1\times 7} \\ \mathbf{0}^{3\times 1} & \mathbf{0}^{2\times 2} & \mathbf{I}^{3\times 3} & \mathbf{0}^{3\times 7} \end{bmatrix}. \tag{5.14}$$

Similar to the offset-free control method on the moment vector, the diagonal gain matrix $\boldsymbol{\ell}_u \in \mathbb{R}^{4\times 4}$ is a corresponding tuning parameter. This parameter controls the magnitude of input correction given by the attitude and relative velocity error with respect to the measurements. If the gain matrix is large, the prediction of the NMPC will be less accurate, meaning much of the desirable attributes of the NMPC can go lost. Also, large values can cause the inputs to oscillate, creating an unstable controller. There will also be manoeuvres this control scheme simply will counteract. This offset-free method is thus more susceptible to tuning-errors and the input should therefore only be corrected for small errors at each sample instance. The NMPC will then be able to keep its attributes, and concurrently keeping input correction to eliminate steady state perturbations. We therefore assign the diagonal of $\boldsymbol{\ell}_u$ with small numbers given in Section 5.3.3.

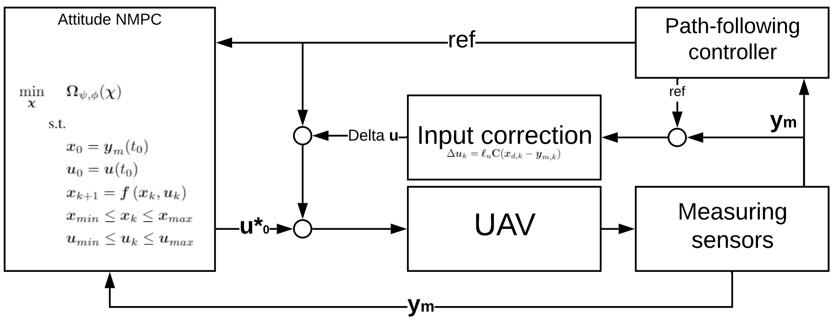

Attitude NMPC
$\min_{\boldsymbol{\chi}} \quad \boldsymbol{\Omega}_{\psi,\phi}(\boldsymbol{\chi})$
s.t.
$\boldsymbol{x}_0 = \boldsymbol{y}_m(t_0)$
$\boldsymbol{u}_0 = \boldsymbol{u}(t_0)$
$\boldsymbol{x}_{k+1} = \boldsymbol{f}(\boldsymbol{x}_k, \boldsymbol{u}_k)$
$\boldsymbol{x}_{min} \leq \boldsymbol{x}_k \leq \boldsymbol{x}_{max}$
$\boldsymbol{u}_{min} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{max}$
ref
Path-following controller
ref
Delta u
Input correction
$\Delta \boldsymbol{u}_k = \ell_u \mathbf{C}(\boldsymbol{x}_{d,k} - \boldsymbol{y}_{m,k})$
ym
u*0
UAV
Measuring sensors
ym

Figure 5.2: Schematically illustration of model correction with a block diagram.

5.2.3 NMPC with Integral Action

We now consider another method of handling unknown disturbances and model errors. Integral action is a well known method of handling offsets in Control Theory. Usually, integral action is used in PID controllers and LQRs. Integral action is implemented adding the total historical state-errors. The OCP solution is therefore changing the inputs to achieve a smaller state deviation respectively. In order to implement integral action in the NMPC, the cost function error state vector is augmented with integral states as discussed in the literature review in Chapter 2. The state-error vector $\tilde{\boldsymbol{x}}^{\psi}$ is concatenated with the integral state vector $\boldsymbol{\zeta} \in \mathbb{R}^4$, forming the new augmented vector $\tilde{\boldsymbol{x}}^{\psi}_a = [\tilde{\boldsymbol{x}}^{\psi} \quad \boldsymbol{\zeta}]^\top \in \mathbb{R}^{n_{\tilde{x}}+n_i}$. The weighting matrices are also concatenated as a result: $\boldsymbol{Q}^{\psi}_{x,a}, \boldsymbol{Q}^{\psi}_{N,a} = diag([q_v \; q_\beta \; q_\alpha \; \boldsymbol{q}^\top_\Theta \; \boldsymbol{q}^\top_\omega \; \boldsymbol{q}^\top_\delta \; \boldsymbol{q}^\top_\zeta])$. Where $\boldsymbol{q}_\zeta$ is the weighting vector of the integrator states respectively. The continuous integral action dynamic in the OCP estimation model in 3.27, is defined as:

$$\dot{\boldsymbol{\zeta}} = \mathbf{C}(\boldsymbol{x}_{nmpc} - \boldsymbol{x}_d) \tag{5.15}$$

$$\boldsymbol{\zeta} = \int_0^t \boldsymbol{C}(\boldsymbol{x}_{nmpc} - \boldsymbol{x}_d)dt \tag{5.16}$$

$$= \begin{bmatrix} \zeta_v & \zeta_\phi & \zeta_\theta & \zeta_\psi \end{bmatrix}^\top, \tag{5.17}$$

where $\boldsymbol{C}$ is given in Equation 5.14, and the initial time step t_0 is given as the initial staring time for the OCP. The integral states are iteratively updated by feedback from the measurements. The initial integral state starting the OCP is given by:

$$\boldsymbol{\zeta}_0 = \int_0^t \mathbf{C}(\boldsymbol{y}_m(t) - \boldsymbol{x}_d(t))dt. \tag{5.18}$$

In order to mitigate the integral action becoming dominating in the NMPC scheme, we utilise a simple integrator-windup strategy. The magnitude-limit of the corresponding integral states is set to $\boldsymbol{I}_{max} = -\boldsymbol{I}_{min} = [9, 3, 3, 3]^\top$. We thus obtain the following NLP scheme:

$$\min_{\boldsymbol{\chi}} \quad \sum_{k=0}^{N-1} \left({\tilde{\boldsymbol{x}}_{a,k}^{\psi}}^{\top} \boldsymbol{Q}_{a,x}^{\psi} \tilde{\boldsymbol{x}}_{a,k}^{\psi} + \boldsymbol{u}_k^{\top} \boldsymbol{Q}_u^{\psi} \boldsymbol{u}_k \right) + {\tilde{\boldsymbol{x}}_{a,N}^{\psi}}^{\top} \boldsymbol{Q}_{a,N}^{\psi} \tilde{\boldsymbol{x}}_{a,N}^{\psi} \tag{5.19a}$$

$$\text{s.t.} \tag{5.19b}$$

$$\boldsymbol{x}_0 = \boldsymbol{y}_m(t_0) \tag{5.19c}$$

$$\boldsymbol{\zeta}_0 = \boldsymbol{f}_{\zeta}^{\phi}(\tilde{\boldsymbol{x}}_a^{\phi}(t_0)) \tag{5.19d}$$

$$\boldsymbol{u}_0 = \boldsymbol{u}(t_0) \tag{5.19e}$$

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) \tag{5.19f}$$

$$\boldsymbol{x}_{min} \leq \boldsymbol{x}_k \leq \boldsymbol{x}_{max} \tag{5.19g}$$

$$\boldsymbol{u}_{min} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{max}. \tag{5.19h}$$

5.3 Pitch-Yaw Attitude NMPC Tuning

This part of the analysis presents the tuning of the disturbance controllers used in this thesis. The original controller is also tuned due to the implemented AoA in the cost function relative to the original controller in (Reinhardt and Johansen, 2019). We consider the tuning- matrices in the given cost functions to obtain a satisfactory tuning. The original controller is only tuned under the influence of a static wind component $\boldsymbol{w}_s^n = [-5, -3, 0]^\top$, due to the uncertainties of the ambient disturbances. Tuning obtained from this isaslo then used throughout the Thesis on all pitch-yaw controllers. The disturbance mitigating controllers are tuned to a less severe ambient condition case than the one presented in Section 5.1.1. We then find appropriate values for the integral weighting matrix, and the model correction behaviour. We can thereby conclude whether the disturbance controllers are effectively increasing robustness of the NMPC framework. All tuning cases are based on the same *Bank to Turn* benchmark manoeuvre. It adequately excites all states commanding both lateral and longitudinal motion simultaneously. The same number of discretisasloation steps and prediction horizon in all tuning cases, $N = 30, T_f = 3s$, are used. Finally, the update-rate of the NMPC remains at $f_{nmpc} = 20Hz$ throughout the entire Thesis.

5.3.1 Pitch-Yaw NMPC

When tuning the pitch-yaw NMPC (original controller), it is important to evaluate which states that are desirable to keep at references. The reference vector is based on trim conditions without ambient disturbances. It will therefore create cases where perfect convergence on all states are impossible. We therefore tune the original controller under the influence of a static wind component only. Considering the cost function in 4.5, $\boldsymbol{Q}_x^\psi, \boldsymbol{Q}_u^\psi$ are assigned values creating an optimal response.

$$\boldsymbol{Q}_x^\psi = diag\left(\begin{bmatrix} q_v & q_\beta & q_\alpha & q_\phi & q_\theta & q_\psi & \boldsymbol{q}_\omega & \boldsymbol{q}_\delta \end{bmatrix}\right) \tag{5.20}$$

$$\boldsymbol{Q}_u^\psi = diag\left(\begin{bmatrix} q_{\dot{\delta}_t} & q_{\dot{\delta}_a} & q_{\dot{\delta}_e} & q_{\dot{\delta}_r} \end{bmatrix}\right) \tag{5.21}$$

Where the subscript of q describes which variable it weights in the OCP. The tuning process is presented in Table 5.2 and 5.3. It is also graphically displayed in Figure 5.3, where the most relevant simulated cases are compared.

$$\Delta_{sim2}^x = 109 \qquad \Delta_{sim3}^x = 112 \qquad \Delta_{sim4}^x = 112 \tag{5.22a}$$

$$\Delta_{sim2}^u = 481 \qquad \Delta_{sim3}^u = 462 \qquad \Delta_{sim4}^u = 470 \tag{5.22b}$$

Figure 5.3 shows the responses of *sim2*, *sim3* and *sim4* respectively. Table 5.2 reveals a significant increase in q_θ, from *sim3* to *sim4*. From previous experience in the Project Thesis, it can be expected that the pitch struggles to converge under rough conditions. Also, the magnitude of pitch-deviations tend to be less than the roll and yaw respectively. Finally, the throttle input deviations are also reduced to prepare the controller for a possible higher drag, making it less costly to apply more throttle. By observing the error-parameters in Equation 5.22, we can see that the difference between *sim3* and *sim4* is small. All simulations in this Thesis therefore use the weighting matrices presented for *sim4*.

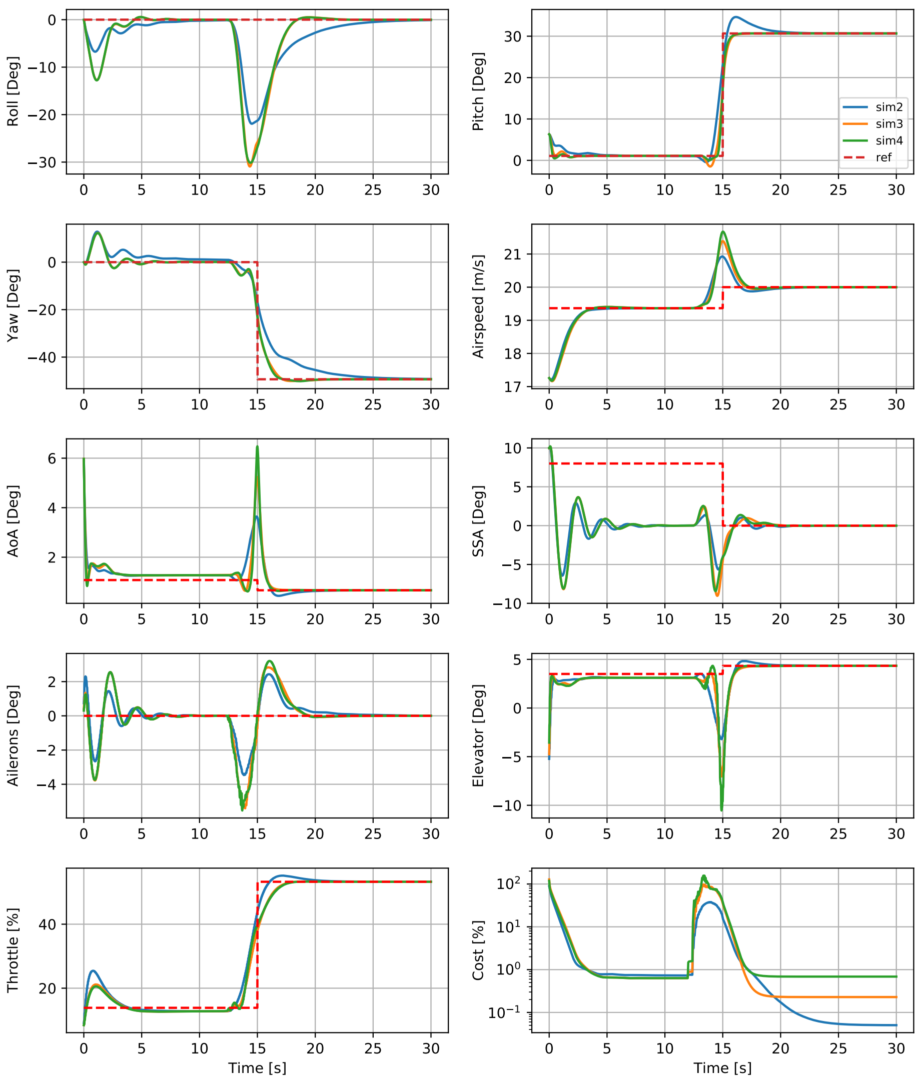


Figure 5.3: Three responses of different tuning matrices for the original NMPC, with a *bank to turn* manoeuvre, under static wind conditions.

Table 5.2: Pitch-yaw tuning parameters.

Sim	q_v	q_β	q_α	q_ϕ	q_θ	q_ψ	$\boldsymbol{q}_\omega^\top$
1.	10^{-3}	10^{-1}	10^{-1}	10^{-1}	10^{-1}	10^{-1}	$[10^{-3}, 10^{-3}, 10^{-3}]$
2.	10^{-3}	10^{-1}	10^{-1}	10^{-1}	10^{-1}	10^{-1}	$[10^{-3}, 10^{-3}, 10^{-3}]$
3.	10^{-3}	10^{-1}	10^{-1}	10^{-1}	4	1	$[10^{-3}, 10^{-3}, 10^{-3}]$
4.	10^{-3}	10^{-1}	10^{-1}	10^{-1}	20	1	$[10^{-3}, 10^{-3}, 10^{-3}]$

Table 5.3: Pitch-yaw tuning parameters.

