TotalControl

Advanced integrated supervisory and wind turbine control for optimal operation of large Wind Power Plants

Title (of deliverable):
Model Predictive Turbine Control
Deliverable no.: 3.4

Delivery date: 28.06.2019

Lead beneficiary: DNV GL

Dissemination level: Public

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 727680

Author(s) information (alphabetical):		
Name	Organisation	Email
Martin Evans	DNV GL	martin.evans@dnvgl.com

Acknowledgements/Contributions:		
Name	Name	Name
Alan Wai Hou Lio, DTU		

Document information

Version	Date	Description			
			Prepared by	Reviewed by	Approved by
1	28.06.2019		Martin Evans	Alan W.H. Lio	Ervin Bossanyi

Definitions

Term	Meaning
SISO	Single Input, Single Output
MPC	Model Predictive Control
LQR	Linear Quadratic Regulator
QP	Quadratic Program
(s)	The Laplace variable
$\mathbb{R}$	The set of real numbers
3P	Three times per rotation of the rotor, i.e. once per blade

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Classical control has been used to regulate and dampen wind turbine behaviour for decades. It has limitations, which turbine designers must spend a great deal of effort to circumvent, for example limits on actuator demands and their derivatives, and tuning only one SISO loop at a time.

MPC is a radically different approach to the control task, whereby the whole linear model, with all inputs and outputs, is used to form predictions into the short time horizon ahead. These predictions are variables, linear in the input (control action) degrees of freedom, which allows the upcoming behaviour to be optimised relative to a simple cost function covering all inputs and outputs.

Additionally, MPC naturally respects constraints, obviating work-arounds like anti-windup control. It does however come at a price – computational complexity. It's fair to say it is also more complex to implement, but experience suggests that the extra effort put into the design of the framework results in much less effort for the engineer tasked with tuning the controller.

The present work focusses on the application of MPC rather than new theoretical findings, but contains an introduction to the theory to help explain the need for some of the more complex elements. Many lesser-known such applied elements are described and motivated, which a 'vanilla' implementation of MPC would typically not include. These elements make the difference between success and failure of MPC, in the same way that the myriad 'tricks' are vital in classical control.

Particularly of value to the success of this implementation are:

- Input blocks, which vastly increase computation speed
- Persistent disturbance rejection, vital to operation with realistic wind
- A single linearisation applied throughout, covering above and below rated conditions
- Stabilising feedback, for linearisations that are not open-loop stable

The author acknowledges that in its current state, this implementation is not ready for installation on a wind turbine. But it provides the foundation for such a venture. Comparisons with classical control on the basis of fatigue loads or energy capture are also not valid at this stage, since the capability to accommodate appropriate signal filters is not yet in place.

THEORY

This chapter on theory is intended to motivate the use of MPC for wind turbine control by showing how its structure arises naturally from the problem definition. While it is not necessary for the reader to understand every stage of the mathematics, they are all essential to the correct functioning of MPC, and can be considered as bolts that hold the framework together. As such, when the framework fails to deliver as expected, typically the underlying mathematics can assist with troubleshooting. Much of the description is derived from the author's thesis, [Evans 2014] but with simplifications.

BACKGROUND TO MPC

Automatic control is found all around us. A thermostat connected to a heater forms a simple on-off control system. More complex than that is a proportional controller, the output of which is calculated as a gain parameter multiplied by the difference between the measurement and a set-point. The behaviour of a system under proportional control will often be preferable to the behaviour of the system under on-off control because the controller does not wait for the measurement to reach a threshold and then slam the control action fully on but rather smoothly applies the control action depending on the deviation from the set-point.

An on-off controller does not cause the plant to settle at a constant steady state. Instead, it will alternate between the on and off thresholds. A proportional controller may result in a constant steady-state, but it may not be at the desired value. We can alter the steady-state of a system under proportional control by applying an offset to the control action, i.e. adding a constant value. However, in many applications, the exact offset required may not be known or may change over time.

Proportional-Integral (PI) control is used to adapt the control offset over time. The controller integrates the difference between measurement and set-point, and the control action is a weighted sum of that integral and the proportional term. A similar form of controller adds in a differential, or derivative, term. Such PID controllers form the basis of the majority of industrial controllers.

Model predictive control (MPC) is a framework for the control of a system by forming predictions of the behaviour of the system and then optimising those predictions. The predictions are variables that depend on the proposed controller actions. That dependence is defined by a model, a mathematical representation of the system to be controlled. The optimisation is posed as a numerical problem constituting the minimisation of a cost function with respect to the proposed controller actions over a finite time horizon into the future. The first predicted inputs of the cost minimising sequence are applied to the system, then a new measurement of the system is taken, which requires the optimisation to be solved again, thus providing a mechanism for incorporating feedback.

What sets MPC apart from other advanced control schemes is its direct ability to account for constraints, not only on the inputs to the system but also on the system's states and outputs. PID can incorporate anti-windup concepts to manage input constraints but there's no clear way to extend that to output constraints. A detailed review of many aspects of MPC is given in [Mayne 2000], a summary written at a time when linear MPC, in discrete time with no uncertainty, was quite mature.

DERIVATION OF MPC

MPC naturally arises from an evolution of control as one introduces discrete time, multi-input multi-output (MIMO) and constraints. It requires much more computation than classical control, but techniques have been developed to make this manageable. Over the last few decades, what constitutes a 'solution' to a control problem has changed from a specific value to any algorithm that poses a numerical problem with a known solution. All MPC in this project require the solution of a quadratic program (QP), that is the minimisation of a quadratic cost function subject to linear constraints.

To introduce the mathematical concept of MPC, firstly consider an unconstrained optimisation. A linear time invariant (LTI) model is defined as follows:

$$x_{k+1} = Ax_k + Bu_k \tag{1}$$

Here $x \in \mathbb{R}^{n_x}$ and $u \in \mathbb{R}^{n_u}$.

The problem of minimising an infinite horizon quadratic cost on the state and inputs, *without* constraints is defined as follows. The abbreviation s.t. means 'subject to'.

$$\min_{u_k} \sum_{k=0}^{\infty} x_k^T Q x_k + u_k^T R u_k \tag{2}$$
$$\text{s.t. (1)}$$

The solution to (2) is given by [Kalman 1960]. It is the linear quadratic regulator (LQR), defined as:

$$u_k = Kx_k, \qquad K = -(\mathrm{R} + \mathrm{B}^{\mathrm{T}}\mathrm{PB})^{-1}\mathrm{B}^{\mathrm{T}}\mathrm{PA} \tag{3}$$

Here P is the solution to the discrete time algebraic Riccati equation. The feedback gain K is optimal for the unconstrained system.

Now we consider a constrained optimisation problem by introducing linear state and input constraints to give a new problem:

$$\min_{\mathbf{u}_k} \sum_{k=0}^{\infty} x_k^T Q x_k + u_k^T R u_k \tag{4}$$
$$\text{s.t. (3)}, \; Fx_k + Gu_k \leq \mathbf{1}$$

Here $\mathbf{1}$ is a vector of ones and $F \in \mathbb{R}^{n_F \times n_x}$, $G \in \mathbb{R}^{n_G \times n_x}$. The new symbol $\mathbf{u}_k$ is a concatenation of $u_{i|k}$ for all i, which is clearly impossible in practice. Indeed, (4) is intractable because there are infinite degrees of freedom. This problem will be address shortly. First some helpful notation is introduced.

During optimisation, future states $x_k, k > 0$ depend on the input sequence, which consists of optimisation variables. The optimisation happens at every time step, so it is worth denoting

optimisation variables and their dependencies in terms of both the time step k and the number of time steps into the future the variable represents, i. Thereby the predicted states are written as $x_{i|k}$ and the predicted inputs are written as $u_{i|k}$, where the bar operator is pronounced 'given'.

