Technical University of Denmark

Bs.c. thesis

Model Predictive Control of a Wind Turbine

Authors:
Dennis Holm
s103240

Hasham Mirza
s103270

June 28, 2013

Technical University of Denmark
Applied Mathematics and Computer Science
Building 303B, DK-2800 Kgs. Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary

The goal of this thesis is to examine the control of a single wind turbine. The demand in power from wind turbines is increasing. But while everyone demands more power production from wind turbines no one wants to have wind turbines in their own backyard. Therefore it is essential that when a turbine is placed somewhere that it will produce the most power with less wear as possible. Another need for controlling the power production of wind turbines is within the implementation of the Smart Grid all over Europe. With the Smart Grid it is wanted to control the flexible demand in power by increasing automatisation, distant control and the exchange of data with other IT systems. By controlling the wind turbines they can be set to steer the power production towards a certain demand and thus help the Smart Grid in doing its best.

A wind turbine model is presented and linearized. The model has been tested by simulations and then verified. The Model Predictive Control toolbox in MATLAB was chosen to help design the controller for the model. Model Predictive Control has been investigated as a way to steer the wind turbine towards the wanted output and still be in the feasible area of the constraints that the model has. It has been shown through various simulations that such a controller is a reliable method.

Preface

This thesis was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfilment of the requirements for acquiring the B.Sc. in Mathematics and Technology.

The thesis deals with control and optimization of a wind turbine. A model for the wind turbine is presented, linearized, simulated and varyfied. The MPC toolbox in MATLAB is used to design a controller for the wind turbine which then can be steered toward a higher power production and less load on the tower.

The thesis consists of four chapters. Chapter 1 which deals with the theory of MSD systems that are used twice in the turbine model. Chapter 2 which describes the wind turbine model, linearizes it and then chapter 3 holds the simulations and varification of the model. Chapter 4 is about the theory in MPC, the use of MPC toolbox in MATLAB and simulations of the turbine with controller implemented.

Lyngby, June 28, 2012

Hasham Mazhar Mirza & Dennis Søren Holm

Acknowledgements

We would like to thanks our supervisor Assoc. Prof., Ph.D., John Bagterp Jørgensen, DTU Compute for help, guidance and inspirations throughout this project.
We would also like to thank M.Sc. student, Rasmus Dalgas Rasmussen, for the detailed answers to our questions about his thesis on the same subject.
Last but not least, we would like to thank Control Engineer, Ph.D., Tobias Gybel Hovgaard, Vestas Wind Systems A/S for the help in getting wind speed data for our simulations.

Contents

List of Figures

1 | Mass-Spring-Damper Systems

This chapter is an introduction to the mass-spring-damper systems, which are a part of the NREL 5MW model. The MSD systems will be described and the stability in a MSD system will be investigated.

1.1 The MSD System

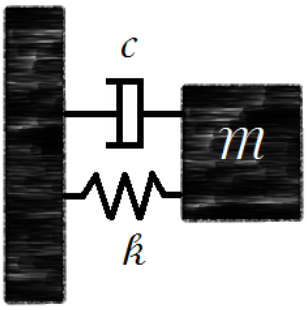


Figure 1.1: Mass-Spring-Damper system (MSD-system)

A mass, m, is attached to a spring of stiffness k, and a viscous damper of damping coefficient c. Recall Hooke's law for a compressed spring

$$F_s = -kx \tag{1.1}$$

The damping force can be described as

$$F_d = -cv = -c\dot{x} \tag{1.2}$$

Applying Newton's second law of motion

$$F_{tot} = ma = m\ddot{x} \tag{1.3}$$

Where x is the displacement of the mass, $\dot{x}$ is the velocity of x and $\ddot{x}$ is the acceleration. Combining (1.1), (1.2) and (1.3) gives the differential equation $F_{tot} = F_s + F_d$ or

$$m\ddot{x} = -kx - c\dot{x} \tag{1.4}$$

By dividing by m this can be rearranged into

$$\ddot{x} + \frac{c}{m}\dot{x} + \frac{k}{m}x = 0 \tag{1.5}$$

Let $\omega_0 = \sqrt{\frac{k}{m}}$ and $\zeta = \frac{c}{2\sqrt{mk}}$ then the differential equation can be written as [Wika]

$$\ddot{x} + 2\zeta\omega_0\dot{x} + \omega_0^2 x = 0 \tag{1.6}$$

It is no coincidence that ω_0 and ζ are chosen that way. Actually ω_0 is the natural frequence of the system and ζ is the damping ratio.

To solve the system, let $x = e^{\gamma t}$ where $\gamma \in \mathbb{C}$ and the characteristic equation becomes

$$\gamma^2 + 2\zeta\omega_0\gamma + \omega_0 = 0 \tag{1.7}$$

Solving this equation will give two roots.

Figure 1.2 shows the result of simulating various systems with constant ζ and diverse values of ω_0.

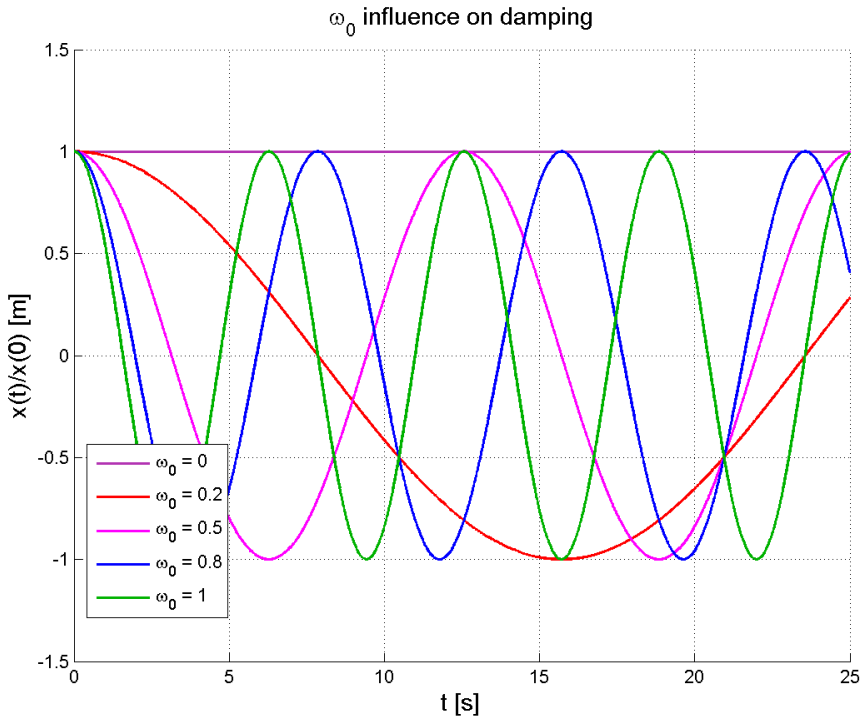


Figure 1.2: The influence of ω_0 in the MSD-system

By examining the oscillations in the figure above, it is easy to see that by increasing ω_0 the frequency of the oscillation increases aswell.

Doing the same simulations but now with diverse ζ with constant ω_0, the influence that ζ has on the system can be explained as the damping ratio.

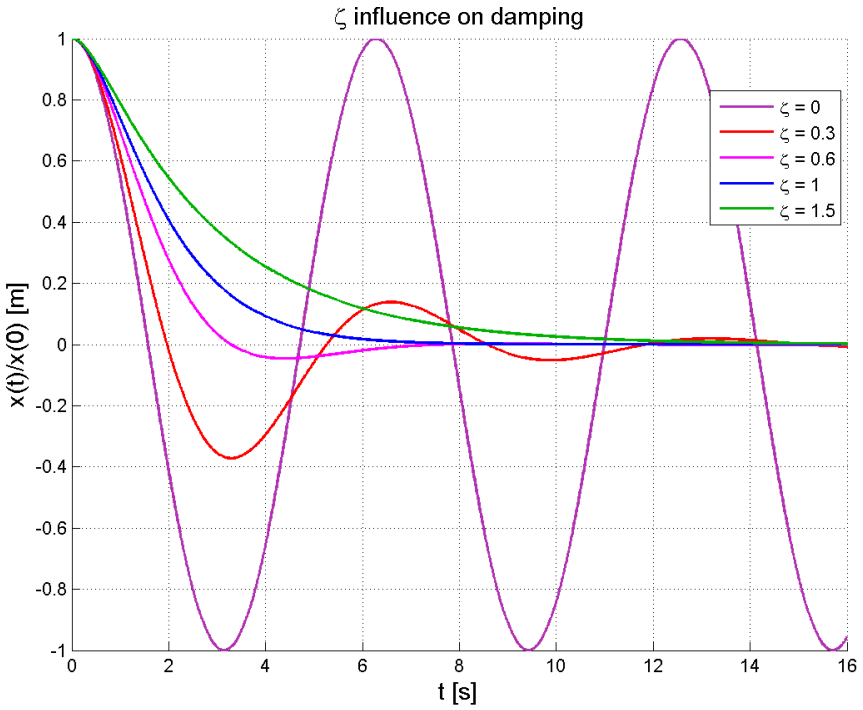


Figure 1.3: The influence of ζ in the MSD-system

The damping can be explained in three different scenarioes. [Wika]

The critical damped system with $\zeta = 1$ and a real double root γ. This system is not oscillating and converges to zero as fast as possible. This can be compared to the damping in a door closer.

The over-damped system where $\zeta > 1$ and there are two different real roots. This system is also not oscillating but converges slower towards zero than the critical-damped system. Related to the door closer example would mean that the door would close slower.

The under-damped system with $0 \leq \zeta < 1$ and two complex roots. For $0 < \zeta < 1$ the system oscillates in the begining but converges to zero. This would mean that the door closer would close fast and the door would hit the door frame with a forceful velocity or continue oscillating in case of a swinging door. For $\zeta = 0$ there is no damping and the system is describing a harmonic oscillation.

1.2 Stability in MSD-systems

The second order differential equation (1.6) can be set up as two coupled first order differential equations

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_0^2 & -2\zeta\omega_0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{1.8}$$

The eigenvalues of the matrix can be used to dermine the stability of the system. It all depends upon the existence of real and imaginary parts of the eigenvalues, together with the sign of the real parts and their actual values. When the eigenvalues are complex numbers, i.e. on the form $a + bi$ where $a \in \mathbb{R}$, $b \in \mathbb{R} \setminus 0$ and i is the complex number $\sqrt{-1}$, there are three cases of oscillatory behavior. The three cases are when a is positive, negative or zero. The system is always oscilatory because $b \neq 0$. [KMNWG]

- When the real part is negative, the system is stable and behaves damped. The amplitude of the oscillations decreases as a function of time.
- When the real part is positive, the system is unstable and the amplitude of the oscillations will increase as a function of time.
- When the real part is zero, the system behaves like an undamped oscillation, i.e. the amplitude is constant.

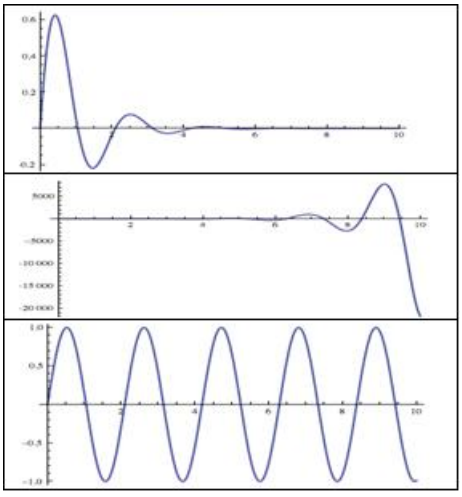


Figure 1.4: Graphs of systems with different eigenvalues. From the top down: 1) Real part is negative 2) Real part is positive 3) Real part is zero.

2 | Wind Turbine Model

The equations used to model a wind turbine will be introduced in this chapter. In this thesis the wind turbine is chosen to be an NREL 5MW wind turbine. The model will then be linearized and set up as a state-space model.

2.1 Aerodynamics

To begin with, the power that is extracted by the wind turbine from the passing wind is described. The following equation describes how much power the turbine extracts from the wind.

$$P = \frac{1}{2}\rho\pi R^2 v^3 C_P \tag{2.1}$$

Here P is the power, ρ is the density of the air, R is the radius of the circle formed by the blades of the wind turbine. v is the wind speed. C_P is the efficiency coefficient defined from the pitch of the blades θ and the tip-speed-ratio of the wings λ . [Hen07]

The aerodynamic torque is then given as

$$Q_r = \frac{P}{\Omega_r} \tag{2.2}$$

Where Ω_r is the angular velocity.
Below is given the thrust force, in this equation C_T can be observed, which is defined by the pitch of the blades θ and the tip-speed-ratio of the wings λ , just like the coefficient C_P.

$$Q_t = \frac{1}{2}\rho\pi R^2 v^2 C_T \tag{2.3}$$

It should be noted that C_P and C_T are not analytically derived. These are specific for each wind turbine model, and are mostly looked up in a table and are derived from measurements. Plots of C_P and C_T as functions of θ and λ are shown below. [AEO]

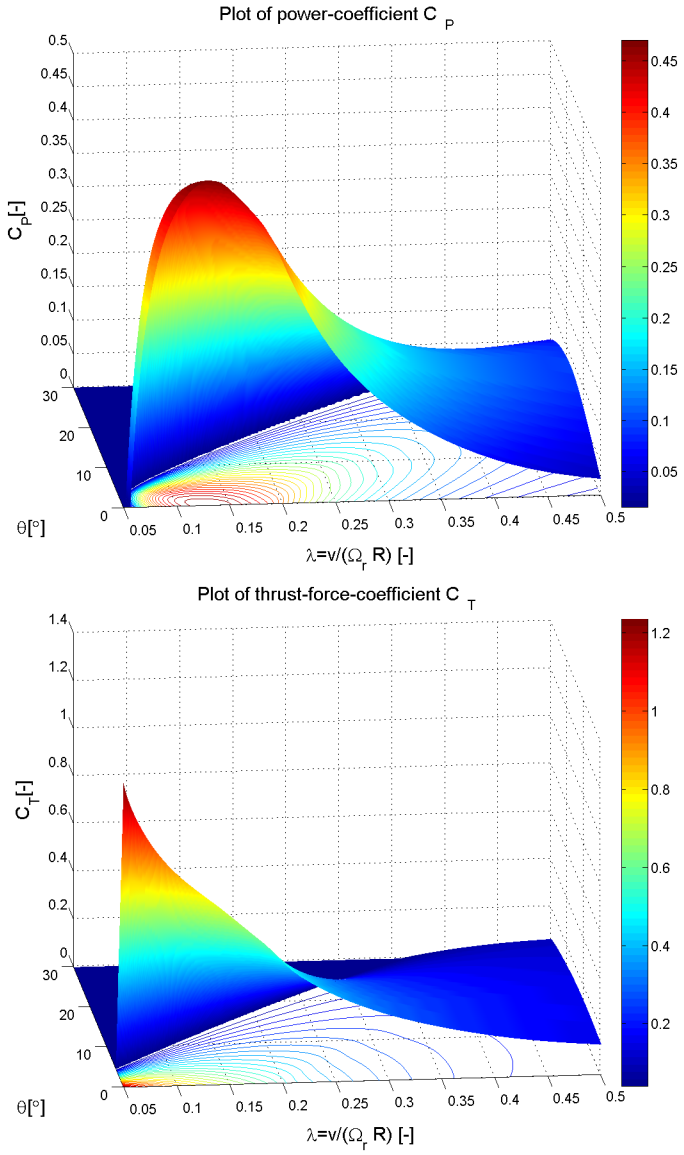


Figure 2.1: C_P and C_T plots

Scripts for plotting are provided in appendix C.1.1.5

2.2 Load on the Tower

The flexible wind turbine experiences wind from every direction that causes it to oscillate from side to side and forth and back, for the sake of simplicity it is assumed that the wind turbine only oscillates forth and back.

