Carl von Ossietzky Universität Oldenburg

ForWind - Center for Wind Energy Research

Bachelor's Thesis:

"Model predictive control for a scaled model wind turbine"

Juan Manuel Boullosa Novo

Matriculation number: 2481927

Supervisors:

Vlaho Petrović

Martin Kühn

Oldenburg, 25th May 2020

CARL VON OSSIETZKY universität OLDENBURG

Table of Contents

Abstract

Many numerical studies have been reporting promising results of MPC for wind turbines, therefore it was considered very important to successfully validate these claims through experimentation. Therefore, the goal of this thesis is to enable experimental validation of MPC in wind tunnel experiments with a scaled wind turbine model. The active set method was chosen as a suitable convex optimisation technique for the MPC and the solver was developed for MoWiTO and implemented on the cRIO control hardware, embedded with the rest of the control software. The attempt to incorporate the solver on-board the FPGA board failed due to hardware constraints, nevertheless, the ASM algorithm on the cRIO proved to be efficient enough to handle any pitch control actions in real time for the wind turbine in the wind tunnel experiments. The experiment proved very successful, achieving to handle numerous strenuous wind conditions settings while obtaining solving times of under 1 ms, with a mean of 558 μs and of 535 μs for the wind gusts and the turbulence experiments, respectively.

1. Introduction

As the climate crisis fueled concerns about the preservation of the environment and the desire to reduce the dependence of fossil fuels in Europe are on the rise, renewable energies have been gaining more attention. In particular, wind energy has become the most important renewable energy source in Germany. Its percentage of shared production has been rapidly increasing in the past recent years in contrast with any other renewable energy sources such as solar, hydro and biomass, rising from 14.5% in 2016 to 24.6% of net public energy generation in Germany in 2019 **(Figure 1.1)**. In the same way, the total wind power capacity has been steadily increasing in Europe, surpassing the 200 GW of total production capacity in 2019 **(Figure 1.2)**.

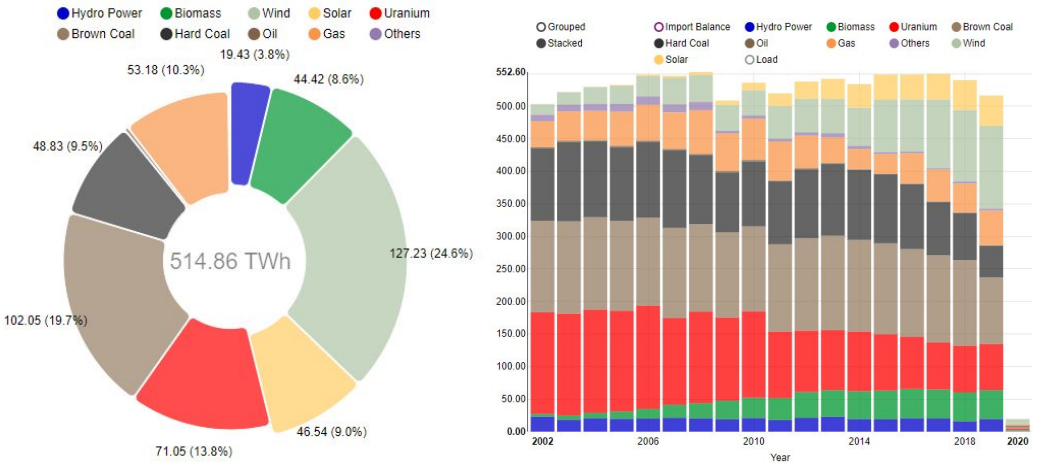

Figure 1.1: B. Burger, Fraunhofer ISE, Source: https://www.energy-charts.de

This surge of wind energy production can be partly attributed to the recent developments in wind turbine technology that allowed the installation of larger and more stable rotors with more efficient energy conversion that could be constructed at a lower cost. However, larger rotor surfaces also lead to increased structural loads that have to be addressed. In order to reduce these loads, while maintaining a stable and high enough power production, more efficient control algorithms have to be designed to regulate the blades pitch and generator torque of the wind turbines.

	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019
Offshore	2	3	4	5	7	8	11	13	16	18	22
Onshore	75	84	93	105	115	127	137	149	162	171	183
Total	77	87	97	110	122	135	148	162	178	189	205

Figure 1.2: Total installed wind power capacity in Europe, Source: WindEurope Annual Statistic 2019

Wind turbines commonly operate in two regions with different control objectives. In low speed winds (i.e. below rated wind speed), there is not enough energy in the wind for the turbine to produce its rated power. Therefore, in this region, the control objective is optimized to prioritize for maximum power production, and this is done by maintaining a constant pitch angle of the turbine blades while the rotor speed is adapted in order to maximize power generation by varying the generator's torque.

On the other hand, 'above-rated' winds are set to minimize the stresses caused on the wind turbine components while keeping a steady power production. This is obtained by setting a constant generator torque, regulating the rotor speed to a rated value by changing the angle of the turbine blades that affects the lift generated by them. Changing the pitch angle of the blades helps to lower the structural loading on the turbine components **(Burton 2011)**.

According to studies, feedforward information caused a much more significant benefit in the reduction of structural loading when there were rapidly changing wind speeds for above rated wind conditions compared with a baseline controller **(Spencer et al. 2012)**. They gained attention with the recent development of lidar technology, which can measure a preview of the wind conditions in front of the turbine. The above-rated case for the pitch-angle optimal control of the wind turbine blades will therefore be the focus of this thesis, using both feedback and feedforward conditions.

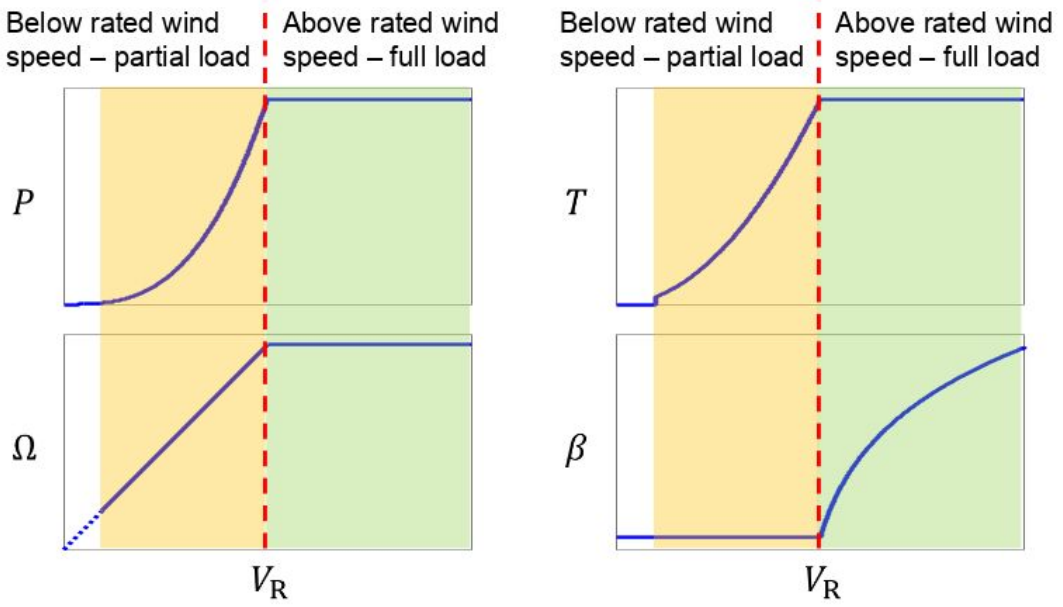

Figure 1.3: Power operating regions for variable-speed wind turbines with pitch/torque control.

In addition, modern controllers are often extended with additional control algorithms and objectives in order to reduce wind turbine loads while keeping a high power production. This leads to an increase of the wind turbine lifetime and a reduction of the maintenance costs associated with it. One of the approaches to integrate multiple control algorithms and objectives is Model predictive control.

Model predictive control (MPC), is an optimal control method that has gained increased attention in recent years as a process that can preview disturbances for wind turbine control, usually attached to a lidar that can measure wind velocity in front of the turbine, which can provide feedforward information to the controller **(Schlipf et al. 2013; Jain et al. 2015; Mirzaei et al. 2016; Gros et al. 2017)**.

MPCs usually involve solving an optimisation problem online, that enable us to combine different control objectives and allows us to include any system of constraints onto the control problem formulation.

However, due to the challenge involved in solving constrained optimisation problems in real time, implementations of MPC typically require significant computational power available. This is why MPC have not been widely used for wind turbine control and were traditionally used only in slow processes. With the

advent of more powerful processors in the last decades, it enabled their applications also for faster processes such as wind turbine disturbance handling.

A simplification is to formulate an unconstrained MPC problem with a quadratic cost function and a linear model of the system. These kinds of controllers, denominated “linear quadratic regulators” (LQR) are commonly used in practice due to their very low computational power requirements. However, due to the inability of LQR to handle system constraints, such as actuator limitators, these controllers have very limited use in wind turbine control.

For the proper operation of modern multi-megawatt wind turbines, it is important to adequately consider constraints, to control the pitch angle actuators in order to limit them from requesting excessive pitch angles or surpass their pitch angle boundaries and pitch angle rates **(Spencer et al. 2012).** MPC problems are often formulated as a quadratic convex constrained optimisation problem that can be addressed and solved efficiently with a variety of optimisation algorithms and for which a mature theory exists.

Experimental testing of novel control system algorithms is important, as numerical testing results are often not reliable enough for real world applications. However, field testing is usually discouraged due to high costs and the risks associated.
At ForWind Oldenburg, a fully controllable scaled wind turbine MoWiTO (Model Wind Turbine Oldenburg 1.8 m) is often used to test the performance of different control algorithms experimentally in the wind tunnel. MoWiTO allows us to test the MPC in a safe environment under reproducible and predictable conditions of wind speed and turbulence. Furthermore, the reduced scale of the model wind turbine allows us to accelerate the rate of testing the experiments significantly, as it undergoes a much faster changing environment in contrast with the full-scaled wind turbines.

The main goal of this thesis is to develop and test an MPC algorithm for MoWiTO. This poses a challenge, as the time is scaled accordingly for the MoWiTO, undergoing a much faster changing environment of around 46 times faster than real-time, which further limits the computational time window for the MPC solver. This is why the MPC solver was implemented onboard a compactRIO board using

LabVIEW. They are equipped with a powerful CPU, an FPGA board and input/output modules that run independently of a computer. This increases the performance of the solver, as the FPGA board is able of theoretically obtaining much faster clock speeds and shorter computational delays due to its high parallelization capabilities that can be utilised in the calculation of the relatively simple linear algebra calculations involved.

The combination of the constrained optimiser MPC solver installed onboard an FPGA could cause an improvement in the control system capabilities for the pitch angle, which will both have a shorter computational delay, a longer prediction horizon with feedforward characteristics, and a system of constraints that limits the maximum and minimum pitch angle turning rate and absolute values, which can in turn enhance the wind turbine performance and safety value.

The thesis is organised as follows:

The methodology is presented in chapter 2, explaining the experimental setup in the wind tunnel, the modelling of the wind turbines, the convex optimisation techniques utilized and the realisation of the solver in the cRIO and the FPGA.
The results are then layed out and explained in chapter 3, together with the performance analysis of the solver and the validation of the MPC in the wind tunnel. Finally, in chapter 4 there is a brief discussion about the efficiency of the algorithms, the timing and the usage of FPGAs for the applications.

2. Methodology

2.1 Wind tunnel setup

At ForWind, University of Oldenburg, a fully controllable scaled wind turbine model MoWiTO can be used to test new control algorithms. This setup allowed to validate the MPC under reproducible conditions of wind speed and stresses coming from wind gusts and turbulences in a safe and controllable closed environment. This is preferable compared to field testing in a real world environment, as they do not offer reproducible wind conditions, making it difficult to obtain conclusive results as well as being more costly and unsafe. Furthermore, numerical simulations, such as aeroelastic simulations, always require some simplifications that might not be sufficiently representative of the real world conditions. The wind tunnel setup allows experimental validation of novel control concepts without or with significantly lower difficulties from field tests. Therefore, the wind tunnel setup presented here is considered as a valuable testing environment for the validation of the control system, which complements the shortcomings of the numerical simulations and the field experiments.

2.1.1 Wind Tunnel

The experimental setup at the WindLab of ForWind in the University of Oldenburg consists of a Göttingen style wind tunnel that can be arranged in a closed or open test section configuration. It is fitted with a cross section of 3 x 3 m and an open test section with a length of 30 m, as seen in **Figure 2.1.1**. The wind tunnel employs an array of four fans with an individual power of 110 kW, summing up to 440 kW of electrical power in total. They can achieve wind speeds of up to 32 m/s with an open section. A combination of honeycomb grids and fine meshes at the nozzle allow for a reduction of the turbulence intensity in the wind tunnel, and a cooling system is introduced to allow the maintenance of a constant temperature in the tube at higher wind speeds. This cooling system has got 192 kW of electric power which results in 406 kW of effective cooling power **(Kröger et al. 2018)**.