The subscript notation introduced here means the predicted value at time step $i+k$, given the known state at time step k, namely x_k. Note that $x_{0|k} \equiv x_k$. The constrained optimisation problem in terms of predicted states is now written as:

$$\min_{\mathbf{u}_k} \sum_{i=0}^{\infty} x_{i|k}^T Q x_{i|k} + u_{i|k}^T R u_{i|k}$$
$$\text{s.t.} \quad x_{i+1|k} = Ax_{i|k} + Bu_{i|k}, Fx_{i|k} + Gu_{i|k} \leq \mathbf{1} \tag{5}$$

To address the intractability of the infinite horizon, the 'dual mode' paradigm is introduced as in [Rossiter 1998]. This provides a way to reduce the problem to instead involve a finite number of degrees of freedom. Firstly, we find a region of state space where the controller (3) satisfies the constraints in (4). This is termed the terminal set X_f, and methods for finding it are discussed in the literature, e.g. [Gilbert 1991]. The remainder of this work assumes that the terminal set is large and is therefore not calculated.

The next stage in dual mode MPC design is to break the horizon into two modes. Mode 1 has degrees of freedom for the inputs:

$$\mathbf{u}_k = [u_{0|k} \dots u_{N-1|k}]$$

and Mode 2 is under autonomous LQR control as per S_1. The cost of the infinite horizon of Mode 2 is given by:

$$x_{N|k}^T \bar{Q} x_{N|k}$$

Here $\bar{Q}$ is the solution of a Lyapunov equation. The dual mode control problem is as follows:

$$\min_{\mathbf{u}_k} \left(x_{N|k}^T \bar{Q} x_{N|k} + \sum_{i=0}^{N-1} x_{i|k}^T Q x_{i|k} + u_{i|k}^T R u_{i|k} \right)$$
$$\text{s.t.} \quad x_{i+1|k} = Ax_{i|k} + Bu_{i|k}, Fx_{i|k} + Gu_{i|k} \leq \mathbf{1}, x_{N|k} \in X_f \tag{6}$$

The horizon length *N* remains choice for the control engineer or can be left as an optimisation problem to be solved in an outer loop. To solve (6) where N is undecided, [Sznaier 1987] suggests choosing an initial value of N, solving (6), checking whether $x_{N|k} \in X_f$ and increasing N if not. In this project we instead choose N based on experience.

The work of [Rawlings 1993] shows that satisfaction of constraints (6) at time at time k implies feasibility for all time after k, a property termed 'recursive feasibility'. Solutions of (6) are optimal if N is long enough that the cost cannot be reduced by increasing N by one. In this work, we do not use X_f and so there is no guarantee of recursive feasibility.

Each subsequent predicted state contains terms in increasing powers of A. To avoid numerical ill-conditioning, we re-state the prediction dynamics in terms of $\Phi = A + BK$ as in [Rossiter 1998]. If

(A, B) is stabilisable then it is possible to find such an LQR gain K. Accordingly, the predicted inputs are decomposed as:

$$x_{i+1|k} = \Phi x_{i|k} + B c_{i|k}$$
$$u_{i|k} = K x_{i|k} + c_{i|k}, i < N \quad (7)$$

This removes $\mathbf{u}_k$ as the optimisation variable and replaces it with $\mathbf{c}_k = [c_{0|k} \dots c_{N-1|k}]$. If the optimal solution touches no constraints for the entire Mode 1 horizon, then $c_{i|k} = 0$ for all i because the problem is equivalent to LQR.

INPUT BLOCKS

It is beneficial for feasibility and optimality to have a long prediction horizon N. But it is computationally expensive to have an optimisation variable for every time step in that horizon. A compromise is possible, which would violate the guarantee of recursive feasibility if one were in place, but which vastly helps reduce computational cost. We now explain a method to reduce the computational cost by reducing the degrees of freedom.

Input blocking constrains some neighbouring input degrees of freedom to be equal within blocks. As the time steps stretch further into the prediction horizon, the temporal resolution is deemed less important, so the blocks start small and grow with i. In this work, the blocks are defined so that they grow by one time step per block, so that:

$$c_{1|k} = c_{2|k}, \quad c_{3|k} = c_{4|k} = c_{5|k}, \quad c_{6|k} = \dots = c_{9|k}, \quad \dots \quad (8)$$

Note that the autonomous part of the control, i.e. $Kx_{i|k}$ is not blocked so the relationships in (8) are not true for $u_{i|k}$.

CONDITIONING

The quadratic program that is created in this type of MPC requires a numerical solver to iterate over valid solutions, which in turn requires the cost function and constraints to be numerically well-conditioned. To visualise the solution of a well-conditioned and ill-conditioned QP, see Figure 1.

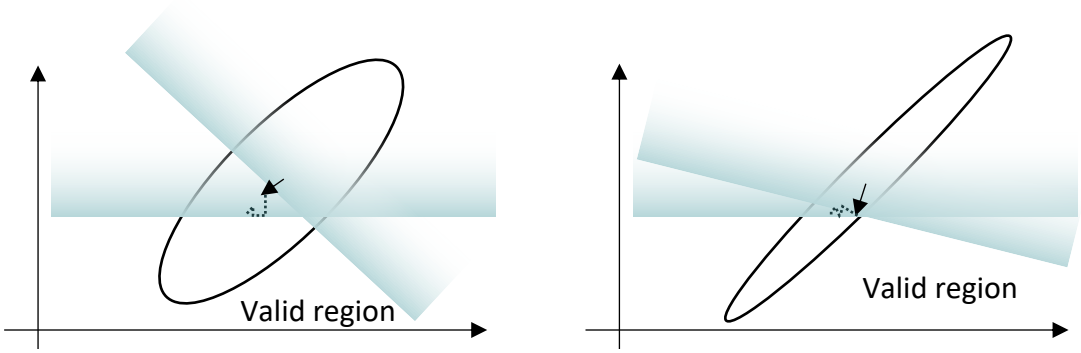

Figure 1: A well-conditioned and an ill-conditioned QP with illustrative iterative solutions. Linear constraints are shown as planes intersecting the space. The cost is quadratic, so a line of constant cost forms an ellipse.

APPLICATION

The theory above is now applied to the problem of controlling a wind turbine in Bladed. A recent review of applications of MPC to wind turbine control is presented in [Lio 2014], including a summary of proven and potential benefits over classical control.

The turbine in this work has the following basic characteristics:

Constant	Value
Rotor diameter	170 m
Hub height	107 m
Rated generator speed	409 rpm
Fine pitch angle	-0.5°
Rated generator torque	177 kNm
Rated wind speed	11 m/s

LINEARISATION

The first step is to linearise the wind turbine model, since we are applying linear MPC. And since our formulation of MPC assumes an LTI plant, only one linearisation point is entered into the Bladed linearisation tool. Much work exists on extending the capabilities of MPC to include multiple linearisation points, e.g. [Kumar 2009].

We aim to operate the new MPC controller in above- and below-rated conditions, so the linearisation point we chose is rated wind speed, specifically the highest wind speed where the steady-state pitch angle is equal to the fine pitch angle.

As usual, dynamic stall and pitch nonlinearities are disabled for the linearisation (not for the Bladed model used for time domain simulation). The linear model is imported into Matlab where the remainder of the design is performed. We return to Bladed for the validation stage.

Note that there will inevitably a difference between the linear model and the full turbine model. This mismatch leads to uncertainty in A, B and that means that uncertainties are multiplied by each other during the prediction horizon. This project does not tackle this problem but there is a thorough discussion with efficient solution in [Evans 2014].