Sim	q_{δ_a}	q_{δ_e}	q_{δ_r}	q_{δ_t}	$q_{\dot{\delta}_a}$	$q_{\dot{\delta}_e}$	$q_{\dot{\delta}_r}$	$q_{\dot{\delta}_t}$
1.	10^{-3}	10^{-3}	0	10^{-3}	10^{-3}	10^{-3}	0	10^{-3}
2.	10^{-3}	10^{-3}	0	10^{-3}	10^{-1}	10^{-1}	0	10^{-1}
3.	10^{-3}	10^{-3}	0	10^{-3}	0.4	0.4	0	0.4
4.	10^{-3}	10^{-3}	0	10^{-4}	0.4	0.4	0	0.4

5.3.2 Model Correction NMPC

We use the same weighting matrices as concluded in Section 5.3.1. In order to tune the estimation model, we only consider the diagonal gain matrix $\boldsymbol{\ell}_d$ in Equation 5.11. Tuning of $\boldsymbol{\ell}_d = diag(\ell_l, \ell_m, \ell_n, \ell_{v_r})$ is done by implementing plant-model errors, thus evaluating the estimator's ability of correct plant-model regarding these changes. We create a less severe measurement model $\boldsymbol{f}_m$ than the one presented in 5.1.1.

- Mass is changed to 4.2 kg (from the original 3.36 kg).
- Change of mass-center $\boldsymbol{r}^b_{b/cm} = [0.01, 0.0, 0.005]^\top$
- Changed lift- and drag coefficients where we change $C_L^{incline}, C_L^{offset}, C_D^{factor}$ to $\{0.6, 0.9, 1.5\}$ respectively. This is further elaborated in Section 5.1.1.

The disturbance model is tuned to account for offsets, constantly updating the plant-model with disturbance input. Tuning values of $\boldsymbol{\ell}_d$ are given in Table 5.4. They are obtained by focusing on finding values of $\boldsymbol{\ell}_d$ resulting in $\boldsymbol{d}_{k+1} - \boldsymbol{d}_k \approx 0$. We therefore seek a relatively constant behaviour. If the values in the observer tuning mattix are too small, it may never converge. This can be seen for d_v in simulation *sim1*, in Figure 5.4. However, if it have large values, the disturbance might oscillate heavily. Figure 5.4 illustrates the individual disturbance components over the entire simulation, performing the same manoeuvre as in Section 5.3.1. Initially all disturbances are noisy. This is because the moving average method starts being effective at $t = 4\Delta t_{nmpc} = 0.4s$. Even if the value of d_v is decreasing over the simulation, it can be seen that the large increase we did from *sim2* to *sim3* only resulted in slightly more oscillations arount d_v^{sim2}. This implicates that the dynamics of $\dot{V}_r$ tends to need more disturbance input over the simulation. Nevertheless, it seems to stabilise at $d_v \approx -2$. Based on this, we choose *sim2* for our tuned disturbance model.

Sim	Comment
1.	Too fast and oscillating deflections of $\delta_a, \delta_e, \delta_t$. Need higher $\boldsymbol{Q}_u^{\psi}$.
2.	Still too fast input dynamics. The pitch and yaw convergences are significantly slower. Need higher $q_\psi, q_\theta, \boldsymbol{Q}_u^{\psi}$.
3.	To prepare the controller of throttle deviations, we chose q_{δ_t} lower. Also, the pitch tends to create offset under the influence of rough conditions. We therefore choose q_θ significantly higher.
4.	This simulation shows fast convergence with acceptable deflection-rates.

Table 5.4: Model correction NMPC tuning parameters.

Sim	ℓ_l	ℓ_m	ℓ_n	ℓ_{v_r}	Δ^x	Δ^u
1.	0.1	0.1	0.1	0.1	147	665
2.	0.1	0.5	0.07	1	143	753
3.	0.05	0.5	0.05	3	142	775

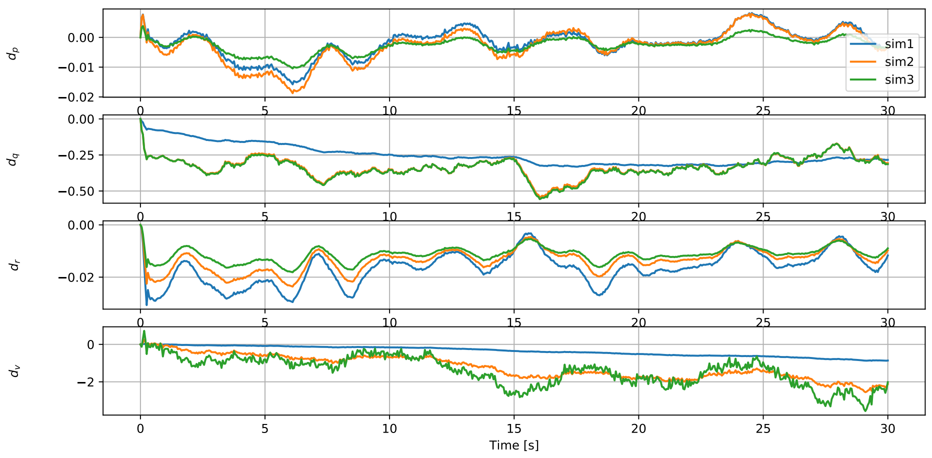


Figure 5.4: Individual disturbance components with varying values of $\boldsymbol{\ell}_d$.

Table 5.5: Input correction NMPC tuning values of $\boldsymbol{\ell}_u$.

Sim	$\ell_{\dot{\delta}_a}$	$\ell_{\dot{\delta}_e}$	$\ell_{\dot{\delta}_r}$	$\ell_{\dot{\delta}_t}$	Δ^x	Δ^u
1.	0.01	-0.01	0	0.05	147	690
2.	0.01	-0.03	0	0.1	146	712
3.	0.01	-0.5	0	0.2	145	754
4.	0.01	-2	0	0.1	144	750
5.	0.01	-5	0	0.1	144	919
6.	0.01	-10	0	0.1	146	1100

Sim	Comment
1.	Slow input correction for throttle and elevator, resulting in small offset mitigation. Need increasing of $\ell_{\dot{\delta}_e}, \ell_{\dot{\delta}_t}$.
2.	Better offset mitigation. However, the throttle starting to oscillate. The airspeed and pitch are still with offsets, we therefore try increasing $\ell_{\dot{\delta}_e}, \ell_{\dot{\delta}_t}$ even further.
3.	Response of both airspeed and pitch is better. Throttle oscillates more heavily. We therefore set $\ell_{\dot{\delta}_t}$ back to 0.1. Also, further increasing $\ell_{\dot{\delta}_t}$ to mitigate pitch-offset.
5.	By increasing $\ell_{\dot{\delta}_e}$, we also introduce elevator oscillations. However, there is still a small pitch-offset.

5.3.3 Input Correction NMPC

We will now tune the following input correction NMPC in 5.2.2. The tuning process is done by evaluating the overall response of the inputs, relative to the ambient disturbances. The input correction works as a P-regulator outside the NMPC, and is therefore expected to struggle correcting large state-offsets. Considering the gain matrix $\boldsymbol{\ell}_u$, we assign values relative to desired input corrections. The system is tested under the same eased ambient disturbances presented in Section 5.3.2. From the tuning process of the input correction NMPC, it can be observed in Figures 5.5 and 5.6 that total offset-elimination is not possible without oscillations on $\delta_a, \delta_e, \delta_t$ respectively. The tuning of this controller is chosen from Table 5.5, for simulation *sim4*.

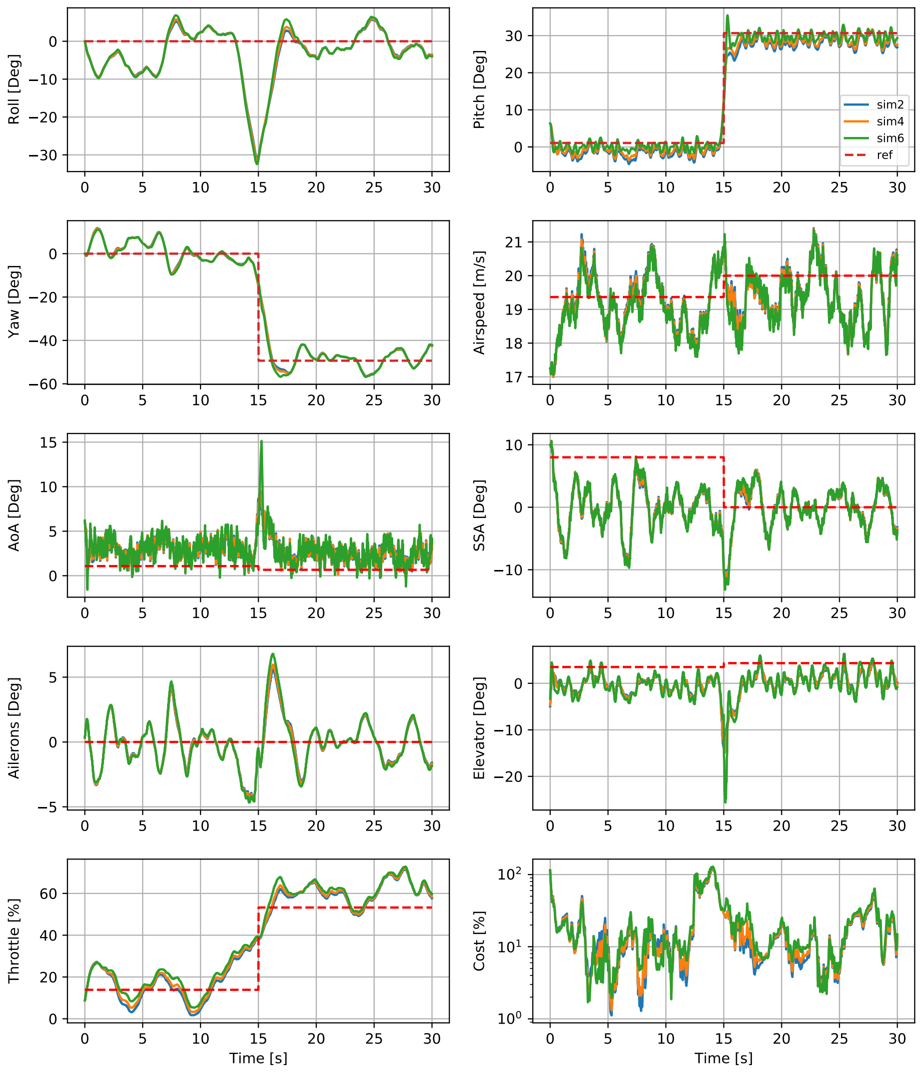


Figure 5.5: Three responses of different tuning configurations of ℓ_u for the input correction NMPC. A *Bank to Turn* manoeuvre under moderate turbulence is displayed with reduced aerodynamic coefficients.

Table 5.6: Tuning of integral action.

Sim	q_{ζ_v}	q_{ζ_ϕ}	q_{ζ_θ}	q_{ζ_ψ}	Δ^x	Δ^u
1.	10^{-6}	10^{-5}	10^{-4}	10^{-5}	150	601
2.	10^{-4}	10^{-4}	10^{-2}	10^{-5}	147	553
3.	10^{-4}	10^{-4}	1	10^{-4}	148	645
4.	10^{-4}	10^{-4}	10	10^{-2}	166	637

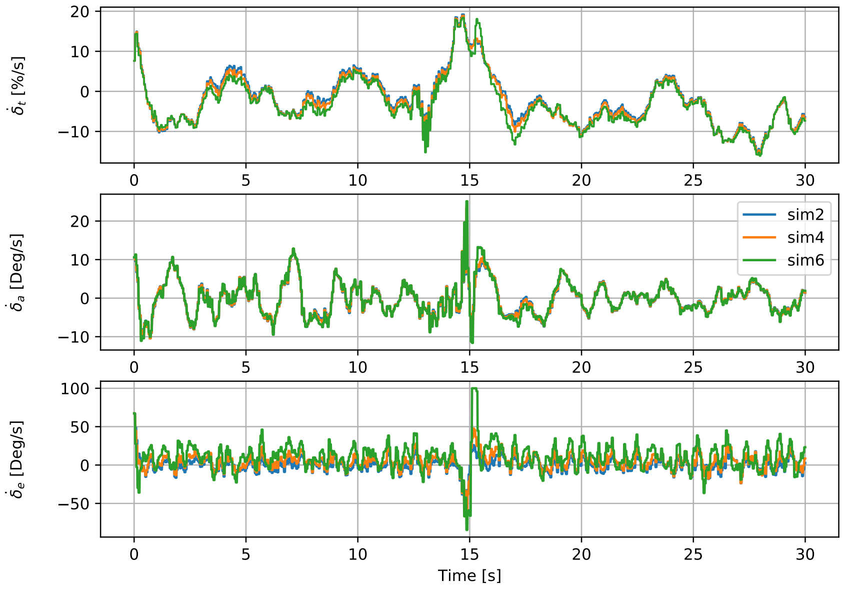


Figure 5.6: Inputs of the input correction NMPCs with varying tuning values of $\boldsymbol{\ell}_u$.

5.3.4 NMPC with Integral Action

Finally, we tune the integral action NMPC by considering the vector $\boldsymbol{q}_\zeta = [q_{\zeta_v}, q_{\zeta_\phi}, q_{\zeta_\theta}, q_{\zeta_\psi}]^\top$, in Section 5.2.3. The offset of the integral defined states is investigated, to determine a good response for the attitude NMPC with augmented integral states. In Table 5.6 and Figure 5.7, we observe the fact that only with large values of q_{ζ_θ}, the pitch-offset is eliminated. The integral state weighting component q_{ζ_θ} has to be large enough to dominate elevator behaviour forcing it to have a bigger offset than previously. We have also chosen the weighting of pitch such that the pitch decreases before the climb to build speed, thus finding an optimal solution forcing pitch convergence. It is then able to keep a relative large airspeed without throttle-rate input throughout the climb. Because of this strong convergence, we choose the weighting corresponding to *sim4*.

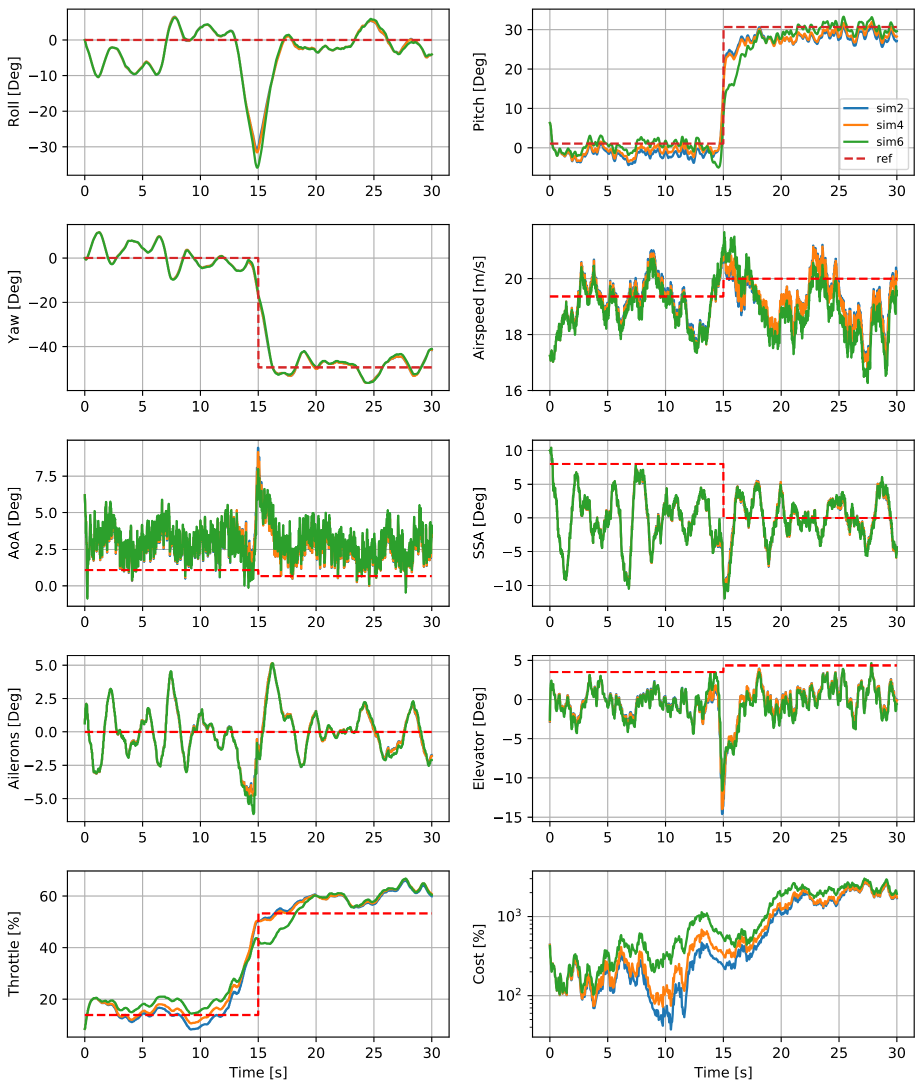


Figure 5.7: Three tuning configurations of $\boldsymbol{q}_\varsigma$ for the attitude NMPC with integral action. A *bank to turn* manoeuvre is performed, under moderate turbulence, with reduced aerodynamic coefficients.