The displacement is denoted x_t. [Hen07]

$$\ddot{x}_t = \frac{1}{M_t}(Q_t - D_t\dot{x}_t - K_t x_t) \tag{2.4}$$

In the above equation M_t is the mass of the tower, D_t is the damping constant and K_t is the spring contant. When formed as a system of two coupled differential equations it can be written as.

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\frac{K_t}{M_t} & -\frac{D_t}{M_t} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{M_t} \end{bmatrix} Q_t \tag{2.5}$$

Now that the swaying is a part of the model, it is necessary to look at the relative velocity because if the wind turbine is moving forward the relative wind speed is higher than the actual wind speed and if the wind turbine is moving backwards the relative wind speed is lower than the actual wind speed.

$$v_r = v - \dot{x}_t \tag{2.6}$$

So when the load on the tower is taken into consideration, v should be replaced by v_r.

2.3 Actuators

Actuators are a necessity because of the lack of the ability to change the pitch and the torque immediately.

2.3.1 Pitch of the Blades

The pitch of the blade is controlled by a motor. The actuator can be described by a second order differential equation, where θ_{ref} is the desired pitch and θ is the actual pitch.

$$\ddot{\theta} = \omega_n^2\theta_{ref} - 2\omega_n\zeta\dot{\theta} - \omega_n^2\theta \tag{2.7}$$

In the above equation ω_n is the natural pitch frequency, θ_{ref} is the desired angle that θ turns towards. ζ is the damping of the pitch actuator. This

second order differential equation can also be written as a system of two coupled first order differential equations. [Hen07] [LM06]

$$\begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_n^2 \end{bmatrix} \theta_{ref} \tag{2.8}$$

2.3.2 The Generator

The electromagnetic torque can be described by a first order differential equation

$$Q_{g,ref} = \tau \dot{Q}_g + Q_g \tag{2.9}$$

To achieve the desired form, this is rewritten

$$\dot{Q}_g = -\frac{1}{\tau} Q_g + \frac{1}{\tau} Q_{g,ref} \tag{2.10}$$

Here Q_g is the actual torque and Q_{ref} is the desired torque that Q_g steers towards. τ is a time constant. [LM06]

2.3.3 The Drive Train

The drive train is the inner gearing of the wind turbine, it is here that power is transformed to the state of electrical energy and this is modelled by three first order differential Equations.

The first equation is

$$\dot{\Omega}_r = \frac{1}{I_r}(Q_r - \Delta\phi K_s - \Delta\dot{\phi} D_s) \tag{2.11}$$

This equation describes the derivative of the angular velocity of the rotor. Here I_r is the inertia of the rotor and $\Delta\phi$ is the torsion angle of the drive shaft, K_s is the spring constant of the drive shaft, and D_s is the damping/friction constant of the driveshaft.

The second equation is

$$\dot{\Omega}_g = \frac{1}{I_g}(-Q_g + \Delta\phi \frac{K_s}{N_g} + \Delta\dot{\phi} \frac{D_s}{N_g}) \tag{2.12}$$

This equation describes the derivative of the angular velocity of the generator. Here I_g is the inertia of the generator and N_g is the gear ratio.

The third equation describes the derivative of the of the torsion angle of the driveshaft.

$$\Delta\dot{\phi} = \Omega_r - \frac{1}{N_g}\Omega_g \tag{2.13}$$

The result is the system of equations shown below. It is a result of inserting (2.13) into (2.12) and (2.11).

$$\begin{bmatrix} \dot{\Omega}_r \\ \dot{\Omega}_g \\ \Delta\dot{\phi} \end{bmatrix} = \begin{bmatrix} -\frac{D_s}{I_r} & \frac{D_s}{I_r N_g} & -\frac{K_s}{I_r} \\ \frac{D_s}{I_g N_g} & -\frac{D_s}{I_g N_g^2} & \frac{K_s}{I_g N_g} \\ 1 & -\frac{1}{N_g} & 0 \end{bmatrix} \begin{bmatrix} \Omega_r \\ \Omega_g \\ \Delta\phi \end{bmatrix} + \begin{bmatrix} \frac{1}{I_r} & 0 \\ 0 & -\frac{1}{I_g} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} Q_r \\ Q_g \end{bmatrix} \quad (2.14)$$

This is how the actuators are described. [LM06]

2.3.4 Power

To calculate the power produced by the wind turbine, it is necessary to look at the torque experienced by the generator.

The generator torque Q_g is adjusted in such a way in the model that $\hat{Q}_g = Q_g \cdot \tau$ is the generator torque.

Therefore the power P produced by the generator i.e. the wind turbine is. [Hen07]

$$P = \Omega_g Q_g = \frac{\Omega_g \hat{Q}_g}{\tau} \quad (2.15)$$

2.4 Linearization

The objective has been to obtain a linear model all along, so that the linear model can be implemented, but the state space model obtained so far has some defects concerning its linearization. The problem is in the tip-speed-ratio, which is dependent on both v_r and Ω_r.

$$\lambda = \frac{v_r}{\Omega_r R} \tag{2.16}$$

The tip-speed-ratio λ is contained in both the aerodynamic torque, Q_r, and the thrust force, Q_t, which both are functions of λ which of course make them non-linear.
Again the objective is to linearize the model. In order to complete that task it is nececary to linearize both Q_r and Q_t. [AEO]

$$Q_r = \frac{P_r}{\Omega_r} = \frac{\frac{1}{2}\rho\pi R^2 v^3 C_p(\lambda,\theta)}{\Omega_r} = \frac{\frac{1}{2}\rho\pi R^2 v^3 C_p(\frac{v_r}{R\Omega_r},\theta)}{\Omega_r} \tag{2.17}$$

The linearization is done by executing the first order Taylor expansion with respect to the variables v_r, ω_r and θ for some linearization points v_{r0}, Ω_{r0} and θ_0

$$Q_r \approx Q_{r0} + \frac{\partial Q_r}{\partial v_r}\bigg|_{v_r 0} \Delta v_r + \frac{\partial Q_r}{\partial \Omega_r}\bigg|_{\Omega_r 0} \Delta\Omega_r + \frac{\partial Q_r}{\partial \theta}\bigg|_{\theta_0} \Delta\theta \tag{2.18}$$

In equation (2.18) Δ means the difference between the variable and a chosen linearized point for instance

$$\Delta v_r = v_r - v_{r0}$$

The individual first order partial derivatives of (2.18) is

$$\begin{aligned}
\frac{\partial Q_r}{\partial v_r}\bigg|_{v_{r0}} &= \frac{1}{\Omega_r}\frac{\partial P_r}{\partial v_r}\bigg|_{v_{r0}} \\
\frac{\partial Q_r}{\partial \Omega_r}\bigg|_{\Omega_{r0}} &= \frac{\frac{\partial P_r}{\partial \Omega_{r0}}\Big|_{\Omega_{r0}} \Omega_{r0} - P_{r0}}{\Omega_{r0}^2} = \frac{1}{\Omega_{r0}}\frac{\partial P_r}{\partial \Omega_r}\bigg|_{\Omega_{r0}} - \frac{P_{r0}}{\Omega_{r0}^2} \\
\frac{\partial Q_r}{\partial \theta}\bigg|_{\theta_0} &= \frac{1}{\Omega_r}\frac{\partial P_r}{\partial \theta}\bigg|_{\theta_0}
\end{aligned}$$

The partial derivatives of P_r of the above equations are

$$\begin{aligned}
\left.\frac{\partial P_r}{\partial v_r}\right|_{v_{r0}} &= \frac{1}{2}\rho\pi R^2\left(3v_{r0}^2 C_{P0} + v_{r0}^3 \left.\frac{\partial C_P}{\partial \lambda}\right|_{\lambda_0}\left.\frac{\partial \lambda}{\partial v_r}\right|_{v_{r0}}\right)\\
\left.\frac{\partial P_r}{\partial \Omega_r}\right|_{\Omega_{r0}} &= \frac{1}{2}\rho\pi R^2 v_{r0}^3 \left.\frac{\partial C_P}{\partial \lambda}\right|_{\lambda_0}\left.\frac{\partial \lambda}{\partial \Omega_r}\right|_{\Omega_{r0}}\\
\left.\frac{\partial P_r}{\partial \theta}\right|_{\theta_0} &= \frac{1}{2}\rho\pi R^2 v_{r0}^3 \left.\frac{\partial C_P}{\partial \theta}\right|_{\theta_0}
\end{aligned}$$

and the partial derivatives with regards to λ are

$$\begin{aligned}
\left.\frac{\partial \lambda}{\partial \Omega_r}\right|_{\Omega_{r0}} &= -\frac{v_{r0}}{R\Omega_{r0}^2}\\
\left.\frac{\partial \lambda}{\partial v_r}\right|_{v_{r0}} &= \frac{1}{R\Omega_{r0}}
\end{aligned}$$

Just as the aerodynamic torque was linearized, the thrust force exerted on the tower has to be linearized. Like with Q_r the linearization variables are v_r, Ω_r and θ and the Taylor expansion is done with respect to some linearization points that are v_{r0}, Ω_{r0} and θ_0

$$Q_t \approx Q_{t0} + \left.\frac{\partial Q_t}{\partial v_r}\right|_{v_r 0}\Delta v_r + \left.\frac{\partial Q_t}{\partial \Omega_r}\right|_{\Omega_r 0}\Delta\Omega_r + \left.\frac{\partial Q_t}{\partial \theta}\right|_{\theta_0}\Delta\theta \tag{2.19}$$

The individual first order partial derivatives of (2.19) is [Hen07]

$$\begin{aligned}
\left.\frac{\partial Q_t}{\partial v_r}\right|_{v_{r0}} &= \frac{1}{2}\rho\pi R^2 2v_{r0}C_{t0} + v_{r0}^2\left.\frac{\partial C_t}{\partial \lambda}\right|_{\lambda_0}\frac{1}{R\Omega_{r0}}\\
\left.\frac{\partial Q_t}{\partial \Omega_r}\right|_{\Omega_{r0}} &= \frac{1}{2}\rho\pi R^2 v_{r0}^2\left.\frac{\partial C_t}{\partial \lambda}\right|_{\lambda_0}\left(-\frac{v_{r0}}{R\Omega_{r0}^2}\right)\\
\left.\frac{\partial Q_t}{\partial \theta}\right|_{\theta_0} &= \frac{1}{2}\rho\pi R^2 v_{r0}^2\left.\frac{\partial C_t}{\partial \theta}\right|_{\theta_0}
\end{aligned}$$

Now only the expressions remaining non-linear are the derivatives of C_P and C_T. The first order derivative approximation is obtained by the finite

differens method. [Ras12]

$$\begin{aligned}
\left.\frac{\partial C_P(\lambda,\theta)}{\partial \lambda}\right|_{\lambda_0} &\approx \frac{C_P(\lambda,\theta_0) - C_P(\lambda_0,\theta_0)}{\Delta\lambda} \\
\left.\frac{\partial C_P(\lambda,\theta)}{\partial \theta}\right|_{\theta_0} &\approx \frac{C_P(\lambda_0,\theta) - C_P(\lambda_0,\theta_0)}{\Delta\theta} \\
\left.\frac{\partial C_t(\lambda,\theta)}{\partial \lambda}\right|_{\lambda_0} &\approx \frac{C_t(\lambda,\theta_0) - C_t(\lambda_0,\theta_0)}{\Delta\lambda} \\
\left.\frac{\partial C_t(\lambda,\theta)}{\partial \theta}\right|_{\theta_0} &\approx \frac{C_t(\lambda_0,\theta) - C_t(\lambda_0,\theta_0)}{\Delta\theta}
\end{aligned}$$

Now the entire state space model is linearized, and is ready to be used.

2.5 Constraints

In order to make to the model as realistic as possible it is also necessary to have some contraints, because of the mechanical limitations. The pitch of the blades cannot be controlled as wished, one of the limitations are for instance regarding how fast the pitch can change, the speed at which the pitch changes is dependent on how fast the engine that is used for that purpose can rotate the wings. The constraints of a realistic wind turbine is given below.

$$\begin{aligned}
\theta_{min} \leq \theta \leq \theta_{max} \\
\dot{\theta}_{min} \leq \dot{\theta} \leq \dot{\theta}_{max}
\end{aligned}$$

Just like the pitch, the generator torque also has some constraints caused by some limitations. [Ras12]

$$\begin{aligned}
Q_{g,min} \leq Q_g \leq Q_{q,max} \\
\dot{Q}_{g,min} \leq \dot{Q}_g \leq \dot{Q}_{q,max}
\end{aligned}$$

These contriants have to be satisfied otherwise the model will not end up useful.

2.6 The Complete Model

The previous derived and explained models can be collected in one big model.