Since we will not know the state x_0 at run-time, it must be estimated from the measurements. Additionally, costs and constraints are more natural applied to measurements than states. These facts motivate an expansion of the notation as follows:

$$\begin{aligned} x_{i+1|k} &= Ax_{i|k} + Bu_{i|k} \\ y_{i|k} &= Cx_{i|k} + Du_{i|k} \end{aligned} \tag{9}$$

Here, $y_{i|k}$ is the predicted measurement at time step $i + k$ given the measurements up to time step k. The state x_k is assumed known, as in the theory, but in reality it is estimated using a Kalman filter to be discussed later. Matrices A, B, C, D are provided in the linear model, but henceforth D will be set to zero and ignored since it is impossible to have an immediate impact on the measurements with the inputs for a plant like this. When constraints and cost are be applied to the measurements

rather than states the QP form of the problem is retained, since $FCx_{i|k}$ is linear and $x_{i|k}^T C^T Q C x_{i|k}$ is quadratic, although F, Q must be redefined for this.

The Bladed linear model has a state called 'rotor rigid body', which represents the azimuthal angle of the rotor and is only useful for controllers that use the azimuth, e.g. individual pitch control. Since this is not the case for this project, the rotor rigid body state is removed with the Matlab function `modred` with the option `'truncate'`.

MEASUREMENTS

The measurements we chose to include in this project are:

1. Measured generator speed
2. Measured pitch angle (collective)
3. Inferred pitch rate (collective)
4. Measured fore-aft acceleration

On some wind turbines, the pitch system can additionally report pitch rate, so inferring that is not required. But for the sake of generality it is treated here as something that we must calculate at run-time based on filtered difference in subsequent pitch angle measurements.

Each of the measurements is a scalar (double) for each time step, which are stacked in a column vector for each time step denoted $\tilde{y}_k$ as introduced above.

Note the wind speed as measured at the anemometer is not one of the measurements. This is quite normal for wind turbine control, at least for algorithms that cover normal operation.

CONTROL ACTIONS

The control actions we chose to include in this project are:

1. Pitch angle demand (collective)
2. Generator torque demand

Note that pitch angle demand certainly does not equal measured pitch angle. The Bladed model includes pitch rate and acceleration limits and a second-order transfer function. The linearisation contains just the transfer function, which introduces a phase lag between demand and measurement.

The output from the MPC optimisation at time k is u_k and the actions applied to the turbine model at run time are $\tilde{u}_k$ as described above.

SCALING AND OFFSETS

Inputs (measurements) and outputs (control actions) are assigned a scale and offset so that in the MPC formulation their typical values are in the range $[-1,1]$ to be well-conditioned. The offsets, which are denoted $\bar{y}, \bar{u}$, are available from Bladed as part of the linear model, but the scales, diagonal matrices denoted S_y, S_u, are chosen based on turbine constants. Raw measurements $\tilde{y}$ are converted for use in MPC, then the solution of the optimisation is converted to a control action $\tilde{u}$, both as follows:

$$y_k = S_y^{-1}(\tilde{y}_k - \bar{y})$$
$$\tilde{u}_k = S_u u_k + \bar{u} \tag{10}$$

Clearly the linear model matrices must also be scaled. Denoting the matrices that come directly from Bladed as A', B', C', the linear model matrices are scaled as:

$$B = B' S_u, \qquad C = S_y^{-1} C' \tag{11}$$

Note $A = A'$ since the states will be numerically conditioned by B, C.

A similar treatment is required when defining constraints. For example, future control actions that must be less than or equal (strict inequalities are discouraged in MPC) to a constant $u_{\max}$ must be posed as:

$$u_{i|k} \leq S_u^{-1}(u_{\max} - \bar{u}) \tag{12}$$

Note that although written differently to the constraints in (4), any linear constraint can be re-written in that format. The transformation is easy to perform by hand, but Matlab tools exist to make it easier, such as YALMIP [Löfberg 2004].

WIND MODEL

The linear model from Bladed has an additional input, hub centre wind speed. This is not a control action, but it is also not a random disturbance, at least not independent identically distributed (IID). A Kalman filter can only reject white noise, so it is the responsibility of the controller to reject anything more persistent, just as it is with a PID controller.

In MPC, persistent disturbances can be rejected by state estimation, and for this, the link between the white noise underlying the turbulence and the hub centre wind speed must be modelled. This is an opportunity to exploit known structures in the structure of turbulence and use that for predictions, as explained in detail in [Liu 2018]. In general, this approach is known as disturbance accommodating control (DAC) and is detailed outside of MPC in [Wright 2004] and [Selvam 2007].

To simplify the process for this work, we assume the wind speed is simply first-order low pass filtered white noise. The filter will be denoted H_v.

$$H_v(s) = \frac{1}{1 + \tau_v s} \tag{13}$$

Here, τ_v is a time constant, which we set as 20 seconds to allow the wind to have enough low frequency content. It has one state, which the Kalman filter attempts to estimate at run time. In the MPC problem formulation, the filter is connected to the linear turbine model using the Matlab function `connect`.

LOW FREQUENCY FEEDBACK

The linear model that is used for predictions in MPC must be stable for the cost to be bounded in the infinite horizon, hence the dual mode paradigm as described in the theory chapter. In general, linear models are unstable at some wind speeds, because faster rotors generate more lift, which accelerates the rotor more. Choosing to stabilise the plant with the LQR gain in feedback is the cost-optimal option, but it isn't necessarily practical. In this project, the stabilising feedback was chosen to be a simple PI loop from measured generator speed to torque demand.

Since the use of a simple PI feedback is not cost-optimal, the MPC solution is not strictly cost-optimal in Mode 2. But for long prediction horizons, Mode 2 cost (calculated using $\bar{Q}$) becomes less significant. This choice of stabilisation also implies that $c_{i|k} = 0$ is not cost-optimal for Mode 1 in the absence of constraints, which can have an impact on the numerical conditioning of the QP. However, importantly, at least one of the constraints we want to apply will normally be active, since we want to discourage pitch activity below rated generator torque.

The PI controller for this purpose clearly has to be tuned, but there is no need to tune it accurately as in classical control because it will be supplemented by the MPC optimisation variables. For the sake of completeness, the format and values in use in this project are as follows, where the torque-speed (QS) stabilisation transfer function is denoted H_{QS}:

$$H_{QS}(s) = \frac{1.2(1+3s)}{s} \tag{14}$$

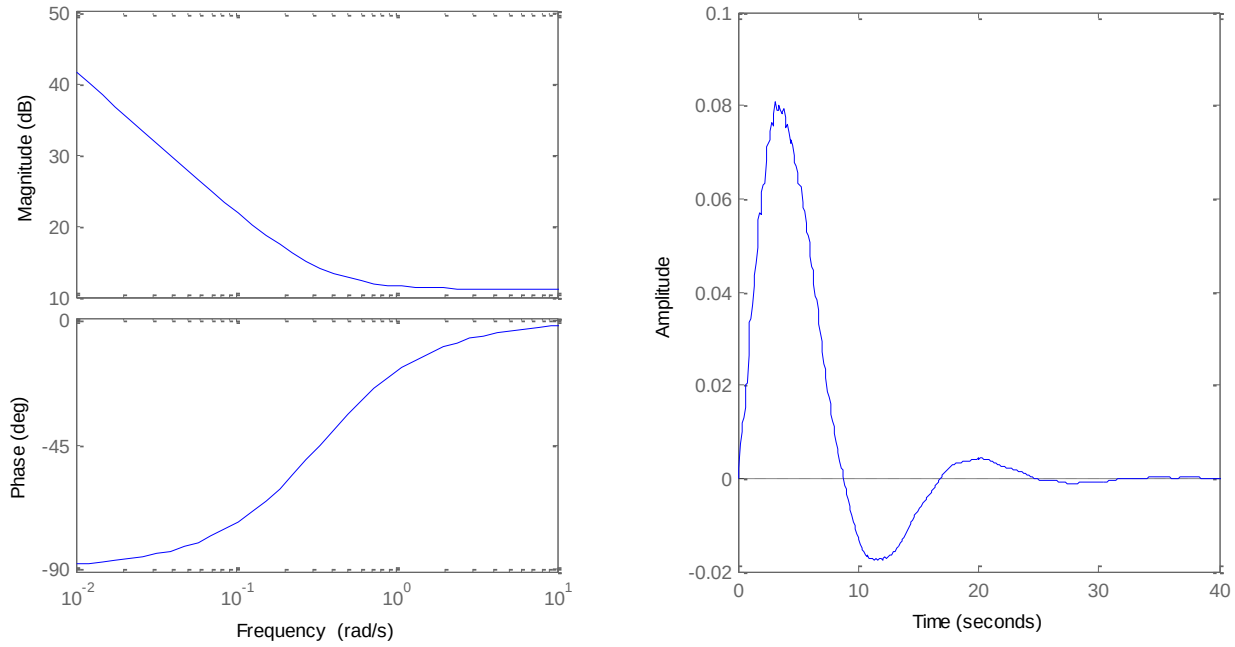

Figure 2: (Left) Bode plot of the torque-speed PI stabilising loop. (Right) Step response of the linear model with the stabilising loop connected, from wind speed to measured generator speed.