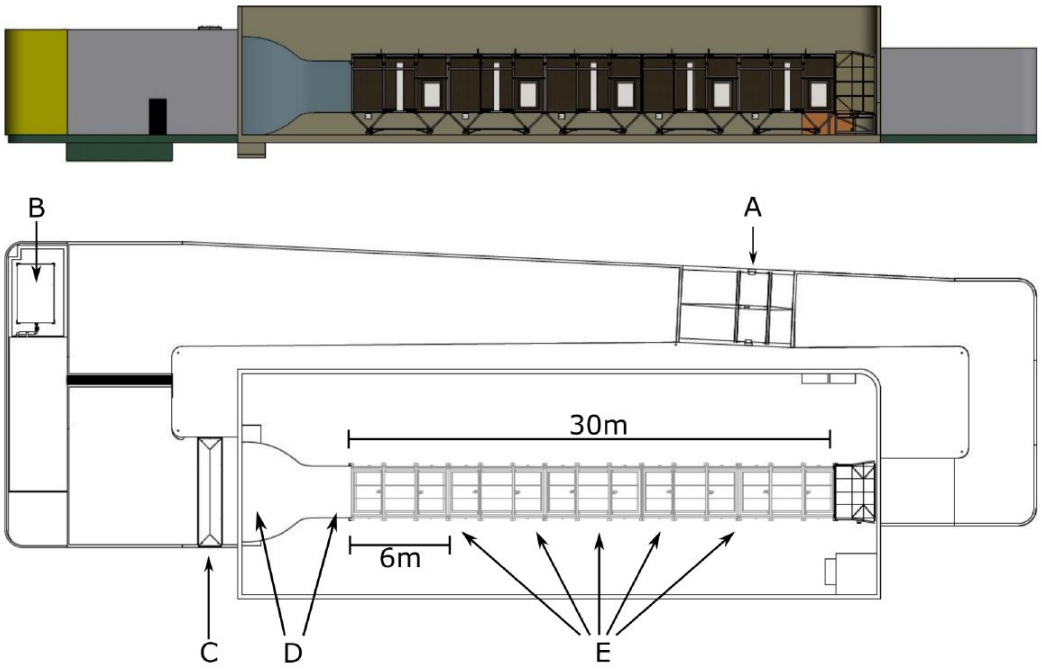

Figure 2.1.1: Diagram of the WindLab wind tunnel of ForWind Oldenburg. Above: side view, Below: Top view. A: array of fans (4 x 110 kW) B: cooling system (406 kW) C: honey combs; D: prantl tubes; E: optional closed section segments **(Kröger 2018)**

2.1.2 Active Grid

In order to modulate the wind outflow, an active grid is mounted to the outlet of the tunnel. This active grid is divided by 80 horizontal and vertical axes with 1.5 m length, spanning half the distance between nozzles. It generates a mesh with 0.14 m width **(Figure 2.1.2)**. The axes rotation are controlled individually in a real time system by servo motors Kollmorgen VLM22C-DLNC-00 as well as Kollmorgen servo drives AKD-P00306-NBEC-0000 as power supplies and monitoring devices. When attached to the nozzle, these move in order to orient a range of square flaps mounted to the axes, with which a range of specific turbulent and transient flow patterns can be generated in a continuous manner into the airflow **(Figure 2.1.3)**. The servo motors and drives are controlled via EtherCAT by a National Instruments Real Time Embedded Controller NI PXIe-8135 and a LabVIEW software interface. This setup is able to create a cross section blockage ranging from a minimum of 21% and a maximum of 92% **(Kröger et al. 2018).** It is also possible to generate a sheared flow and wind gusts, which allows for testing of the control systems in various conditions. All of these patterns are scaled in size and speed to match the model wind turbine scaling **(Petrovic et al. 2019).**

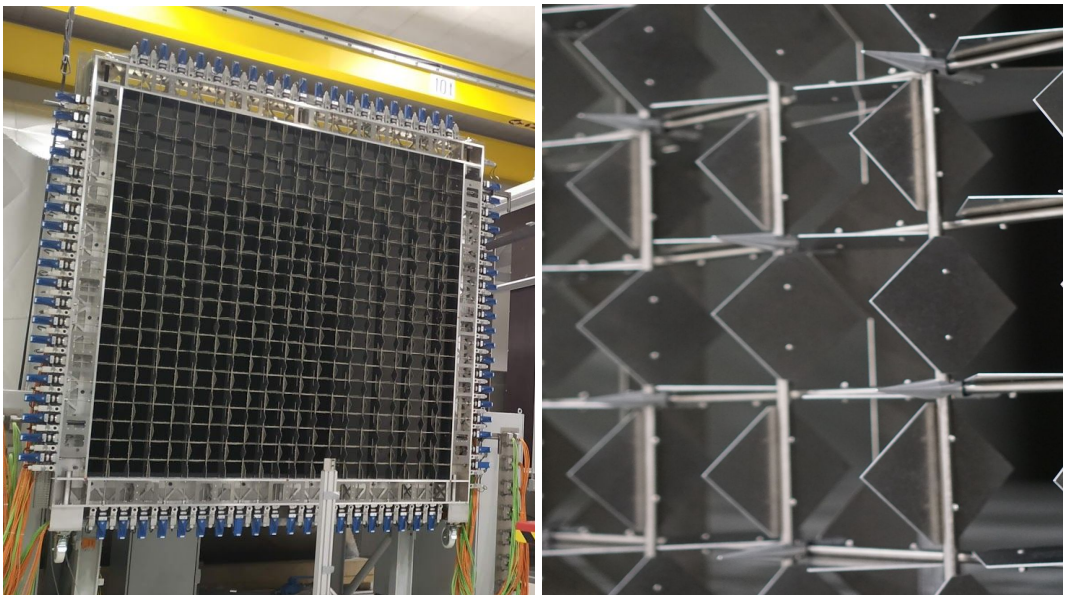

Figure 2.1.2, 2.1.3: Active grid mounted to the wind tunnel nozzle and close up of the active grid flaps and axes. **(Kröger 2018)**.

2.1.3 MoWiTO - Model Wind Turbine

The ForWind Model Wind Turbine Oldenburg (MoWiTO), as shown in **Figure 2.1.4**, is a fully controllable scaled wind turbine with a rotor diameter of 1.8 m, with generator torque control and individual blade pitch control, designed after the generic offshore NREL 5MW reference turbine **(Jonkman et al. 2009).**

It is scaled in order to maintain the design tip speed ratio of 7.5 of the reference turbine, while keeping a representative lift distribution on the blades. The rotor diameter $\mathbf{D_{scaled}}$ = 1.8 m was chosen according to the wind tunnel open test section nozzle size of 3 x 3 m, leading to the following length scaling compared to the diameter of the reference turbine $\mathbf{D_{reference}}$ = 126 m : $\mathbf{n_L} = D_{scaled} / D_{reference}$ = 1 / 70; as per equation **(1)**. The rated rotational speed is limited to $\mathbf{n_{scaled}}$ = 600 rpm, whereas the full scale value is $\mathbf{n_{reference}}$ = 12.11 rpm, thus leading to the time scaling factor $\mathbf{n_T}$ = 49.6, as calculated by equation **(2)**. 600 rpm is the capable MoWiTO rotor speed, but a lower rated rotor speed of 480 rpm is chosen instead for the experimentation. A list of all the scaling parameters can be observed in **table 2.1.1**. For more details refer to **(Berger et al. 2018).** In addition, an array of sensors are located in the turbine, which allows for measurements of blade, shaft and tower loads, as well as pitch angles, electrical torque and rotor speed **(Petrovic et al. 2019).**

$$n_L = \frac{D_{scaled}}{D_{reference}} = \frac{1.8m}{126m} = \frac{1}{70} \quad \textbf{(1)} \qquad n_r = \frac{n_{scaled}}{n_{reference}} = \frac{600\ rpm}{12.11\ rpm} = 49.6 \quad \textbf{(2)}$$

Rated values	Factor	Reference	Scaled
Rotor diameter	n_L	126m	1.8m
Rotor speed	n_T	12.1 rpm	600 rpm
Tip speed	$n_L \bullet n_T$	80m/s	57m/s
Power	$n_L^5 \bullet n_T^3$	5MW	363W
Torque	$n_L^5 \bullet n_T^2$	3.95MNm	5.8Nm
Wind speed	$n_L\ n_T$	11.4m/s	8.1m/s
Thrust	$n_L^4 \bullet n_T^2$	700kN	72N
Max. chord Reynolds number	$n_L^2 \bullet n_T$	$1.1 * 10^7$	$1.1 * 10^5$

Table 2.1.1: Scaling of main turbine parameters **(Petrovic 2019)**

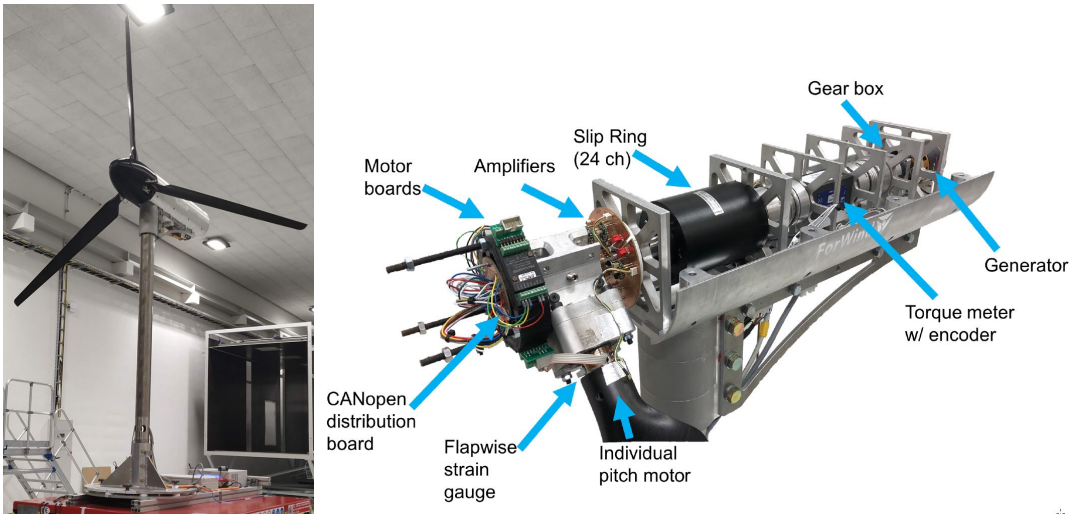

Figure 2.1.4: Model wind turbine MoWiTo and nacelle of the MoWiTo **(Petrovic 2019)**

2.1.4 Wind speed measurement

A hot-wire anemometer is positioned 2.7 m in front of the rotor centre in order to measure the wind speeds. The hot-wire has to be constantly calibrated to the current temperature and pressure conditions of the wind tunnel before performing the experiment. It is important to measure the wind speeds in front of the turbine, which can be used for the data analysis as well as for the feedforward controller.

2.1.5 Modelling

A computerized medium-fidelity simulation model of MoWiTO was implemented using FAST v8 (fatigue, aerodynamics, structures and turbulence) in a Simulink model, in order to verify and tune the controller. FAST is a wind turbine modeling package developed by NREL, USA, that links the aerodynamic, elastic, mechanical and electrical behaviors of the turbine, and allows for time-domain simulations in turbulent and changing inflow wind situations.

The wind turbine is often modelled after a nonlinear first-order system as per equation **(3)**. The nonlinearity is contained in the aerodynamic part, as it can be seen from the equation for the aerodynamic torque in equation **(4)**. This model has three inputs, wind velocity *v* (which is also the disturbance), collective pitch angle β (assumes all turbine blades are pitched identically) and generator torque M_g.

$$J\omega' = M_a - M_g \qquad \textbf{(3)}$$

J = *rotor inertia*
ω = *rotor speed*
M_a = *aerodynamic torque produced by the blades*
M_g = *generator torque*

$$M_a = \tfrac{\pi}{2}\rho R^3 C_q(\lambda,\beta) v_w{}^2 \qquad \textbf{(4)}$$

ρ = *air density*
R = *rotor diameter*
v_w = *wind speed*
β = *pitch angle*
$\lambda = \frac{\omega R}{v_w}$, *tip speed ratio* (TSR)
$C_q(\lambda,\beta)$ = *torque coefficient*

After linearisation, the equation for aerodynamic torque **(4)** becomes:

$$\Delta M_a = c_w \Delta v_w + c_\beta \Delta\beta + c_\omega \Delta\omega \qquad \textbf{(5)}$$

$$c_w = \frac{\partial M_a}{\partial v_w}, \qquad c_\beta = \frac{\partial M_a}{\partial \beta}, \qquad c_\omega = \frac{\partial M_a}{\partial \omega} \qquad \textbf{(6)}$$

In equations **(5)** and **(6)**, Δ represents a deviation about the operating point around which the linearisation is made.

If equation **3** is linearised, the following equation is obtained:

$$J\Delta\omega' = c_w * \Delta v_w + c_\beta \Delta\beta + c_\omega \Delta\omega - \Delta M_g \quad \textbf{(7)}$$

MPC is implemented using a linear state-space model of the plant in discrete time, First, the state space is expressed more in the continuous time with equation **(8)**, where $x'(t)$ is the time derivative of x, and A_c, B_c are the model matrices.

$$x'(t) = A_c x(t) + B_c u(t) \quad \textbf{(8)}$$

Since the pitch control is developed in this thesis, and the generator torque is kept constant at its rated value, i.e. ΔM_g = 0. The disturbance caused by the wind is regarded as unknown, therefore it is not modeled and it will not be considered any further in the thesis. Besides the rotor speed $\Delta\omega$, its integral in the state vector x is also included in equation **(9)**. In this way, integral behaviour is introduced in the system, thus avoiding steady state errors in the presence of disturbance. Collective pitch angle (β) is considered to be the only controllable input of the model.

$$x = [\Delta\omega, \int \Delta\omega]^T, \; u = \Delta\beta \quad \textbf{(9)}$$

The model matrices for such state space model are:

$$A_c = \begin{bmatrix} \frac{c_\omega}{J} & 0 \\ 1 & 0 \end{bmatrix}, \qquad B_c = \begin{bmatrix} \frac{c_\beta}{J} \\ 0 \end{bmatrix} \quad \textbf{(10)}$$

As the goal is to design a discrete-time controller with a sampling time of T_S=10 ms, the model has to be discretized, to obtain the state space representation as shown with equation **(11)**. For linearisation and controller tuning, an operating point defined with the wind speed of 7.2 m/s was chosen as shown in **table 2.1.2**, and the model matrices as per equation **(12)** are obtained.

$$x(k+1) = Ax(k) + Bu(k) \quad \textbf{(11)}$$

Wind speed (v_w)	7.2 m/s
Rotor speed (ω)	480 rpm
Pitch angle (β)	4.5º

Table 2.1.2: Operating points used for the model linearisation.

$$A = \begin{bmatrix} 0.971426 & 0 \\ 0.00985644 & 1 \end{bmatrix}, B = \begin{bmatrix} -0.853190 \\ -0.00428656 \end{bmatrix}, \quad \textbf{(12)}$$

This is the discrete-time state space model that is used to formulate the MPC optimisation problem.

2.1.6 Baseline MoWiTO controller

In region 2 (below-rated conditions), a baseline torque controller is in charge of regulating the generator torque of the turbine. The main objective of the torque controller is to enable the optimal energy conversion in the partial load conditions (below rated wind speed), and to keep the torque at the rated value in the full load conditions (above rated wind speed). The first objective can be achieved by mapping a look-up table of the rotor speed values to the optimal torque values that would lead to the optimal energy conversion in the steady-state.