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{E}v$$
$$\mathbf{y} = \mathbf{C}\mathbf{x}$$

Here $\mathbf{x}$ is the state vector. $\mathbf{A}, \mathbf{B}, \mathbf{E}$ and $\mathbf{C}$ are matrices defining the system. $\mathbf{u}$ and v are the input to the model and finally $\mathbf{y}$ is the output. From the previously derived bits of the model we define. [LM06]

$$\mathbf{x} = \begin{bmatrix} \Omega_r \\ \Omega_g \\ \Delta\phi \\ x_t \\ \dot{x}_t \\ \theta \\ \dot{\theta} \\ \hat{Q}_g \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} \theta_{ref} \\ Q_{g,ref} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} \Omega_r \\ \Omega_g \\ \Delta\phi \\ x_t \\ Q_g \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} \frac{-D_s}{I_r} + \frac{1}{I_r}\frac{\partial Q_r}{\partial \Omega_r} & \frac{D_s}{I_r N_g} & -\frac{K_s}{I_r} & 0 & -\frac{1}{I_r}\frac{\partial Q_r}{\partial v} & \frac{1}{I_r}\frac{\partial Q_r}{\partial \theta} & 0 & 0 \\ \frac{D_s}{I_g N_g} & -\frac{D_s}{I_g N_g^2} & \frac{K_s}{I_g N_g} & 0 & 0 & 0 & 0 & -\frac{1}{I_g \tau} \\ 1 & -\frac{1}{N_g} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{1}{M_t}\frac{\partial Q_t}{\partial \Omega_r} & 0 & 0 & -\frac{K_t}{M_t} & -\frac{D_t}{M_t} - \frac{1}{M_t}\frac{\partial Q_t}{\partial v} & \frac{1}{M_t}\frac{\partial Q_t}{\partial \theta} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\omega_n^2 & -2\zeta\omega_n & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tau} \end{bmatrix}$$

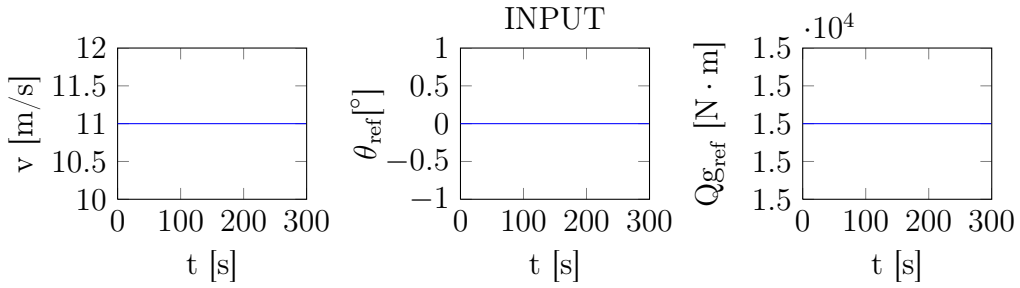
$$\mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \omega_n^2 & 0 \\ 0 & \frac{1}{\tau} \end{bmatrix} \qquad \mathbf{E} = \begin{bmatrix} \frac{1}{I_r}\frac{\partial Q_r}{\partial v} \\ 0 \\ 0 \\ 0 \\ \frac{1}{M_t}\frac{\partial Q_t}{\partial v} \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\tau} \end{bmatrix}$$

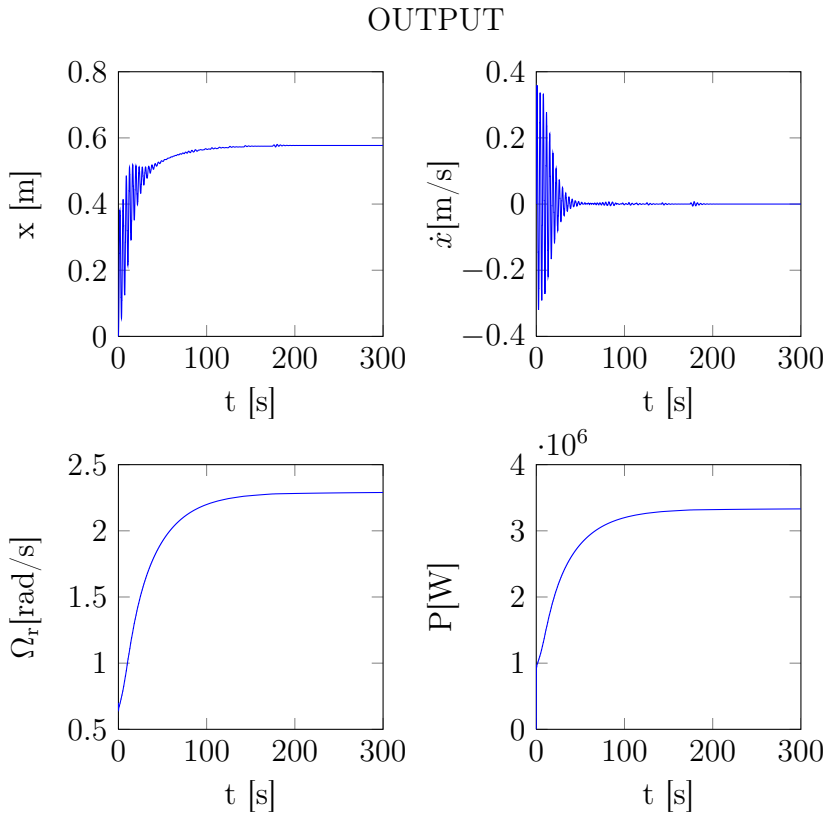
3 | Simulations

This chapter will show various simulations of the NREL 5MW wind turbine. The simulations are done in MATLAB *with* ODE45*. The model is then varified.*

In chapter 2 a model for the NREL 5MW wind turbine was described. In this chapter the model has been implemented and simulated in MATLAB, the implementation of the model i shown in appendix C.1.1.7. To get the coefficients C_T and C_P the tables are downloaded from [AEO] and they are implemented for look-up in script C.1.1.3. The look-up tables and other NREL 5MW specifications are implemented in script for loading, see appendix C.1.1.4. The input of the simulations are v, θ_{ref} and $Q_{g,raef}$, all vectors and the simulations are done by using MATLAB function ODE45, see the script in appendix C.1.1.8. The input wind v should be between $4m/s$ and $11.4m/s$. The starting point is chosen as $\mathbf{x}_0 = \mathbf{0}$ except for $\Omega_{r,0}$ which should be chosen between $0.7\frac{rad}{s}$ and $1.2\frac{rad}{s}$ [Ras12]. The simulation results in plots of the input and output but only a selection of the output plots are shown in here. The simulations does not output the power production directly but it is known from (2.15) that $P = \Omega_g Q_g$.

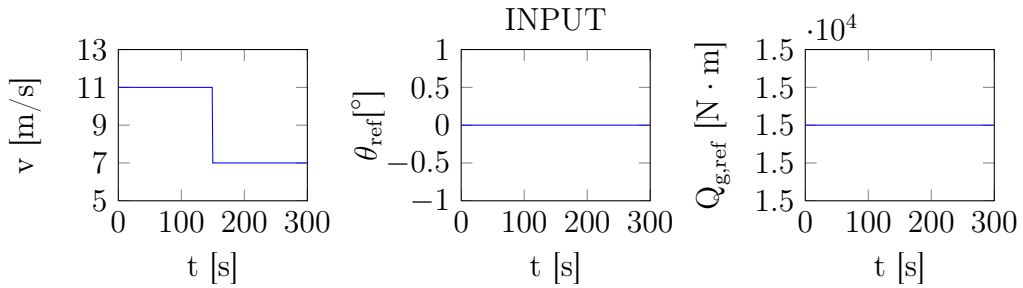
In the first simulation the inputs are kept stable for 300 seconds. The wind speed is chosen to be at $11m/s$, a strong breeze in the upper end of the boudary.

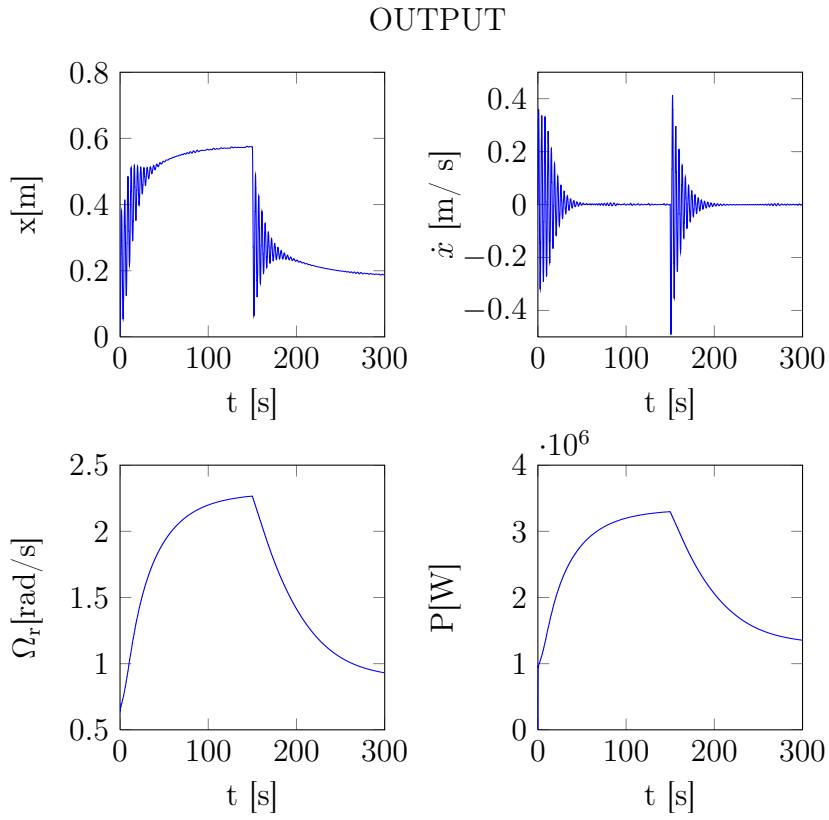




It is seen on the output that the tower displacement begins with an oscillation but then it converges towards a specific point. This is expected as the real parts of the eigenvalues of tower displacement equations (2.4) are both negative for the NREL 5MW turbine. As seen in section 1.2 this leads to a stable damped system. The same goes for tower velocity $\dot{x}$. All in all it is satifiable that every output converges.

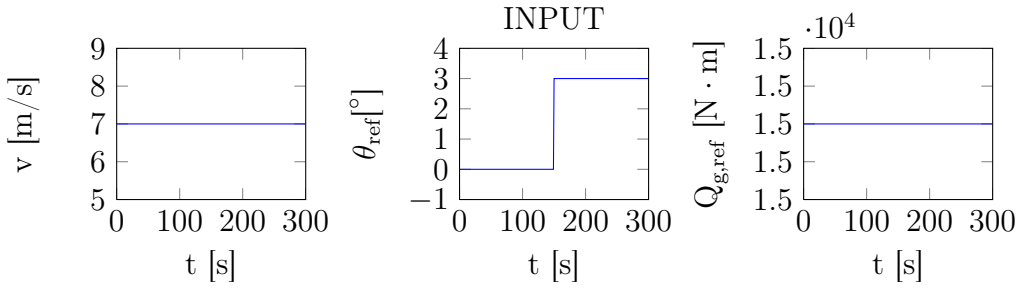
For the next simulation it is tried to change the wind speed from a strong breeze of $11m/s$ to a moderate breeze of $7m/s$ when the system has stabilized. Still the other inputs are constant.

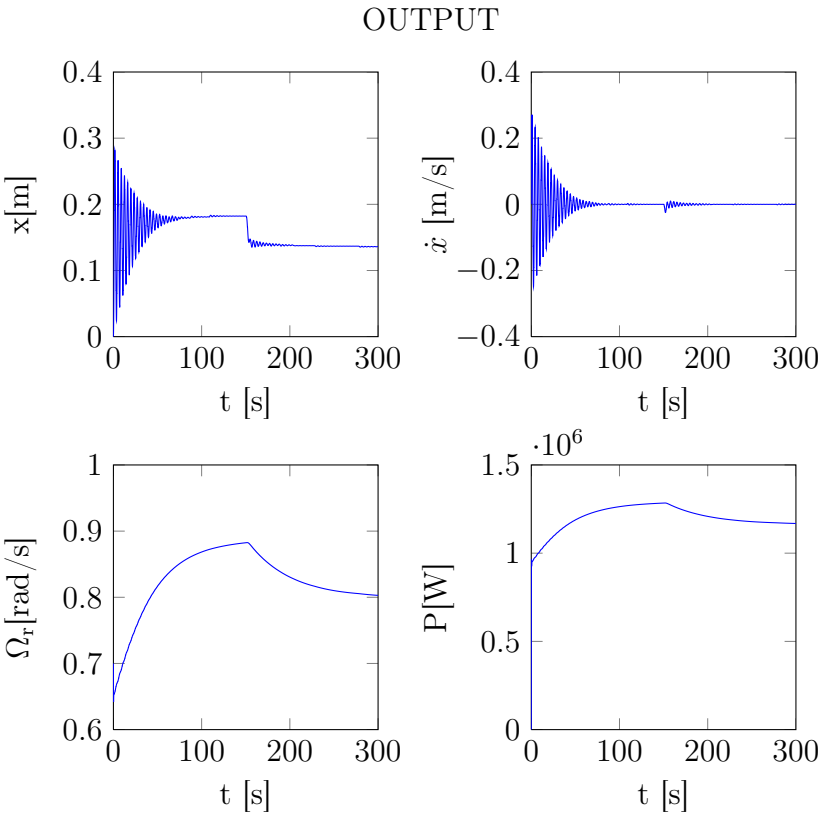




Like the previous simulation the systems stabilizes for the wind speed of $11m/s$ and when the wind speed then drops at 150 sec the tower begins to oscillate again. It stabilizes now at a lower point as it would be expected.