As Figure 2 shows, the torque-speed stabilisation is not aggressive, and certainly not optimal. Its primary purpose is to allow the Mode 2 horizon cost to be finite. The optimisation variables will drive down the cost in Mode 1, allowing for arbitrarily aggressive speed control, through the selection of output costs, to be described later.

H_{QS} is connected to the linear model using the Matlab function `feedback`. The new linear model is then checked for stability before proceeding to the next steps. Some configurations of Bladed have very low damping on edgewise rotor modes, which at some wind speeds can appear as slightly negative damping. If the model is still unstable after the rotor speed stabilisation, we check which specific modes are unstable and correct them in the Bladed model through modal damping parameters, then re-linearise.

The theoretical background and practical benefits of feedback stabilisation on an inner loop, over which the MPC problem is posed, is discussed in [Lio 2017]. During the TotalControl task leading to this deliverable, many alternative configurations of linear models and MPC problem definitions have been tried. Only this configuration, as per that paper, with stabilisation and Kalman filter state estimation has produced a working MPC controller.

MODEL REDUCTION

The linear model exported from Bladed and stabilised with H_{QS} has 67 states. This is too many to process in a QP at run time without an advanced solver. Besides, most of those states are low energy structural vibrations, which are not necessarily observable with the selected measurements, or controllable with the selected actions. Therefore, the model is reduced with the Matlab function `balred`. This allows the designer to choose the number of states preferred in the simplified model, which in this project was set to six. Note that this step happens before the wind model H_v is applied. This adds one state.

Seven states, only six of which relate to the turbine, two of which serve the pitch model, leaves only four states for the structural modes, i.e. position and velocity for only two modes. This is clearly a large simplification and future work will experiment with less swingeing model reduction. Less reduced models are more computationally expensive and harder to design Kalman filters for. But to ever compete with classical design, which can have any number of SISO filters, clearly an MPC would have to have more states.

DISCRETISATION

The reduced linear model is discretised. In this project we use a time step of 100 ms, which is five times longer than the time step of full commercially implemented controllers, to allow extra time for the solution of the QP. Two major technological improvements can be made to improve performance based on this selection.

Firstly, the torque-speed stabilisation happens outside the MPC controller, so that can run at 20 ms. This introduces a major and minor time step paradigm, where the QP is solved once per major times step and simpler loops can be added on with a shorter step. For example, drivetrain damping typically impacts only a narrow range of frequencies, so can be added afterwards without interfering. These performance improvements are outside the scope of this work but are not seen as high risk omissions.

Discretisation uses the Matlab function `c2d`, with the default option ‘zero order hold’ to best represent the way measurements are processed in a turbine. No delays are added to the model at this stage, but LTI-based MPC does support delays in principle.

Figure 3 shows the Bode diagram for one input and one output of the discrete plant against the continuous reduced order plant and the stabilised full order plant.

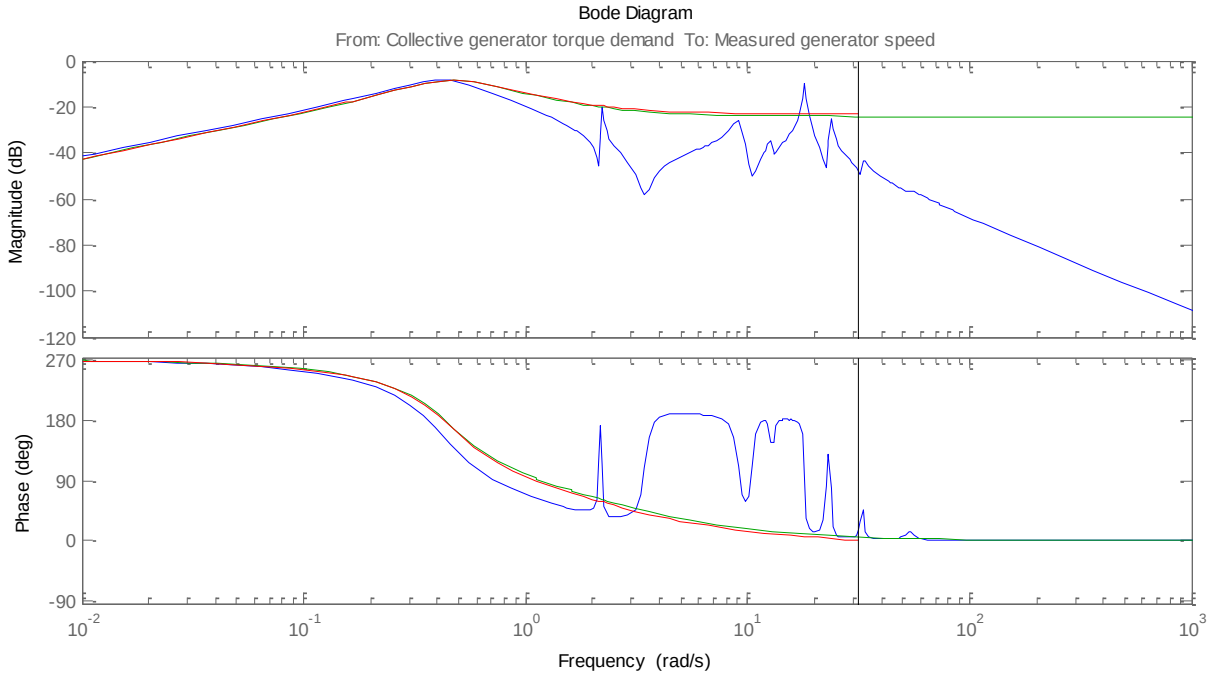

Figure 3: Bode diagram from generator torque demand to measured generator speed for the stabilised linear model (blue), the reduced model (green) and the discrete model (red, with Nyquist limit shown as a black bar). Clearly there are many modes ignored by the reduced model, which are thereby uncontrolled.

KALMAN FILTER

When given a sensible, well-conditioned plant model, making a Kalman filter can be quite easy. Of course, it is often more challenging in practice. This project takes the simplifying assumption of a fixed uncertainty on the measurements and plant, rather than monitoring the difference between measurements and filtered measurements to continuously update the Kalman gain matrix in run time. That improvement can be applied at a later date without impacting the MPC design.

There are four measurements, so the covariance of the uncertainty is 4×4. We assume the uncertainty is independent (which might not be strictly true for pitch angle and rate, see paragraph above). So the covariance matrix R_{KF} is diagonal and populated with numbers based on experience:

$$y_k = Cx_k + v, \qquad \mathrm{E}(vv^T) = R_{KF} \tag{15}$$

Here, E is the expectation operator and v is the measurement noise (assumed white). R_{KF} has 10^{-4} on its diagonal. There is also uncertainty on the wind speed state update:

$$x_{k+1} = Ax_k + Bu_k + Gw, \qquad \mathrm{E}(ww^T) = Q_{KF} \tag{16}$$

Here, G is an 'allocator' matrix that distributes the state update uncertainty w across the states. We set G such that w is scalar and only applies to the wind state, meaning Q_{KF} is 1×1 and has the value 10.