5.4 Pitch-Yaw NMPC Compared to a PID Controller

To inspect the effect of the severe icing conditions in Section 5.1.1, we test the original pitch-yaw attitude NMPC towards the PID controller in Section 4.2. The weighting matrices $\boldsymbol{Q}_x^{\psi}, \boldsymbol{Q}_u^{\psi}$ are obtained from the tuning process in 5.3. We use these matrices throughout this thesis for all the pitch-yaw NMPC configurations. Optimally, we want to design a controller that can be used in multiple designs in varying environments, also under rough conditions. The tuning of the controller is therefore constant, given all UAV applications. The weighting matrices are then obtained to be:

$$\boldsymbol{Q}_x^{\psi} = diag\left(\begin{bmatrix} q_v & q_\beta & q_\alpha & \boldsymbol{q}_\Theta^\top & \boldsymbol{q}_\omega^\top & \boldsymbol{q}_\delta^\top \end{bmatrix}\right) \tag{5.23a}$$

$$\boldsymbol{Q}_u^{\psi} = diag\left(\begin{bmatrix} 0.4 & 0.4 & 0.4 & 0 \end{bmatrix}\right) \tag{5.23b}$$

$$q_v, q_\beta, q_\alpha = \{10^{-3}, 10^{-1}, 10^{-1}\} \tag{5.23c}$$

$$\boldsymbol{q}_\Theta = \begin{bmatrix} 10^{-1} & 2\cdot 10^{1} & 10^{0} \end{bmatrix}^\top \tag{5.23d}$$

$$\boldsymbol{q}_\omega = \begin{bmatrix} 10^{-1} & 10^{-1} & 10^{-1} \end{bmatrix}^\top \tag{5.23e}$$

$$\boldsymbol{q}_\delta = \begin{bmatrix} 10^{-4} & 10^{-3} & 10^{-3} & 0 \end{bmatrix}^\top \tag{5.23f}$$

Where $diag(x)$ denotes the diagonal square matrix with vector x as a diagonal. The vector $\boldsymbol{q}_\Theta$, weights roll, pitch and yaw related to the importance of each state convergence. The NMPC therefore strives to find solutions with low errors on these states respectively.

Initially, we want to evaluate the pitch-yaw NMPC against a PID controller. PID controllers are often the common choice when it comes to small unmanned aerial vehicles. In order to test the controllers, we define a benchmark manoeuvre; The *Bank to Turn* manoeuvre. It excites all states creating an ideal case-study regarding performance under rough conditions. A *Bank to Turn* is when an aerial vehicle turns by exciting roll and pitch at the same time, leading to a turning manoeuvre. After the *Bank to Turn* manoeuvre is finished, the UAV will continue to climb relative to a positive pitch-value. The initial conditions are the same for all simulations in this thesis, and are obtained from initial trim conditions.

$$\boldsymbol{w}_s^n = \begin{bmatrix} -5 & -3 & 0 \end{bmatrix}^\top (m/s) \tag{5.24a}$$

$$V_{r,0} = 17.24\ (m/s) \tag{5.24b}$$

$$\beta_0 = 9.98\ (Deg) \tag{5.24c}$$

$$\alpha_0 = 6.43\ (Deg) \tag{5.24d}$$

$$\boldsymbol{\Theta}_0 = \begin{bmatrix} -0.03 & 6.31 & -0.02 \end{bmatrix}^\top (Deg) \tag{5.24e}$$

$$\boldsymbol{\omega}_{nb,0}^s = \begin{bmatrix} -5.24 & -2.17 & -3.52 \end{bmatrix}^\top (Deg/s) \tag{5.24f}$$

$$\boldsymbol{\delta}_0 = \begin{bmatrix} 9(\%) & 0.83 & -2.05 & 0 \end{bmatrix}^\top (Deg) \tag{5.24g}$$

The references creating the benchmark manoeuvre are a step function activated at 15 seconds:

$$\boldsymbol{x}_{d,k} = \begin{cases} \boldsymbol{x}_{d,0:} & \forall \quad t < 15s \\ \boldsymbol{x}_{d,15:} & \forall \quad t \geq 15s \end{cases} . \tag{5.25}$$

Where the reference steps are obtained from trim conditions as:

$$\boldsymbol{x}_{d,0:} \approx [19.4,\ 8.0,\ 1.1,\ 0.0,\ 1.0,\ 0.0,\ 0.0,\ 0.0,\ 0.0,\ 0.0,\ 3.5,\ 0.0,\ 0.14]^\top \tag{5.26a}$$

$$\boldsymbol{x}_{d,15:} \approx [20.0,\ 0.0,\ 0.0,\ 30.1,\ -49.3,\ 0.0,\ 0.0,\ 0.0,\ 0.0,\ 0.0,\ 4.3,\ 0.0,\ 0.31]^\top . \tag{5.26b}$$

The NLP for the pitch-yaw NMPC (original NMPC) is formulated in Equation 4.11. Figure 5.8 graphically illustrates the pitch-yaw NMPC and the PID controller under icing conditions, with increased mass, under moderate turbulence. First, we consider the PID controller's response on the first time interval $t < 15s$. From Figure 5.8, the pitch angle can be observed to have an offset similar to the NMPC. This offset comes from the drastically changed parameters in the model due to icing. The implemented integral action accumulates as a result of a persistent pitch error. The integral action in the PID controller forces it to add more inputs, leading to the slow approach to the pitch reference. The PID controller's update-rate is 100 Hz compared to the NMPC's 20 Hz, giving it a five times faster update-rate. This makes the output from the PID more noisy. As previously discussed in Section 4.2, the actuator limitations are not implemented other than a simple low pass filter giving it unrealistically high and fast inputs. As can be seen in Figure 5.9. This effect can be mitigated by reducing the proportional gains of the PID. It is therefore able to follow the airspeed reference exceptionally well.

To summarise, the PID controller's performance, reaches the desired references, yet slowly, and under the process of applying very fast and varying inputs respectively. Due to the fast actuator responses we expect that the PID needs more tuning before it is ready for real-time implementation.

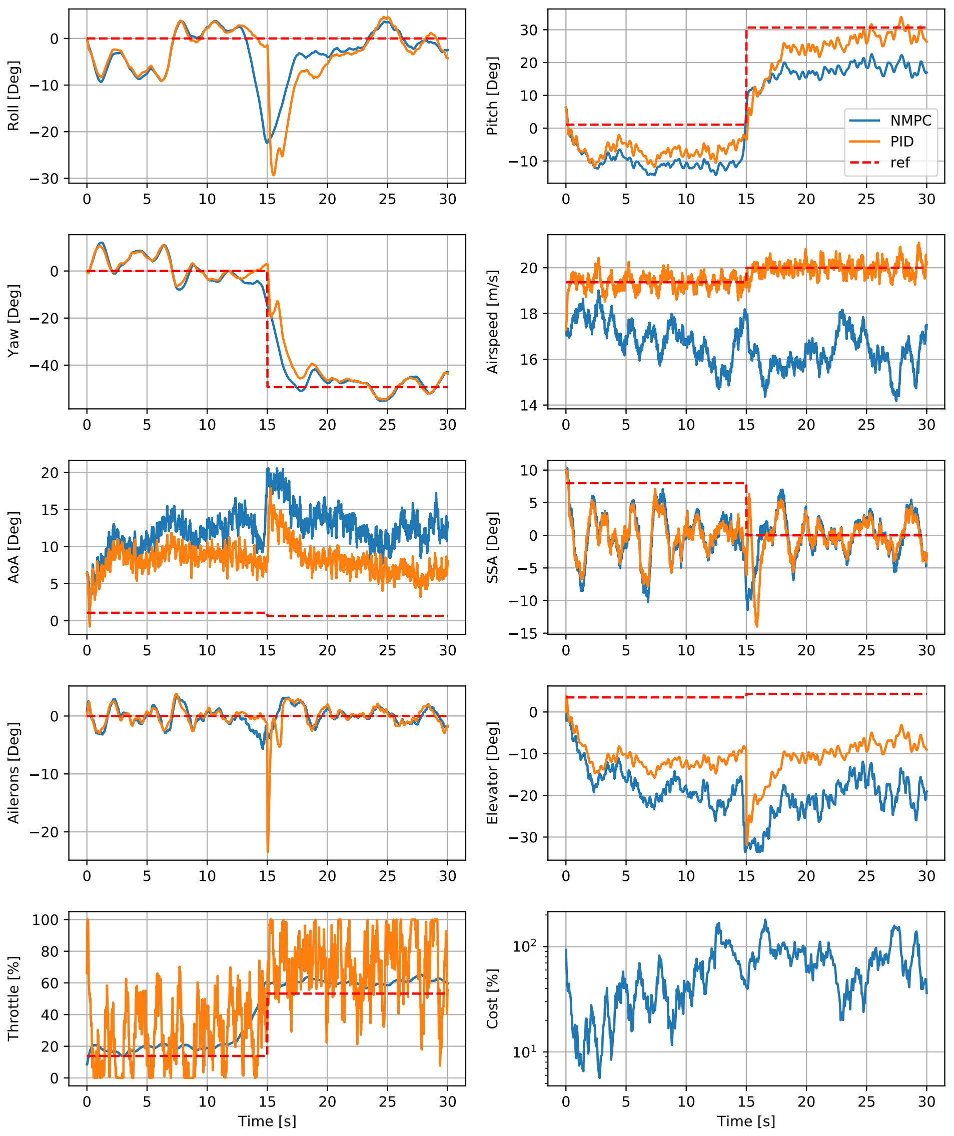


Figure 5.8: Pitch-yaw NMPC and PID controller states with severe case of icing accretions, under moderate turbulence, performing a *Bank to Turn* manoeuvre.

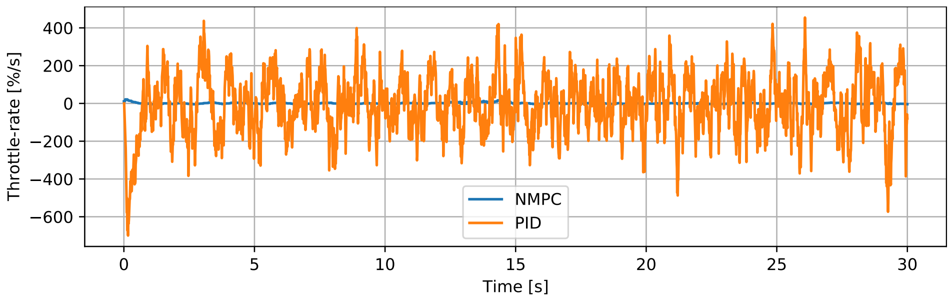


Figure 5.9: Throttle rate of the NMPC and PID controller.

We will now consider the performance of the pitch-yaw NMPC under these extreme conditions in Figure 5.8. The airspeed of the NMPC never reaches the desired value due to the increased drag and mass. From the throttle plot, we see that even if the throttle is persistently above the desired vales, it is not enough to counteract the harsh environment, and bring the airspeed to the desired reference. The OCP is formulated based on the plant-model in Equations 3.27a-3.27e, thus not with the ambient disturbances. It therefore finds input sequences based on these equations, resulting in a convergence of values substantially off the desired pitch and airspeed. The OCP is not tuned to account for these ambient unknown disturbances. This can be verified by considering the cost function, where its moving average can be seen to stabilise at approximately 80, indicating it has reached its objective. The OCP persistently predicts with a less accurate model, finding input sequences that are too low to meet the real unmodeled ambient disturbances. It therefore finds the lowest cost scenario of all the presented states relative to the trim conditions; finally settling on a compromise. The pitch-yaw NMPC shows a rather poor performance under these extreme conditions due to the inaccurate plant-model, and incorrect tuning for this case. We cannot tune the controller to tolerate ambient disturbances due to their random nature. Instead, we introduce model correction, input correction or integral action to account for the disturbances. These scenarios are simulated in Section 5.2.

5.5 Comparisons of Ambient Disturbance Mitigation Controllers

Figure 5.11 graphically displays the performance of the model correction NMPC, the input correction NMPC and the NMPC with integral action under the same extreme conditions as in Figure 5.8. The responses of each method are systematically discussed in this section. Simulation parameters are presented in 5.27a-5.27c.

$$\boldsymbol{\ell}_d = diag(\begin{bmatrix}0.1 & 0.5 & 0.1 & 1\end{bmatrix}) \tag{5.27a}$$

$$\boldsymbol{\ell}_u = diag(\begin{bmatrix}-0.3 & 0.1 & 10 & 0\end{bmatrix}) \tag{5.27b}$$

$$\boldsymbol{q}_\zeta = \begin{bmatrix}10^{-3} & 10^{-3} & 10 & 10^{-1}\end{bmatrix}^\top \tag{5.27c}$$

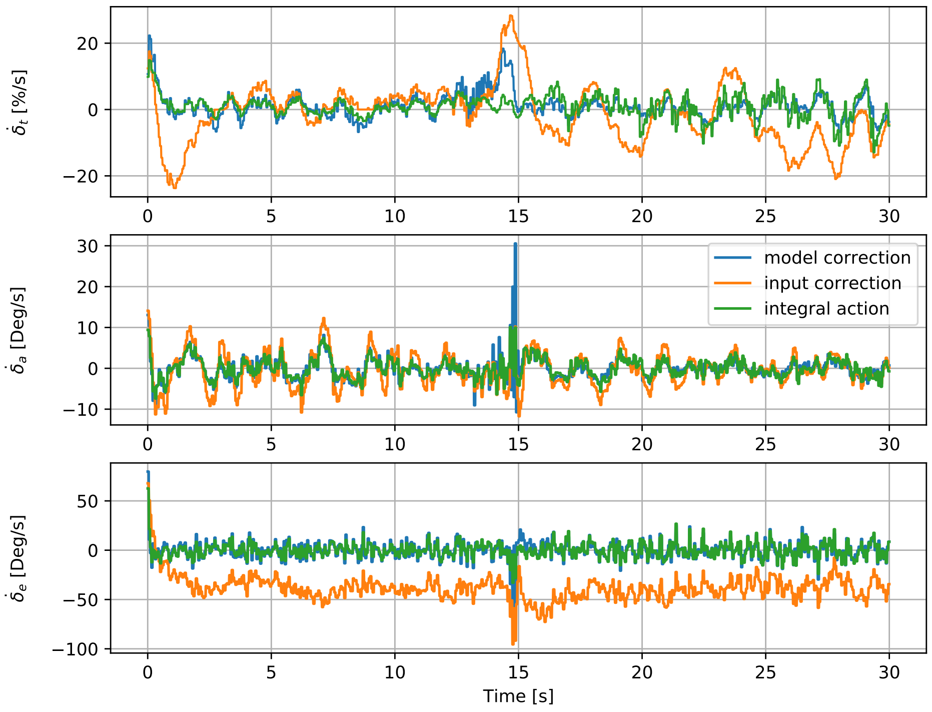


Figure 5.10: Input comparison for model correction NMPC, input correction NMPC and the NMPC with integral action respectively.