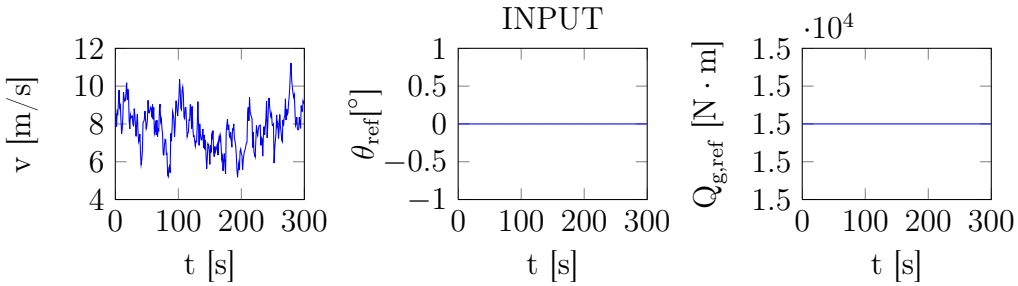
It would also be interesting to see what influence the pitch of the blades will have on the output. For this simulation the pitch is changed from 0° to 10° and the wind speed kept steady at $7m/s$

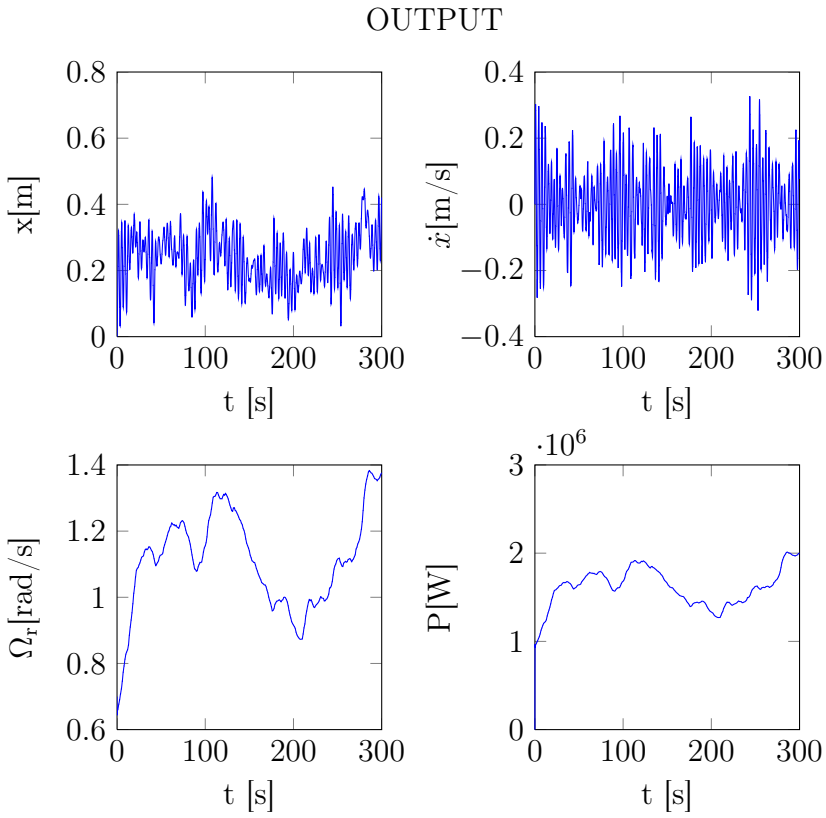




As expected all the shown values drops. This is due to the fact that the thrust force Q_t becomes lower as it is depending on the pitch. According to (2.4) the displacement and the velocity of the tower is depending on the thrust force. Of course this also leads to a decrease in the power production and the angular velocity of the rotor.

For this last simulation the wind input is stochastic. It is generated with the NREL 5MW simulink simulator from [AEO]. The wind has a speed varying from $5.2m/s$ to $11.2m/s$.





It is seen that all of the outputs are affected by this. Both the displacement of the tower, the velocity of the rotor and the power production seems to follow the changes in the wind speed.

Simulations of various input values and changes in the inputs has been made. From the output of those simulations the model has been varified at suitable for further progress

4 | Model Predictive Control

In this chapter it will be explained what Model Predictive Control (MPC) is and why it can be beneficial to use it. In MATLAB *there exists a MPC toolbox and some of its features will be examined through this chapter aswell. At last some simulations with the MPC implemented will be shown.*

4.1 MPC as a tool

Model predictive control is a process control method. It predicts the changes in the dependent variables. There are two categories of the independent variables, the first one is, the ones that can be adjusted by the controller also called the manipulated variables, and the second category is of the variables that cannot be adjusted by the controller and the latter ones is called measured disturbance. [Wikb]
A simplified explanation of how the MPC works is that the model is given some setpoints to how the outputs are desired; from those setpoints to the actual results a cost function is constructed. The aim is to minimize the cost function as much as possible so to obtain results as close to the desired setpoints. The predictions of events are given by the following equations

$$\hat{\mathbf{x}}_{k+i+1|k} = f\left(\hat{\mathbf{x}}_{k+i|k}, \mathbf{u}_{k+i|k}\right) \tag{4.1}$$

$$\hat{\mathbf{y}}_{k+i|k} = g\left(\hat{\mathbf{x}}_{k+i|k}, \mathbf{u}_{k+i|k}\right) \tag{4.2}$$

An example of this could be the linear formulation

$$\left.\begin{aligned}\hat{\mathbf{x}}_{k+1} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{E}\mathbf{v}_k \\ \hat{\mathbf{y}}_{k+1} &= \mathbf{C}\hat{\mathbf{x}}_{k+1} + \mathbf{D}\mathbf{u}_{k+1}\end{aligned}\right\} \text{ for } k = \{0, 1, \ldots, N-1\} \tag{4.3}$$

In the calculations of the predictions of NREL 5MW $\mathbf{D} = \mathbf{0}$ will be used. For discrete time the predictions can be found iteratively. The predictions can be expressed as a loop as shown below.

$$\begin{aligned}
\hat{\mathbf{x}}_{k+1|k} &= \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{E}\mathbf{v}_k \\
\hat{\mathbf{y}}_{k+1} &= \mathbf{C}\hat{\mathbf{x}}_{k+1|k} \\
&= \mathbf{C}\mathbf{A}\mathbf{x}_k + \mathbf{C}\mathbf{B}\mathbf{u}_k + \mathbf{C}\mathbf{E}\mathbf{v}_k \\
\hat{\mathbf{x}}_{k+2|k} &= \mathbf{A}\hat{\mathbf{x}}_{k+1|k} + \mathbf{B}\mathbf{u}_{k+1} + \mathbf{E}\mathbf{v}_{k+1} \\
&= \mathbf{A}\left(\mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{E}\mathbf{v}_k\right) + \mathbf{B}\mathbf{u}_{k+1} + \mathbf{E}\mathbf{v}_{k+1} \\
&= \mathbf{A}^2\mathbf{x}_k + \mathbf{A}\mathbf{B}\mathbf{u}_k + \mathbf{A}\mathbf{E}\mathbf{v}_k + \mathbf{B}\mathbf{u}_{k+1} + \mathbf{E}\mathbf{v}_{k+1} \\
\hat{\mathbf{y}}_{k+2} &= \mathbf{C}\hat{\mathbf{x}}_{k+2|k} \\
&= \mathbf{C}\mathbf{A}^2\mathbf{x}_k + \mathbf{C}\mathbf{A}\mathbf{B}\mathbf{u}_k + \mathbf{C}\mathbf{A}\mathbf{E}\mathbf{v}_k + \mathbf{C}\mathbf{B}\mathbf{u}_{k+1} + \mathbf{C}\mathbf{E}\mathbf{v}_{k+1} \\
\hat{\mathbf{x}}_{k+3|k} &= \mathbf{A}\hat{\mathbf{x}}_{k+2|k} + \mathbf{B}\mathbf{u}_{k+2} + \mathbf{E}\mathbf{v}_{k+2} \\
&= \mathbf{A}(\mathbf{A}^2\mathbf{x}_k + \mathbf{A}\mathbf{B}\mathbf{u}_k + \mathbf{A}\mathbf{E}\mathbf{v}_k + \mathbf{B}\mathbf{u}_{k+1} + \mathbf{E}\mathbf{v}_{k+1}) + \mathbf{B}\mathbf{u}_{k+2} + \mathbf{E}\mathbf{v}_{k+2} \\
&= \mathbf{A}^3\mathbf{x}_k + \mathbf{A}^2\mathbf{B}\mathbf{u}_k + \mathbf{A}^2\mathbf{E}\mathbf{v}_k + \mathbf{A}\mathbf{B}\mathbf{u}_{k+1} + \mathbf{A}\mathbf{E}\mathbf{v}_{k+1} + \mathbf{B}\mathbf{u}_{k+2} + \mathbf{E}\mathbf{v}_{k+2} \\
\hat{\mathbf{y}}_{k+3} &= \mathbf{C}\hat{\mathbf{x}}_{k+3|k} \\
&= \mathbf{C}\mathbf{A}^3\mathbf{x}_k + \mathbf{C}\mathbf{A}^2\mathbf{B}\mathbf{u}_k + \mathbf{C}\mathbf{A}^2\mathbf{E}\mathbf{v}_k + \mathbf{C}\mathbf{A}\mathbf{B}\mathbf{u}_{k+1} + \mathbf{C}\mathbf{A}\mathbf{E}\mathbf{v}_{k+1} + \mathbf{C}\mathbf{B}\mathbf{u}_{k+2} + \mathbf{C}\mathbf{E}\mathbf{v}_{k+2} \\
&\vdots \\
\hat{\mathbf{x}}_{k+N|k} &= \mathbf{x}_{k+N-1|k} + \mathbf{B}\mathbf{u}_{N-1} + \mathbf{E}\mathbf{v}_{N-1} \\
&= \mathbf{A}^N\mathbf{x}_k + \mathbf{A}^{N-1}\mathbf{B}\mathbf{u}_k + \mathbf{A}^{N-1}\mathbf{E}\mathbf{v}_k + \mathbf{A}^{N-2}\mathbf{B}\mathbf{u}_{k+1} + \mathbf{A}^{N-2}\mathbf{E}\mathbf{v}_{k+1} + \ldots \\
&+ \mathbf{B}\mathbf{u}_{N-1} + \mathbf{E}\mathbf{v}_{N-1} \\
\hat{\mathbf{y}}_{k+N} &= \mathbf{C}\hat{\mathbf{x}}_{k+N|k} \\
&= \mathbf{C}\mathbf{A}^N x_k + \mathbf{C}\mathbf{A}^{N-1}\mathbf{B}\mathbf{u}_k + \mathbf{C}\mathbf{A}^{N-1}\mathbf{E}\mathbf{v}_k + \mathbf{C}\mathbf{A}^{N-2}\mathbf{B}\mathbf{u}_{k+1} + \mathbf{C}\mathbf{A}^{N-2}\mathbf{E}\mathbf{v}_{k+1} + \ldots \\
&+ \mathbf{C}\mathbf{B}\mathbf{u}_{N-1} + \mathbf{C}\mathbf{E}\mathbf{v}_{N-1}
\end{aligned}$$

The above predictions can be expressed in matrix form as shown below.

$$\begin{bmatrix} \hat{\mathbf{x}}_{k|k} \\ \hat{\mathbf{x}}_{k+1|k} \\ \hat{\mathbf{x}}_{k+2|k} \\ \vdots \\ \hat{\mathbf{x}}_{k+N|k} \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^N \end{bmatrix} x_k + \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{B} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{A}\mathbf{B} & \mathbf{B} & \mathbf{0} & \ldots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}^N\mathbf{B} & \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & \ldots & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u}_k \\ \mathbf{u}_{k+1} \\ \mathbf{u}_{k+2} \\ \vdots \\ \mathbf{u}_{k+N} \end{bmatrix} \ldots$$

$$+\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{E} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{AE} & \mathbf{E} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}^N\mathbf{E} & \mathbf{A}^{N-1}\mathbf{E} & \mathbf{A}^{N-2}\mathbf{E} & \dots & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v}_k \\ \mathbf{v}_{k+1} \\ \mathbf{v}_{k+2} \\ \vdots \\ \mathbf{v}_{k+N} \end{bmatrix}$$

Just like the prediction are written in a matrix-system, the outputs can also be written as a matrix system

$$\bar{\mathbf{y}} = \mathbf{\Phi}\mathbf{x}_0 + \mathbf{\Gamma}\bar{\mathbf{u}} + \mathbf{\Lambda}\bar{\mathbf{v}} \tag{4.4}$$

In this case the vector and matrices are defined as

$$\bar{\mathbf{y}} = \begin{bmatrix} \hat{\mathbf{y}}_1 \\ \hat{\mathbf{y}}_2 \\ \hat{\mathbf{y}}_3 \\ \vdots \\ \hat{\mathbf{y}}_N \end{bmatrix}, \mathbf{\Phi} = \begin{bmatrix} \mathbf{CA} \\ \mathbf{CA}^2 \\ \mathbf{CA}^3 \\ \vdots \\ \mathbf{CA}^N \end{bmatrix}, \mathbf{\Gamma} = \begin{bmatrix} \mathbf{CB} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{CAB} & \mathbf{CB} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{CA}^2\mathbf{B} & \mathbf{CAB} & \mathbf{CB} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{CA}^N\mathbf{B} & \mathbf{CA}^{N-1}\mathbf{B} & \mathbf{CA}^{N-2}\mathbf{B} & \dots & \mathbf{CB} \end{bmatrix}$$

,

$$\bar{\mathbf{u}} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \vdots \\ \mathbf{u}_N \end{bmatrix}, \mathbf{\Lambda} = \begin{bmatrix} \mathbf{CE} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{CAE} & \mathbf{CE} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{CA}^2\mathbf{E} & \mathbf{CAE} & \mathbf{CE} & \dots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{CA}^N\mathbf{E} & \mathbf{CA}^{N-1}\mathbf{E} & \mathbf{CA}^{N-2}\mathbf{E} & \dots & \mathbf{CE} \end{bmatrix}, \bar{\mathbf{v}} = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \vdots \\ \mathbf{v}_N \end{bmatrix}$$

Now a cost function is needed in order to have some kind of measurement on how close the result are from the optimal solution. The cost will increase the further the results deviate from the optimal solution, therefore the aim is to minimize the cost function. The cost function is a sum of all the outputs diviation from the optimal solution. As shown below takes the form of a quadric norm.

$$\frac{1}{2}\sum_{k=0}^{N} ||\mathbf{y}_{k+1} - \mathbf{r}_{k+1}||^2_{\mathbf{Q}} \tag{4.5}$$

The quadric sum is multiplied by $\frac{1}{2}$, which is for practical reasons, it does not have any effect on the minimization of the cost function which means that it is permissible to do so.