The Kalman filter is then created with the Matlab function `kalman`. Now how can we know whether it's estimating the states correctly? The states themselves have no physical meaning because they

are generated by model reduction. But the estimated states, multiplied by C give estimated measurements, which should converge to the actual measurements over time.

An impulse is applied to the white noise wind input to the reduced order model. The outputs of that model are recorded over a period of 10 seconds. Four of those outputs are measurements, which are fed into the Kalman filter. The other output is the wind speed, which we don't have access to in the turbine but want to know that its state is being estimated correctly.

The output from the Kalman filter is multiplied by C to give the estimated measurements, i.e. the measurements that would arise if the state that has been estimated is correct. Note that we are not measuring anything about the wind speed or its white noise disturbance with the Kalman filter. The wind speed is estimated using the Kalman filter, which is based on the reduced linear model. This is a great result because it provides the persistent disturbance rejection that the integral gain provides in classical control. The results are shown in Figure 4 below.

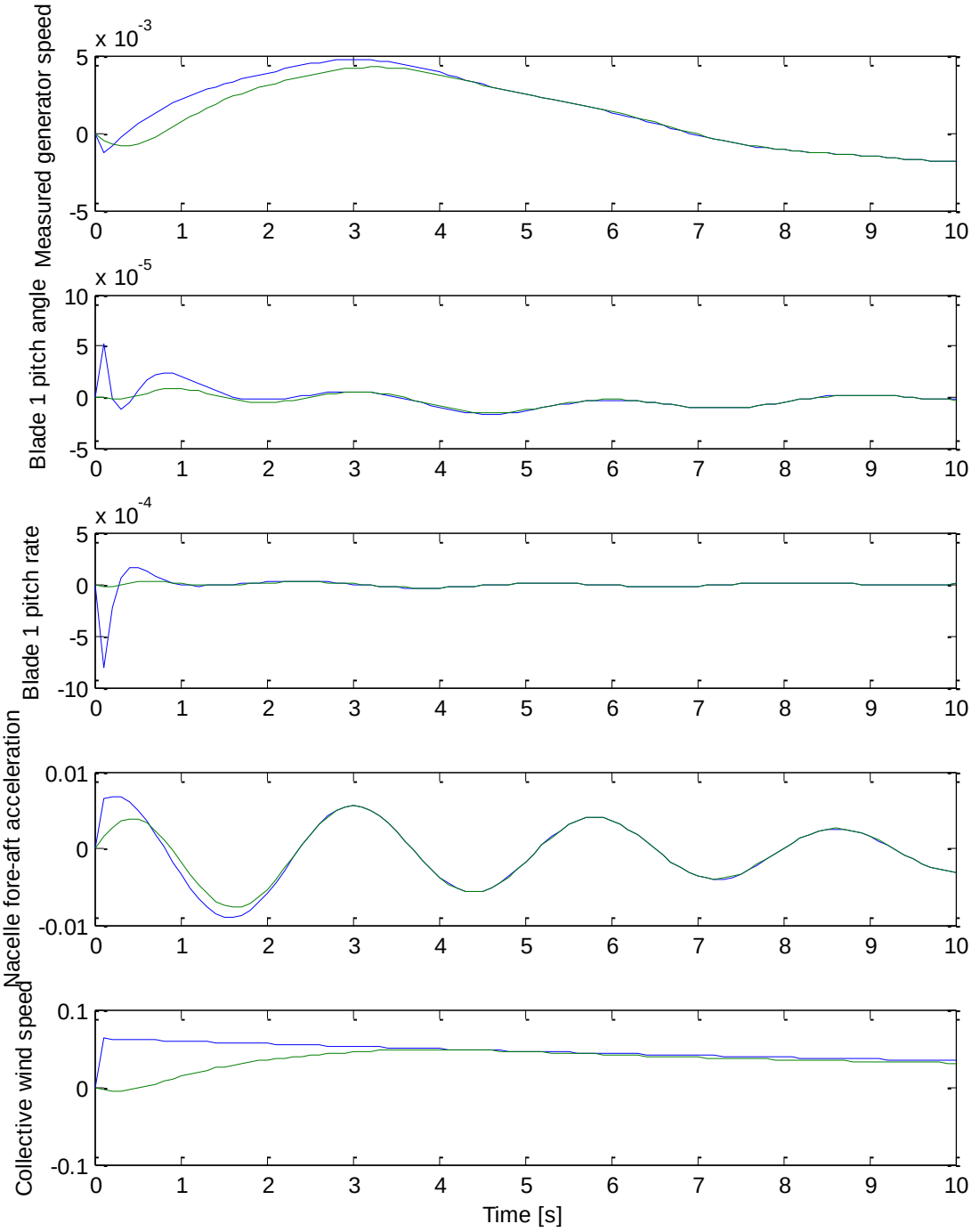

Figure 4: The response of the full linear turbine and wind model (blue) to an impulse in wind white noise disturbance. The Kalman filter estimates the states, which are mapped back onto the measurements for comparison (green). Note the wind speed is correctly estimated after 4 s, purely from the four measured signals.

COST FUNCTION

We now have a workable prediction model. We need a cost function and constraints. We use the cost function format from (6) and the constraint format from (12). We do not want to penalise pitch angle because that is used to reject high wind speeds. But the other measurements are penalised,

as well as the control actions. The cost applied to the predicted measurements is pre- and post-multiplied by C to form the cost on the predicted states:

$$Q = C^T Q_y C \tag{17}$$

Tuning can be performed by modifying the values on the diagonal of Q_y. The element corresponding to pitch angle is set to zero, because we do not want to penalise quasi-steady pitch action. This is because for a given wind speed above rated, there will be a steady-state pitch angle. The wind speed is not known at run-time so, like with a PI controller, we let the controller find the steady-state pitch rather than enforce one. Penalising deviation from the steady-state pitch angle at the wind speed where the linearisation was performed will result in poor low frequency wind rejection.

There is no cost applied to the stabilising torque because that is persistent disturbance rejection below rated wind speed. The diagonal matrix R has zero for pitch angle demand, for the same reason. The value for torque discourages large torque responses.

As per (6), there is a terminal cost applied to the final predicted state. The terminal cost matrix is the solution to the discrete-time Lyapunov equation $A^T \bar{Q} A - \bar{Q} + Q = 0$ using the Matlab function `dlyap`. Remember $Q = C^T Q_y C$ and note that A is the state evolution matrix of the reduced discrete stabilised system and includes the wind model. The reason we calculate $\bar{Q}$ using the Lyapunov equation can be demonstrated with a formulation of the infinite horizon Mode 2 cost J.

$$J = \sum_{i=N}^{\infty} x_{i|k}^T Q x_{i|k} \tag{18a}$$

$$= \sum_{i=N}^{\infty} \left(x_{i|k}^T \bar{Q} x_{i|k} - x_{i+1|k}^T \bar{Q} x_{i+1|k} \right) \tag{18b}$$

$$= x_{N|k}^T \bar{Q} x_{N|k} \tag{18c}$$

The final equality arises from all the terms in (18)b cancelling out, except for the first expression, for $i = N$. We can now use $\bar{Q}$ to calculate J directly, provided we can calculate it so that (18)a equals (18)b. Since the sums are over the same range, the requirement can be re-posed in terms of its summands, with the $x_{i|k}$ terms cancelling out on each side of the equation:

$$Q = \bar{Q} - A^T \bar{Q} A \tag{19}$$

CONSTRAINTS

The constraints we apply at each prediction step are that the pitch demand must be more positive than the fine pitch angle, and that the torque demand (plus stabilising torque) must be between minimum and maximum torque.