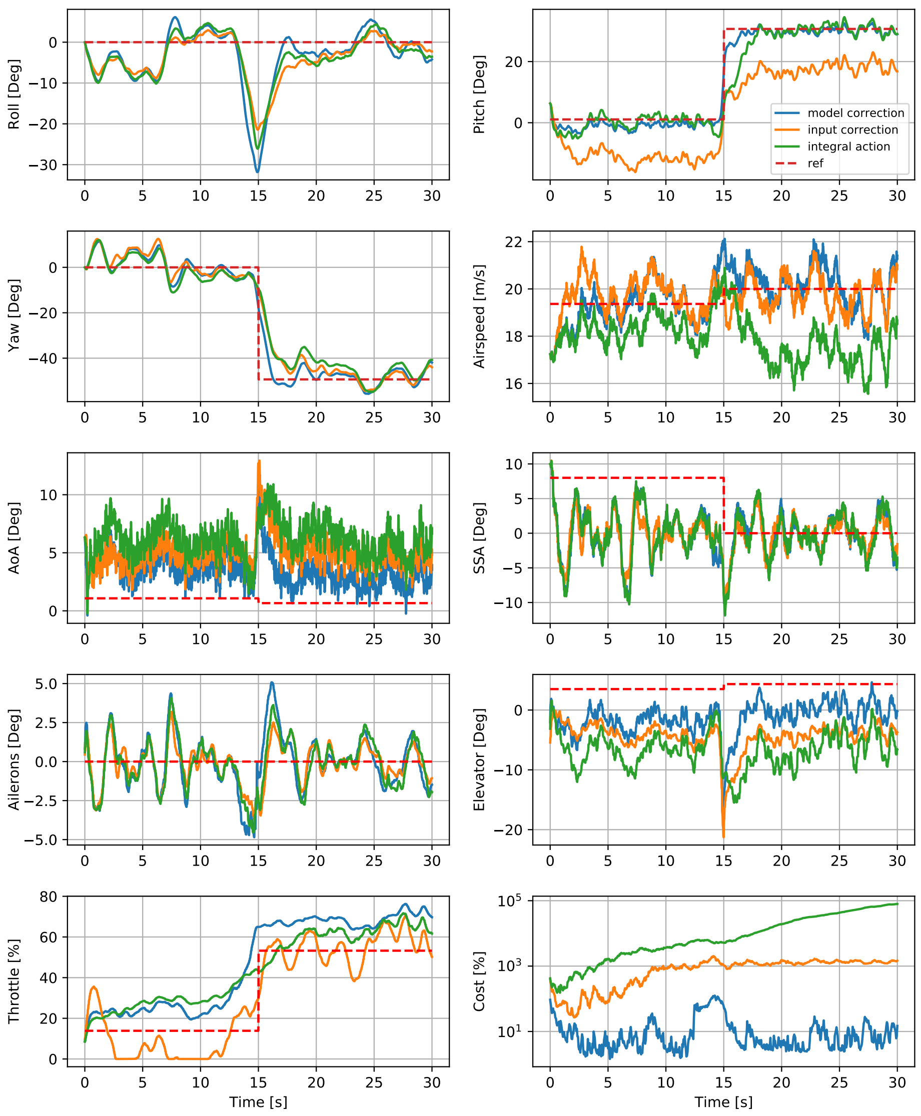


Figure 5.11: Pitch-yaw NMPC comparing the three presented offset-mitigation controller techniques: model correction NMPC (blue), input correction NMPC (orange) and NMPC with integral action (green). The states are plotted under the severe case of icing accretions, in moderate turbulence, performing a *Bank to Burn* manoeuvre.

The model correction NMPC systematically corrects the plant-model mismatch in the NMPC. This unique property takes advantage of the predictability of the NMPC, simultaneously as not affecting the estimations if they are accurate. The in-loop model correction changes the system constraints in the NLP via feedback. When designing the disturbance model in 5.11, we base the framework on the one presented in (Morari and Maeder, 2012). In order to correct purely attitude estimations, we use the angular rate vector. The angular rates are easily accessible from measurements in real-time applications, concurrently as being dynamically coupled to the attitude by an integrator manoeuvre. To form an optimal accurate estimation model, we also corrects the airspeed with a disturbance term. From this, it can be assumed that both the attitude and the velocity models improve in-flight. The remaining untampered states $\dot{\beta}, \dot{\alpha}, \dot{\mathbf{\Theta}}$, will implicitly be corrected through these disturbances. This creates a fully defined corrected plant-model that is more accurate towards the actual system behaviour. Predictions of the NMPC will as a result become more accurate, and the overall performance will improve. We now observe the throttle, pitch and airspeed dynamics in Figure 5.11. First, we see that the throttle starts increasing together with the other controller modifications at $t \approx 12s$. The predictive capabilities of the NMPC make it violating current references, building speed for the upcoming pitch and airspeed climb. Due to the disturbance input in the model corrected NMPC, its mathematical predictions resolve that it needs more throttle than the two other modifications. The airspeed enlargement is considerable higher as a result. The outcome is that the model corrected NMPC has an impressively fast convergence to the pitch reference. The previous large pitch-offset seen in the regular NMPC in Figure 5.9 is now entirely gone. The disturbance vector has corrected the model so that it can evaluate the ambient dynamic disturbances in the model, thus counteracting them in the predictions of the NMPC. This represents an elegant and simple solution of handling plant-model mismatch and ambient disturbances.

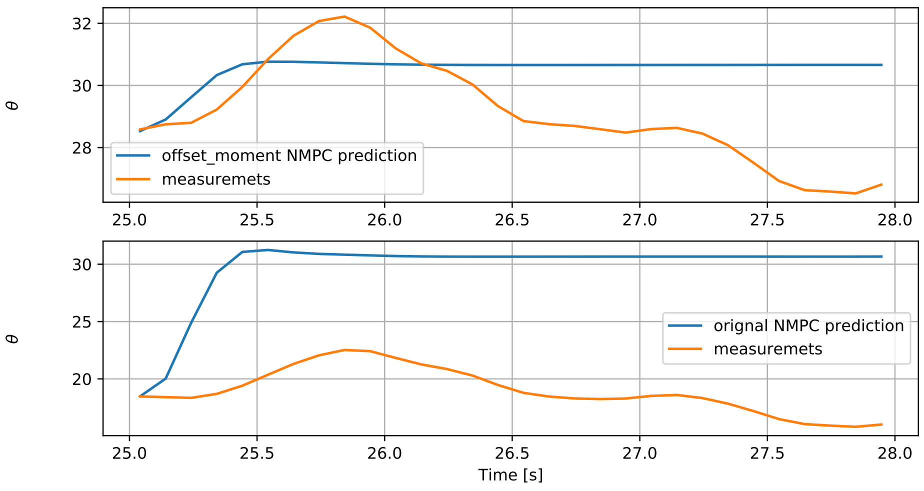


Figure 5.12: Prediction of the NMPC with model correction and the original NMPC at $t = 25s$, compared to the actual behaviour of the UAV (measurements).

However, there are problems with this method. We assumed the disturbance model to converge to a mean value after a period of flight. In the same set of assumptions we also assume it to be sufficient to add a linear disturbance term to the moment and airspeed over the entire prediction horizon T_f. Assuming moment and relative airspeed represent a good place to insert model correction, there are still issues solely related to the disturbance model in 5.11. Due to the nonlinear dynamics of both the moment and airspeed, the disturbance term might not be an actual constant, but rather a variable defined by the system states. The predictions deteriorate over the horizon as a result. This can be seen in Figure 5.12.

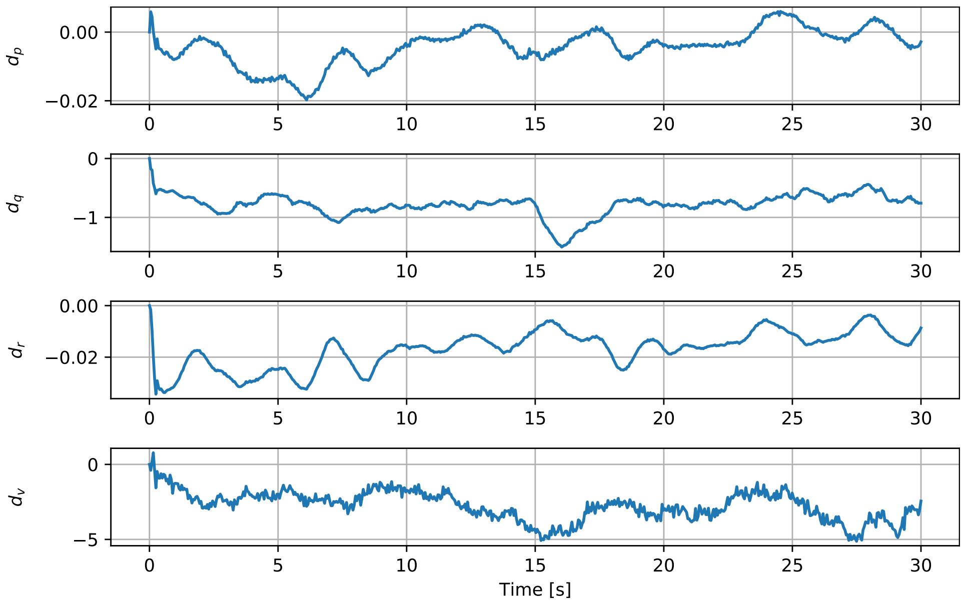


Figure 5.13: Individual disturbance values of the components in $\boldsymbol{d}_k$, over the simulation.

We apply a worst case scenario based on the icing case in 5.1.1, that implements drastic dynamical changes to the simulations. If the lift curve for instance changes in flight, the disturbance term will not find a constant value. Also, the model dynamics tends to behave differently relative to the attitude of the aircraft. Figure 5.12 illustrates this effect by comparing the predictions of the NMPC in both the regular case (Figure 5.8), and the plant-model corrected NPMC. It can clearly be seen that the overall estimation of the model correction NMPC is better. Nevertheless, its accuracy deteriorates over the prediction horizon. Therefore, it could be argued the prediction horizon could be shorter, still achieving a similar performance. Considering the disturbance vector $\boldsymbol{d}_k = [d_p \; d_q \; d_r \; d_v]^\top$, we can observe the individual disturbance values in Figure 5.13. A moving average method is applied to the disturbance vector in Equation 5.11 working as a low-pass filter. This is to prevent oscillations, and help the disturbances converge towards an approximately small constant area. By observing Figure 5.13, we draw the conclusion that all the presented disturbance values remain within a bounded area and are relatively stable. The tuning process in Appendix 5.3.2, specifically investigated whether the disturbance-rate stagnated or increased. Small values of $\boldsymbol{\ell}_d$ result in slow or no convergence of these disturbances respectively. It is necessary to find a compromise, since large values create oscillations. Disturbance component $d_r \in \mathbb{R}$ in Figure 5.13 drops from 0 to approximately -0.029. It thereby proceeds to increase and to find a relatively constant operating range. This is typical P-controller behaviour. A stable operating range at approximate $t = 2.5s$ can also be observed. Further tuning of the disturbance model might increase performance even more. There are more complex model estimation methods that create estimation models on a sophisticated scale; A Kalman Filter makes a covariance matrix, representing the disturbance-change relative to state values. This is iteratively improved by a stochastic model giving disturbance terms based on historical state-errors at state values. This method might be relevant when real-flight disturbance and process noise are implemented in the model.

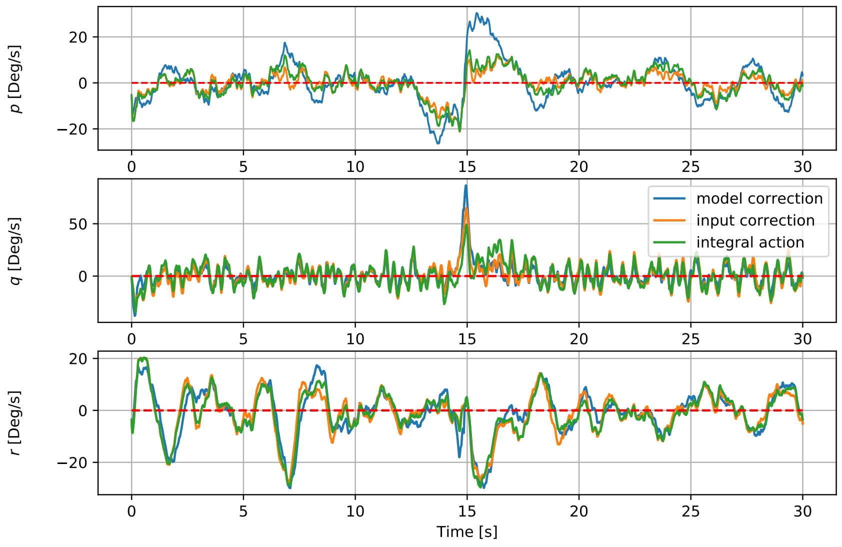


Figure 5.14: Angular velocity of the model correction NMPC, input correction NMPC, and the NMPC with integral action.

The input correction NMPC directly uses the feedback from the sensors ($\boldsymbol{y}_m$) to modify the input applied to the plant. Its input correction therefore bypasses the NMPC. If the model in the NMPC deviates significantly from the real system dynamics, the input correction might have to be large to dominate the predicted behaviour of the NMPC; constantly counteracting the calculated response from the NMPC. This can cause several issues. It is therefore important to find a compromise. In figure 5.11, we can see that the pitch-offset is slightly smaller than in the original NMPC's response (Figure 5.8). The altered elevator input is not corrected enough to converge to the reference. With higher weighting on the elevator the controller becomes unstable, due to the coupled pitch climb and reduction of airspeed. The throttle is also oscillating, to keep the airspeed close to the reference.