Apart from that the fact is that it is not well wished that there should be big changes in the inputs, this makes a basis for an extension of the cost

function so that the bigger the change is in the inputs, the bigger penalty is received in form of the cost function. The extension is as before mentioned based on the change in the inputs, as demonstrated below.

$$\frac{1}{2}\sum_{k=0}^{N-1}||\Delta\mathbf{u}_{k+1}||^2_{\mathbf{R}} \tag{4.6}$$

Here $\Delta\mathbf{u}_{k+1}$ is $\mathbf{u}_{k+1} - \mathbf{u}_k$

By combining the two pieces of the cost functions the result obtained is

$$\phi = \frac{1}{2}\sum_{k=0}^{N}||\mathbf{y}_{k+1} - \mathbf{r}_{k+1}||^2_{\mathbf{Q}} + \frac{1}{2}\sum_{k=0}^{N-1}||\Delta\mathbf{u}_{k+1}||^2_{\mathbf{R}} \tag{4.7}$$

Now a vector of the setpoints is defined and a matrix with weights for (4.5) or the first part of (4.7) [Hen07]

$$\mathbf{r}_0 = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_N \end{bmatrix}, \mathbf{Q} = \begin{bmatrix} Q & 0 & 0 & \dots & 0 \\ 0 & Q & 0 & \dots & 0 \\ 0 & 0 & Q & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & Q \end{bmatrix}, \mathbf{R} = \begin{bmatrix} R & 0 & 0 & \dots & 0 \\ 0 & R & 0 & \dots & 0 \\ 0 & 0 & R & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & R \end{bmatrix}$$

The MPC-Toolbox asks for the setpoints as well as the weights as shown above, in order to have the nessesary data to minimize the cost function. The built in cost function is slightly bigger, but the neglegtet part of it is not relevant for the criterion on which the above cost function is built on [Mat].

4.2 MPC in MATLAB

In MATLAB there exists an MPC toolbox. By typing MPCTOOL in MATLAB's commando prompt, the control and estimation tool manager pops up, see fig 4.1.

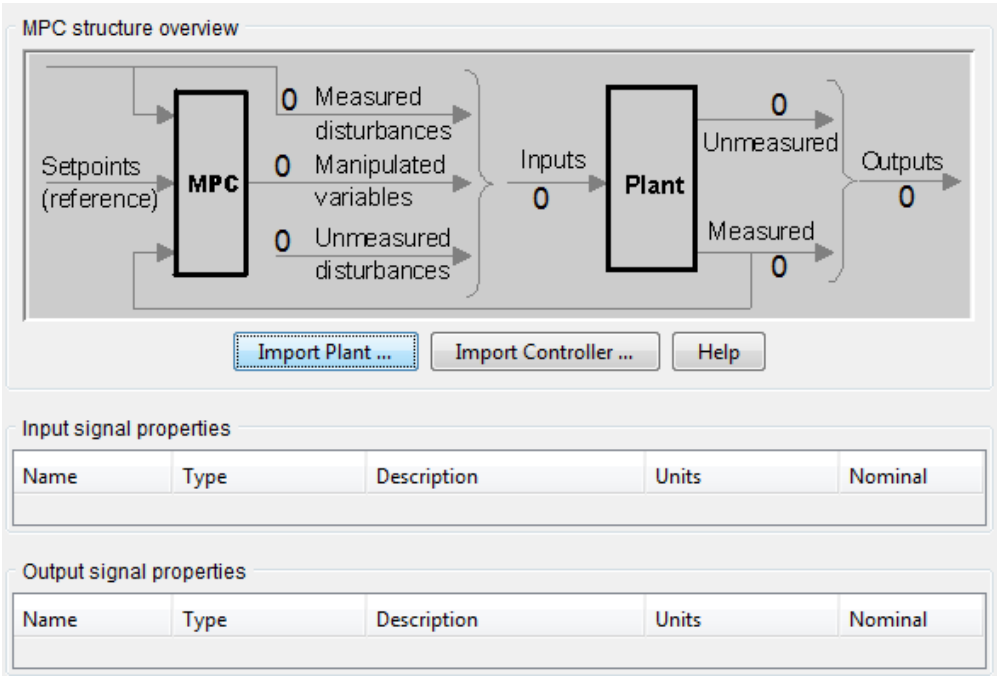


Figure 4.1: The control and estimation tool manager (CETM) in MATLAB

The CETM requires a plant which should be either in the workspace or in a .mat file. The plant should be a linear model of what should be controlled. In this case the plant is the model of the NREL 5 MW from section 2.6. One way of defining a plant model is to define a state-space model. In MATLAB state-space models are of the form:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$$
$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}$$

It seems like this model does not allow any disturbances, which means that the wind speed cannot be implemented in the system. To deal with this situation a new matrix is defined as $\mathbf{H} = \begin{bmatrix} \mathbf{E} & \mathbf{B} \end{bmatrix}$. And the plant will be defined as the continious state-space model:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{H}\bar{\mathbf{u}} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} + \mathbf{D}\bar{\mathbf{u}} \end{aligned}$$

Where $\bar{\mathbf{u}}$ is defined as the combination of measured distubances and manipulated variables

$$\bar{\mathbf{u}} = \begin{bmatrix} \mathbf{v} & \mathbf{u} \end{bmatrix} = \begin{bmatrix} v & \theta_{ref} & \Omega_{r,ref} \end{bmatrix}$$

The plant is then discretised

$$\begin{aligned} \mathbf{x}_{[k+1]} &= \mathbf{A}\mathbf{x}_{[k]} + \mathbf{B}\bar{\mathbf{u}}_{[k]} \\ \mathbf{y}_{[k]} &= \mathbf{C}\mathbf{x}_{[k]} + \mathbf{D}\bar{\mathbf{u}}_{[k]} \end{aligned}$$

It can now be imported into the CETM. The CETM automaticly generates a controller for the plant model. Anyhow it is desireble to make one yourself with the right specifications. The sampling time was set when the discrete model was created to be $T_s = 2$, this means that the controller computes new manipulated variables every two seconds. The prediction horizon and control horizon is set to be $p = 10$ and $m = 3$ which means that the controller optimizes over 10 future sampling periods and calculates 3 future moves. Finally the controller can be made with the MPC-command and the constraints are added aswell, see C.1.1.10.

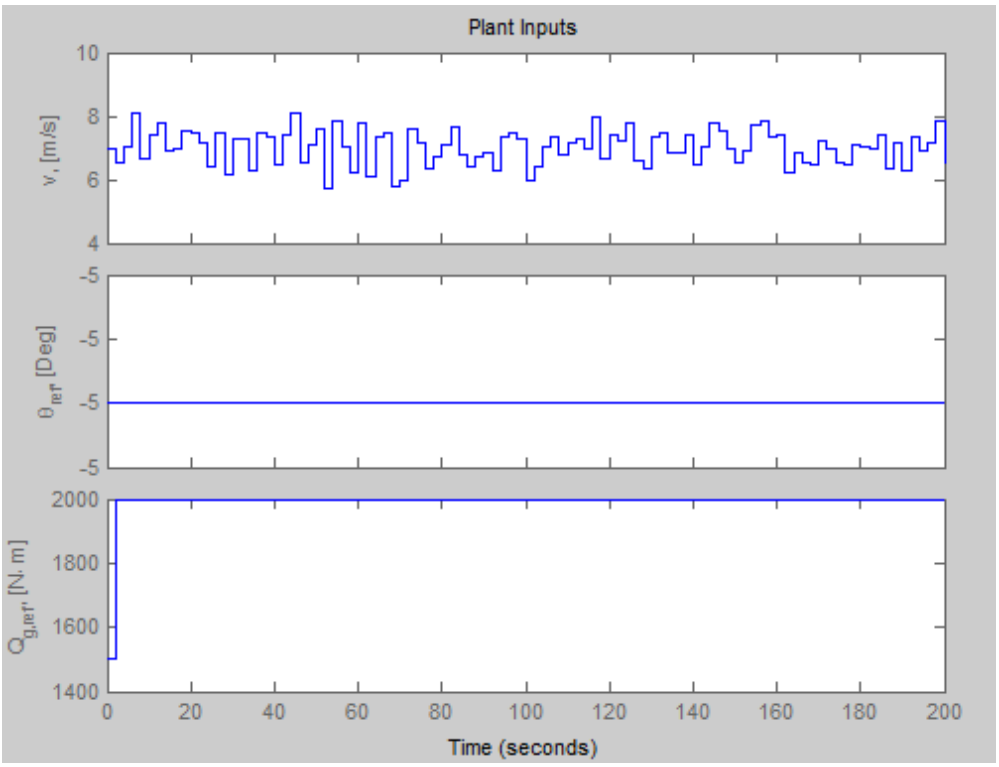
4.3 MPC simulations

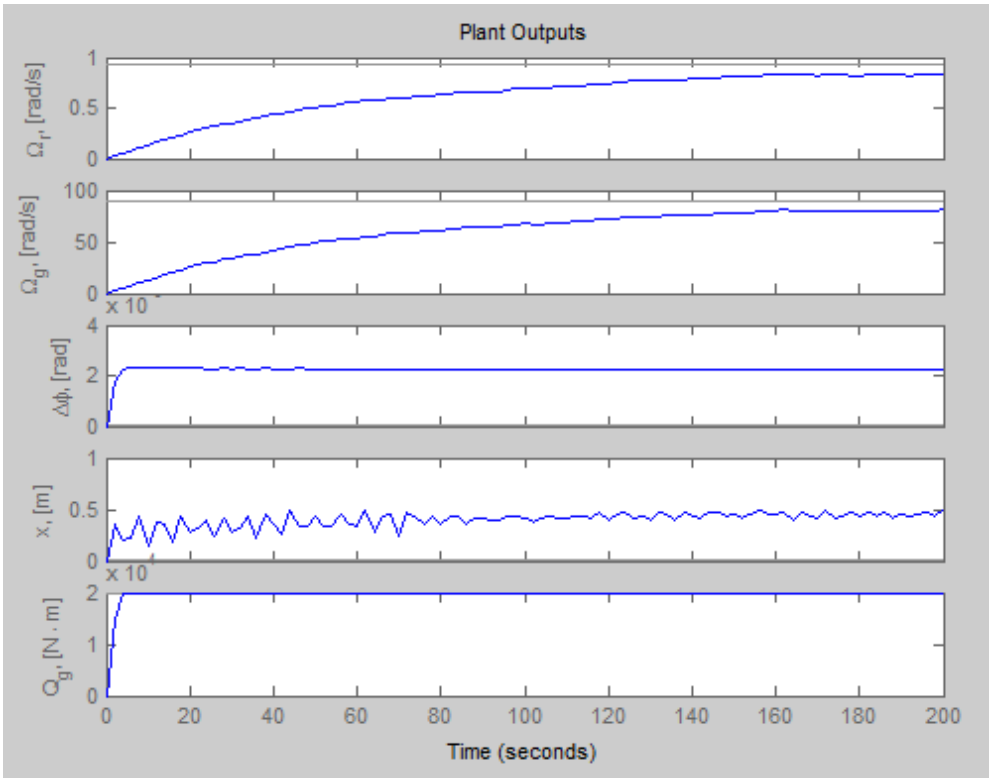
The simulations of a NREL 5 MW turbine is done by by setting up the state-space model and the mpc described in section 4.2 and implementet in script C.1.1.10. The simulations are performed in the CETM by entering setpoints and measured disturbance. The duration of the simulations is 200 seconds. The first thing to do when simulating is to define the setpoints and the weights. The setpoints for x_t and $\Delta\phi$ are chosen to be low, so that the tower will be more stable and the possibility that the driveshaft will cause the generator to fail is smaller. It is known that the maximum power production of the turbine is $5MW$ and $Q_{g,max} = 47.4kNm$ this implies that $\Omega_{g,max} \approx 105\frac{rad}{s}$ and for the simulations the setpoint is chosen a little lower. Ω_r is dependent of Ω_g and the setpoint is chosen so that it fits the setpoint for Ω_g. The setpoint of Q_g is ofcourse not the same as the maximum. The setpoint are as follows

$$\mathbf{r}_0 = \begin{bmatrix} 0.928 \\ 90 \\ 10^{-6} \\ 10^{-6} \\ 2 \cdot 10^4 \end{bmatrix}$$

This means that the controller aims for a power production of $90\frac{rad}{s} \cdot 20kN \cdot m = 1.8MW$. The wind speed is chosen to be of random numbers with mean 7 and standard deviation 0.5. The weights and the rate weights of the manipulated variables that are used for this simulation are as follows

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

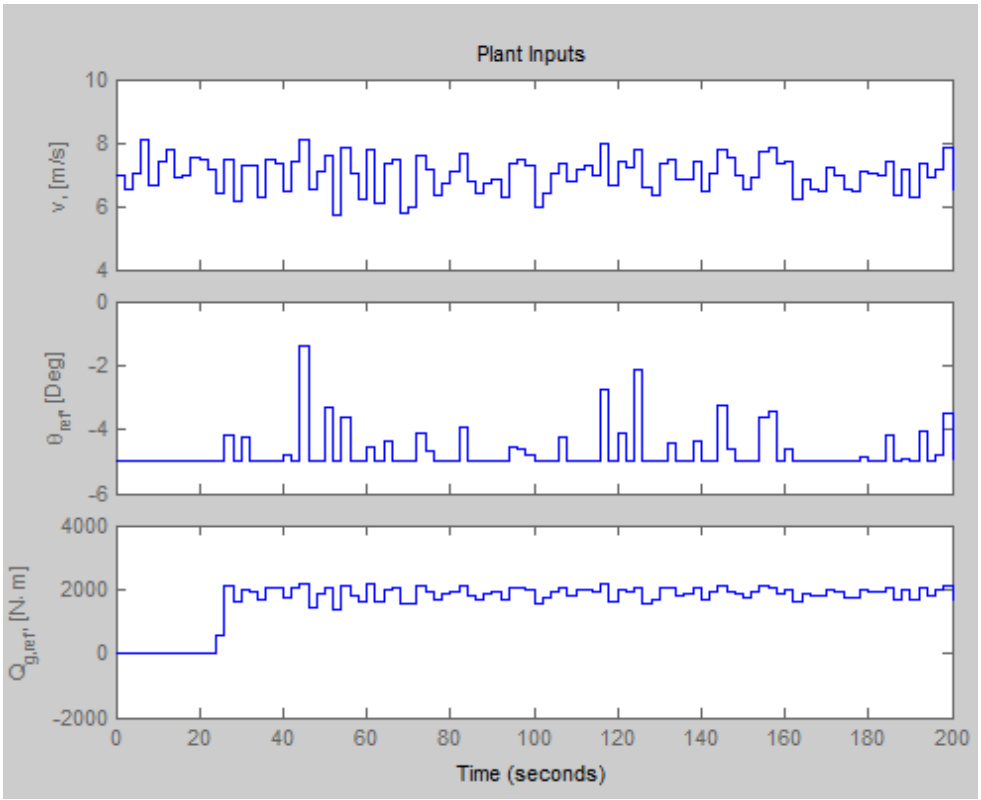

Plant Inputs
v, [m/s]
θref [Deg]
Qg,ref [N. m]
Time (seconds)