Upcoming work will apply pitch rate and acceleration constraints. Pitch rate is a measurement, but acceleration isn't. However, it can be approximated by a finite difference on predicted pitch rate

measurements. Constraints on outputs (measurements) must in general be applied carefully, with some technique to soften them if necessary to ensure feasibility.

SUMMARY OF MODEL ADJUSTMENTS

Previous sections have described some necessary differences between the prediction model and the raw linearisation from Bladed. These adjustments are the key differentiator between this work and previous works and are vital to the success of the scheme. Figure 5 shows a flow chart of the above-mentioned changes to draw them together for reference.

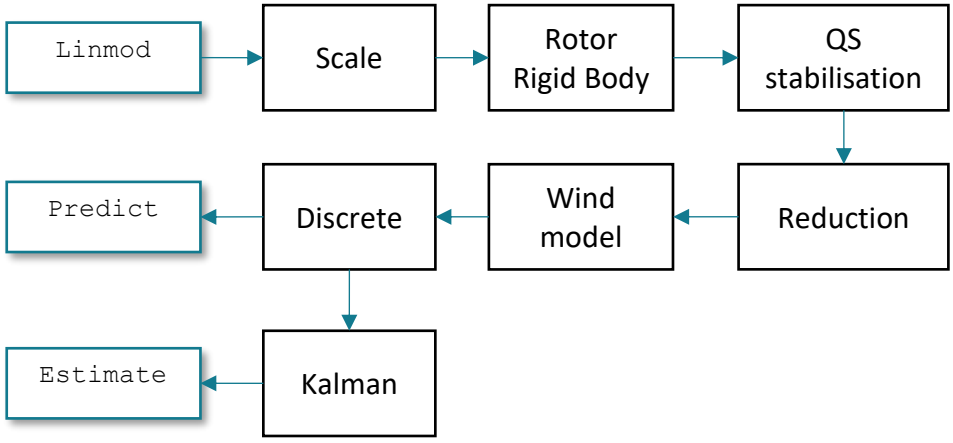

Figure 5: Adjustments made to the raw linear model to prepare it for predictions and estimation

QUADRATIC PROGRAM

Quadratic programs are favoured in MPC because they have known methods of solution and they represent the underlying aims well. That is to say that the square of a deviation of a state, action or measurement from its steady-state value is a natural way to penalise that deviation, akin to minimising variance, and most constraints can be posed linearly in the optimisation variables.

The choice of QP solver is a topic for future work. Some algorithms are more suited to certain QP properties than others. This work does not investigate the choice of solver, and instead uses a commonly applied QP solver for Matlab, called `sedumi` (Self-Dual Minimisation) [Sturm 1999].

The solver has a parameter `eps`, which represents how close two values in the state space have to be to be considered roughly equal, which clearly has a large influence on the precision and computational cost of the solution. If that parameter is set too large, the constraints might be de facto too tight and the problem appears infeasible. If it's too small, the solver continues optimising well after the point where it becomes unnecessary, i.e. many decimal places that are irrelevant. This project sets `eps` to 10^{-5} but it's highly recommended to tune it.

To form the QP, we use YALMIP [Löfberg 2004]. This parses symbols in Matlab as if they were normal variables but then allows the QP to be saved as a dedicated object for run time solutions. We use the YALMIP function `optimizer` for this. We pass in the cost and constraint expressions, and let it know which solver to use and over which variables the cost can be optimised ($c_{i|k}$). The optimisation object is also given variables that are not to be optimised over, but are to be defined at run time, which in our case are:

- The output of the Kalman filter at time k, x_k
- The stabilising torque, which is set outside MPC and used for in the constraints at run time

The `optimizer` object can also be exported in matrix format, allowing implementation outside of Matlab. This is out of scope for this work but will useful later in TotalControl when MPC is implemented in C++. The intention is to still design the MPC in Matlab, but then export the run time data in elementary (matrix) form.

RUN TIME CONSIDERATIONS

Control loops outside of MPC, e.g. the torque-speed controller and any extras desired like drivetrain damping, must be implemented in the run time code. All filters, including the Kalman filter, are saved in state-space form, so that the state can be retained from one time step to the next.

When implementing filters outside of MPC, care must be taken not to invalidate the prediction model, especially due to phase differences introduced by the filter. A much better approach ideally is to ensure the prediction model contains the modes that you wish to dampen, and let MPC achieve this via the cost function. Increasing the fidelity of the prediction model without causing over-fitting or making the QP too hard to solve is left for future work. In the simulations reported presently, a 3P notch filter is applied to both generator speed and nacelle acceleration measurements. The filter was set quite shallow so as not to introduce much phase lag.

After each MPC optimisation is complete, although the prediction horizon is $N = 45$, only the temporally first optimised control action is applied. Both actions (pitch and torque) are scaled and offset as per (10) and in the case of torque, the stabilising torque is added.

The controller, constituting the QP and the pre- and post-processing of signals, was attached to Bladed via a message-passing system to allow the numerical calculations at each time step to be executed in Matlab, while benefiting from the realism of the full Bladed model for testing. All the results in the following section are produced in this way.

On an Intel i7, using only one core, in Matlab, the optimisation step takes around 40 ms. It's assumed running the optimiser in C++ will speed this up, although we must consider the computational power of the hardware that the MPC algorithm will eventually run on in the turbine. Exporting the QP from Matlab is made relatively easy by the YALMIP function `export`.

RESULTS

STEP SIMULATIONS

The first test of the controller was to run a deterministic wind case, where four steps are applied to the incoming wind speed as shown in Figure 6. Then the cost function was modified, and the same simulation run again to compare the impact of the cost function values. The blue lines in all the following figures show the results of the base case, i.e.

$$Q_y = \begin{bmatrix} 1 & & & \\ & 0 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}, \qquad R = \begin{bmatrix} 0 & \\ & 1 \end{bmatrix} \tag{20}$$

The matrices in (20) can be interpreted as applying equal cost to generator speed, pitch rate, nacelle acceleration and torque demand, with the obvious caveat that these variables are all subject to scaling and they represent different units. Zero cost is applied to the pitch angle measurement and pitch angle demand.

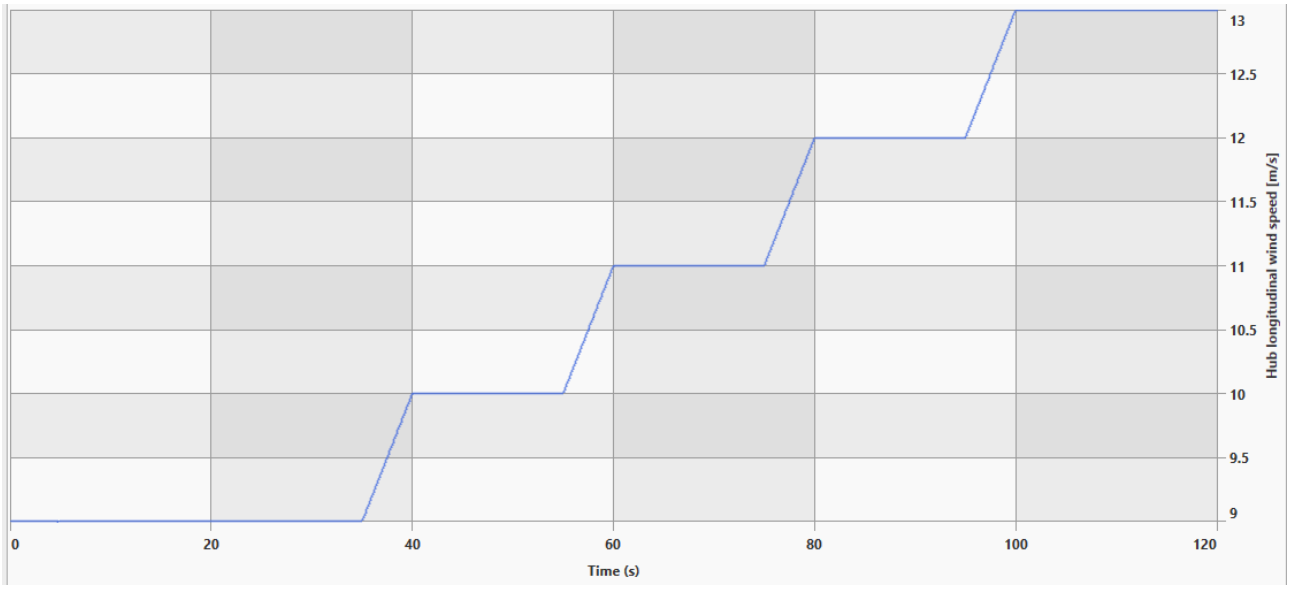

Figure 6: Steps in wind speed from 9 m/s to 13 m/s covering control regions of variable toque and pitch angle