However, to deal with transitions between the partial and the full load operation close to the rated wind speed, a PI controller is introduced. The PI controller uses measured rotor speed as feedback, and tries to keep the rotor speed constant 10 rpm under the rated value by setting the rated generator torque.

The transitioning between operating regions is achieved through the controller limits. The upper limit is set to the rated torque value, ensuring that the torque will be kept constant when the turbine operates around the rated rotor speed in the full load conditions. The lower limit is set to the optimal torque for partial load conditions, that is constantly changing together with the rotor speed.

When the wind speed increases further and the torque alone is not enough to keep the rotor speed below its rated value, the PI controller sets the rated torque, and the pitch controller takes over the wind turbine control.
Since the transition between the full and partial load conditions is done only based on the rotor speed measurement, it is possible that the torque controller reacts to sudden drops in wind speed even when the pitch controller is still active. Therefore, the torque controller will assist the pitch controller in keeping the rotor speed at the rated value during strong dips in wind speed, at the expense of an additional loss of power.

In region 3, the baseline pitch controller was substituted by the MPC while keeping the torque controller unchanged.

Since the torque controller is supposed to be saturated at the rated torque value (2.8 Nm) in full-load conditions, an anti-wind-up technique is implemented. Similarly, the pitch controller has to be saturated at β_{min} in partial-load conditions, which is handled by defining the angle constraints directly in the MPC formulation.

2.2 Formulation of the MPC problem

Model predictive control is a control system technique that runs on-line and uses predictions based on a model as described in **section 2.1.5.** These predictions are drawn across a receding prediction horizon which determines how far into the future the control actions are being planned, while only executing part of the plan, as seen in **Figure 2.2.1**.

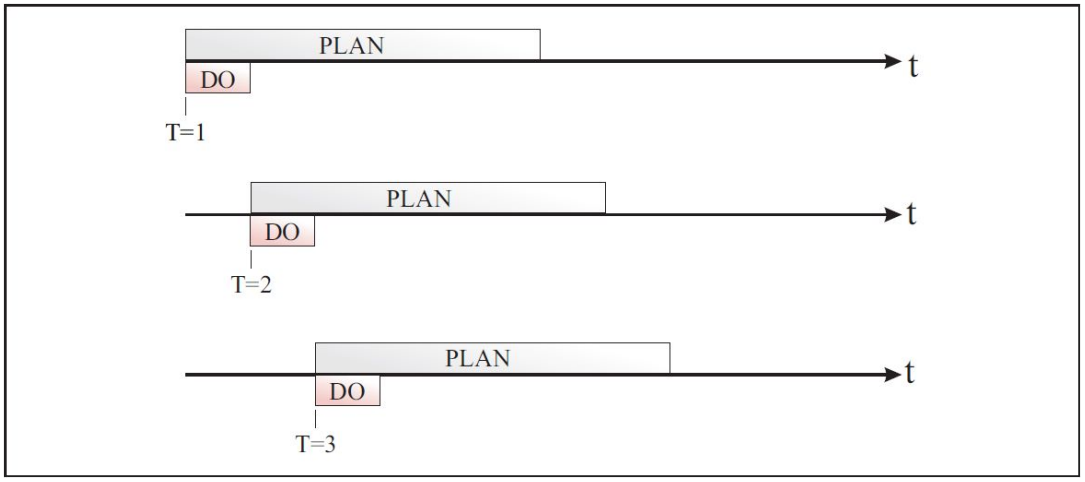

Figure 2.2.1: Model predictive control receding prediction horizon **(Lofberg 2010)**

The MPC problem is defined by an optimal control problem that consists of a cost function and a system of constraints, and solved optimisation technique as can be seen in **chapter 2.3.** In this case, it is assumed that the model is as per equation **(11).** And that the quadratic MPC cost function is given by equation **(13),** where *x(i)* and *u(i)* are the predicted state and control at a future time step *i*. Q and R are the cost function positive definite weighting matrices, that describe the minimization objective penalising certain behavior, e.g.: minimize rotor speed and pitch angle changes, and N is the prediction horizon length. J and g are the system of inequality constraints, which represent the physical system constraints, such as the pitch rate and the minimum and maximum angles.

$$\min_u \sum_{k=0}^{N-1} x(k)^T Q x(k) + u(k)^T R u(k) \qquad \textbf{(13)}$$

$$\text{s.t.} \quad x(0) = \hat{x}$$

$$Jx(k) \leq g$$

Two vectors containing all of $x(k)$ and $u(k)$ are formed as follows

$$X = [x(1); x(2); x(3);\ \ldots x(N)], \quad U = [u(0); u(1);\ \ldots u(N-1)] \textbf{ (14)}$$

And the model equation from **(11)** can be rewritten as equation **(15)**:

$$X = A_m \hat{x} + B_m U \qquad \textbf{(15)}$$

Only the first step u(0) of the sequence is implemented, after which, a new initial state $\hat{x}$ is used to calculate the new solution. Equation **(13)** is often written in a matrix-vector form. The new cost function in equation **(16)** contains all states and inputs along the prediction horizon as follows.

$$\begin{aligned} &\min_U \ X^T Q_m X + U^T R_m U \qquad \textbf{(16)} \\ &\text{s.t.} \ J_m X + E U \leq g \end{aligned}$$

It is then condensed by substituting the model **(15)** into the cost function and constraints **(16)** in order to obtain the final form of the optimisation problem in equation **(17).** In it, $h = h_{x_0} * x_0$, since this is how it has to be calculated in real time.

$$\begin{aligned} &\min_u \ U^T H U + 2h^T U \qquad \textbf{(17)} \\ &\text{s.t.} \ \ GU \leq l \end{aligned}$$

In which

$$\begin{aligned} &H = {B_m}^T Q_m B_m + R_m, \ h = {B_m}^T Q_m A\hat{x}(k) \qquad \textbf{(18)} \\ &G = J_m B_m + E, \ l = g - J_m \ A_m \hat{x}(k) \end{aligned}$$

Two kinds of constraints are considered: a blade pitch rate constraint and a blade pitch angle constraint. Both are constraints on the control input *u*. The rate constraint is given by a simple box constraint, with β as the blade pitch rate limit

and $T_s = 10\ ms$ as the controller sampling time. These parameters can be seen in **table 2.2.1**.

$$u_k \geq 0, \quad k = 0, ..., N \tag{19}$$

$$-0.86º \leq u_{k+1} \leq 0.86º, \ k = 0, ..., N-1 \tag{20}$$

$$-0.86º \leq u_0 - \widehat{\beta} \leq 0.86º \tag{21}$$

The angle constraints are given by equation **(19),** and describe that the pitch angle should always be equal or larger than 0 at any given point of the prediction horizon. The rate constraint given by equation **(20)** corresponds to a maximum rate of change of $\pm 86º/s$ due to the sampling time $T_s = 10\ ms$. In order to prevent the control signal in the first time step k=0 from violating the rate limit, the blades have a current pitch value described by $\widehat{\beta}$, so the very first action in the prediction horizon would not differ from that too much, as in equation **(21)**.

Prediction horizon length N	10
Controller sampling time [ms]	10
Blade pitch angle limit β_{min} [º]	0
Blade pitch rate limit β [º/s]	$\pm$86

Table 2.2.1: MPC parameters

The MPC is in charge of controlling exclusively the blade pitch angle β happening in region 3, with above-rated winds. Meanwhile, the generator torque and the rpm and kept constant at the rated value $\tau = 2.8\ Nm$.

2.3 Convex optimisation techniques

Quadratic programming (QP) methods are commonly used in applications of model predictive control problems (MPC), based on a linear model with a quadratic objective function and physical constraints. In order to develop an efficient model predictive control, different recent optimization techniques were compared with the purpose of solving problems with quadratic objectives subject to linear constraints online in a reasonable amount of time, with the opportunity of later implementing it in a Field Programmable Gate Array platform (FPGA).

Different approaches were considered. Two of the most common methods are the Interior point method (IPM) and Active set method (ASM) algorithms.

$$\min_{x} \left\{ \frac{1}{2} x' Q x + c' : Gu \leq l \right\} \quad \textbf{(22)}$$

The above equation based off of equation **(12),** is a MPC problem in which Q is a NxN positive matrix, in which N is the size of the prediction horizon, c is an N x 1 vector and G and l have the sizes N_c x N and N_c respectively, where N_c is the number of constraints of the problem.

According to the paper **(Lau et al. 2009)**, that compared the efficiency, speed and scalability of these two methods in the FPGA environment, the ASM method rendered much better results for convex optimisation due to its parallelization capabilities and lower computation complexity.

2.3.1 Interior point method

The idea of Interior point method (IPM) is to approach the solution of the optimisation problem by successive descent steps from the interior of the feasible set. Each descent step is obtained from solving a system of linear equations. **(Boyd 2004) (Figure 2.3.1)**

These systems of linear equations are usually very expensive to compute but can iteratively make significant progress towards the solution **(Lau et al. 2009).**

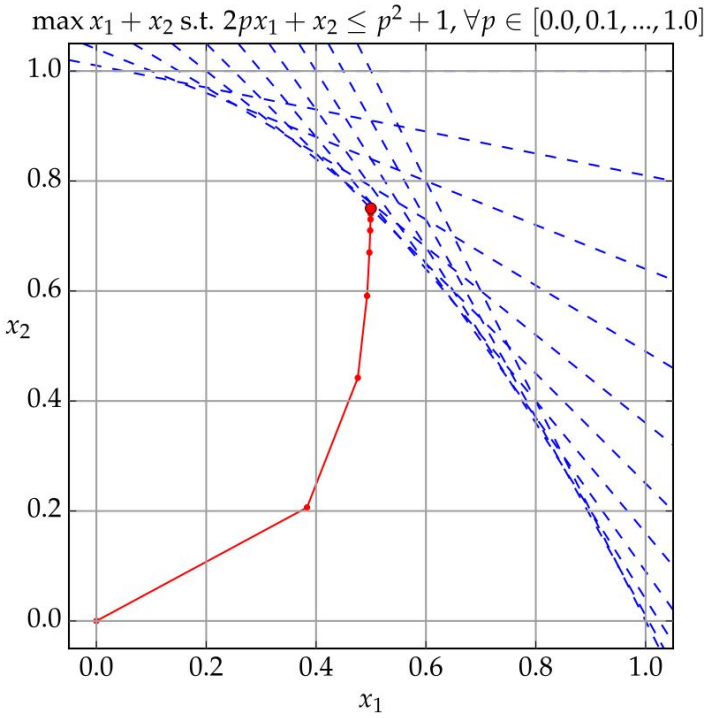

Figure 2.3.1: Example solution of IPM (https://commons.wikimedia.org/wiki/File:Karmarkar.svg)

In IPM, the infeasible region is penalised in the cost function by assigning it an infinite value. Ideally, the original cost function would be left unchanged and only assigned an infinite value to the infeasible region, but that would cause differentiability issues, as step functions cannot be derived. Therefore, gradual slopes that rise slowly to the infinite values are introduced in the cost function.
This way, the problem can be solved iteratively, making the slope of added components to the cost function steeper in each subsequent iteration until converging, in order to avoid numerical issues.

The first iteration will find a solution somewhere in the middle of the feasible region, making it easy to converge if the real solution happens to be in the vicinity. If the solution is located in the middle of the feasible region, i.e. far from any constraints, the algorithm will converge fast. However, it will take a large number of iterations in case the solution is close to some of the constraints, modifying the cost function in each step as it approaches the infeasible region.

Due to this continuous modification of the cost function in each iteration, knowing the previous controller step does not provide any useful information to help in finding the solution in the current step, which is why IPM is often a much more computationally intensive alternative to the Active Set Method.

2.3.2 Active set Method

The Active set Method (ASM) algorithm works by identifying the active set of constraints that are being used in a QP problem by making an initial guess W_0 (working set) and iteratively removing or adding such constraints to the working set until the right combination is found. As stated before when describing the IPM, ASM might be the better choice for an optimisation algorithm due to its ability to retain its cost function every step along its converging steps. Also, according to the paper that compared the efficiency and speed of the IPM and the ASM **(Lau et al, 2009),** ASM rendered much better results for convex optimization due to its parallelization capabilities and lower computational complexity. Therefore ASM was chosen as the optimisation algorithm.

As seen in **Figure 2.3.2**, which highlights the working procedure of the optimisation by the active set method in a random quadratic, inequality constrained function, a first initial guess z_0 is chosen in the feasible region at point 1., then, the active set is calculated (constraint 1), and z_0 is moved by Δz towards it until it is at the active constraint itself at point 2. Lastly, z is moved along the active constraint until it is closest to the global minimum at point 3.