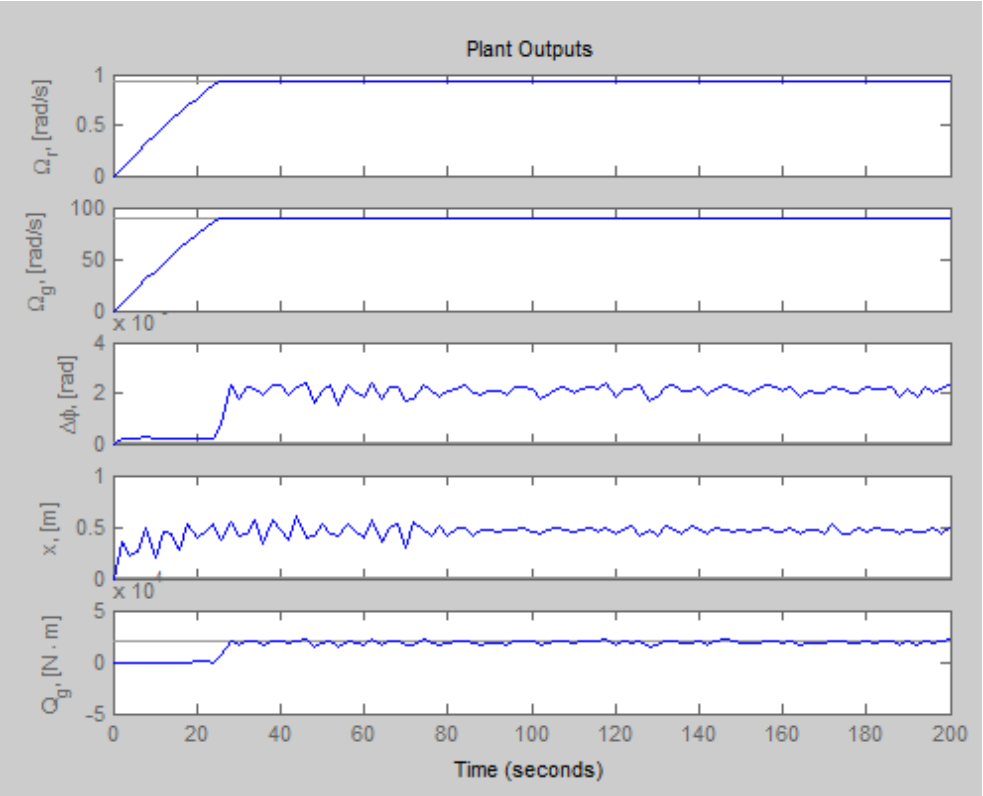

Plant Outputs
Ωr, [rad/s]
Ωg, [rad/s]
x 10
Δψ, [rad]
x, [m]
x 10
Qg, [N . m]
Time (seconds)

It it seen on the output that Q_g hits its setpoint but Ω_g struggels to get to the $90\frac{rad}{s}$ This means that the power production never reaches the $1.8MW$ that is aimed for. Instead it has a production of $1.63MW$ at the 200 seconds mark of simulation. It is seen that the lack of power production is caused by Ω_g. So for the next simulation it is tried to change the weight matrix, so that the it penalises the cost function when Ω_g is away from its setpoint. This should set the controllers main objective to be power production.

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 10^5 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

For the next simulation it is only $\mathbf{Q}$ that has been changed.



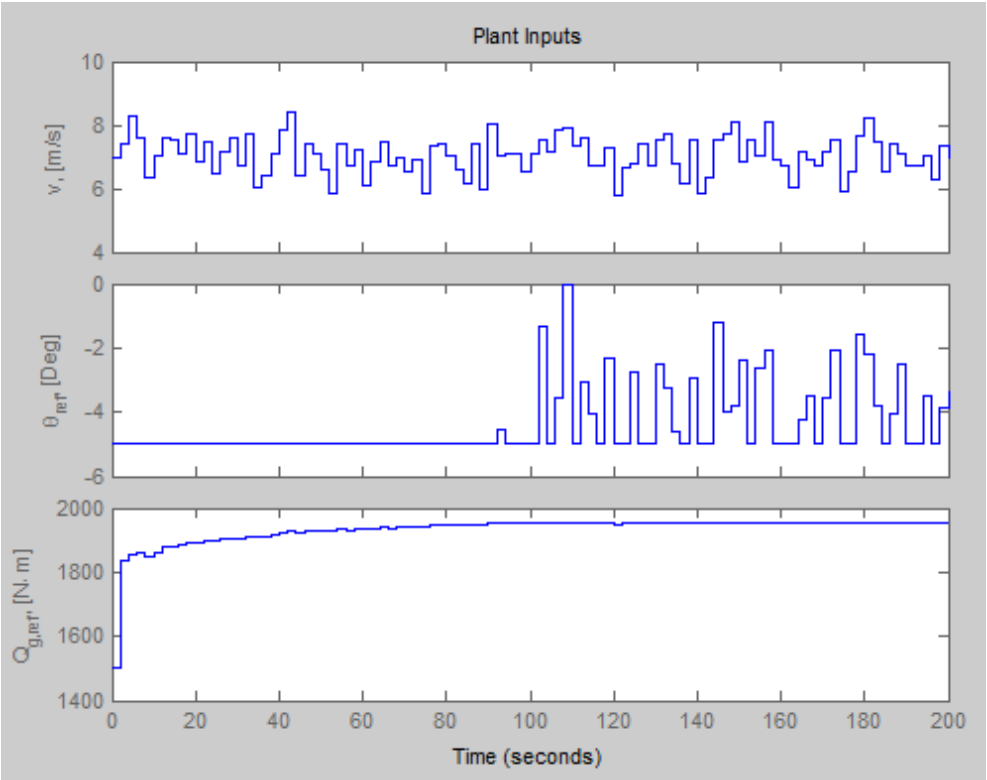


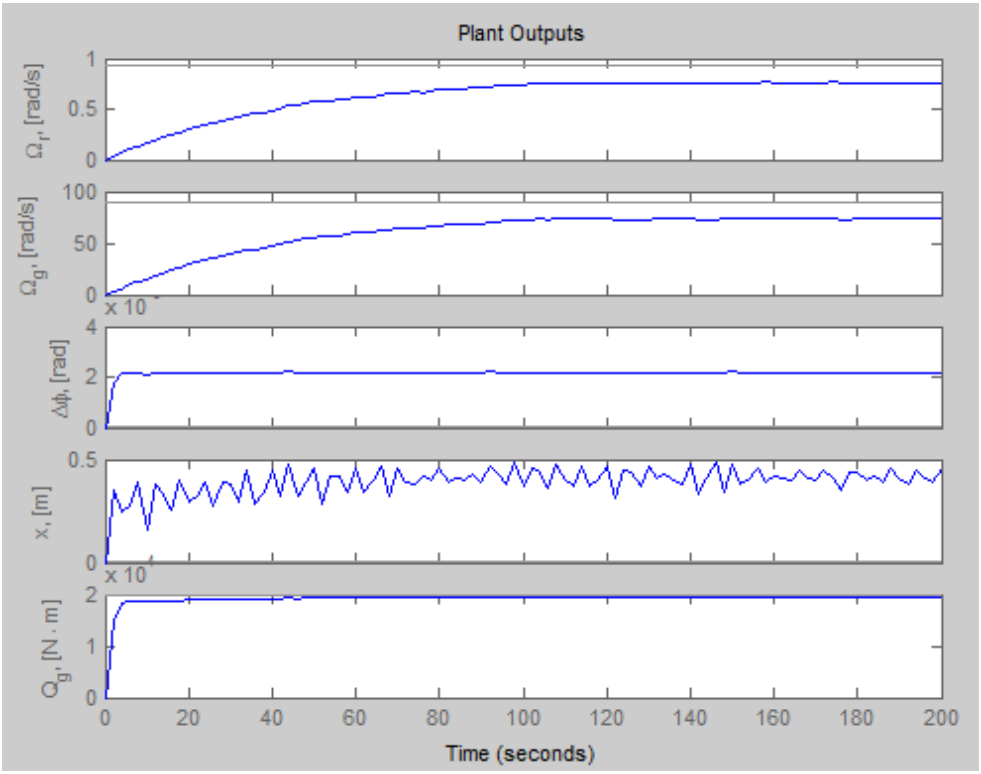
This solution leeds to the desired power production of $1.8MW$ in aprroximateliy 28 seconds. Also it leeds to an increase in the mean of x_t. It could be desireble to have a wind turbine that did not oscillate as much. By changing the weights again, it might be optained.

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 10^2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 10^4 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

With this weight matrix both the power production and the displacement of the tower has become main objectives for the controller.

For the next simulation it is only $\mathbf{Q}$ that has been changed.


Plant Inputs
v, [m/s]
θref [Deg]
Qg,ref, [N. m]
Time (seconds)


Plant Outputs
Ωr, [rad/s]
Ωg, [rad/s]
x 10
Δψ, [rad]
x, [m]
x 10
Qg, [N . m]
Time (seconds)

It is seen that the MPC works as intended. The displacement has decreased, but unfortunately the cost of doing so is that the power production decreased aswell.

The previous simulations and tests of the MPC shows that the wind turbine model can be controlled in a way that one would prefer by changing the weight such that the output will be satisfiable.

5 | Conclusion

A model for the NREL 5MW wind turbine was set up. The model was then implemented in MATLAB and then tested through simulations with the ODE45. Thereby the model of the NREL 5MW was verified. MATLAB was chosen because it can prove the concept and is easy when handling matrices. It also already had an inbuilt MPC toolbox with the function needed.
The model was then linearized and set up as a state-space model. With the state-space model we were able to set up a model predictive controller for the wind turbine with the MPC toolbox in MATLAB. The model combined with the MPC was then put through several tests, which showed that it was able to control the turbine and steer it towards the desired outputs. This means that the MPC controls the manipulated variables of generator torque and pitch angle of the blades in a way such that the turbine produces the wanted power with the less force on the tower. By that the MPC was proven to be an excellent way to control the turbine and thereby controlling the profit.

A | Notation

A.1 Mathematical Notatio

Scalars are in italic style: x_1, Y, z where $x_1, Y, z \in \mathbb{R}$

Vectors are in lower case boldface style $\mathbf{a}, \mathbf{x}, \lambda$

$$\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \text{ where } \mathbf{x} \in \mathbb{R}^n$$

The vector of the $k'th$ iteration is denoted $\mathbf{x}_{[k]}$
The $i'th$ element of $\mathbf{x}$ is denoted x_i
The derivative of vectors and scalars are denoted with a dot, such that the derivative of $\mathbf{x}$ is denoted $\dot{\mathbf{x}}$ and the derivative of x is $\dot{x}$. The derivative is with respect to the time if nothing else is mentioned e.g.

$$\dot{\mathbf{x}} = \frac{\partial \mathbf{x}}{\partial t} = \begin{bmatrix} \frac{\partial x_1}{\partial t} \\ \frac{\partial x_2}{\partial t} \\ \vdots \\ \frac{\partial x_n}{\partial t} \end{bmatrix}$$

Matrices are in upper case boldface style, $\mathbf{A}, \mathbf{X}, \mathbf{\Lambda}$

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & \ldots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \ldots & a_{n,m} \end{bmatrix} \text{ where } \mathbf{A} \in \mathbb{R}^{n \times m}$$

The tranpose of a vector or matrix is denoted with an upper T like
$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$

The idendity matrix is denoted $\mathbf{I}$
The zero-matrix is denoted $\mathbf{0}$ no matter the size of it

$$\mathbf{0} = \begin{bmatrix} 0 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 0 \end{bmatrix}$$

A.2 Acronyms

- **NREL**: **N**ational **R**enewable **E**nergy **L**aboratory
- **MSD** (system): **M**ass-**S**pring-**D**amper (system)
- **MPC**: **M**odel **P**redictive **C**ontrol(ler)
- **CETM**: **C**ontrol and **E**stimation **T**ool **M**anager
- **MIMO**: **M**ultiple-**I**nput and **M**ultiple-**O**utput

B System Parameters

B.1 Variables and data for NREL 5MW wind turbine

Data is taken from [Ras12]

P		W	Power
ρ	1.25	kg/m^3	mass density of air
R	63	m	Lenght of rotor blade
v		m/s	Wind speed
v_r		m/s	Relative wind speed
C_P		$-$	Efficiency coefficient
λ		$-$	Tip speed ratio
θ		$^\circ$	Pitch angle of the blades
$\dot{\theta}$		$^\circ/s$	Pitch angle velocity
$\ddot{\theta}$		$^\circ/s^2$	Pitch angle acceleration
θ_{ref}		$^\circ$	Reference pitch angle
ω_n	0.88	rad/s	Natural pitch frequence
ζ	0.9	$-$	Damping of the pitch
x		m	Displacement of tower
$\dot{x}$		m/s	Displacement of tower velocity
$\ddot{x}$		m/s^2	Displacement of tower acceleration
M_t	4.4642e5	kg	Mass of the tower
D_t	2.0213e3	$N/(M \cdot s)$	Tower damping constant
K_t	1.6547e6	N/m	Tower spring constant
Q_r		$N \cdot m$	Aerodynamic torque
Ω_r		rad/s	Angular velocity of rotor
Q_t		N	Thrust force
C_T		$-$	Thrust coefficient
Q_g		$N \cdot m$	Generator torque
$Q_{g,ref}$		$N \cdot m$	Reference generator torque
τ	0.1	s	Time constant for the generator
I_r	5.9154e7	$kg \cdot m^2$	Inertia of the rotor
$\Delta\phi$		rad	Torsion angle of the driveshaft
$\Delta\dot{\phi}$		rad/s	Torsion angle velocity of the driveshaft
Ω_g		rad/s	Angular velocity of the generator
I_g	500	$kg \cdot m^2$	Inertia of the generator
N_g	97	$-$	Gear ratio

The constraints used in the MPC [Ras12]