Optimal input sequence from the NMPC is therefore constantly corrected, resulting in an non-cooperative behaviour. It is not possible to choose $\boldsymbol{\ell}_u$ in such a way that references are reached. Now consider the input correction vector $\Delta\boldsymbol{u}$ in Equation 5.13. Figure 5.15 illustrates the individual input correction components. The input corrections are a P-controller, only assigning values as a function of the error and the proportional gain matrix $\boldsymbol{\ell}_u$. Oscillations of the throttle are therefore due to the oscillating behaviour of Δu_v in Figure 5.15. The values of Δu_θ are as predicted, close to constant, as a function of the pitch-offset in Figure 5.11. To entirely eliminate this offset an integrating behaviour in the input correction could be utilised, making it a PI-controller. Performance might also improve with a thorough tuning process (further increase $\ell_{\dot{\delta}_e}$). Nevertheless the non-cooperative behaviour makes this controller less relevant. The main purpose of this contribution is to create a robust controller scheme. It is therefore difficult to keep this controller and guarantee robustness in a wide application range. We therefore disregard this controller for further analysis.

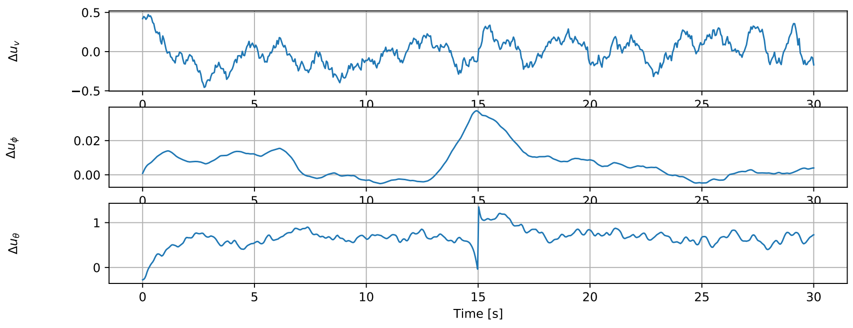


Figure 5.15: Individual input correction propagation $\Delta\boldsymbol{u}_k$, for the input correction NMPC.

Integral action is implemented in the OCP, augmenting the state error vector with an integral state vector in Equation 5.15. It is added an extra cost based on the total predicted airspeed and the Euler Angle state deviation respectively. It can be seen in Figure 5.11, that the NMPC with integral action starts the simulation $(0.5s \leq t \leq 3s)$ with a considerable pitch offset. When the integral state representing pitch deviation q_θ has accumulated a large enough value, it slowly eliminates the offset. The pitch convergence to the new step reference, is also slow compared to the other two control methods. This is partly because the OCP has to be dominated by the integral states in order to converge. Because of the airspeed accumulated integral value, it is not able to find a solution of increasing the airspeed before the climb. This forces it to do the pitch step slower, mitigating total deviation as much as possible. Observing the cost function of the NMPC with integral action reveals its ongoing increase. This can implicate that the integral states are weighted to heavily. Differently from the model correction method, the NMPC with integral action do not have model correction and therefore predicts with a poor plant-model. The integral action predictions in the OCP are based on the presumed accumulation of integral states. These accumulated values forces the OCP to find solutions violating other state references, in order to achieve convergence of the cost function. The UAV will as a result, have an enforced convergence of the states affected by the integral action ($V_r, \mathbf{\Theta}$). It serves as a kind of guarantee of convergence. This introduces the problem of completely overshadowing the other desired cost related states. By observing Figure 5.11, we can see that AoA, elevator, and throttle all experience large deviations as a result of converging to the integral defined states. Some of these deviations might be necessary, but in some cases it can introduce the problem of getting too big deviations on important states (i.e AoA). It can be difficult to tune the integral effect to an extent that is functional for all cases. For example, a small oscillation on the throttle can be observed, due to the constant desire of reaching the constant airspeed reference. It is constantly striving towards an impossible task due to the big ambient disturbances. The cost function of the NMPC with integral action will therefore continue to grow to a large value where the integral states overshadow the other states. The OCP struggles to find a value that satisfies the constraints, at the same time as the rapidly increasing integral state vector. This is partly the tendency we see in the cost function plot in Figure 5.11. Magnitudes of integral states weighted in the cost function are a direct function of the weighting vector $\boldsymbol{q}_\zeta \in \mathbb{R}^4$, prediction horizon T_f, model-mismatch, reference error and the integrator limits we define in Section 5.2.3. However, performance of this controller might improve with a proper tuning sequence.

5.6 Optimal Robust Broad-Spectre Pitch-Yaw Attitude NMPC

Based on the controller performances in Section 5.5, we can see that both the model correction NMPC and the NMPC with integral action have their separate advantages. An optimal controller should have the model correction from the model correction NMPC, and the integral action from the integral action controller. This controller will thus have more accurate predictions, concurrently keeping the convergence guarantee of the integral action. Such a controller could possibly tolerate an even broader range of applications. In this section, we implement such a controller and compare it to the model correction NMPC and the NMPC with integral action.

$$\boldsymbol{q}_\zeta^{NMPC} = \begin{bmatrix}10^{-6} & 10^{-3} & 1 & 10^{-2}\end{bmatrix}^\top \tag{5.28}$$

In Figure 5.16, we compare the controllers with model correction and integral action towards a ”full offset NMPC” with integral action and model correction. As discussed in Section 5.2, both the model correction and integral action alone can have separate problems. The integral states could in some cases accumulate large values, becoming dominating in the OCP scheme. Because a model correction method is implemented, we reduce the weighting of the integral actions in the full offset correction NMPC. The new values are given in Equation 5.28. Considering the full offset NMPC’s cost function, it can be seen that its value-increase stagnates. The OCP is then finding trim-flight input sequences that is stable and constant. The improved model of the full offset NMPC, allows the predictions with the integral actions to become more accurate. This makes the pitch convergence of the full offset NMPC to be slower than the model correction NMPC. For comparisons we consider the error-parameters:

$$\Delta^x_{offset\_moment} = 144 \qquad \Delta^x_{offset\_NMPC} = 159 \qquad \Delta^x_{integral} = 201 \tag{5.29a}$$

$$\Delta^u_{offset\_moment} = 928 \qquad \Delta^u_{offset\_NMPC} = 906 \qquad \Delta^u_{integral} = 760. \tag{5.29b}$$

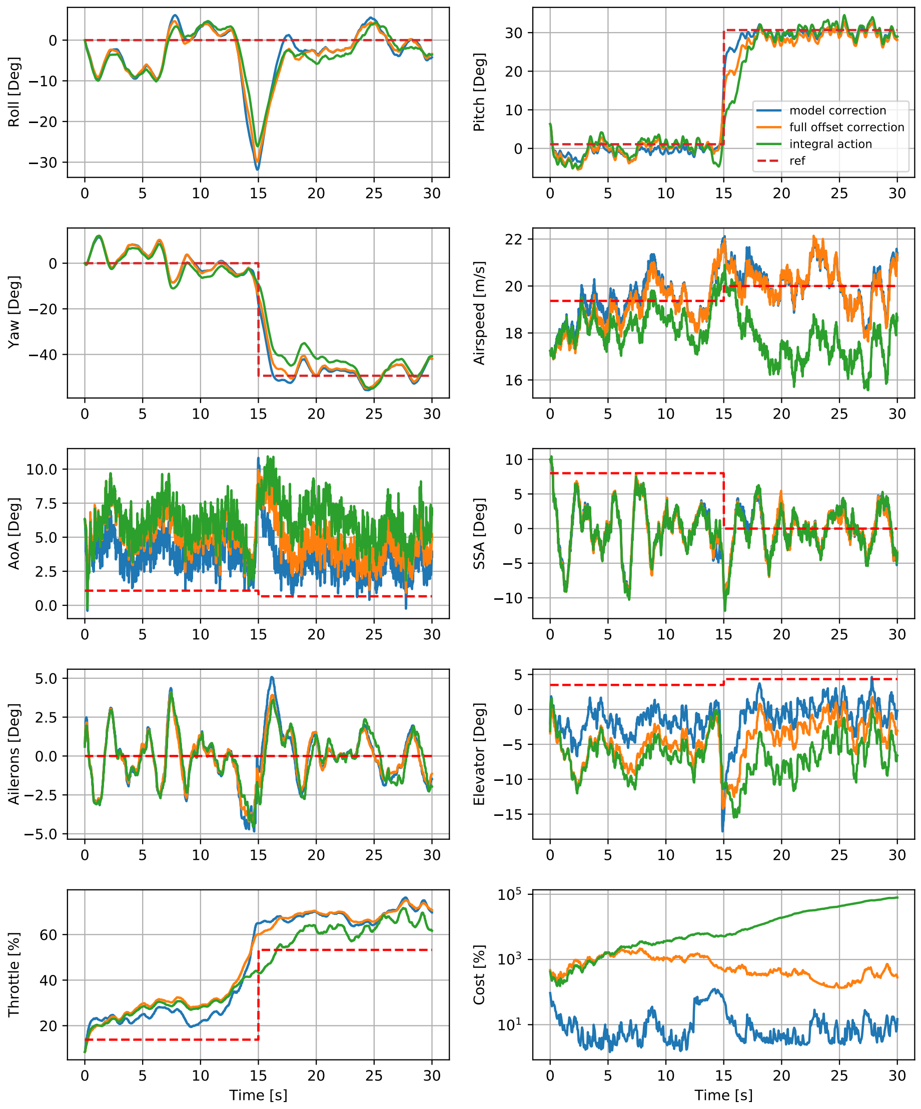


Figure 5.16: Pitch-yaw NMPC comparing: model correction NMPC (blue), full offset NMPC with integral action and model correction (orange), and NMPC with integral action (green). The states are plotted under the severe case of icing accretions, under moderate turbulence, performing a *Bank to Turn* manoeuvre.

Equation 5.29 shows that the model correction NMPC has a lower error-parameter. The implemented integral action slows down the dynamics, leading to a slower response to the step pitch reference. Arguably, the integral action slows the controller, but ensures convergence in severe cases. This further strengthens the controller's ability of handling a broad range of harsh conditions. It can also be seen that the model correction NMPC has a higher energy usage, respectively. Additional plots of the angular-rate and input are given in Appendix D.1, in Figure D.1-D.2. The full offset correction NMPC proves to be stable and reliable in theoretical severe, icing conditions.

5.7 Optimal Offset Correction NMPC With a Roll-Pitch NMPC Formulation

To extend the robustness analysis of the full offset correction NMPC proposed in Section 5.6, we now evaluate a lower level attitude controller with a different objective function and reference vector, towards the severe icing case. This controller is based on the controller proposed in (Reinhardt et al., 2020). To achieve higher level control objectives (path-controller), autopilots tend to rely on a lower level control of climb and turn rates. A roll-pitch controller will purposely be a lower level attitude controller than the pitch-yaw controller. The pitch-yaw controller is penalising all Euler Angles so that both attitude and heading (ψ) are ensured. The pitch-yaw controller also penalises state deviation from the trim states, making the controller obtaining solutions relative to the trim conditions. To formulate the lower level roll-pitch controller, it is fascinating to investigate the performance weighting the roll-pitch attitude as well as airspeed. This controller will then purposely have references that all can be satisfied. The OCP therefore do not have to find a compromise in between. We try the same algorithm architecture proposed in Section 5.6, for the new lower level roll-pitch attitude controller. The controller therefore finds solutions complying with the same constraints, but with a different objective.

By closer inspecting the reduced error vector in Equation 4.9, we conclude that it is invariant to rotation about $\boldsymbol{k}^n$ and thus independent of yaw-angles. For the roll-pitch NMPC with integral action, we define the integral states with the same architecture as in Equation 5.15. We want to implement the integral action on the reference attitudes and airspeed respectively (ϕ, θ, V_r). Integral dynamic is then obtained as:

$$\dot{\boldsymbol{\zeta}}^{\phi} = \tilde{\boldsymbol{x}}^{\phi} \tag{5.30}$$

$$\boldsymbol{\zeta}^{\phi} = \boldsymbol{f}_{\zeta}^{\phi}(\tilde{\boldsymbol{x}}^{\phi}(t)) = \int_0^t \tilde{\boldsymbol{x}}^{\phi} dt, \tag{5.31}$$

where the OCP initialising integrator state is defined by:

$$\boldsymbol{\zeta}_0^{\phi} = \boldsymbol{f}_{\zeta}^{\phi}(\tilde{\boldsymbol{x}}^{\phi}(t_0)). \tag{5.32}$$

The following NLPs for the original roll-pitch controller, and the roll-pitch controller with the full offset correction are then obtained as:

$$\min_{\boldsymbol{\mathcal{X}}} \quad \sum_{k=0}^{N-1}\left(\tilde{\boldsymbol{x}}_k^{\phi\top}\boldsymbol{Q}_x^{\phi}\tilde{\boldsymbol{x}}_k^{\phi}+\boldsymbol{u}_k^{\top}\boldsymbol{Q}_u^{\phi}\boldsymbol{u}_k\right)+\tilde{\boldsymbol{x}}_N^{\phi\top}\boldsymbol{Q}_N^{\phi}\tilde{\boldsymbol{x}}_N^{\phi} \tag{5.33a}$$

$$\text{s.t.} \tag{5.33b}$$

$$\boldsymbol{x}_0=\boldsymbol{y}_m(t_0) \tag{5.33c}$$

$$\boldsymbol{u}_0=\boldsymbol{u}(t_0) \tag{5.33d}$$

$$\boldsymbol{x}_{k+1}=\boldsymbol{f}\left(\boldsymbol{x}_k,\boldsymbol{u}_k\right) \tag{5.33e}$$

$$\boldsymbol{x}_{min}\leq\boldsymbol{x}_k\leq\boldsymbol{x}_{max} \tag{5.33f}$$

$$\boldsymbol{u}_{min}\leq\boldsymbol{u}_k\leq\boldsymbol{u}_{max}, \tag{5.33g}$$

$$\min_{\boldsymbol{\mathcal{X}}} \quad \sum_{k=0}^{N-1}\left(\tilde{\boldsymbol{x}}_{a,k}^{\phi\;\top}\boldsymbol{Q}_{a,x}^{\phi}\tilde{\boldsymbol{x}}_{a,k}^{\phi}+\boldsymbol{u}_k^{\top}\boldsymbol{Q}_u^{\phi}\boldsymbol{u}_k\right)+\tilde{\boldsymbol{x}}_{a,N}^{\phi\;\top}\boldsymbol{Q}_{a,N}^{\phi}\tilde{\boldsymbol{x}}_{a,N}^{\phi} \tag{5.34a}$$

$$\text{s.t.} \tag{5.34b}$$

$$\boldsymbol{x}_0=\boldsymbol{y}_m(t_0) \tag{5.34c}$$

$$\boldsymbol{\zeta}_0=\boldsymbol{f}_{\zeta}^{\phi}(\tilde{\boldsymbol{x}}_a^{\phi}(t_0)) \tag{5.34d}$$

$$\boldsymbol{u}_0=\boldsymbol{u}(t_0) \tag{5.34e}$$

$$\boldsymbol{d}_0=\boldsymbol{f}_d(\boldsymbol{d}_{z:z-3},\boldsymbol{x}_z,\boldsymbol{y}_{m,z}) \tag{5.34f}$$

$$\boldsymbol{x}_{k+1}=\boldsymbol{f}\left(\boldsymbol{x}_k,\boldsymbol{u}_k,\boldsymbol{d}_k\right) \tag{5.34g}$$

$$\boldsymbol{d}_{k+1}=\boldsymbol{d}_k \tag{5.34h}$$

$$\boldsymbol{x}_{min}\leq\boldsymbol{x}_k\leq\boldsymbol{x}_{max} \tag{5.34i}$$

$$\boldsymbol{u}_{min}\leq\boldsymbol{u}_k\leq\boldsymbol{u}_{max}. \tag{5.34j}$$

Where the augmented error-state vector $\tilde{\boldsymbol{x}}_{a,k}^{\phi}=[\tilde{\boldsymbol{x}}_k^{\phi\top}\quad\boldsymbol{\zeta}^{\phi\top}]^{\top}\in\mathbb{R}^8$ is with the implemented integral action on the parameterised attitude and airspeed respectively. The corrected model estimation is implemented identically as in the pitch-yaw controller in Section 5.2.1. The full offset correction roll-pitch NMPC therefore has the same model correction as previously, with new integral action dynamics. Finally, the augmented weighting matrices $\boldsymbol{Q}_{a,x}^{\phi}=\boldsymbol{Q}_{a,N}^{\phi}\in\mathbb{R}^{8\times8}$, and the original weighting matrices $\boldsymbol{Q}_x^{\phi}=\boldsymbol{Q}_N^{\phi}\in\mathbb{R}^{4\times4}$, are thus obtained from tuning.