First, a prototype of the algorithm given in **Figure 2.3.4** was realized in Matlab for the previous QP problem formulation **(21).**

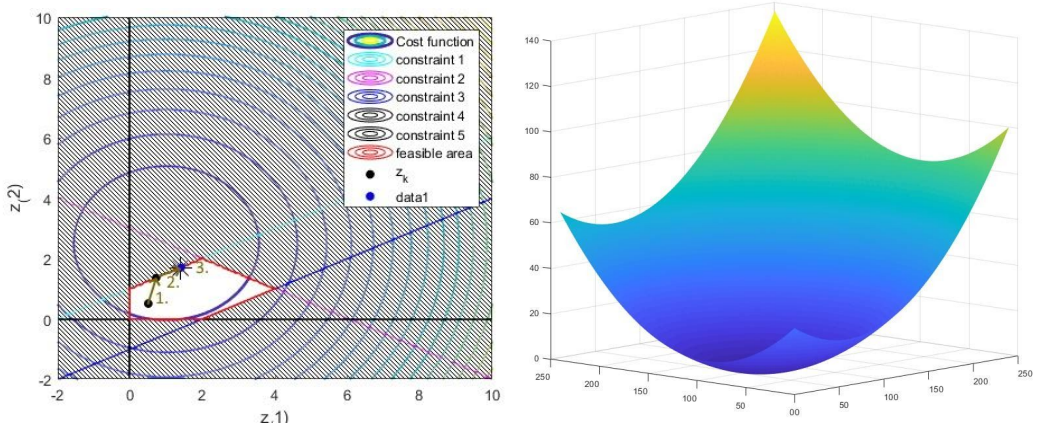

Figure 2.3.2, 2.3.3; Matlab plotting of the constrained cost function and the ASM algorithm calculation results for example cost function and constraints

1: Compute a feasible initial point z_0.
2: Let the initial working set $\mathcal{W}_0$ be empty.
3: **for** $k = 0, 1, 2, \ldots$ **do**
4: Solve $\begin{bmatrix} Q & J'_{\mathcal{W}_k} \\ J_{\mathcal{W}_k} & 0 \end{bmatrix} \begin{bmatrix} \Delta z_k \\ \lambda \end{bmatrix} = \begin{bmatrix} -c - Qz_k \\ 0 \end{bmatrix}$ for $(\Delta z_k, \lambda)$
5: **if** $\Delta z_k = 0$ **then**
6: **if** All entries of λ are not negative **then**
7: Terminate with solution $z^* = z_k$
8: **else**
9: Remove from $\mathcal{W}_k$ an index that corresponds to the smallest entry of λ, and then $z_{k+1} \leftarrow z_k$
10: **end if**
11: **else**
12: $\mathcal{D}_k \leftarrow \left\{ i \notin \mathcal{W}_k : J_i \Delta z_k > 0, \frac{g_i - J_i z_k}{J_i \Delta z_k} < 1 \right\}$
13: **if** $\mathcal{D}_k$ is empty **then**
14: $z_{k+1} \leftarrow z_k + \Delta z$ and $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k$
15: **else**
16: $\alpha \leftarrow \min_{i \in \mathcal{D}_k} \left\{ \frac{g_i - J_i z_k}{J_i \Delta z_k} \right\}$ and $z_{k+1} \leftarrow z_k + \alpha \Delta z$
17: Create $\mathcal{W}_{k+1}$ by adding one element of $\mathcal{D}_k$ to $\mathcal{W}_k$ I
18: **end if**
19: **end if**
20: **end for**

Figure 2.3.4; Active set method algorithm **(Lau, Mark S. K., et al, 2009)**

In the algorithm described above **(Figure 2.3.4)**, k is the iteration number, W_k is the working set, Q and c are as described by equation **(21),** J and g are the constraints weight matrices, described by G and I in equation **(21),** with J_i being the i th row of J, and J_{W_k} the matrix obtained by removing the i th row from J for all $i \Subset W_k$ .If all entries of g are positive, then (0,0,...,0)' is feasible. z_k is the numerical solution for the iteration k, and z_0 is the initial feasible point, that is found by checking the feasibility of $z_0 = [0\ 0\ \ldots]'$ prior to the algorithm and obtaining a new value if the chosen value is proven infeasible.

2.4 ASM with cRIO realisation in LabVIEW

After realising the ASM algorithm in the matlab implementation, it was programmed in the LabVIEW environment in order to prepare the transition to the cRIO and a possible FPGA implementation.

A new cRIO hardware module was connected and installed to the host computer, and the previous ASM software could run natively into the cRIO embedded cpu. The same algorithm previously implemented in Matlab **(chapter 2.3.2),** was realised in LabVIEW as per **Figures 2.4.1 to 2.4.8**.

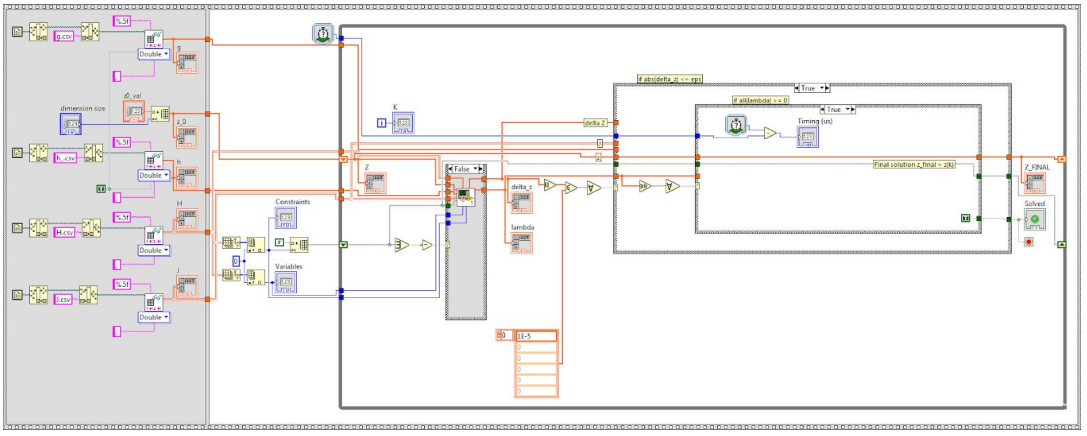

Figure 2.4.1; LabVIEW implementation of the ASM

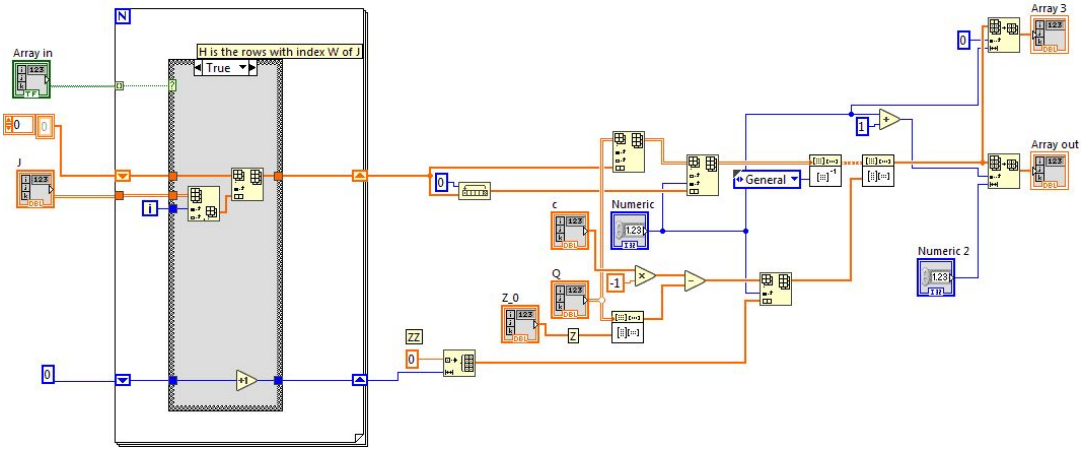

Figure 2.4.2; SubVI 1 detail from Figure 2.4.1

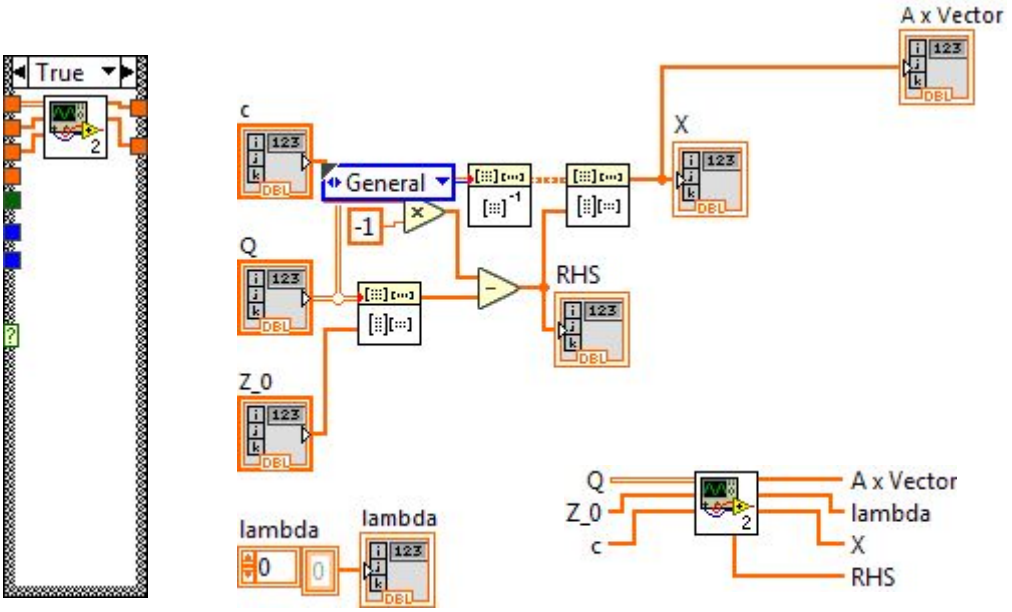

Figure 2.4.3;Detail from Figure 2.4.1 with SubVI 2.

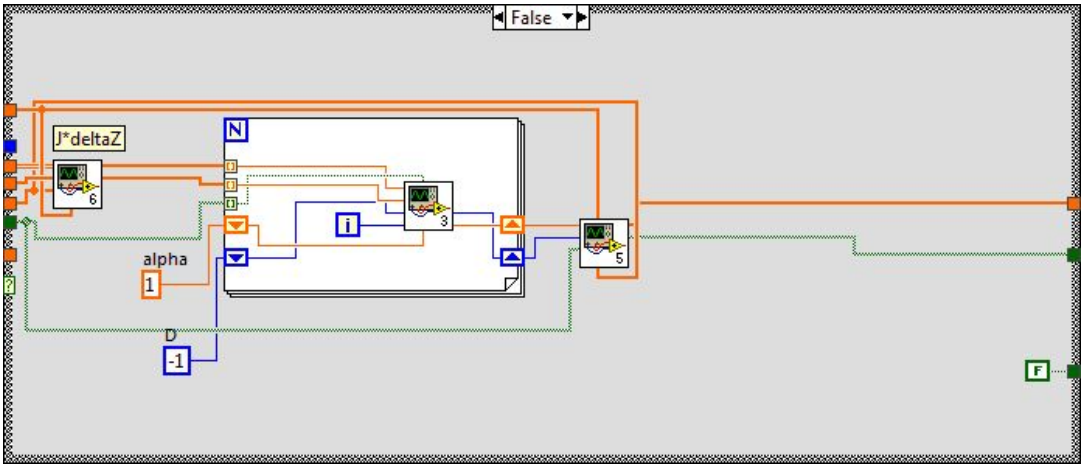

Figure 2.4.4;Detail from Figure 2.4.1.

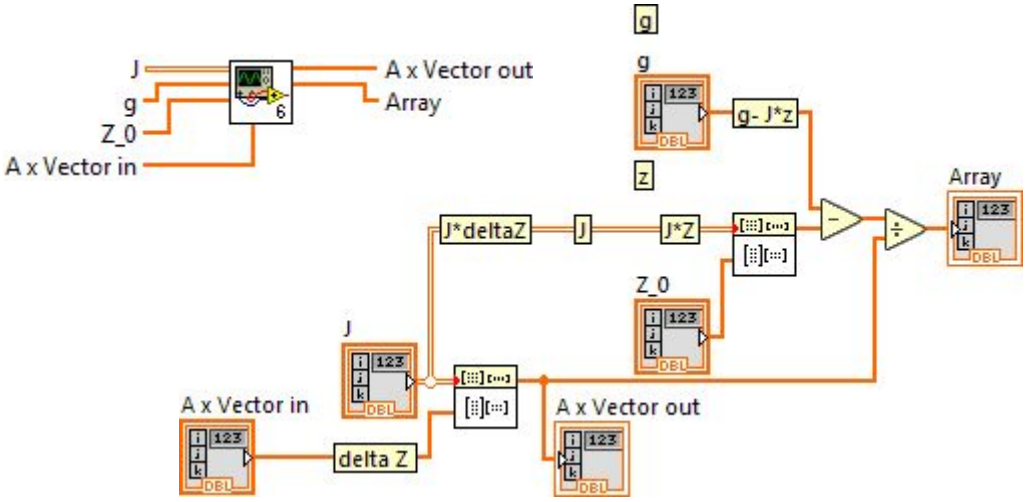

Figure 2.4.5; Detail from Figure 2.4.4, SubVI 6.

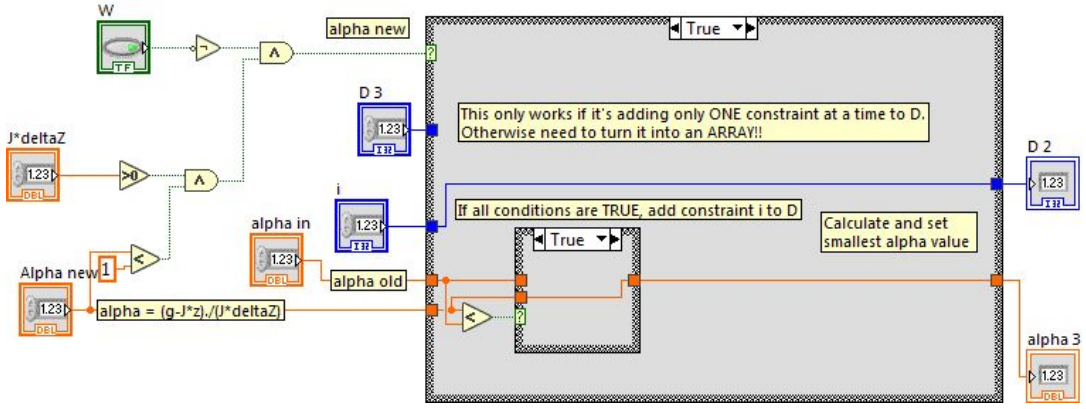

Figure 2.4.6; Detail from Figure 2.4.4, SubVI 3

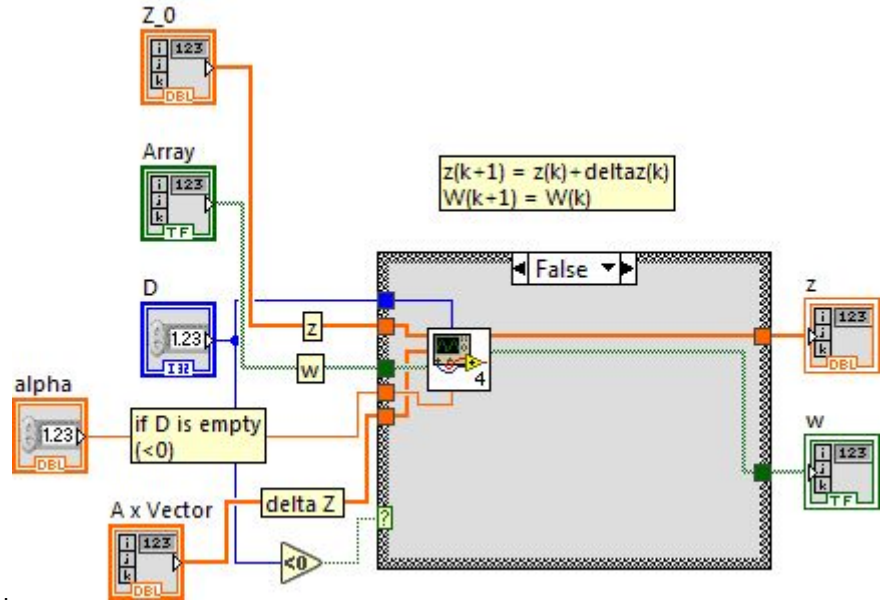

Figure 2.4.7; Detail from Figure 2.4.4, SubVI 5.