θ_{min}	-5	$^\circ$	Minimum pitch angle
θ_{max}	25	$^\circ$	Maximum pitch angle
$\dot{\theta}_{min}$	-8	$^\circ$	Minimum pitch angle velocity
$\dot{\theta}_{max}$	8	$^\circ$	Maximum pitch angle velocity
$Q_{g,min}$	0	$N \cdot m$	Minimum generator torque
$Q_{g,max}$	47,4	$kN \cdot m$	Maximum generator torque
$\dot{Q}_{gmin}$	-15	$kN \cdot m$	Minimum generator torque velocity
$\dot{Q}_{gmax}$	15	$kN \cdot m$	Maximum generator torque velocity

C | Implementation

C.1 Scripts

C.1.1 MATLAB scripts

C.1.1.1 Script testprime

```
function dydt = testprime(t,y,zeta,omega0)
% MSD system of the form:
%       \ddot y = 2*omega*zeta*\dot y - omega^2*y
%
dydt(1,1)=y(2);
dydt(2,1)=-2*zeta*omega0*y(2)-omega0^2*y(1);
```

C.1.1.2 Script testODE

```
% This script is made to see the influence of
% zeta and omega in a % mass-spring-damper
% system by plotting the same system
% with various zeta and omega values

%%
%Initializing constants
zeta = [0 0.3 0.7 1 1.5];
colors = {[0.7,0.2,0.7], 'r', 'm', 'b', [0,0.7,0]};
omega0 = [0, 0.2, 0.5, 0.8, 1];
t = zeros(length(zeta),1);
y = zeros(length(zeta),1);

%% zeta influence
%Creating new figure
f1 = figure('units', 'normalized', 'position', [.1 .08 .6 .6]);
xlabel('t [s]', 'Fontsize', 14);
ylabel('x(t)/x(0) [m]', 'Fontsize', 14);
title('\zeta influence on damping', 'Fontsize', 14);

%Plotting the graphs for varying zeta
hold on
for i=1:length(zeta)
    [t, y]=ode45(@testprime,0:0.01:16,[1 0],[],zeta(i), omega0
   (5));
    plot(t,y(:,1),'Color',colors{i}, 'LineWidth', 1.8)
end

legend('\zeta = 0', '\zeta = 0.3', '\zeta = 0.6', '\zeta = 1',
   '\zeta = 1.5',...
    'Location', 'Best')
grid on;
hold off;
saveas(f1, 'zetainfluence.png');

%% omega influence
%Creating new figure
```

```
36 f2 = figure('units', 'normalized', 'position', [.1 .08 .6 .6]);
37 xlabel('t [s]', 'Fontsize', 14);
38 ylabel('x(t)/x(0) [m]', 'Fontsize', 14);
39 title('\omega_0 influence on damping', 'Fontsize', 14);
40
41 %Plotting graphs for varying omega0
42 hold on
43 for i=1:length(omega0)
44     [t, y]=ode45(@testprime,0:0.01:25,[1 0],[],zeta(1), omega0(
       i));
45     plot(t,y(:,1),'Color',colors{i}, 'LineWidth', 1.8)
46 end
47
48 legend('\omega_0 = 0', '\omega_0 = 0.2', '\omega_0 = 0.5', '\
       omega_0 = 0.8', '\omega_0 = 1',...
49     'Location', 'Best')
50 grid on;
51 hold off;
52 saveas(f2, 'omega0influence.png');
```

C.1.1.3 Script getTables

```
function tables = getTables()
% This function loads the C_P and C_T tables
% downloaded from Aeolus website

%Basic parameters
rho = 1.25; %Air density
R = 63; %Rotor radius

omega = 12.1/60*2*pi; %Rotor speed used in wt_perf

%Load CP and thrust files
cp = load('nrel_cp.tsv', '-ascii');
F = load('nrel_thrust.tsv', '-ascii');

theta = cp(2:end,1); %Extract pitch
lambda = cp(1, 2:end); %Extract tip speed ratio
cp = cp(2:end, 2:end); %Extract power coefficient
F = F(2:end, 2:end)*1e3; %Extract thrust

uu = omega*R./lambda; %Derive wind speeds

%Compute thrust coefficient for each lambda
for i = 1:size(F,2)
    ct(:,i)=F(:,i)./(.5*rho*pi*R^2*uu(i)^2);
end

%Remove negative coefficients
cp(cp<0) = 0;
ct(ct<0) = 0;

%Collecting the tables
tables = {cp, ct, lambda, theta};
end
```

C.1.1.4 Script LoadParameters

```
function par = LoadParameters( )
%   This function loads all the parameters
%   for an NREL 5MW wind turbine
%%
% Loading the Cp and Ct tables
tables = getTables();
par.cp = tables{1};
par.ct = tables{2};

% Loading the constants
par.Kt = 1.6547e6;  % Tower Spring const. [N/m]
par.Dt = 2.0213e3;  % Tower damping conts. [N/(m*s)]
par.Mt = 4.4642e5;  % Tower mass [Kg]
par.rho = 1.25;     % Air density [kg/m^3]
par.R = 63;         % Roter blade length [m]
par.on = .88;       % natural pitch frequency [rad/s]
par.zeta = .9;      % Pitch damping
par.tau = .1;       % Time const. [s]
par.Ir = 5.9154e7;  % Rotor initia [kg*m^2]
par.Ks = 8.7354e8;  % Spring-const. of driveshaft [N/(rad)]
par.Ds = 8.3478e7;  % Damping-const. of driveshaft [N/(rad*s)]
par.Ig = 500;       % Generator initia [kg*m^2]
par.Ng = 97;        % Gear ration [-]
end
```

C.1.1.5 Script CP_CT_plots

```
%Plot coeffiencients CP and CT

%NREL 5MW parameters
rho = 1.25; %Air density
R = 63; %Rotor radius

omega = 12.1/60*2*pi; %Rotor speed used in wt_perf

tables = getTables();
cp = tables{1};
ct = tables{2};
lam = tables{3};
th = tables{4};

[L,T] = meshgrid(1./lam(20:200),th(1:150));

%Plot CP
f1 = figure(1);
hold on
surf(L,T,cp(1:150,20:200),'Edgecolor', 'none', 'FaceColor', '
    interp');
contour(L,T,cp(1:150,20:200),30); colorbar;
hold off
xlabel('\lambda=v/(\Omega_r R) [-]', 'Fontsize', 14);
ylabel('\theta[{\circ}]', 'Fontsize', 14);
zlabel('C_P[-]', 'Fontsize', 14);
title('Plot of power-coefficient C_P', 'Fontsize', 14)
grid on
view(-6,20)
saveas(f1, 'cpplot.png');

%Plot CT
f2 = figure(2);
hold on
surf(L,T,ct(1:150,20:200),'Edgecolor', 'none', 'FaceColor', '
    interp');
contour(L,T,ct(1:150,20:200),30); colorbar;
```

```
hold off
xlabel('\lambda=v/(\Omega_r R) [-]', 'Fontsize', 14);
ylabel('\theta[{\circ}]', 'Fontsize', 14);
zlabel('C_T[-]', 'Fontsize', 14);
title('Plot of thrust-force-coefficient C_T', 'Fontsize', 14)
grid on
view(-6,20)
saveas(f2, 'ctplot.png');
```

C.1.1.6 Script GetCpAndCt

```
function [Cp, Ct] = GetCpAndCt( vr, Omega_R, theta, par )
% This function is used to look-up the C_T and C_P values
% for given values of theta and Omega_r
%Input:     vr      - Relative wind speed
%           Omega_R - Angular velocity of rotor
%           theta   - Pith andgle
%           par     - NREL 5 MW parameters
%Output:    Cp      - Efficiency coefficient
%           Ct      - Thrust coefficient

%Unpacking parameter for use
R = par.R;
cp = par.cp;
ct = par.ct;

%Computing lambda
lambda = (Omega_R*R)/vr;

%Getting indexes from values
i = round(theta*5)+1;
j = min(max(round(lambda*10),1),249);

%Look-up in tables
Cp  = cp(i,j);
Ct = ct(i,j);
end
```

C.1.1.7 Script WTM

```
function xdot = WTM(t,x,u,d,par)
%% This is the NREL 5MW model implementation
%Input:     t    - time
%           x    - vector of states
%           u    - vector of manipulated variables
%           d    - vector of disturbances
%           par  - NREL 5MW parameters
%Output:    xdot - measured output from model
%%
% Unpack states
xt    = x(1);       % Tower position [m]
xtdot = x(2);       % Tower velocity [m/s]
theta = x(3);       % Pitch angle [deg]
thetadot = x(4);    % Pitch angle velocity [deg/s]
Qg = x(5);          % Generator torque [N*m]
Omegar = x(6);      % Angular velocity of rotor [rad/s]
Omegag = x(7);      % Angular velocitu of generator [rad/s]
dphi = x(8);        % Torsion angular velocity of driveshaft [
   rad/s]


% Unpack manipulated variables
thetaref = u(1);    % Reference pitch angle [deg]
Qgref = u(2);       % Reference generator torque [N*m]

% Unpack disturbances
v = d;              % Wind speed [m/s]

% Unpack parameters
Kt = par.Kt;        % Tower Spring const. [N/m]
Dt = par.Dt;        % Tower damping conts. [N/(m*s)]
Mt = par.Mt;        % Tower mass [Kg]
rho = par.rho;      % Air density [kg/m^3]
R = par.R;          % Roter blade length [m]
omegan = par.on;    % natural pitch frequency [rad/s]
zeta = par.zeta;    % Pitch damping
tau = par.tau;      % Time const. [s]
```

```
Ir = par.Ir;        % Rotor initia [kg*m^2]
Ks = par.Ks;        % Spring-const. of driveshaft [N/(rad)]
Ds = par.Ds;        % Damping-const. of driveshaft [N/(rad*s)]
Ig = par.Ig;        % Generator initia [kg*m^2]
Ng = par.Ng;        % Gear ration [-]

%%
% Computing relative windspeed
vr = v - xtdot;

% Get the Cp and Ct values from table
[Cp, Ct] = GetCpAndCt(vr, Omegar, theta, par);

% Trust force
Qt = (rho * pi * R^2 * vr^2 * Ct)/2;

% Power
P = (rho * pi * R^2 * vr^3 * Cp)/2;

% Aerodynamic torque
Qr = P/Omegar;

%% Model
nx = length(x);
xdot = zeros(nx,1);

% Tower
Fs = Kt*xt;          % Spring force
Fd = Dt*xtdot;       % Damper force
xdot(1) = xtdot;
xdot(2) = (Qt - Fs - Fd)/Mt;

% Pitch
dtheta = thetaref - theta;
xdot(3) = thetadot;
xdot(4) = omegan*(omegan*dtheta - 2*zeta*thetadot);

% Generator
xdot(5) = (Qgref - Qg)/tau;

% Drive train
```

```
dphidot = Omegar - Omegag/Ng;
Fs2 = dphi*Ks;      %Spring force
Fd2 = dphidot*Ds;   %Damper force

xdot(6) = (Qr - Fs2 - Fd2) / Ir;
xdot(7) = (-Qg + (Fs2 + Fd2)/Ng ) / Ig;
xdot(8) = dphidot;
```

C.1.1.8 Script WTMsimulation

```
%% This script is used to simulate the NREL 5MW wind turbine
%% Initializing constants
par = LoadParameters();
load simwindspeed.mat
% Set manual windspeed
windspeed1 = 7;
windspeed2 = 7;
% Set manual pitch reference value
thetaref1 = 0;
thetaref2 = 0;
% Set manual Qg ref
Qgref1 = 15000;
Qgref2 = 15000;

% Set simualtion time
n = 300;

% Defining input vectors
d = [windspeed1*ones(1,n/2), windspeed2*ones(1,n/2)]; % use for
    manual wind
%d = v; % use for simulated wind speed
u = [[thetaref1*ones(n/2,1); thetaref2*ones(n/2,1)],...
    Qgref1*ones(n/2,1); Qgref2*ones(n/2,1)];

%% Simulating
odeopt = odeset('abstol',1e-6,'reltol',1e-6);

[t, y]=ode45(@WTM,0:0.1:1,[0 0 0 0 0 0.7 0 0]',odeopt,u(1,:),d
   (1),par);
```

```
for i = 2:n
    [t1, y1]=ode45(@WTM,i-0.99:0.1:i,y(end,:),odeopt,u(i,:),d(i),par);
    t = [t; t1];
    y = [y; y1];
end

%% Input plot
time = 0:n-1;

f1 = figure('units', 'normalized', 'position', [.1 .3 .8 .5]);
subplot(1,3,1)
plot(time,d)
set(gca,'YLim',[4 12])
set(gca,'YTick',[4:2:12])
set(gcf,'PaperPositionMode','auto')
xlabel('t [s]');
ylabel('v [m/s]');

subplot(1,3,2)
plot(time,u(:,1));
set(gca,'YLim',[-1 1])
set(gca,'YTick',[-1:0.5:1])
set(gcf,'PaperPositionMode','auto')
xlabel('t [s]');
ylabel('\theta_{ref}[^\circ]');

subplot(1,3,3)
plot(time,u(:,2));
set(gca,'YLim',[14995 15005])
set(gca,'YTick',[14995:2.5:15005])
set(gcf,'PaperPositionMode','auto')
xlabel('t [s]');
ylabel('Qg_{ref} [N \cdot m]');
suptitle('INPUT');
matlab2tikz('inputWind.tikz', 'height', '\figureheight', 'width', '\figurewidth');

%% Output plot
f2 = figure('units', 'normalized', 'position', [.1 .08 .8 .8]);