5.7.1 Tuning of Roll-Pitch NMPC

We tune the roll-pitch NMPC in a similar methodical way as in Section 5.3. It is desired to obtain roll, pitch and airspeed convergence with realistically fast inputs. Considering the weighting matrices of the roll-pitch NMPC to have the following form: $\boldsymbol{Q}_x^{\phi}=diag([q_v,\boldsymbol{q}_{\Gamma}^{\top}])$, $\boldsymbol{Q}_u^{\phi}=diag([q_{\dot{\delta}_t},q_{\dot{\delta}_a},q_{\dot{\delta}_e},q_{\dot{\delta}_r}])$, Table 5.7 gives the corresponding tuned values. As earlier, we expect the pitch to deviate at severe icing conditions. It can therefore be seen that *sim3* has a higher weighting of pitch deviation.

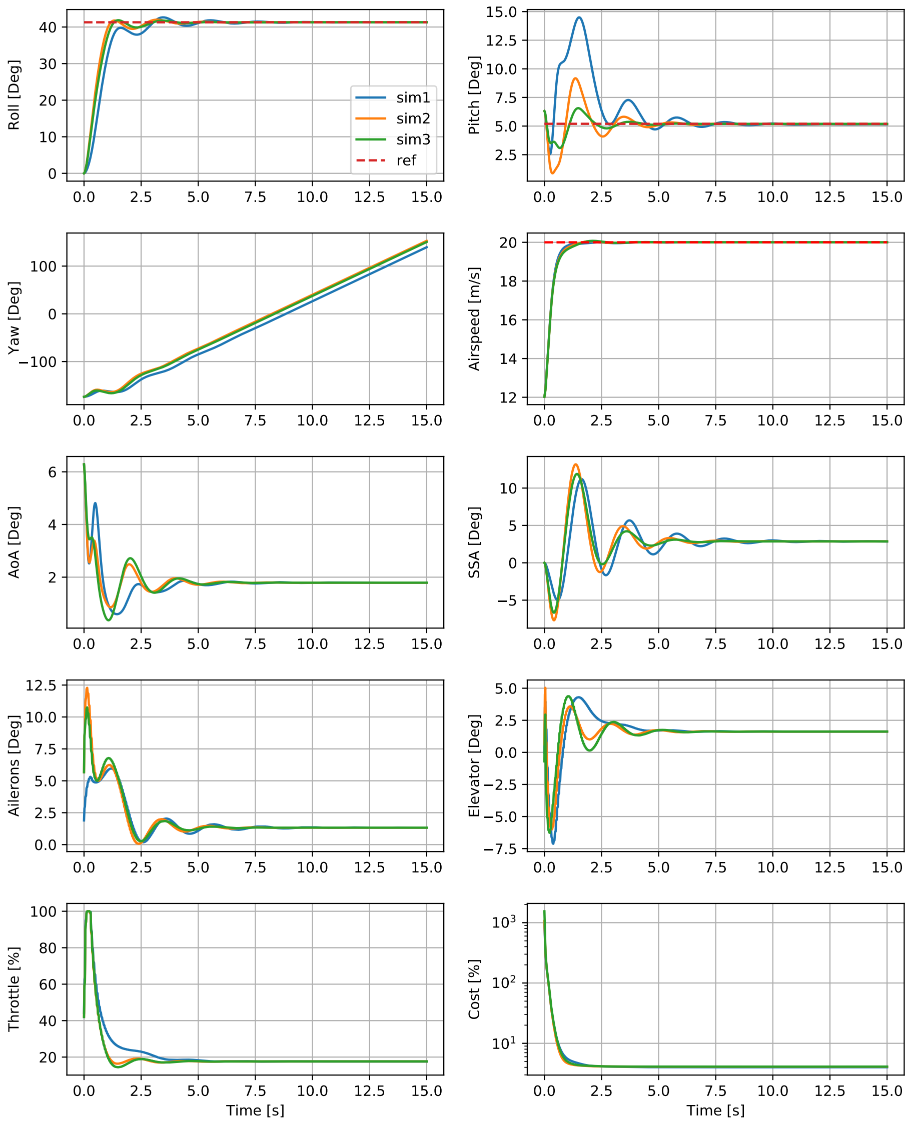


Figure 5.17: Two roll-pitch controller versions with (orange) and without (blue) offset correction. Also, includes the original controller without icing conditions (green) in moderate turbulence.

In Figure 5.17, we observe that the increase of pitch-weighting in the cost function leads to a faster pitch and elevator response respectively. We therefore settle with the performance of *sim3*, and choose the following weighting matrices.

Table 5.7: Roll-pitch NMPC tuning parameters.

Sim	q_v	$\boldsymbol{q}_\Gamma^\top$	$\boldsymbol{q}_{\dot{\delta}}$
1.	10^{-1}	$[10^{-1}\ 10^{-1}\ 10^{-1}]$	$[10^{-1}\ 10^{-1}\ 10^{-1}]$
2.	10^{-1}	$[10^{0}\ 10^{0}\ 10^{0}]$	$[10^{-1}\ 10^{-1}\ 10^{-1}]$
3.	10^{-1}	$[10^{1}\ 10^{0}\ 10^{0}]$	$[10^{-1}\ 10^{-1}\ 10^{-1}]$

$$\boldsymbol{Q}_x^\phi = diag\left(\begin{bmatrix}10^{-1} & 10^{-1} & 10^{-1} & 10^{-1}\end{bmatrix}\right) \tag{5.35a}$$

$$\boldsymbol{Q}_u^\phi = diag\left(\begin{bmatrix}10^{-1} & 10^{-1} & 10^{-1} & 10^{-1}\end{bmatrix}\right) \tag{5.35b}$$

5.7.2 Tuning of Offset Correction Roll-Pitch NMPC

When tuning the integrator action for the roll-pitch NMPC, we consider the input dynamics of the UAV and offset-elimination capabilities. The estimator remains unchanged from Section 5.3.2, due to the same model constraints. Finally, we implement integral action in the new cost function in 5.34a, thus obtaining the values for $\boldsymbol{q}_\zeta^\phi \in \mathbb{R}^4$ with a similar process as in Section 5.3.4. We therefore take the liberty of not plotting this tuning process. This gives the following integral action weighting:

$$\boldsymbol{q}_\zeta^\phi = \begin{bmatrix}10^{-5} & 10^{2} & 10^{0} & 10^{0}\end{bmatrix}^\top \tag{5.36}$$

$$\boldsymbol{\ell}_d = \begin{bmatrix}0.1 & 0.5 & 0.1 & 1\end{bmatrix} \tag{5.37}$$

5.7.3 Offset Correction NMPC tested in a Roll-Pitch NMPC Scheme

We are now ready to compare the new controller with the proposed offset correction architecture from 5.6 to the roll-pitch controller without integral action and model correction. The new reference is constant and will lead the UAV to going in circles.

$$V_{r,d} = 20[m/s] \tag{5.38a}$$
$$\phi_d = 41.76[Deg] \tag{5.38b}$$
$$\theta_d = 8.5[Deg] \tag{5.38c}$$
$$\boldsymbol{\Gamma}_d(\phi_d, \theta_d) = \begin{bmatrix}-0.1478 & 0.6585 & 0.7379\end{bmatrix}^\top \tag{5.38d}$$
$$V_{r,0} = 12.0[m/s] \tag{5.38e}$$
$$\theta_0 = 4.22[Deg] \tag{5.38f}$$
$$\beta_0 = 6.55[Deg] \tag{5.38g}$$
$$\delta_{t,0} = 44.3[\%] \tag{5.38h}$$
$$\boldsymbol{\omega}^s_{nb} = \begin{bmatrix}6.31 & -174 & 0.63\end{bmatrix}^\top (Deg/s) \tag{5.38i}$$
$$\delta_{a,0} = 1.9[Deg] \tag{5.38j}$$
$$\delta_{e,0} = 0.1[Deg] \tag{5.38k}$$
$$\delta_{r,0} = 0[Deg] \tag{5.38l}$$

Figure 5.18 graphically illustrates the states for the original roll-pitch NMPC given by NLP 5.33 and the roll-pitch offset correction NMPC with the NLP 5.34, under icing conditions. By observing the yaw angle in Figure 5.18, an approximately linear increase as a result of excited roll and pitch angles can be observed. The lower level controller is not considering the yaw-dynamics in the objective function, and will therefore be able to increase freely. The original roll-pitch NMPC can still be observed to experience severe pitch-offsets. Even with significantly less states weighted in the cost function, extreme conditions create pitch-offsets under large plant-model mismatches. The OCP finds the optimal input sequence and that is too small to overcome the severe conditions. By using the same framework for the full offset correction NMPC as presented in Section 5.6, we can clearly observe the offset to be severely mitigated. The disturbance input to the plant-model aids the OCP to find a solution that it is beneficial to excite more input to achieve the desired state values. The cost function of the full offset correction NMPC is also observed to be converging. This monotone behaviour proves that even with integral weighting, the controller is stable. In fact, the cost function for the offset correction NMPC is seen to render lower values than the original. This further implicates that the poor plant-model for the original NMPC scheme deviates significantly from the measured output.

Finally, we observe the elevator dynamic of the full correction NMPC. It is slightly more noisy than the original NMPC, due to the weighted integral states. In real flight, we must expect slow actuator exciting, due to the real delay between applied voltage and applied force on the UAV. Aileron, elevator and throttle are assumed to give instant forces in this model. This can possibly affect the performance of the algorithm. Nevertheless, the disturbance input constantly correcting the model, might help mitigating some of this effect. It can therefore be argued that the integral action is too heavy weighted in this particular case. It could be expedient to rely more on the model correction, relative to the integral action.

From the robustness analyses performed in Section 5.6 and 5.7, we conclude the combination of model correction and integral action prove to be an effective way of handling unknown plant-model mismatches and ambient disturbances. We perform a time-study in Section 5.8 to further investigate the possibility of real-time implementation.

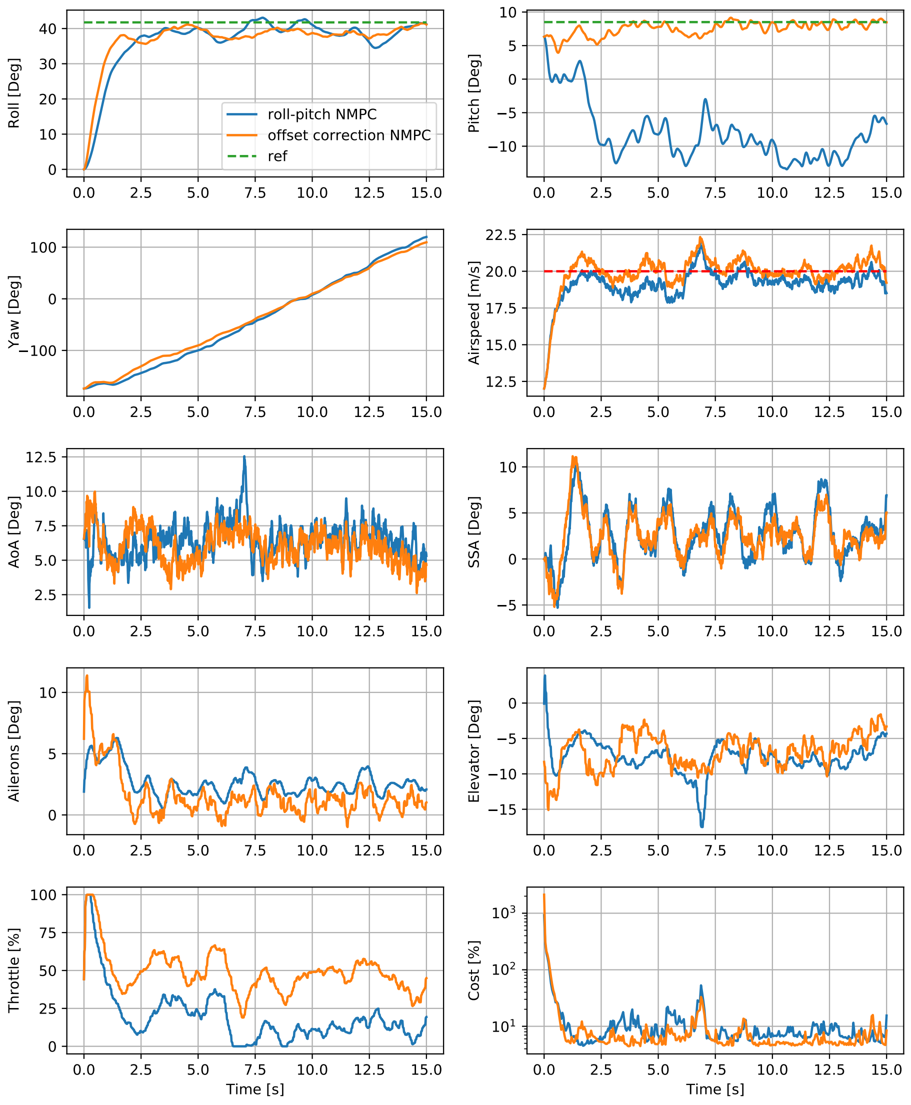


Figure 5.18: Roll-pitch NMPC (blue) compared towards a full offset correction roll-pitch NMPC (orange). The states are plotted under the severe case of icing accretions, in moderate turbulence, performing a turning manoeuvre.

Name	Dell Optiplex 7070 Micro
Platform	Ubuntu 18.04.4 LTS
Memory	32 GB
Processor	Intel® Core^{TM} i7-8700 CPU 3.20 GHz $\times$ 12
Graphics	Intel® UHD Graphics 630 (Coffeelake 3x8 GT2)
GNOME	3.28.2
OS type	64-bit
Disk	469,4 GB

Table 5.8: Dell Optiplex specifications.

5.8 Computational Time-Study

In order to ensure realisation of the controllers presented in Section 5.2, we perform a time-study. This time study is to determine whether the controllers can be implemented in a real-time system. Purposely, it could validate the proposed controller's ability of finding solutions efficiently. Their stability and performance with varying number of discretization steps are also considered. As previously mentioned in Section 3.5, the fast coupled dynamics of the UAV introduces high demands of computational power. The UAV presented in this paper (*Skywalker X8*) will be mounted with the on-board micro-controller Odroid XU-4 (Figure 5.19). The Odrioid XU-4 is considered to be a strong micro controller with a low weight. It is therefore an ideal candidate to test this algorithm on. Because of limited access to an Odroid XU-4, the time-study is performed using a *Dell Optiplex 7070*. The specifications of the computer are given in Table 5.8. We will then be able to present a controller that is both robust towards disturbances and real-time realisable on a theoretical level.