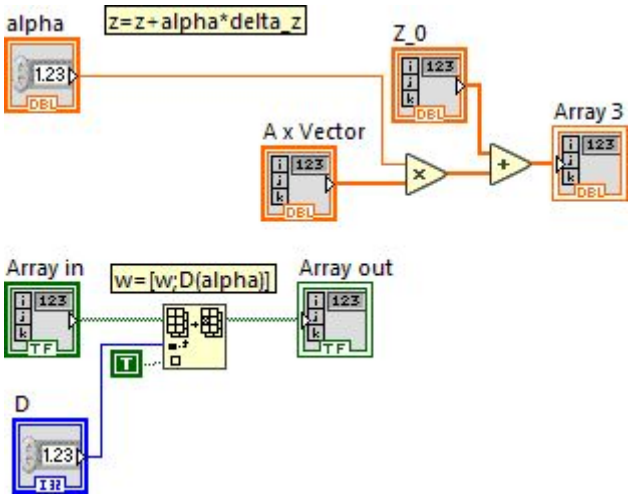

Figure 2.4.8; Detail from Figure 2.4.7, SubVI 4.

As seen in **Figure 2.4.1,** the algorithm inputs the matrices H, h, J, g and z_0 (corresponding to Q, c, G, I and x_0 respectively from equation **(21)** and outputs Z_FINAL (min of x), Computation timing (μs), Solved status (YES or NO) and K (loops required in the calculation), which will later be used for the integration with the wind turbine controller in the wind tunnel validation **(chapter 3.2)**.

2.4.1 FPGA Implementation

The algorithm was also attempted to be compiled in the FPGA module of the cRIO board unsuccessfully for any models with big enough prediction horizons, as the FPGA resources were deemed insufficient to cover the requirements to compute the expensive linear algebra equations required with the large enough matrices, in particular the matrix inversion, which was attempted to be done by following an SPMI algorithm given by the paper **(Xu, Li, Xi, Lan, & Jiang 2018)** as it could theoretically exponentially increase the efficiency of the matrix inversion due to its ability to exploit the high parallelization capabilities of the FPGA.

The SPMI algorithm is given by **figure 2.4.9.**

$\alpha \leftarrow rem(N, \sigma)$
for $k = 0 \rightarrow N$ **step** σ **do**
  $l \leftarrow \sigma$
  $n \leftarrow k + \sigma$
  if $n = \sigma$ **then**
    Calculate the inverse of sub-matrix $A_{n,n}^{1,1}$ by its adjoint matrix
  else
    if $n > N$ **then**
      $l \leftarrow \alpha$
      $n \leftarrow N$
    end if
    $\Sigma_{l\times l} \leftarrow A_{n,n}^{k+1,k+1} - A_{n,k}^{k+1,1}(A_{k,k}^{1,1})^{-1}A_{k,n}^{1,k+1}$
    $B_{n,n}^{k+1,k+1} \leftarrow {\Sigma_{l\times l}}^{-1}$
    $B_{k,n}^{1,k+1} \leftarrow -(A_{k,k}^{1,1})^{-1}A_{k,n}^{1,k+1}B_{n,n}^{k+1,k+1}$
    $B_{n,k}^{k+1,1} \leftarrow (B_{k,n}^{1,k+1})^{T}$
    $B_{k,k}^{1,1} \leftarrow (A_{k,k}^{1,1})^{-1} - (A_{k,k}^{1,1})^{-1}A_{k,n}^{1,k+1}B_{n,k}^{k+1,1}$
    $(A_{n,n}^{1,1})^{-1} \leftarrow \begin{bmatrix} B_{k,k}^{1,1} & B_{k,n}^{1,k+1} \\ B_{n,k}^{k+1,1} & B_{n,n}^{k+1,k+1} \end{bmatrix}$
  end if
end for
return $B_{N,N}^{1,1}$

Figure 2.4.9; SPMI algorithm for matrix inversion. **(Xu, Li, Xi, Lan, & Jiang 2018)**

In the algorithm in **figure 2.4.9** there are as inputs a positive definite symmetric matrix A with dimensions N x N and the iteration step size $\sigma(\sigma = 1, 2 \; or \; 3)$. And as output we obtain the inverse of matrix A, denoted as B.

The algorithm works by dividing the matrix A into submatrices of size $\sigma \mathrm{x} \sigma$, as illustrated in figure **2.4.10**, then calculating the matrix inverse of those submatrices in sequence by iteration using another simple method of inversion and then combining them by assuming equation **(23)**

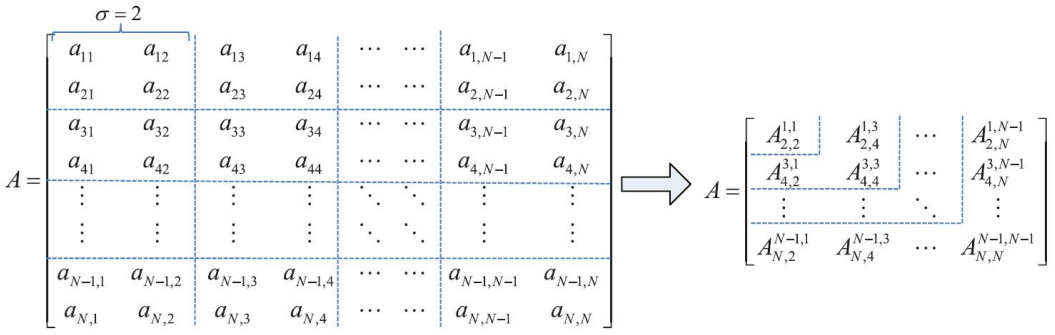

Figure 2.4.10;Division of matrix A whose dimension is divisible by $\sigma = 2$. **(Xu, Li, Xi, Lan, & Jiang 2018)**

$$\begin{bmatrix} A_{k,k}^{1,1} & A_{k,n}^{1,k+1} \\ A_{n,k}^{k+1,1} & A_{n,n}^{k+1,k+1} \end{bmatrix} \times \begin{bmatrix} B_{k,k}^{1,1} & B_{k,n}^{1,k+1} \\ B_{n,k}^{k+1,1} & B_{n,n}^{k+1,k+1} \end{bmatrix} = I_{n \times n} \qquad \textbf{(23)}$$

In the end, only N/σ iterations and division operations would be necessary to compute the matrix inverse of A with this method, which makes it theoretically very efficient compared to other methods, as seen in **figure 2.4.11.**

Matrix Dimension	CD Inversion	The SPMI Algorithm		
		$\sigma = 1$	$\sigma = 2$	$\sigma = 3$
6×6	0.552	0.348	0.218	0.121
12×12	2.052	0.821	0.540	0.342
24×24	7.808	2.088	1.346	0.898
36×36	17.507	4.157	2.557	1.762

Figure 2.4.11;Computing time of the CD inversion and the SPMI algorithm compared in matlab (unit: ms) **(Xu, Li, Xi, Lan, & Jiang 2018)**

Due to hardware limitations already for problem dimensions of 20, however, the FPGA implementation was unrealisable for the testing purposes, therefore was deemed unnecessary to pursue an alternate method for matrix inversion, and was reverted to the LabVIEW implementation by default in the cRIO.

Nevertheless, the cRIO solution proved to be fast enough for all testing purposes and the real time validations in the wind tunnel, as proven in **chapter 3.2**. However, there is room for improvement in the FPGA algorithm optimisation that could lead to, as well as with using a larger FPGA, to future utilisation of these resources that make use among other things, of the parallelisation abilities of the FPGA in order to theoretically lowering even more the computation times of the solver, if deemed necessary.

3. Results

3.1 Performance analysis

Following the procedure described in the sections **2.1.5** and **2.2**, an MPC problem was formulated using a range of different prediction horizons and solved using the same Active set method solver described in section **2.4**, using the cRIO implementation with LabVIEW. 8 different prediction horizons (**N**) were used in 6 different settings, including a fully constrained case that includes both limit constraints as per equation **(18)** and rate constraints as in **(19)** and **(20)**.

Subsequently, each case was calculated 10 times, and the results averaged and recorded as per **tables 3.1.1 and 3.1.2** , then plotted in **Figures 3.1.1 and 3.1.2**. The difference between the three basic constrained cases is the initial position $\mathbf{z_0}$, and the initial state vector $\hat{x}$, which can influence the amount of constraints activated, and the number of iterations in the process, **k**. As it can be observed, there is a strong direct correlation between the computational timing and the number of iterations, as well as the size of the prediction horizon **N**. **'std'** stands for the standard deviation observed in the 10 measured samples.

In **table 3.1.1** there are gathered the results for the standard case of $\mathbf{x_0=[30;0]}$, meaning that the initial rotor speed is 30 rpm higher than the nominal value. "Fully constrained" stands for the cases in which both the angle constraints and the rate constraints are being modelled, while "Angle constrained" stands for the cases in which only the angle constraints are.

	Fully constrained, z_0 = [0.5 0.5 …]			Angle constrained, z_0 = [0.5 0.5 …]			Angle constrained, z_0 = [0 0 …]		
N	**k**	**Avg [µs]**	**Std[µs]**	**k**	**Avg [µs]**	**std[µs]**	**k**	**Avg [µs]**	**Std[µs]**
2	3	633	44	2	354	16	1	164	6
5	3	743	38	2	354	10	1	169	3
10	3	852	37	3	845	24	1	203	32
20	3	1491	118	3	1516	64	1	391	26
50	3	8236	154	3	8638	242	1	2581	73
100	3	73822	972	3	79208	524	1	26145	67
200	3	641934	5633	3	697393	5737	1	227033	2836
500	3	12594636	12785	3	13753352	9772	1	4520316	4138

Table 3.1.1: Limit and Rate constraints for the MPC solver for $x_0 = [30; 0]$

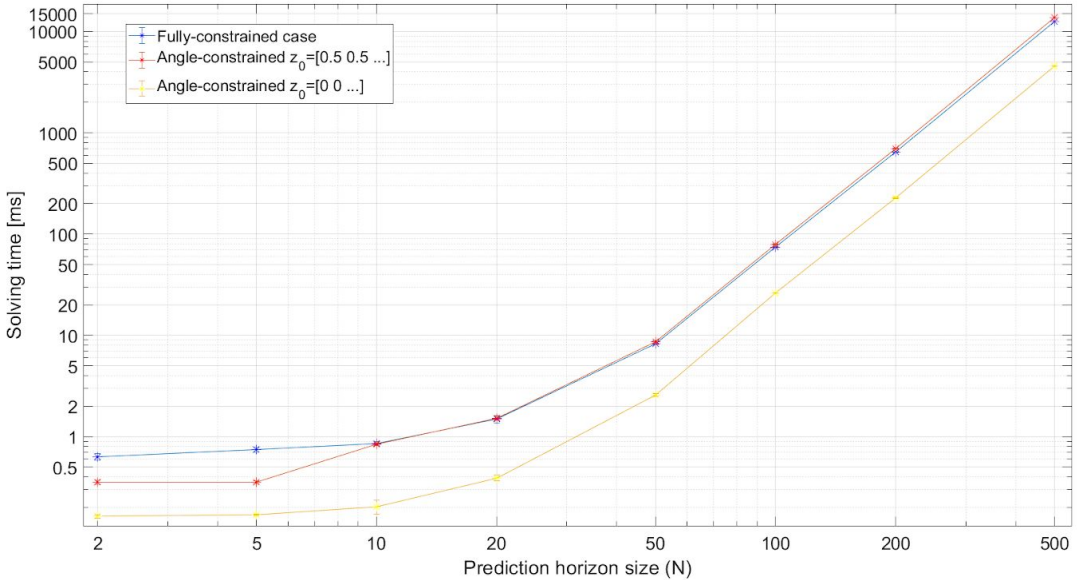

Figure 3.1.1: Solving time vs prediction horizon size performance analysis for the solver in the cRIO for $x_0 = [30; 0]$.

It can be observed in **Figure 3.1.1** (plotted in a log-log axis), how increasing the prediction horizon size derives into an exponential increase of the solving time for the MPC. For the current purposes it is not advised to use prediction horizon sizes larger than 20 or 30, as it often results in computational times larger than 1 ms. This makes sense as the active set method is sensitive to dimensional changes, particularly due to the high computational cost of performing large matrix inverse operations in real time.