```

```
67 subplot(3,3,1)
68 plot(t,y(:,1));
69 xlabel('t [s]');
70 ylabel('x[m]');
71
72 subplot(3,3,2)
73 plot(t,y(:,2));
74 xlabel('t [s]');
75 ylabel('\xdot[m/s]');
76
77 subplot(3,3,3)
78 plot(t,y(:,3));
79 xlabel('t [s]');
80 ylabel('\theta[^\circ]');
81
82 subplot(3,3,4)
83 plot(t,y(:,4));
84 xlabel('t [s]');
85 ylabel('\theta \dot[^ \circ /s]');
86
87 subplot(3,3,5)
88 plot(t,y(:,5));
89 xlabel('t [s]');
90 ylabel('Qg[N \cdot m]');
91
92 subplot(3,3,6)
93 plot(t,y(:,6));
94 xlabel('t [s]');
95 ylabel('\Omega_{r}[rad/s]');
96
97 subplot(3,3,7)
98 plot(t,y(:,7));
99 xlabel('t [s]');
100 ylabel('\Omega_{g}[rad/s]');
101
102 subplot(3,3,8)
103 plot(t,y(:,8));
104 xlabel('t [s]');
105 ylabel('\Delta\phi[rad/s]');
106
107 % Calculate the power production
```

```
108 P = y(:,5).*y(:,7);
109
110 subplot(3,3,9)
111 plot(t,P);
112 xlabel('t [s]');
113 ylabel('P[W]');
114 suptitle('OUTPUT');
115
116 %% Simplified output plot
117 figure('units', 'normalized', 'position', [.1 .3 .5 .5]);
118
119 subplot(2,2,1)
120 plot(t,y(:,1));
121 xlabel('t [s]');
122 ylabel('x[m]');
123
124 subplot(2,2,2)
125 plot(t,y(:,2));
126 xlabel('t [s]');
127 ylabel('xdot [m/ s]');
128
129 subplot(2,2,3)
130 plot(t,y(:,6));
131 xlabel('t [s]');
132 ylabel('\Omega_{r}[rad/s]');
133
134 subplot(2,2,4)
135 plot(t,P);
136 xlabel('t [s]');
137 ylabel('P[W]');
138 suptitle('OUTPUT');
139 matlab2tikz('outputWind.tikz', 'height', '\figureheight', '
    width', '\figurewidth');
```

C.1.1.9 Script setUpLinearModel

```
function [A,B,E,C,D] = setUpLinearModel(ssp)
% This function sets up the
% the state-space model of the NREL 5 MW turbine
% such that:
%       \dot x = Ax + Bu + Ev
%            y = Cx + Du
%
%Input:    ssp       - steady-state point
%Output:   A,B,C,D,E - Matrices defining the state-space model
%% Get the NREL 5 MW data
par = LoadParameters();
% Unpack parameters
Kt = par.Kt;        % Tower Spring const. [N/m]
Dt = par.Dt;        % Tower damping conts. [N/(m*s)]
Mt = par.Mt;        % Tower mass [Kg]
rho = par.rho;      % Air density [kg/m^3]
R = par.R;          % Roter blade length [m]
wn = par.on;        % natural pitch frequency [rad/s]
zeta = par.zeta;    % Pitch damping
tau = par.tau;      % Time const. [s]
Ir = par.Ir;        % Rotor initia [kg*m^2]
Ks = par.Ks;        % Spring-const. of driveshaft [N/(rad)]
Ds = par.Ds;        % Damping-const. of driveshaft [N/(rad*s)]
Ig = par.Ig;        % Generator initia [kg*m^2]
Ng = par.Ng;        % Gear ration [-]

% Loading the tables
tables = getTables();
cp = tables{1};
ct = tables{2};
lamb = tables{3};
thet = tables{4};

%% Steady state
% Extracting steady state points
vr0 = ssp(1);
Or0 = ssp(2);
```

```
38
39 % Calculating steady state pitch angle
40 if vr0<11
41     thet0 = 0;
42 else
43     co = [-0.0258 2.0590 -16.1587];
44     thet0 = co(1)*vr0^2+co(2)*vr0+co(3);
45 end
46
47 % Calcualting steady state tip speed ratio
48 lamb0 = vr0/(R*Or0);
49
50 % Look up cp and ct for steady state
51 i = round(thet0*5)+1;
52 j = min(max(round(1./lamb0*10),1),249);
53 CP0 = cp(i,j);
54 CT0 = ct(i,j);
55
56 %%
57
58 % Calculating the first order derivative approximations
59 dCPdl = (cp(i,j+1)-cp(i,j))/(1/lamb(j+1)-1/lamb(j));
60 dCPdth = (cp(i+1,j)-cp(i,j))/(thet(i+1)-thet(i));
61 dCTdl = (ct(i,j+1)-ct(i,j))/(1/lamb(j+1)-1/lamb(j));
62 dCTdth = (ct(i+1,j)-ct(i,j))/(thet(i+1)-thet(i));
63
64 %calculate derivatives
65 dQrdOr = 1/Or0*(1/2*rho*pi*R^2*vr0^3*dCPdl*(-vr0/(Or0^2*R)))
       -...
66      (1/2*rho*pi*R^2*vr0^3*CP0)/Or0^2;
67 dQrdv = 1/Or0*(1/2*pi*R^2*vr0^2*CP0+vr0^3*dCPdl*1/(Or0*R));
68 dQrdth = 1/Or0*(1/2*rho*pi*R^2*vr0^3*dCPdth);
69
70 dQtdOr = 1/2*rho*pi*R^2*vr0^2*dCTdl*(-vr0/(Or0^2*R));
71 dQtdv = 1/2*rho*pi*R^2*vr0*CT0+vr0^2*dCTdl*1/(Or0*R);
72 dQtdth = 1/2*rho*pi*R^2*vr0^2*dCTdth;
73
74 %% Setting up the matrices
75
76 A = ...
```

```
77  [-Ds/Ir+1/Ir*dQrdOr  Ds/(Ir*Ng)  -Ks/Ir      0  -dQrdv/Ir
       dQrdth/Ir  0     0
78     Ds/(Ig*Ng)  -Ds/(Ig*Ng^2)  Ks/(Ig*Ng) 0     0    0         0
        -1/(Ig*tau)
79     1             -1/Ng           0       0     0          0
          0     0
80     0               0             0       0     1          0
          0     0
81     dQtdOr/Mt       0          0    -Kt/Mt  (-Dt-dQtdv)/Mt
       dQtdth/Mt  0 0
82     0               0             0       0     0          0
          1     0
83     0               0             0       0     0        -wn^2
       -2*zeta*wn 0
84     0               0             0       0     0          0
       0  -1/tau];
85
86
87
88  B = zeros(8,2);
89  B(7,1) = wn^2;
90  B(8,2) = 1/tau;
91
92
93  C = zeros(5,8);
94  C(1:4,1:4) = eye(4);
95  C(end,end) = 1/tau;
96
97
98  D = zeros(5,3);
99
100 E = zeros(8,1);
101 E(1,1) = dQrdv/Ir;
102 E(5,1) = dQtdv/Mt;
103 end
```

C.1.1.10 Script MPC

```
% This script sets up the NREL 5MW model
% as a discrete state-space model
% and then a MPC is constructed

%% Setting up the plant
ssp=[11,0.925];
tables = getTables();
[A,B,E,C,D] = setUpLinearModel(ssp);
H = [E,B];

model = ss(A,H,C,D);
clear A B C D E H ssp

Ts=2;                       %Sampling time
model=c2d(model,Ts);        %Convert to discrete time

% Defining the input
model.InputGroup.MV = 2; %Manipulated variables
model.InputGroup.MD = 1; %Measured distubance

% Specifying I/O names and units
model.Inputname = {'v', '\theta_{ref}', 'Q_{g,ref}'};
model.InputUnit = {'[m/s]' '[Deg]' '[N\cdot m]'};
model.OutputName = {'\Omega_r', '\Omega_g', '\Delta\phi', 'x',
    'Q_g'};
model.OutputUnit = {'[rad/s]' '[rad/s]' '[rad]' '[m]' '[N \cdot
     m]'};
model.StateName = {'\Omega_r', '\Omega_g', '\Delta\phi', 'x', '
    \dot x',...
     '\theta', '\dot \theta', '\hat{Q}_g'};


%% Setting up the MPC
% Define input constraints
clear InputSpecs OutputSpecs
InputSpecs(1)=struct('Min',-5,'Max',25,'RateMin',-8,'Ratemax'
    ,8);
```

```
34 InputSpecs(2)=struct('Min',0,'Max',4750,'RateMin',-1500,'
       Ratemax',1500);
35
36 % Define weights on manipulated and controlled variables.
37 Weights=struct('ManipulatedVariables',[0 0],...
38     'ManipulatedVariablesRate',[1 1],...
39     'OutputVariables',[1 1 1 1 1]);
40
41 % Define prediction and control horizons, and set up the MPC
       object.
42 p=10; %Prediction horizon
43 m=3;  %Control horizon
44 MPCobj=mpc(model,Ts,p,m,Weights,InputSpecs);
45
46 %% Opens the CETM
47 mpctool
```

D | Matlabs mpctool

D.1 Guide to Matlabs mpctool

When running the scrip in appendix C.1.1.10 the wind turbine model is setup as a plant and a controller is setup aswell. The window in figure 4.1 is then shown (the figure is reshown below).

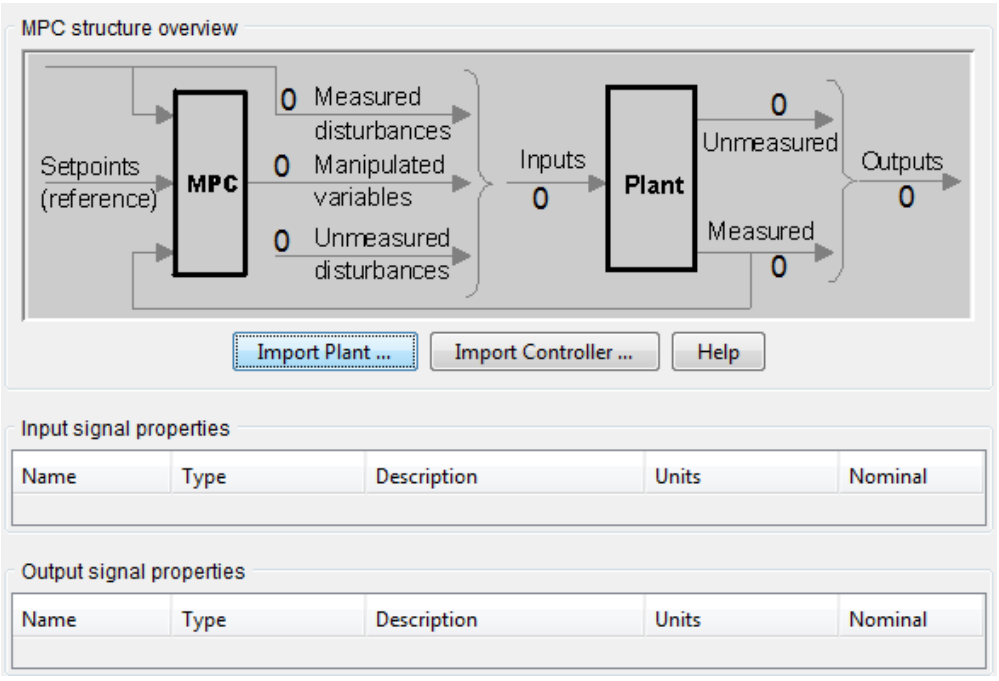


Figure 4.1: The control and estimation tool manager (CETM) in Matlab

First thing to do is to import the plant. Press the import plant button and import the plant named model, then press close. Then import the controller named MPCobj the same way by clicking import controller.

D.1.1 Editing the controller

To change anything about the controller click the plus (+) in the left side menu of the window (not shown in figure). There are two controllers to choose. The one, MPC1, is autogenerated by MPCTOOL when the plant was imported, the other one,MPCOBJ, is the one that was made with the script. Click the MPCOBJ one. In the first tab the control interval, prediction horizon and control horizon can be changed. Those were already set by the script and was not changed. If wished the constraints can be changed in the constraints tab.

To change the weight matrices go to the weight tuning tab. There are two tables. For the input weights the weight should be zeros as they are not considered in this reports simulations. The rate wieght is the diagonal of $\mathbf{R}$ and are chosen to be ones for the simulations. The weights on the output however, are the ones that are changed. The values that are in the outputs weights tables weight column are the diagonal of $\mathbf{Q}$

D.1.2 Simulating

Click the plus (+) next to the scenarios folder in the left side menu. A scenario named SCENARIO1 is displayed, click it. In the top it is possible to change the controllers and the plants. As there is only one plant, do not change that. Instead change the controller to MPCOBJ and change the duaration from 10 to 200, for a 200 second simulation.
The setpoints will be set here with the type constant and the initial value should be equal to the elements in $\mathbf{r}_0$ in section 4.3.
In the mesured distubances table the input of the wind speed is specified, it can be chosen as constant, step, random and other. To have random wind speeds choose the Gaussian type, the initial values field will then be the mean of the random numbers and the size values will be the standard deviation.

$$y = y_0 \text{ for } 0 \leq t \leq t_0 \text{ where } y_0 = \text{ Initial value and } t_0 = \text{ Time}$$
$$y = y_0 + Mrandn \text{ for } t \geq t_0 \text{ where } M = \text{ Size}$$

For the simulation in this thesis the initial value was chosen to be 8 the size was 0.5 and time was 1.
To simulate with the controller make sure that the box next to Close loops is checked. When ready click the simulated button.

Bibliography

[AEO] AEOLUS. Aeolus simwindfarm. World Wide web, http://www.ict-aeolus.eu/SimWindFarm/model-turbine.html. [Online; accessed 04-June-2013].

[BJSB11] T. Burton, N. Jenkins, D. Sharpe, and E. Bossanyi. *Wind Energy Handbook*. Wiley, 2011.

[Hen07] Lars Christian Henriksen. Model predictive control of a wind turbine. Master's thesis, Technical University of Denmark, 2007.

[KMNWG] D. Katzman, J. Morene, J. Noelanders, and M. Winston-Galant. Eigenvaluestability - the michigan process dynamics and controls open text book. World Wide Web, https://controls.engin.umich.edu/wiki/index.php/EigenvalueStability. [Online; accessed 08-May-2013].

[LM06] A. Larsen and T. Mogensen. Individuel pitchregulering af vindmolle. Master's thesis, Technical University of Denmark, 2006.

[Mat] Mathworks. Optimization problem — mathworks, homepage. World Wide Web, http://www.mathworks.se/help/mpc/ug/optimization-problem.html#bsc6bhk. [Online; accessed 20-June-2013].

[Ras12] Rasmus Dalgas Rasmussen. Wind turbine control for wind parks, 2012.

[unkxx] unk. *PDE bogen*. unk, xxxx.

[Wika] Wikipedia. Damping — wikipedia, the free encyclopedia. World Wide Web, http://en.wikipedia.org/wiki/Damping. [Online; accessed 08-Mayl-2013].

[Wikb] Wikipedia. Mpc — wikipedia, the free encyclopedia. World Wide Web, http://en.wikipedia.org/wiki/Model_predictive_control. [Online; accessed 20-June-2013].