The following figures show the impact of changing just one element in either Q or R and the corresponding change of behaviour of the turbine, as illustrated by the most relevant time series from the results.

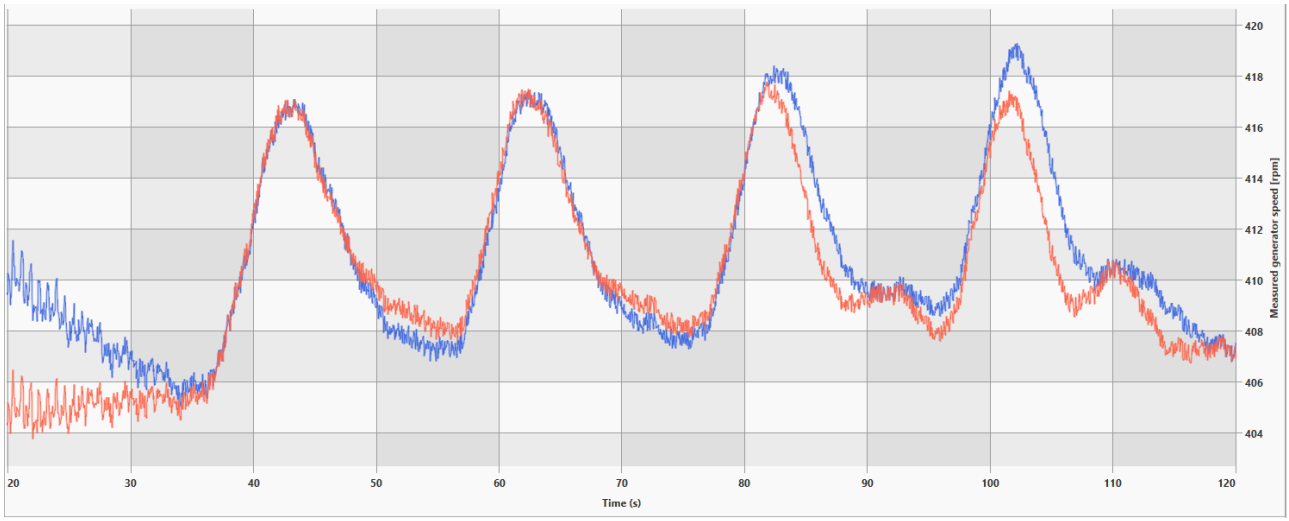

Figure 7: Generator speed response to steps in wind speed for base case (blue) and with higher cost on generator speed (red). The initial response (t<30s) can be ignored due to simulation initialisation transients. Higher cost has a clear impact on tightening rotor control, e.g. 55 s and 102 s with less overshoot.

The first cost function adjustment was to double the cost on generator speed deviations. The results in Figure 7 show that the turbine operates closer to its nominal speed (409 rpm) with this cost function. Note that the error is not halved by doubling the cost, because that would not represent the optimal solution, since it would require extreme control activity.

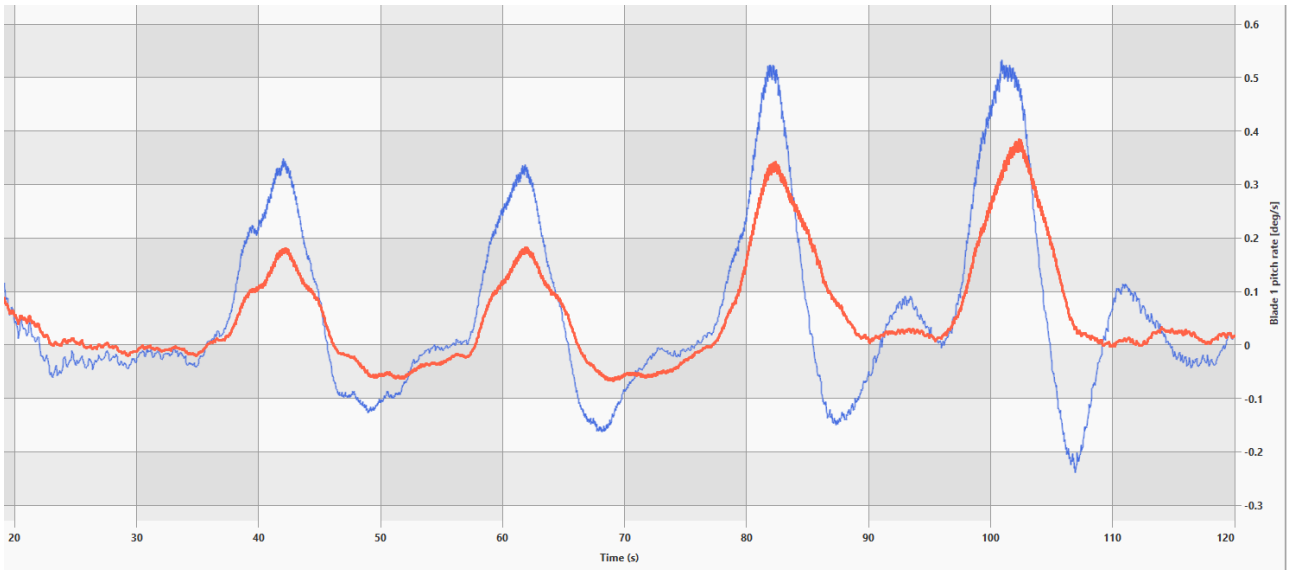

Figure 8: Pitch rate response to steps in wind speed for base case (blue) and with higher cost on pitch rate (red). The impact is very clear, with much higher damping and much lower overshoot.

The next test was to reset the cost on generator speed and instead penalise pitch rate by doubling its cost function value. Figure 8 clearly shows the impact of the change to the cost function.