	Fully constrained, z_0 = [0.5 0.5 …]			Angle constrained, z_0 = [0.5 0.5 …]			Angle constrained, z_0 = [0 0 …]		
N	**k**	**Avg [µs]**	**Std[µs]**	**k**	**Avg [µs]**	**std[µs]**	**k**	**Avg [µs]**	**Std[µs]**
2	2	356	14	2	373	44	1	166	3
5	3	736	34	3	711	26	1	166	4
10	3	945	56	3	859	34	1	192	10
20	3	1644	110	3	1520	69	1	388	71
50	3	8964	107	3	8709	186	1	2589	100
100	3	82247	851	3	79118	532	1	25825	403
200	3	701014	5443	3	694729	11186	1	226805	3743
500	3	13865136	10759	3	13754944	8464	1	4529866	7814

Table 3.1.2: Limit and Rate constraints for the MPC solver for $x_0 = [0; 30]$

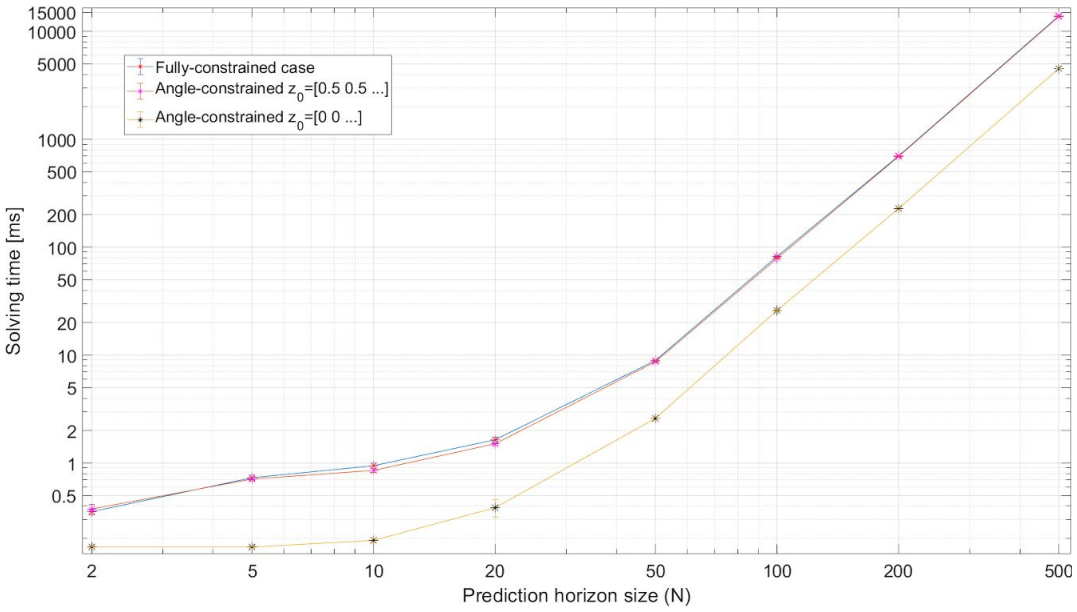

Figure 3.1.2: Solving time vs prediction horizon size performance analysis for the solver in the cRIO for $x_0 = [0; 30]$.

For the case with **x_0=[0;30]** in **table 3.1.2** and **Figure 3.1.2,** it can be observed that it requires a slightly higher computational time on average than in the previous case. Also, the fully constrained and the angle constrained cases observe almost identical computation times, although the change observed by varying the initial state vector is not too significant.

3.2 Validation in the wind tunnel

On the 2nd of December 2019, the MPC algorithm was tested using MoWiTO in the Wind Tunnel at ForWind Oldenburg. A protocol was used in combination with the active grid to produce a combination of 10 mexican hat wind gusts and an approximately 2 minutes long turbulent setting.

3.2.1 Wind Gusts

A total of 10 mexican hat wind gusts were created by the active grid following the protocol as seen in **Figure 3.2.1**. The MPC reacted accordingly in order to keep the rotor speed at the rated value. The loads, generator torque and the rotor speed can be observed in **Figure 3.2.2.** In **Figure 3.2.3,** the same can be observed in detail for one wind gust.

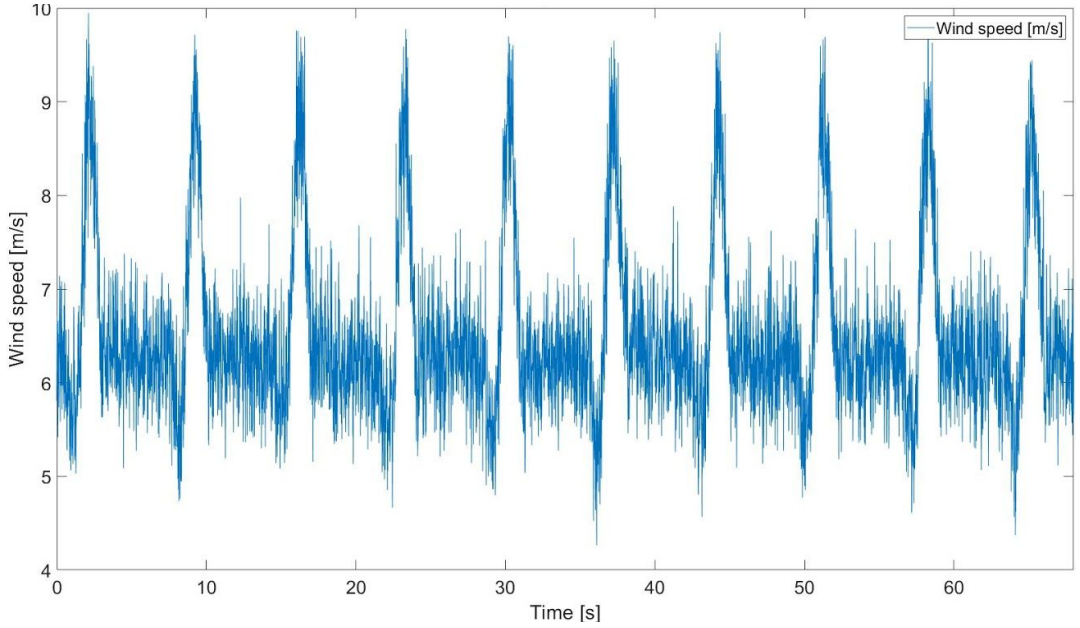

Figure 3.2.1: Wind velocity showing the successive 10-gust test

As it can be observed, the turbine blade flapwise bending moment increases sharply with the wind gusts. This is due to the increase in wind speed. Pitching the turbine blades is done in order to decrease those loads.

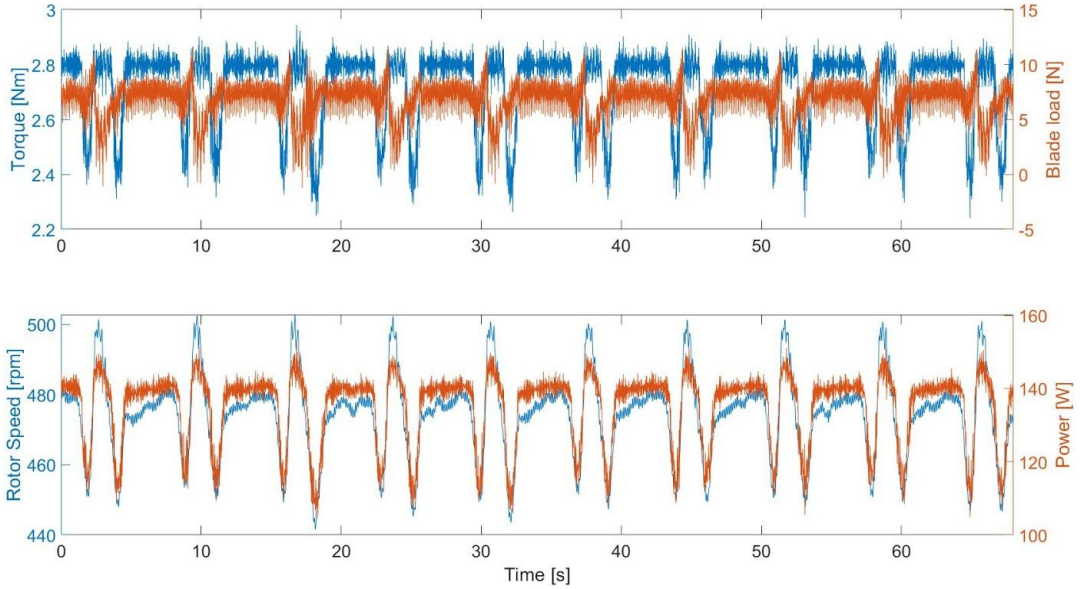

Figure 3.2.2: Detail of torque, loads, rotor speed and power of the turbine for the 10 gusts period of the experiment.

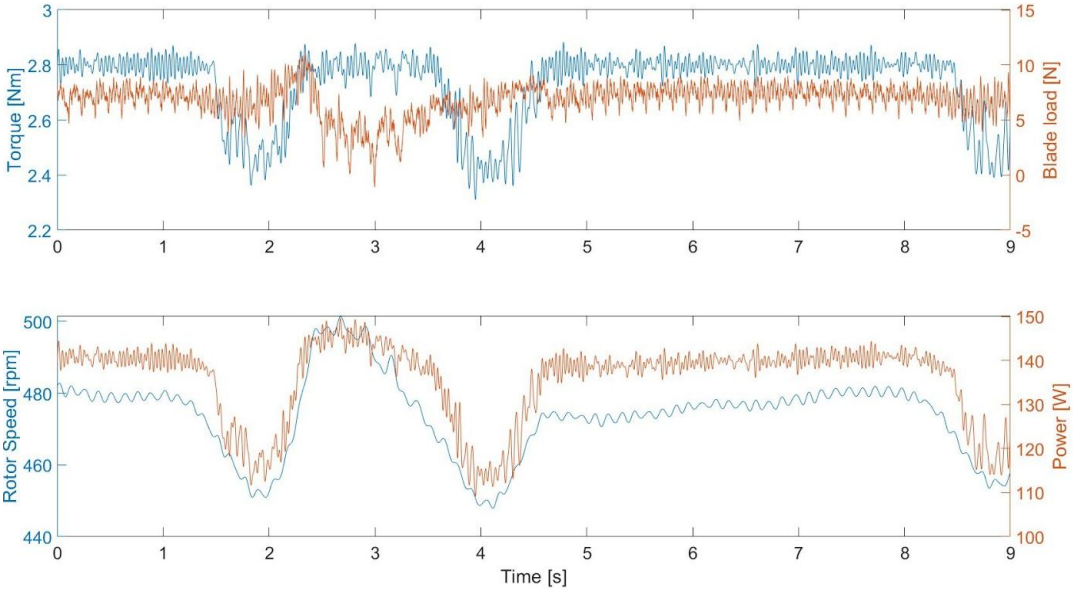

Figure 3.2.3: Detail of torque, loads, rotor speed and power of the turbine for the first gust.

Furthermore, a direct correlation between the RPM of the turbine and the generator torque can be observed. In the tested conditions, the turbine operates mainly in region 3, meaning that the controller keeps a constant generator torque rated value of 2.8 Nm and attempts maintaining a constant rotor speed of 480 rpm, while varying the turbine blades pitch angle in order to keep a steady power generation (140 W). However, a rapid change in the wind speed during the gusts can lead to a drop in the rotor speed. In that case, the baseline torque controller reacts, reducing the torque in order to keep a steady rpm. Transitioning between region 2 and region 3 represents demanding conditions for a wind turbine

controller because it involves transitions between the torque control performed by the baseline controller and the pitch control done by the MPC**.**

The goal of the MPC is to calculate the pitch rate in order to keep the pitch angle of the 3 turbine blades at the rated value, and then communicate this information to the pitch actuators. In **Figure 3.2.4,** the planned pitch angle for turbine blade 1 calculated by the MPC in real time over the measured wind speed for the wind gusts can be observed. As it can be seen, the MPC can react accordingly. There is a short delay caused by the MPC waiting for the rotor speed to catch up before it begins to change the pitch angle.

In **Figure 3.2.5** it can be further observed the comparison between the reference pitch angle and the actual pitch angle. These two parameters are followed very closely with a very small difference, often caused by the mechanical limitations of the motors. One can see how the constraints of the MPC are working since the pitch angle constraint is saturated by the limit at 0º, as per equation **(18)**, which indicates the transition area between region 2 and region 3.

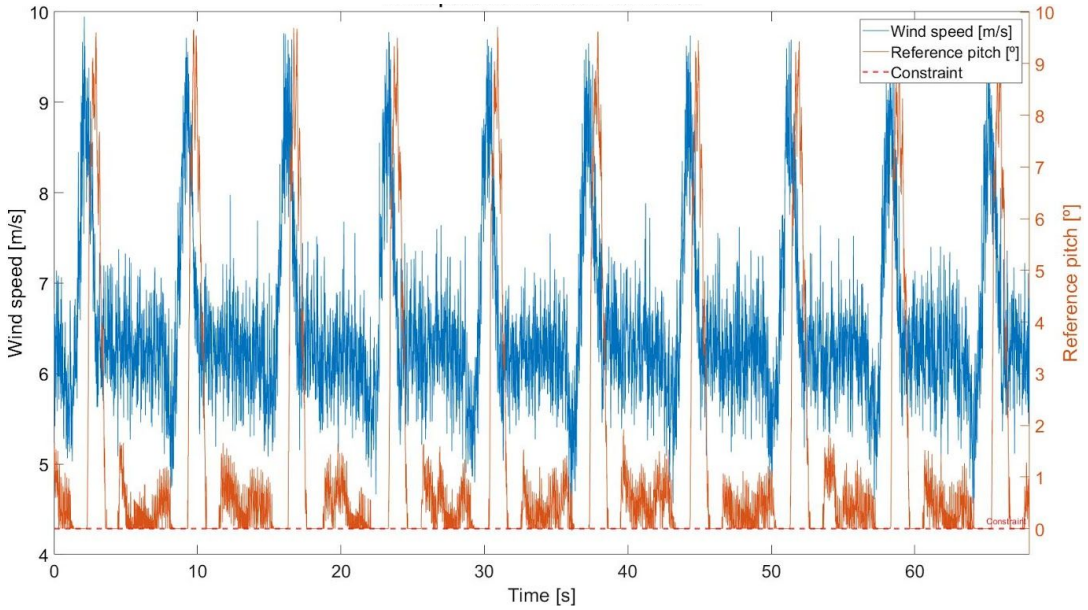

Figure 3.2.4: Planned pitch angle (beta) over measured wind speed for the 10 wind gusts experiment.

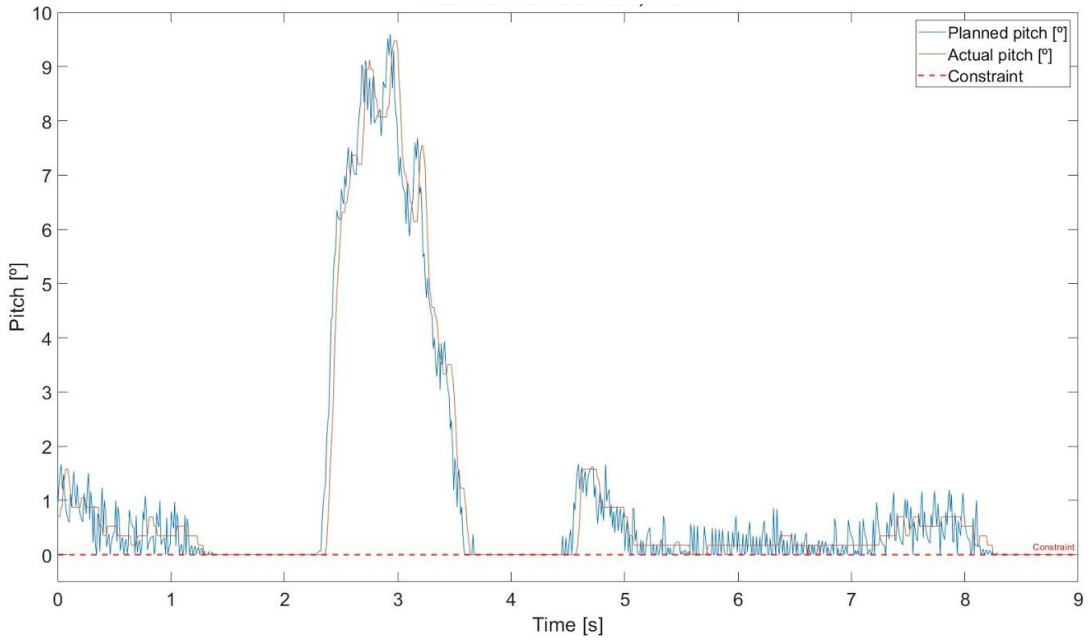

Figure 3.2.5: Detail of the rotor blade pitch (orange) over the planned rotor pitch (blue) from the Test 2 section of the wind gust number 1 in the wind tunnel validation of the MPC.