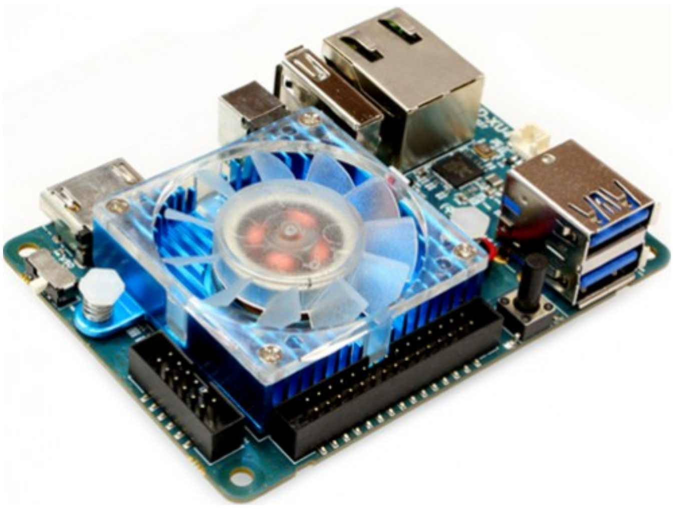


Figure 5.19: Odroid-XU4 micro controller (UK, 2020)

Table 5.9: Time-study of the three proposed offset correction NMPCs with varying number of discretization steps, under constant update rate and horison step size.

Controller	N	T_f[s]	t_{peak}[ms]	Δ^x	Δ^u
Original NMPC	15	1.5	6.79	227	583
	30	3.0	19.8	193	702
	50	5.0	21.7	187	829
	70	7.0	30.4	184	945
Model correction NMPC	15	1.5	7.14	184	999
	30	3.0	14.7	160	893
	50	5.0	26.3	167	875
	70	7.0	35.8	172	879
NMPC with integral action	15	1.5	8.30	210	691
	30	3.0	14.5	201	760
	50	5.0	25.7	202	825
	70	7.0	33.1	202	888
Full offset correction NMPC	15	1.5	7.85	190	918
	30	3.0	13.80	162.9	906
	40	4.0	19.4	161.0	913
	50	5.0	25.9	165	923
	70	7.0	31.5	164.0	943

We include a solver-time requirement $t_{max} = \frac{1}{f_{nmpc}}$, representing the maximum allowed OCP-solver time. This is thus a function of the update-rate of the NMPC. It is crucial that the solver finds a solution to the OCP with a substantial margin before next optimisation starts. In a real-time application, the NMPC will have several time-delays between components. There will be a time delay from the sensors feeding real-time data to the micro-controller. Also, it can safely be assumed that the actuators will have a delay of power output relative to the signal input. For simplicity, we simply consider the OCP solving time. Based on the knowledge that the controller needs a safety margin, we can make a rough estimate whether the controller is able to be implemented for real-time applications.

OCP solver times tend to vary for nonlinear optimisation problems, due to the constraints having multiple solutions. In theory, only the first input of the calculated input sequence is applied to the plant, while the rest is discarded. Often times, the presumed discarded predicted input values are used in the solving algorithm. They are used as initialisers for the NLP making it locate a feasible initial point, starting the optimisation problem. Situations where several local minimas are present, the OCP also tends to have a slower solver time testing local minimas to find the global optimum.

The process of choosing an adequate number of discretization steps that ensures stability and computational efficiency can be a rather complex endeavour. To simplify this process, we consider only significant values of discretization steps $N \geq 15$. However, this topic is thoroughly considered in (Grüne and Pannek, 2016). We start the time-study by investigating the controllers stability and simulation time at a constant NMPC time-step

($\Delta t_{nmpc} = 0.1s$) and update-rate ($f_{nmpc} = 20Hz$). A desirable prediction horizon based on the presented simulation setup, can be found. Controller stability is affected by its update rate, as well as the number of discretization steps. Too few discretization steps, give a piece-wise grid of constant inputs with a coarse sampling rate based on a plant-model. Fast UAV dynamics with ambient disturbances can accumulate large errors based on this coarse input grid. The controller becomes more reliant of the plant-model predictions if a small number of discretisation is used. However, OCP solving time tends to be quicker. At the same time, large N gives more predictability, and allows the UAV to positioning it self before acting. It is therefore important to choose N adequately so that the NMPC algorithm yields stability and computational efficiency. In Table 5.9 all three controllers experience higher computational peaks (t_{peak}) relative to the increasing number of control discretization steps respectively. Considering the relation between OCP-solver time and error parameter values, it seems that $N = 30$ is the optimal choice based on Table 5.9. We therefore investigate the OCP-solver times as a grid over the entire $30s$ simulation in Figure 5.20.

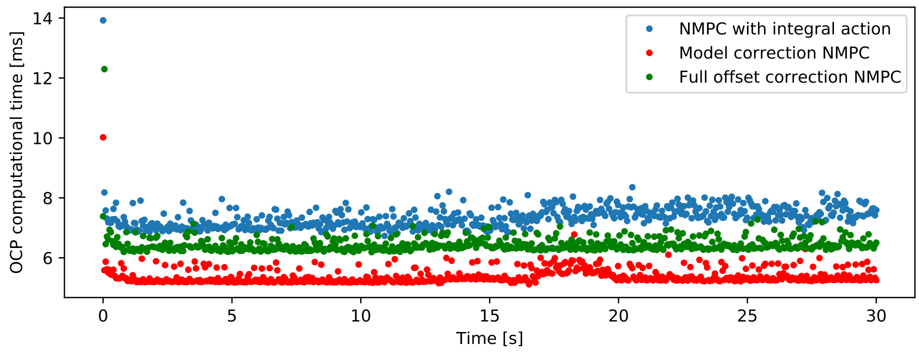


Figure 5.20: OCP-solving time over the entire simulation for the Pitch-yaw attitude NMPCs presented in Section 5.5.

Since the entire solving scheme is based on a computer dependent on heat accumulation, air flow, available sources etc; these might be sources of error. As expected, all controllers have the highest solving-times at the starting point of the simulation ($t = 0$). We also see, all OCP solver times are clustered along a thick line throughout the simulation after the initialisation. The integral action NMPC has a broader clustering-line than the two other controllers. This can be traced to an inaccurate model, giving poor predictions, meaning the OCP initialisation is further off the optimal solution. It can clearly be seen that we have a solid margin to the maximum simulation time $t_{max} = 50ms$. The individual expected component delay (fundamental theory) is considered in (Grüne and Pannek, 2016)[p.313-316]. Because we do not have the ability to test the real system, it is sufficient to base our analysis initially on a solid solver-time margin.

We now implement the full offset correction algorithm proposed in this thesis on the Odroid XU4. We can further strengthen our time-study ensuring real-time implementation of the controller.

5.9 Chapter Summary

This Chapter explores several methods of implementing features in attitude NMPCs to guarantee a robust controller. The pitch-yaw model correction NMPC under icing conditions prove to start the simulations ($t < 2s$) with a similar offset as the NMPC without model correction. It takes approximately $2.5s$ before the disturbance corrected model is efficiently employed in the NMPC scheme. The model correction NMPC therefore converges to the attitude and airspeed references, exploiting the NMPC framework with more accurate predictions due to the disturbance modelling in Equation 5.11. The prediction accuracy of the pitch dynamic is depicted in Figure 5.12. The predictions of the model corrected NMPC are clearly enhanced, but over a longer horizon span, deteriorate quickly. Nevertheless, the model correction NMPC has the quickest convergence rate of all presented controllers in this Thesis, and proves that a simple model correction method increases robustness towards plant-model mismatch and unknown ambient disturbances.

The attitude NMPC with implemented integral action also converges to the desired attitude references. However, it struggles to converge to the airspeed as a result. It has a generally slow convergence as well as problems stabilising the cost function. Because of the large plant-mismatches in the plant-model, the integral states keep increasing. The integral action NMPC shows the ability of ”forcing” the OCP finding solution that makes the user-defined attitude references converge. Integral action in the NMPC is therefore a convergence-enforcement method on the integral defined states. This allows the user to strengthen the UAV’s ability of converging to certain states, also enforcing the robustness towards plant-model mismatches and unknown ambient disturbances.

The input correction NMPC alters the optimal input from the OCP based on reference offset of certain states directly correlated to the inputs. This method therefore does not cooperate with the NMPC scheme, and quickly gives oscillating inputs. Also, it does not converge to the pitch-reference under the severe icing conditions. We therefore discard the controller due to low performance and an overall bad cooperative framework.

We then propose a broad-range full correction offset NMPC algorithm that uses the desired attributes of model correction and integral action to account for plant-model mismatch and unknown ambient disturbances. This model has a slightly slower convergence than the model correction NMPC, due to the implemented integral action. The cost function of this controller stabilises and all defined states converge to the desired references. This controller proves to be robust towards a theoretical icing case. Under the same icing-condition, the same controller algorithm architecture is tested using a different lower level attitude controller. This is to determine whether it could be used in several controller architectures. We compare the lower level attitude controller towards a similar lower level attitude controller based on our full offset correction framework. It is found that the full correction algorithm ensures convergence in a similar method as in the pitch-yaw controller framework. The attitude NMPC without offset correction struggles to converge and ended up with a large pitch offset and a small airspeed offset.

The time-study reveals that the integral action tends to increase the OCP solver time, due to the increased state-vector dimension and a generally larger OCP. Nevertheless, with the optimal number of discretization steps (N=30), the solver-time of the full offset correction proves to have a significant margin towards the maximum solver time. Therefore, We conclude the controller to be ready for real-time implementation.

Chapter 6

Conclusion

This Master Thesis has conducted a robustness analysis of a pitch-yaw attitude NMPC and a roll-pitch attitude NMPC, based on the controllers in (Reinhardt and Johansen, 2019) and (Reinhardt et al., 2020) respectively. Nonlinear Model Predictive Controllers hinge on a mathematical model and use optimisation-techniques to find an optimal input sequence based on the model constraints. In cases where this plant-model is deviating significantly due to ambient factors, the performance might be considerably affected. UAVs are often used in arctic conditions where turbulence and added research equipment affect the physical plant, and thereby possibly the plant-model. A severe icing test scenario is made, based on the effect of iced air-foils discussed in (Hann et al., 2017)(Dalmau, 2018) and (Winter, 2019). Some disturbance mitigating methods are then tested through a benchmark manoeuvre, under these icing conditions, investigating the effect of plant-model mismatch. We then propose a full offset correction attitude NMPC algorithm that methodically uses model correction and integral action to account for ambient disturbances. Both reference convergence and OCP-simulation time are discussed, bringing this algorithm ready to be implemented on the Odroid XU-4 and in the *Skywalker X8*. The full correction offset NMPC shows promising results, and as a next step, needs to be evaluated in-flight to demonstrate its capabilities in experiments.

Chapter 7

Future Work

This Master Thesis has conducted robustness analysis of several attitude NMPCs, and proposed a robust NMPC framework towards ambient disturbances and plant-model mismatches. This Chapter highlights remaining work of the project and interesting topics to consider in the future.

7.1 Adaptive Tuning

Throughout this thesis it has been discussed how plant-model mismatch and ambient disturbances can affect an NMPC's robustness. Large plant-model mismatch and severe conditions, can lead to significant state to reference offsets. We have also discussed that tuning for such unknown environmental impacts are not possible, due to the random and varying attributes of these disturbances. A UAV will have a broad spectre of manoeuvres that may need different tuning parameters to perform optimally. For future developing of the proposed algorithm in this Thesis, it can be interesting to investigate an adaptive tuning full offset correction NMPC framework. An adaptive tuning NMPC will be able to choose varying tunings for certain flight situations. Ultimately, this might lead to a lower energy consumption as well as better performance in pre-defined flight paths. We can for example consider a simple steady state trim flight. The UAV's objective is to find a stable low energy attitude and airspeed complying with the references. It is then able to perform better in certain situations. An adaptive tuning method allows a more aggressive NMPC tuning for certain in-flight manoeuvres. The controller will recognise manoeuvres based on the references and thus change tuning parameters more appropriately. There are several ways of implementing adaptive tuning in an NMPC framework. Commonly, it is often considered varying number of discretisation steps (N) and objective functions. In (Turki et al., 2018), it is considered a single-input-single-output (SISO) MPC with adaptive tuning of the number of discretization steps (N). This method ensures stability and can further optimise prediction horizon for the UAV control algorithm. However, for our control design it can be more relevant to consider adaptive weighting matrices as in (Bagheri et al., 2011). Ultimately an adaptive offset correction NMPC will be able to account for unknown ambient disturbances as well as plant-model mismatches. Also, because of tuning attributes that are pin pointed to an exact manoeuvre, it will handle these better.

7.2 More Sophisticated Disturbance Dynamic

We have in this Thesis proved that the disturbance term in Equation 5.11 gave better prediction for the NMPC under the influence of unpredictable environmental disturbances. However, it was also found that due to the constant disturbance term over the prediction horizon, the prediction accuracy quickly deteriorated. For future work, it could be interesting to evaluate a more sophisticated disturbance modelling. There are several methods to be considered. One could be giving it stochastic predictability. It will then map disturbance behaviour based on state values and previous measuring errors corresponding to these. This is not far from how a *Kalman Filter* works. Finally, the disturbance can be altered throughout the prediction horizon also based on a stochastic model. Ultimately, this might lead to even more accurate predictions, and further increase the UAV controller's robustness and performance.

7.3 Experimental Testing

We have tested several NMPC architectures in this Thesis. In order to determine whether the full correction offset NMPC is able to handle real-time flight, experimental flights should be considered. This will increase knowledge of real-time delays, tuning of controller, and potential real-time plant-model mismatches.

Bibliography

Airelectronics, . X8 flying wing URL: `https://www.airelectronics.es/products/x8_brochure.pdf`.

Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., Diehl, M., 2019. Casadi – a software framework for nonlinear optimization and optimal control. Mathematical Programming Computation 11, 1–36. doi:`10.1007/s12532-018-0139-4`.

Bagheri, P., Mardanlou, V., Fatehi, A., 2011. Multiple model predictive control of multivariable ph process using adaptive weighting matrices. doi:`10.3182/20110828-6-IT-1002.02940`.

Beard, R.W., McLain, T.W., 2012. Small Unmanned Aircraft: Theory and Practice. volume 2. Princeton University Press.

Bemporad, A., Morari, M., 2007. Robust model predictive control: A survey. volume 245. pp. 207–226. doi:`10.1007/BFb0109870`.

Borrelli, F., Morari, M., 2007. Offset free model predictive control, in: 2007 46th IEEE Conference on Decision and Control, pp. 1245–1250.

Dalmau, E.S., 2018. Study of the effects of in-flight icing on fixed wing uav's aerodynamic performance.

Egeland, O., Gravdahl, T., 2002. Modeling and Simulation for Automatic Control. volume 2. Marine Cybernetics.

Fitzpatrick, P., 2013. Calculation of thrust in a ducted fan assembly for hovercarft. Hovercraft Club of Great Britain .

Fortuna, J., Ferreira, F., Gomes, R., Ferreira, S., Sousa, J., 2013. Using low cost open source uavs for marine wild life monitoring - field report .