It can also be corroborated that the rate constraints for the pitch-rate are working when **Figure 3.2.6** is analysed. As it can be seen, the pitch rate never surpasses 85.94 º/s.

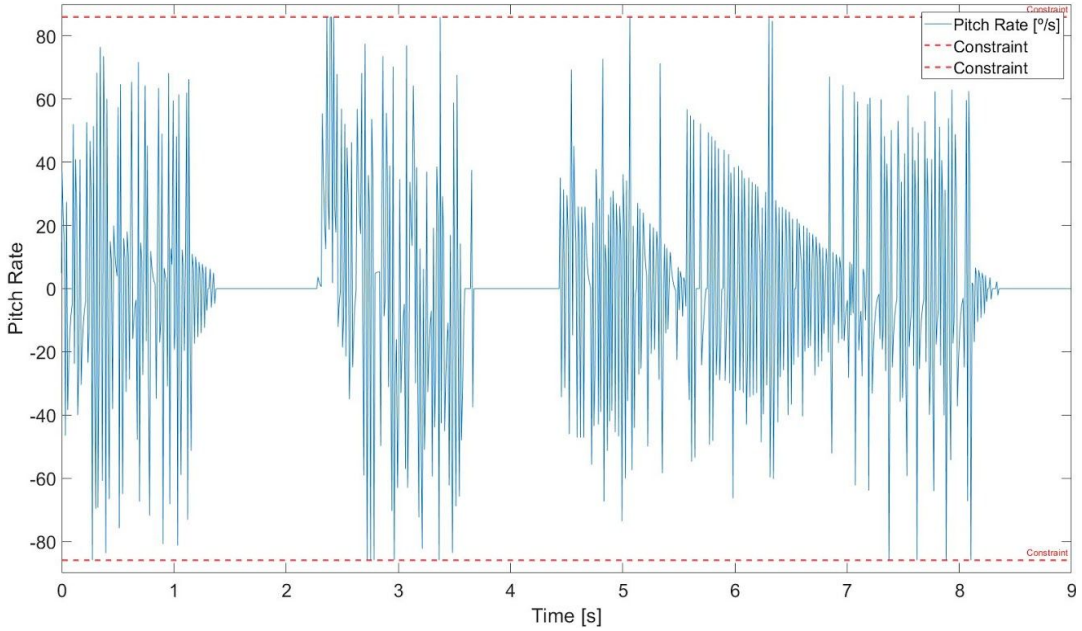

Figure 3.2.6: Pitch-rate for 1-gust section of the experiment with constraints.

In **Figure 3.2.7**, the MPC computing time is plotted on the y-axis for 1 wind gust. The MPC on the compactRIO turned out to be very efficient. Only surpassing 1 ms 0.0441 % of the time, with a maximum of 1082 µs, as detailed in **Table 3.2.1**. The median is close to the minimum, indicating that the execution is fast most of the time, but there are some iterations with much higher execution time, as it can be observed by the mean being higher than the median. That is fast enough to consider increasing the prediction horizon of the MPC. Most of the execution times are in the range shown by the mode of 335 µs, which can be seen by the highest bin in the histogram from **Figure 3.2.8.**

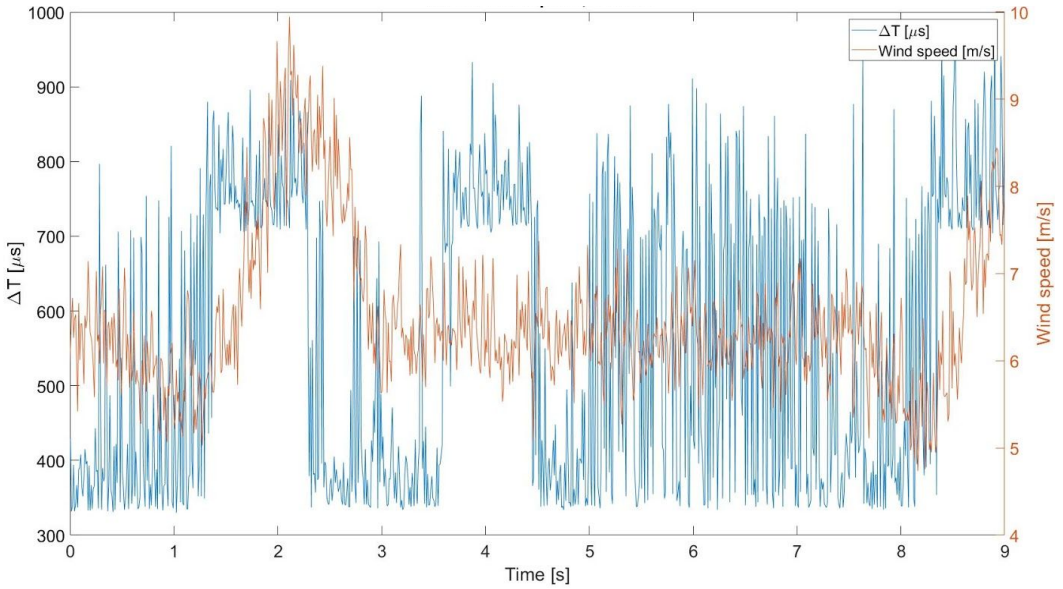

Figure 3.2.7: Timing of the MPC calculations against wind speed for 1 wind gust of the experiment.

Min	Max	Mean	Median	Mode	Std. Dev.
325 µs	1082 µs	558.44 µs	460 µs	335 µs	202.70 µs

Table 3.2.1: Data from the t-solve timing for the 10-gusts section of the wind tunnel experiment

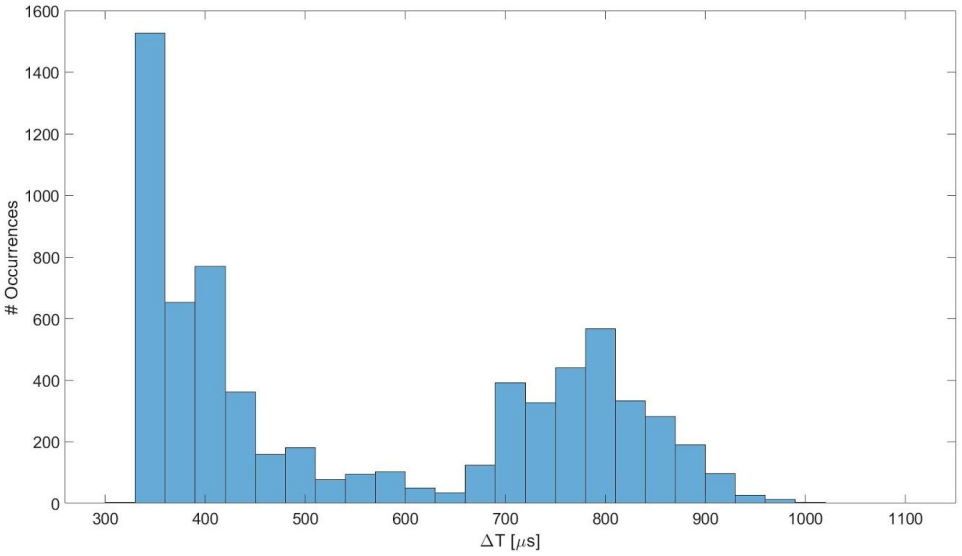

Figure 3.2.8: Histogram of the probability of occurrence for different execution times in the wind gusts test.

3.2.2 Turbulent inflow

The experiment consisted of approximately 200 seconds of turbulent wind conditions generated with the wind tunnel's active grid, as observed in **Figure 3.2.9** and in more detail for a small excerpt of the turbulent period in **Figure 3.2.10**. In these two plots can also be noted that the MPC reacts accordingly and follows the base constraint at 0º.

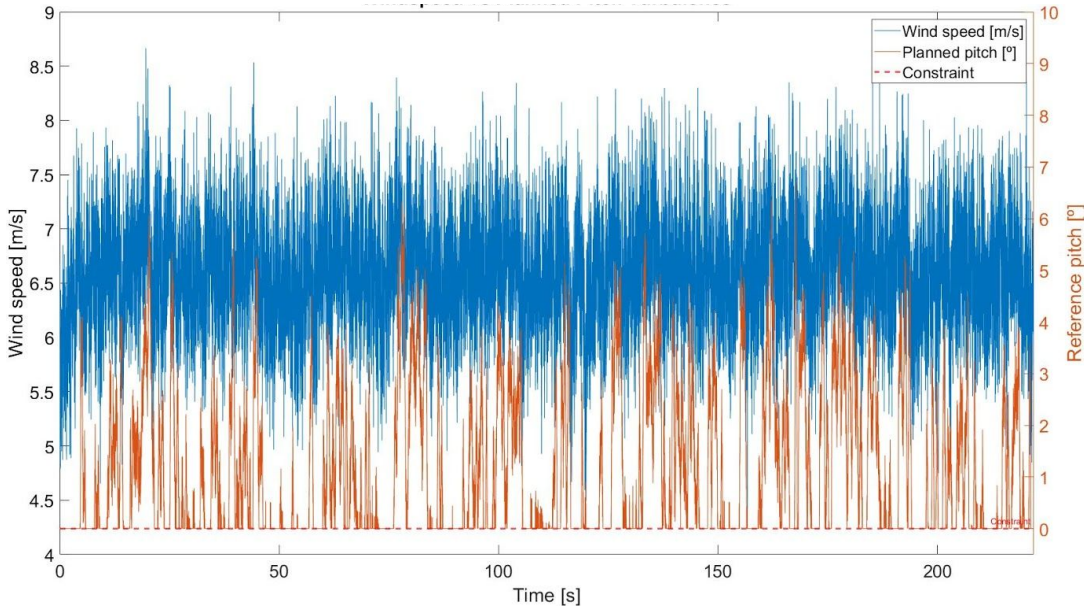

Figure 3.2.9: Planned pitch angle (beta) over measured wind speed for the turbulence test.

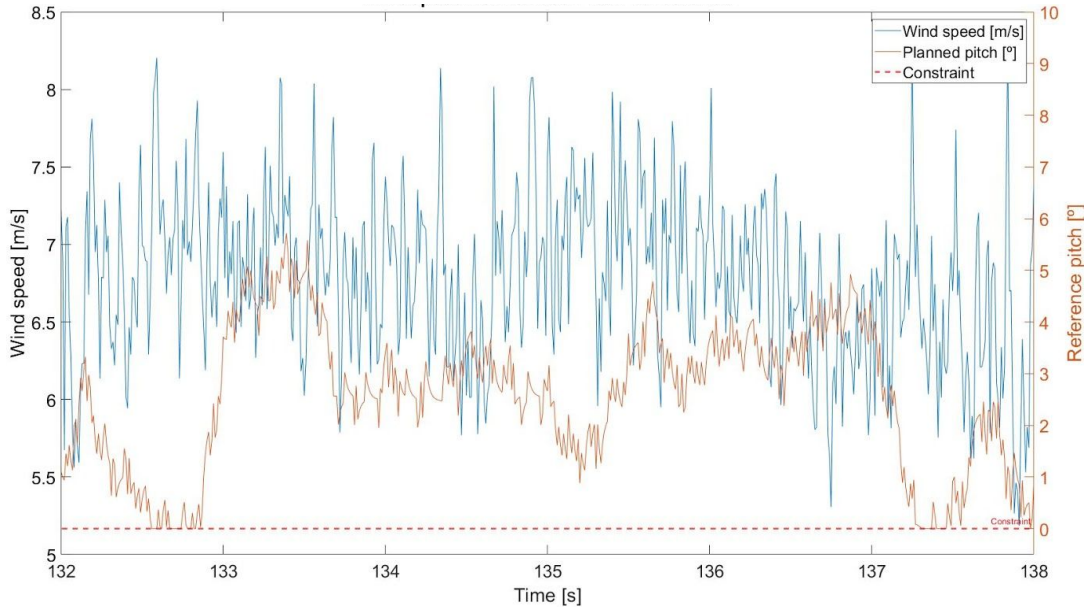

Figure 3.2.10: Planned pitch angle (beta) over measured wind speed for detail of the turbulence test.

By plotting the difference of the pitch angle over the sampling time, **Figure 3.2.11** is obtained, which also proves the correct functioning of the MPC in establishing the rate constraints. In **Figure 3.2.12,** it can also be observed the behavior of the MPC in the turbine with its torque, blade loads, rotor speeds and power plotted against time for a small excerpt of the turbulence experiment. The occasional transitions between region 2 and region 3 can be observed, while the controller keeps the rotor speed around its rated value, indicating a correct controller behavior.

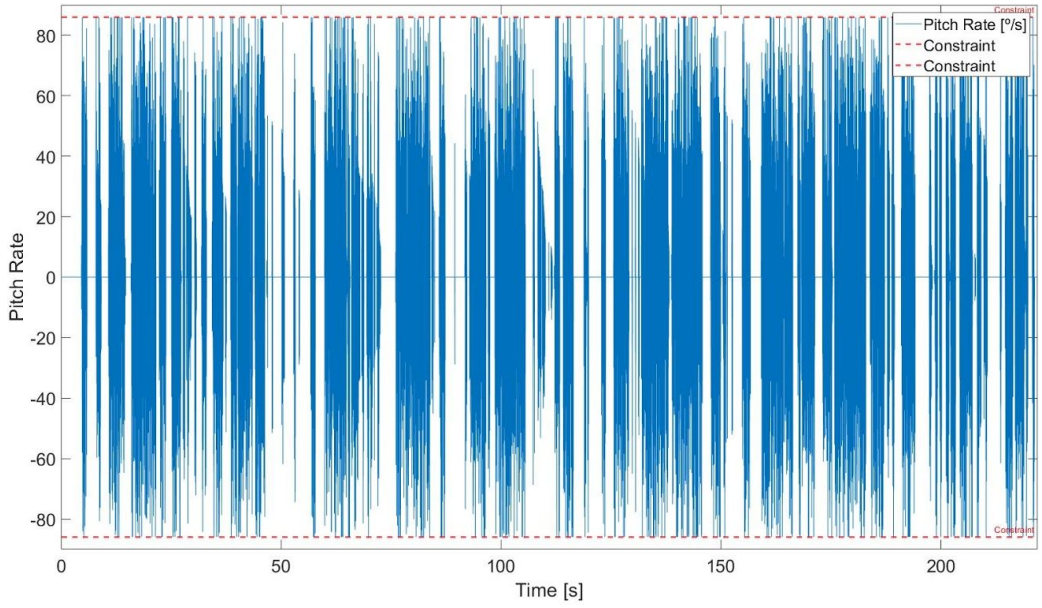

Figure 3.2.11: Pitch-rate for the turbulent section of the experiment with constraints.

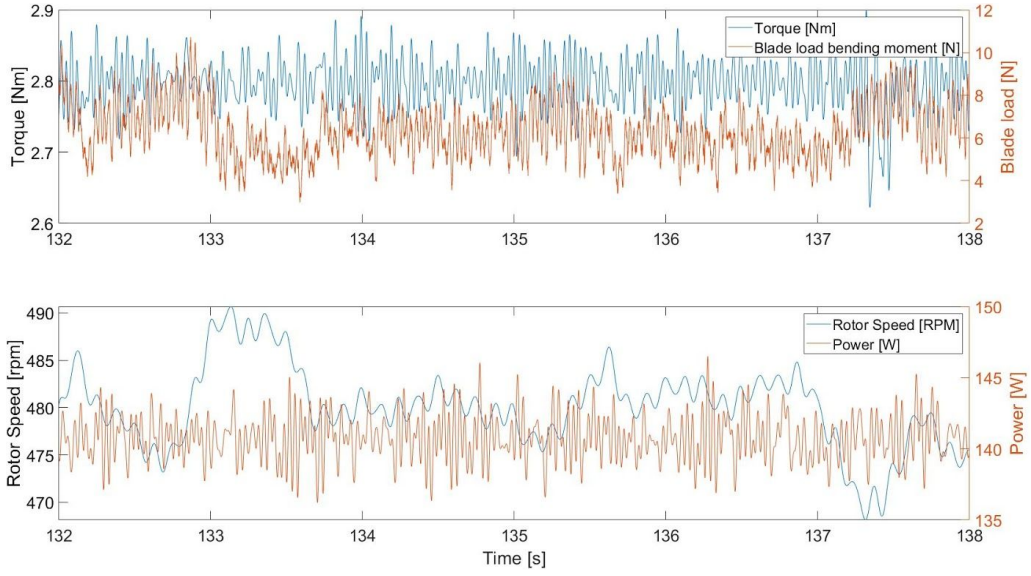

Figure 3.2.12: Detail of torque, loads, rotor speed and power of the turbine for the turbulence period of the experiment.

Finally, the MPC execution time is calculated and subsequently plotted in **Figure 3.2.13.** Surprisingly, the MPC reacts slightly faster on average, with lower mean, minimum, maximum and lower standard deviation timings than in the gusts case, while the mode (most common value) remains quite stable, as shown in **table 3.2.2**, which shows how this erratic behavior can be handled easier by the MPC than the extreme mexican hat gusts from the earlier experiment.

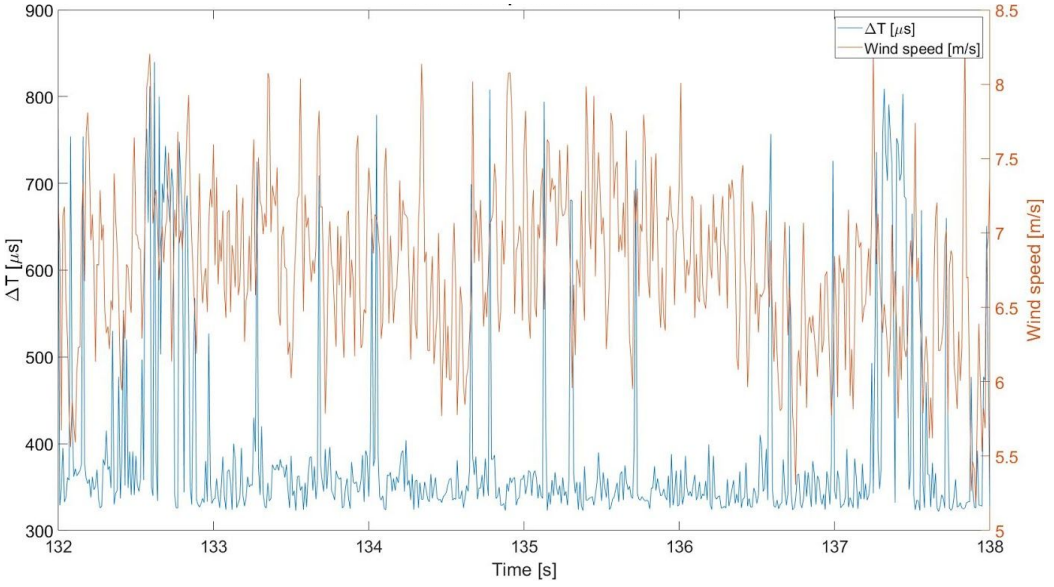

Figure 3.3.13: T-solve of the MPC vs measured wind speed for the turbulent section of the experiment..

Min	Max	Mean	Median	Mode	Std. Dev.
319 µs	982 µs	534.87 µs	472 µs	327 µs	189.74 µs

Table 3.2.2: Data from the t-solve timing for the turbulent section of the wind tunnel experiment

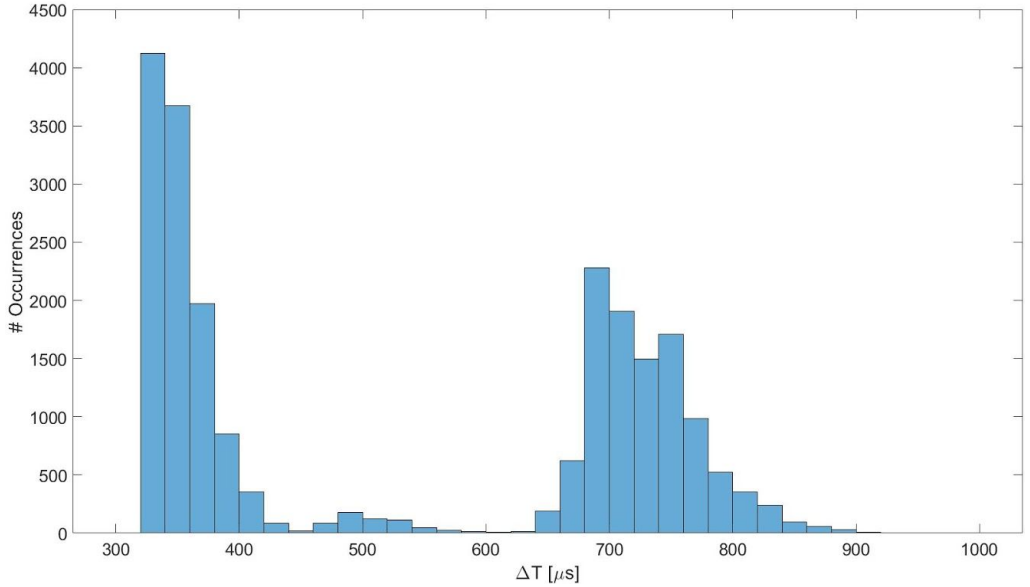

Figure 3.2.14: Histogram of the probability of occurrence for different execution times in the turbulent inflow test.

4. Discussion and conclusion

A solver was implemented using the active set method as an MPC controller that models a model wind turbine successfully onboard a cRIO board using labVIEW software and integrated with the rest of the MoWiTO control software, then experimentally tested in the wind tunnel as well as numerically tested using matlab and LabVIEW.

Overall it can be concluded from the testing performed in the performance analysis **(chapter 3.1),** that the cRIO CPU offers fast enough solving time for problems with reasonable sizes. This was further corroborated with the empirical results obtained during the validation experiments in the wind tunnel with MoWiTO **(chapter 3.2).** An ASM solver onboard a cRIO is enough for efficiently controlling the blade pitch angles of a wind turbine with conditions of turbulence or highly stressing wind gusts.

The computation times remain sufficiently low for all the performed experiments and operations, with a maximum of 1086 μs and a mean of 558 μs for the wind gusts experiments, and a maximum of 986 μs with a mean of 535 μs for the turbulence experiment, which is much lower than the maximum expected value of 10 ms given by the controller sampling time. This gives a lot of room for change, such as increasing the solver prediction horizon even further, which increases the computational time drastically as seen in **chapter 3.1.**

It was also attempted to implement the code onboard the FPGA module, as it is explained in **chapter 2.4,** and while theoretically would work even more efficiently than the cRIO solution, in the end was unable to compile it due to a lack of hardware capabilities in the used FPGA to implement even for smaller optimisation problems. There is a possibility that with both the usage of a bigger FPGA module and with further code optimisation this might be possible, however this was deemed unnecessary as the cRIO code managed to fulfill all the prior expectations.

As previously seen in **table 3.1.1** and later observed in the data from the validation in **chapter 3.2**, the number of activated constraints highly affects the computational time required for the MPC to find optima. This is due to the ASM requiring additional loops in order to deal with the extra constraints being activated, as it adds one constraint at a time, requiring to repeat the calculations a number of times. In the example this is good, as constraints are not being activated very frequently, and there is rarely a case when angle constraints and rate constraints are being activated at the same time, which keeps the computational times sufficiently low.

Although the performance of the MPC was tested experimentally to great success, no work has been done in proving the stability conditions. It can be assumed that it is stable, but further work could be done in order to theoretically establish that the MPC is stable.

Furthermore, the developed solver is not only limited for pitch control, and it could be extended to different wind turbine control applications including torque, turbine speed or even to wind farm control operations with future work.

5. References

1. Berger F, Kröger L, Onnen D, Petrović V and Kühn M 2018 *Journal of Physics: Conference Series* 1104 012026
2. Boyd, Stephen., Vandenberghe, Lieven. “Convex Optimization” Cambridge University Press 2004
3. Burton, Tony. “*Wind Energy Handbook” . Wiley, 2011*
4. C. V. Rao, S. J. Wright and J. B. Rawlings. “Application of Interior-Point Methods to Model Predictive Control” 1998
5. Fraunhofer ISE “Net public electricity generation in Germany in 2018” www.energy-charts.de
6. Jain A, Schildbach G, Fagiano L, Morari M “On the Design and Tuning of Linear Model Predictive Control for Wind Turbines.” Renewable Energy , vol. 80, 2015, pp. 664–673., doi:10.1016/j.renene.2015.02.057.
7. Jonkman J, Butterfield S, Musial W and Scott G 2009 Definition of a 5-MW reference wind turbine for offshore system development Tech. Rep. NREL/TP-500-38060 NREL, National Renewable Energy Laboratory
8. Kröger L, Frederik J, van Wingerden J W, Peinke J and Hölling M 2018 *Journal of Physics: Conference Series* 1037
9. Lau, Mark S. K., Yue S. P.,Ling K. V. Maciejowski J. M.. “A Comparison of Interior Point and Active Set Methods for FPGA Implementation of Model Predictive Control.” 2009 European Control Conference (ECC) , 2009, doi:10.23919/ecc.2009.7074396.
10. Lofberg, Johann et al. “Model Predictive Control” Lecture slides, ETH Zürich, 2010.
11. Mirzaei, Mahmood, and Morten H. Hansen. “A LIDAR-Assisted Model Predictive Controller Added on a Traditional Wind Turbine Controller.” 2016 American Control Conference (ACC) , 2016, doi:10.1109/acc.2016.7525110.
12. Petrović, V, Berger F, Neuhaus L, Hölling M, Kühn M. “Wind Tunnel Setup for Experimental Validation of Wind Turbine Control Concepts under Tailor-Made Reproducible Wind Conditions.” *Journal of Physics: Conference Series* , vol. 1222, 2019, p. 012013., doi:10.1088/1742-6596/1222/1/012013.

13. Rajendran S, Jena D. “Control of Variable Speed Variable Pitch Wind Turbine at Above and Below Rated Wind Speed.” Journal of Wind Energy, Hindawi, 22 Oct. 2014, www.hindawi.com/journals/jwe/2014/709128/.
14. S. Gros and A. Schild. “Real-time economic nonlinear model predictive control for wind turbine control.” Int. Jrnl. of Control , 90(12):2799-2812, 2017.
15. Schlipf D., Schlipf D. J., Kühn M., “Nonlinear Model Predictive Control of Wind Turbines Using LIDAR.” Wind Energy , vol. 16, no. 7, 2012, pp. 1107–1129., doi:10.1002/we.1533.
16. Sinner, Michael N., Petrović, Vlaho et al. “Experimental Testing of a Preview-enabled Model Predictive Controller for Blade Pitch Control of Wind Turbines”. (Manuscripting)
17. Sinner, Michael N., Petrović, Vlaho et al. “Wind Tunnel Testing of an Optimal Feedback/Feedforward Control Law for Wind Turbines.” (Manuscripting)
18. Spencer, Martin D. et al. “Model predictive control of a wind turbine using short-term wind field predictions”
19. Windeurope “Wind energy in Europe in 2019 - Trends and statistics” windeurope.org
20. Wright, Stephen J. “Applying new optimization algorithms to model predictive control”
21. Xu, Yunwen, Li D.,Xi Y., et al. “An Improved Predictive Controller on the FPGA by Hardware Matrix Inversion.” IEEE Transactions on Industrial Electronics , vol. 65, no. 9, 2018, pp. 7395–7405., doi:10.1109/tie.2018.2798563