Foss, B., Heirung, T.A.N., 2016. Merging Optimization and Control.

Foundation, P.S., 2018. Python. URL: `https://www.python.org/downloads/release/python-368/`.

Gryte, K., Hann, R., Alam, M., Roh, J., Johansen, T.A., Fossen, T.I., 2018. Aerodynamic modeling of the skywalker x8 fixed-wing unmanned aerial vehicle .

Grüne, L., Pannek, J., 2016. Nonlinear Model Predictive Control Theory and Algorithms Second Edition. Springer.

Hann, R., Wenz, A., Gryte, K., Johansen, T.A., 2017. Impact of atmospheric icing on uav aerodynamic performance, in: 2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS), pp. 66–71.

HSL, . http://www.hsl.rl.ac.uk/. [Online; accessed 15-Dec-2019].

Insider, B., 2020. Commercial unmanned aerial vehicle (uav) market analysis – industry trends, forecasts and companies. https://www.businessinsider.com/commercial-uav-market-analysis?r=USIR=T. [Online; accessed 27-May-2020].

Ltd, S.T.C., 2020. URL: `http://skywalkermodel.com/en/76.html#inspire-1-pro-overview-s4`.

Maeder, U., Borelli, F., Morari, M., 2009. Linear offset-free model predictive control. Automatica 45, 2214 – 2222.

Malkapure, H.G., Chidambaram, M., 2014. Comparison of two methods of incorporating an integral action in linear quadratic regulator. 3rd International Conference on Advances in Control and Optimization of Dynamical Systems (2014) 47, 55–61.

Morari, M., Maeder, U., 2012. Nonlinear offset-free model predictive control. Automatica 48, 2059 – 2067.

Nocedal, J., Wright, S.J., 2006. Numerical Optimization Second Edition. Springer.

Reinhardt, D., Coates, E.M., Johansen, T.A., 2020. Hybrid control of fixed-wing uavs for large-angle attitude maneuvers on the two-sphere, in: 2020 IFAC 21st World Congress.

Reinhardt, D., Johansen, T.A., 2019. Nonlinear model predictive attitude control for fixed-wing unmanned aerial vehicle based on a wind frame formulation. Centre for Autonomous Marine Operations and Systems, Department of Engineering Cybernetics, NTNU , 10.

Ruscio, D., 2013. Model predictive control with integral action: A simple mpc algorithm. Modeling, Identification and Control: A Norwegian Research Bulletin 34, 119–129. doi:`10.4173/mic.2013.3.2`.

Ryan, J.C., Hubbard, A.L., Box, J.E., Todd, J., Christoffersen, P., Carr, J.R., Holt, T.O., Snooke, N., 2014. Uav photogrammetry and structure from motion to assess calving dynamics at store glacier, a large outlet draining the greenland ice sheet .

Róbert, S., 2009a. Modelling of dynamical systems. Review of the Air Force Academy , 350–361.

Róbert, S., 2009b. Stochastic noises affecting dynamic performances of the automatic flight control system , 350–361.

Stadheim, P.A., 2019. Robustness analysis of a nonlinear model predictive controller for a fixed-wing uavs , 1 – 46.

Stevens, B.L., Lewis, F.L., Johnson, E.N., 2016. AIRCRAFT CONTROL AND SIMULATION DYNAMICS, COTROLS DESIGN, AND AUTONOMOUS SYSTEMS.

Turki, M., Langlois, N., Yassine, A., 2018. An analytical tuning approach for adaptive mpc parameters applied to ltv siso systems, in: 2018 Annual American Control Conference (ACC), pp. 1534–1539.

UK, O., 2020. Odroid xu4. URL: `https://www.odroid.co.uk/hardkernel-odroid-xu4/odroid-xu4`.

Verschueren, R., Frison, G., Kouzoupis, D., van Duijkeren, N., Zanelli, A., Novoselnik, B., Frey, J., Albin, T., Quirynen, R., Diehl, M., 2019. acados: a modular open-source framework for fast embedded optimal control. arXiv preprint URL: `https://arxiv.org/abs/1910.13753`, `arXiv:1910.13753`.

Verschueren, R., Frison, G., Kouzoupis, D., van Duijkeren, N., Zanelli, A., Quirynen, R., Diehl, M., 2018. Towards a modular software package for embedded optimization, in: Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (NMPC).

Wächter, A., Biegler, L.T., 2006. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Mathematical Programming 106, 25–57.

Wenz, A., Johansen, T.A., Cristofaro, A., 2016. Combining model-free and model-based angle of attack estimation for small fixed-wing uavs using a standard sensor suite, in: 2016 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 624–632.

Winter, A., 2019. Iced airfoils in uavs - entwurf.

Appendix

Appendix A

Assumptions

The assumptions made creating the mathematical model and performing robustness analysis are presented in this Appendix.

A.1 General Assumptions

- **Sensors** feeding the controller with real-time data are **noise free**.
- **Actuator** voltages to applied forces are instant.

A.2 Propeller Torque

The toque from the actuator applied on the propeller for thrust will have an opposite component reacting on the UAV. This effect will, by the assumption that the propeller is placed in the $\boldsymbol{i}^b\boldsymbol{k}^b$-plane, generate a small amount of torque in the lateral direction. It is easily compensated by applying a small aileron deflection. In (Beard and McLain, 2012) this effect is considered by adding a moment component to the total moment vector:

$$\mathbf{M}_p = \begin{bmatrix} -k_{T_p}(k_\Omega \delta_t)^2) & 0 & 0 \end{bmatrix}^\top, \tag{A.1}$$

where $\Omega = k_\Omega \delta_t \in \mathbb{R}$ is the propeller speed, and $k_{T_p} \in \mathbb{R}$ is a constant determined by experiments.

A.3 Rigid body and UAV symmetry

The mathematical model uses the moment of inertia matrix assuming the form:

$$\boldsymbol{J}^b = \begin{bmatrix} I_{xx} & 0 & I_{xz} \\ 0 & I_{yy} & 0 \\ I_{xz} & 0 & I_{zz} \end{bmatrix}. \tag{A.2}$$

By definition in (Egeland and Gravdahl, 2002), the moment of inertia matrix is defined by the equation:

$$\mathbf{J}^b = \int_b \begin{bmatrix} y^2 + z^2 & -xy & -xz \\ -xy & x^2 + z^2 & -yz \\ -xz & -yz & x^2 + y^2 \end{bmatrix} dm \tag{A.3}$$

Where dm is defining the position of mass elements relative to the center of mass. The coordinates is defined in the body-frame coordinate system $x, y, b = x^b, y^b, z^b$. If we assume the MAV to have a rigid body and symmetry about the $\boldsymbol{i}^b\boldsymbol{k}^b$-plane, A.3 takes the form in A.2. Where a rigid body is defined having a constant moment of inertia matrix in body $\frac{d}{dt_b}\mathbf{J} = 0$.

Appendix B

Simulation Parameter Values

Parameter	Value	Description
m	3.364 kg	Mass of X8.
$J_{xx}, J_{yy}, J_{zz}, J_{xz}$	$\{1.229, 0.1702, 0.8808, 0.9343\}$	Inertia matrix components in body.
g,c,b	$9.81m/s^2, 0.3571, 2.1$	Gravitational acceleration, span and chord.
ρ	$1.225kg/m^3$	Air density.
S	$0.75m^2$	UAV wing surface area.
$C_{D,0}, C_{D,\alpha}, C_{L,0}, C_{L,\alpha}$	$\{0.0197, 0.079, 0.0867, 4.0203\}$	Drag and lift linearised air coefficients.
$C_{Dq}, C_{D\delta_e}, C_{Y\delta_a}, C_{Y,\delta_r}$	$\{0.0, 0.0633, 0.0433, 0\}$	Taylor force linearised parameters.
$C_{Y,0}, C_{Y,\beta}, C_{Y,p}, C_{Y,r}$	$\{0.0, -0.2239, -0.1379, 0.0839\}$	Taylor force linearised parameters.
$C_{Lq}, C_{L\delta_e}$	$\{3.87, 0.2781\}$	Taylor force linearised parameters.
S_p, C_p, k_motor	$\{0.1018, 1.0, 40.0\}$	Fitzpatrick parameters.
$C_{l,0}, C_{l,\beta}, C_{l,\delta_a}, C_{l,\delta_r}$	$\{0.0, -0.0849, 0.1202, 0\}$	Taylor moment linearised parameters.
$C_{l,p}, C_{l,r}, C_{m,0}, C_{m,\alpha}$	$\{-0.4042, 0.0555, 0.0227, -0.4629\}$	Taylor moment linearised parameters.
$C_{m,\delta_e}, C_{m,q}, C_{n,0}, C_{n,\beta}$	$\{-0.2292, -1.3012, 0.0, 0.0283\}$	Taylor moment linearised parameters.
$C_{n,\delta_a}, C_{n,\delta_r}, C_{n,p}, C_{n,r}$	$\{-0.0034, 0.0, 0.0044, -0.0720\}$	Taylor moment linearised parameters.
$\sigma_u, \sigma_v, \sigma_w$	$\{2.12, 2.12, 1.4\}(m/s)$	Dryden turbulence parameters.
L_u, L_v, L_w	$\{200, 200, 50\}(m)$	Dryden turbulence parameters.

Appendix C

Kinematic Relations

$$\mathbf{v}_r^b = \mathbf{v}_{nb}^b - \mathbf{R}_v^b(\mathbf{\Theta})\boldsymbol{w}_s^n \tag{C.1}$$

$$\frac{d}{dt_b}\mathbf{v}_r^b = \frac{d}{dt_b}\mathbf{v}_{nb}^b - \frac{d}{dt_b}\mathbf{R}_v^b(\mathbf{\Theta})\boldsymbol{w}_s^n \tag{C.2}$$

$$\frac{d}{dt_b}\mathbf{v}_r^b = \frac{1}{m}\mathbf{F}^b - S(\boldsymbol{\omega}_{nb}^b)\mathbf{v}_r^b \tag{C.3}$$

$$\frac{d}{dt_w}\mathbf{v}_r^b = \frac{1}{m}\mathbf{F}^b - S(\boldsymbol{\omega}_{nb}^b)\mathbf{v}_r^b - S(\boldsymbol{\omega}_{bw}^b)\mathbf{v}_r^b \tag{C.4}$$

$$\dot{\mathbf{v}}_r^w = \frac{1}{m}\mathbf{R}_b^w(\alpha,\beta)\mathbf{F}^b - \left(\mathbf{R}_b^w(\alpha,\beta)(S(\boldsymbol{\omega}_{nb}^b) + S(\boldsymbol{\omega}_{bw}^b)\right)\mathbf{v}_r^w \tag{C.5}$$

Where $\boldsymbol{\omega}_{bw}^w = \mathbf{R}_b^w(\alpha,\beta)\boldsymbol{\omega}_{bw}^b \in \mathbb{R}^3$ is the angular-rate of the wind-frame relative to the body-frame. Knowing from the frame definitions in 3.2.5, that this is defined by $\dot{\alpha}, \dot{\beta}$ leads to the following expression.

$$\boldsymbol{\omega}_{bw}^w = \dot{\beta}\boldsymbol{k}^w - \dot{\alpha}\mathbf{R}_s^w(\beta)\boldsymbol{j}^s \tag{C.6}$$

Combining the terms in Equations C.5 and C.6 yield the following dynamical equation:

$$\dot{\mathbf{v}}_r^w + \mathbf{S}(\boldsymbol{\omega}_{bw}^w)\mathbf{v}_r^w = \begin{bmatrix}\dot{V}_r & \dot{\beta}V_r & \dot{\alpha}V_r\cos\beta\end{bmatrix}^\top \tag{C.7}$$

Equations 3.27a - 3.27c, are then obtained.

Appendix D

Additional Plots From Simulations

D.1 Disturbance mitigating method comparisons

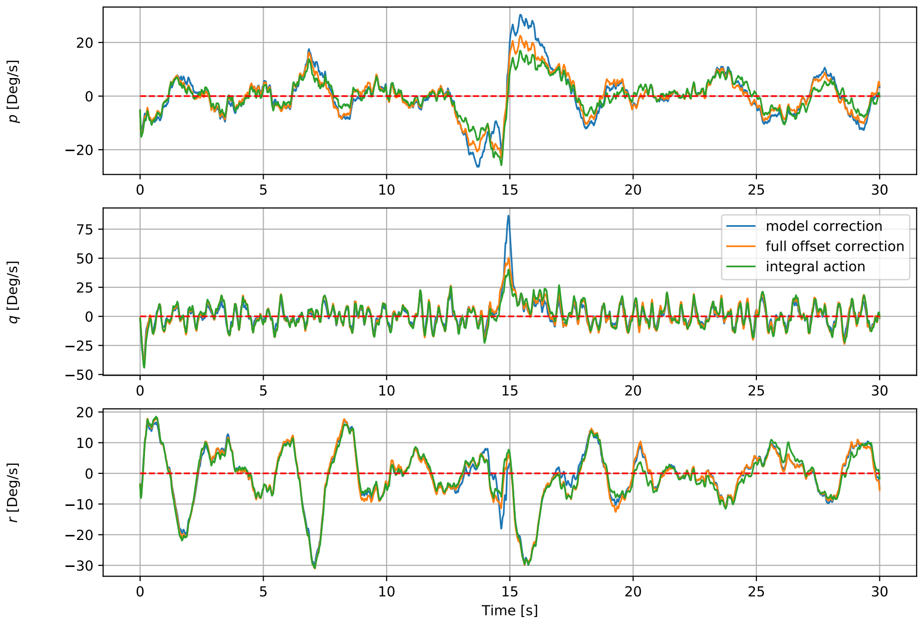


Figure D.1: Three responses of different tuning configurations of ℓ_u for the offset-free NMPC with input correction. There are displayed a *bank to turn* manoeuvre under moderate turbulence with reduced aerodynamic coefficients.

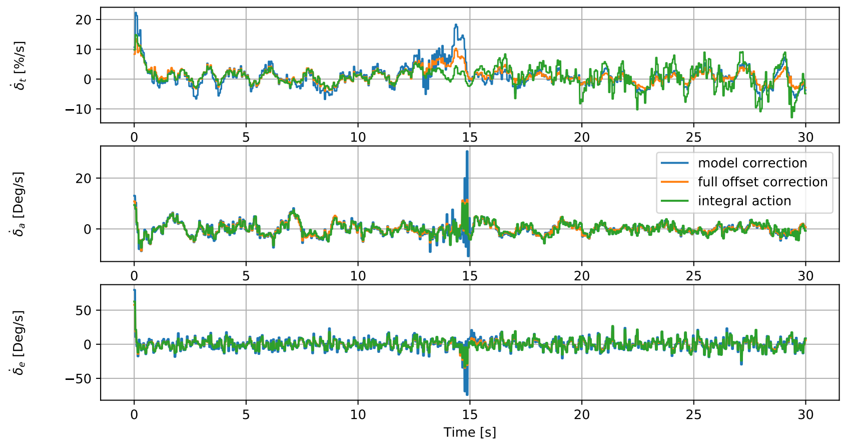


Figure D.2: Three responses of different tuning configurations of ℓ_u for the offset-free NMPC with input correction. There are displayed a *bank to turn* manoeuvre under moderate turbulence with reduced aerodynamic coefficients.