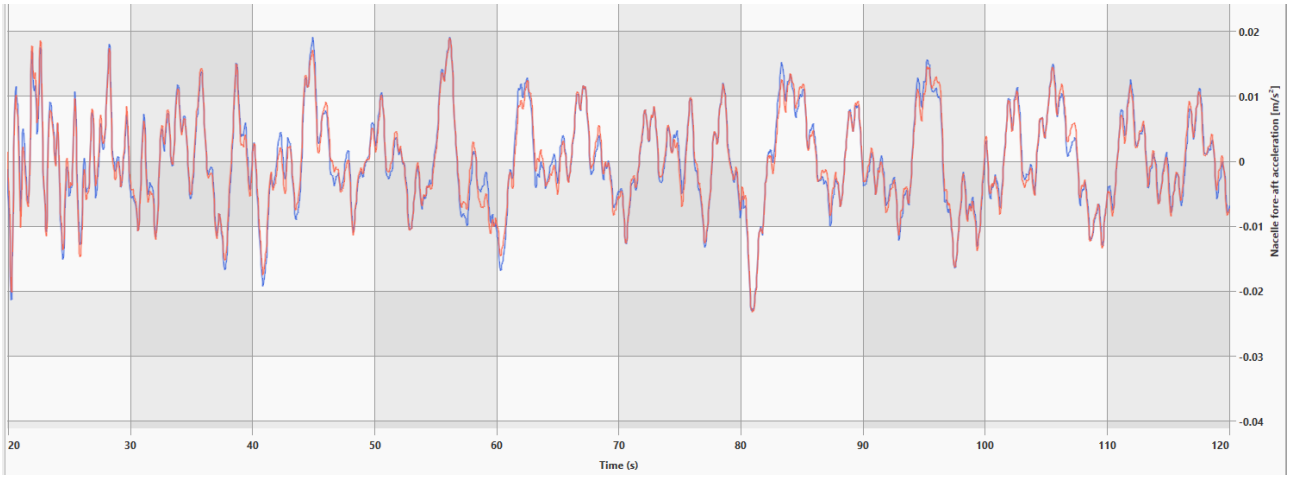

Figure 9: Nacelle fore-aft acceleration response to steps in wind speed for base case (blue) and with higher cost on nacelle acceleration (red). The impact is very subtle but occasionally lower overshoot is evident.

The next test was to reset the cost and then penalise nacelle acceleration by doubling its cost function value. Figure 9 shows that the ability of this implementation of the controller to dampen nacelle motion is limited. The reason is clearer when looking at the same results in the frequency domain as in Figure 10. Nacelle acceleration is highly affected by many structural modes that are removed by the model reduction stage when creating the prediction model. It is therefore not a surprise that the cost function has no impact on these modes. Indeed, it can be considered a success that only the first tower mode is dampened by MPC with only that mode modelled.

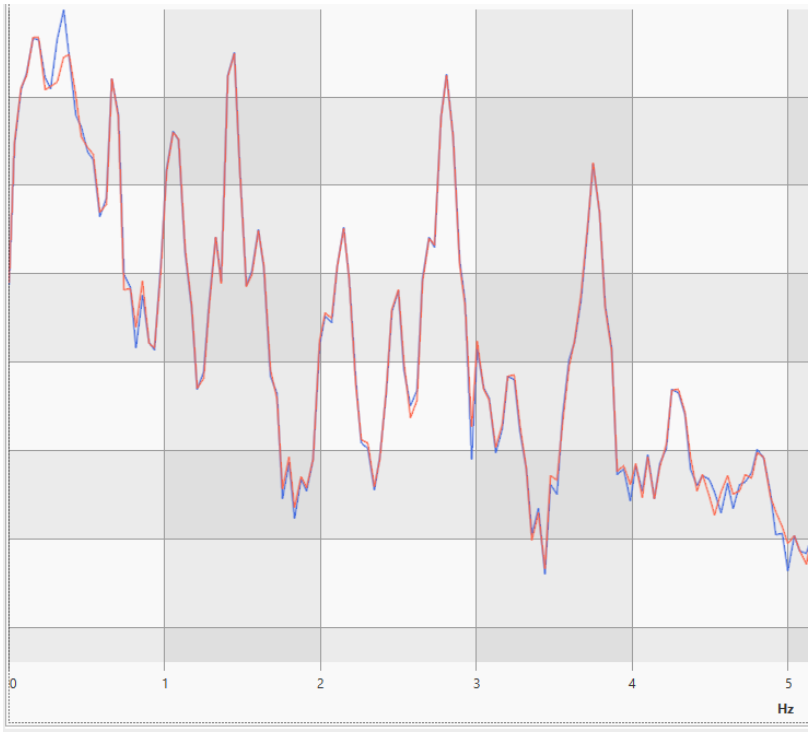

Figure 10: Power spectral density of naccelle acceleration of the results from the test of increasing cost for that measurement. A clear reduction is seen at 0.34 Hz – the 1st tower mode. 3P falls at 0.55, where there is very little activity, suggesting the filtering works. But many other modes are visible, which are not in the prediction model and are therefore outside the capability of the cost function to reduce.

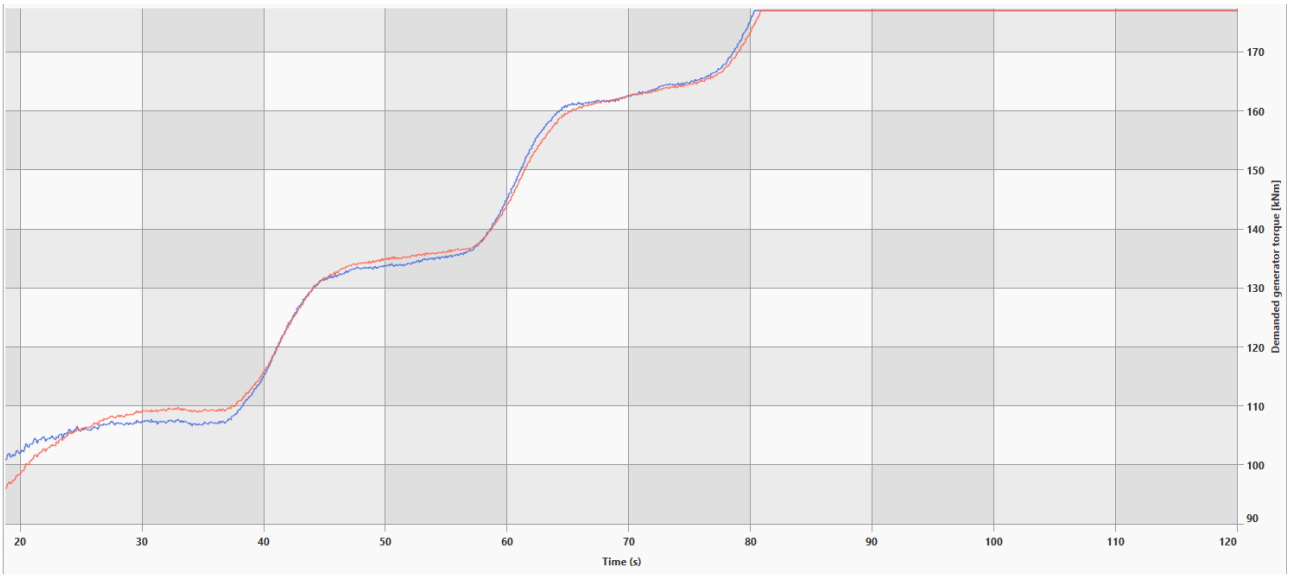

Figure 11: Demanded generator torque response to steps in wind speed for base case (blue) and with higher cost on torque demand (red). Beyond 80 s the generator torque hits its limit but before then, there are signs that the actions are less aggressive when the cost is higher, e.g. 64 s and 78 s.

Finally, the Q matrix was reset to that in (20) but now the cost on generator torque demand was doubled. Figure 11 shows that the torque response is less aggressive with the higher cost. However, the torque demand is also driven by the stabilising torque-speed loop, which is not included in the cost function because its integral part is used to reject persistent disturbance of wind speeds below rated. MPC therefore has limited authority to control torque activity scale, but succeeds at responding quickly to transients while respecting constraints.

TURBULENT SIMULATION

The step tests above are useful for checking closed-loop damping, overshoot and persistent disturbance rejection but clearly the ability to handle turbulent wind fields is essential to turbine control. The following results are from a 400 second simulation with a Kaimal turbulence wind field of mean 10 m/s at hub height and turbulence intensity (longitudinal) of 21%. There is also a standard wind shear with exponent 1/7. The wind speed varies above and below rated, exercising all the applied constraints. The cost function is as in (20).

The results are very promising, especially considering there has been no tuning applied. Taking the findings from the step tests, a designer or engineer could easily apply more cost to the variables they want to tighten.

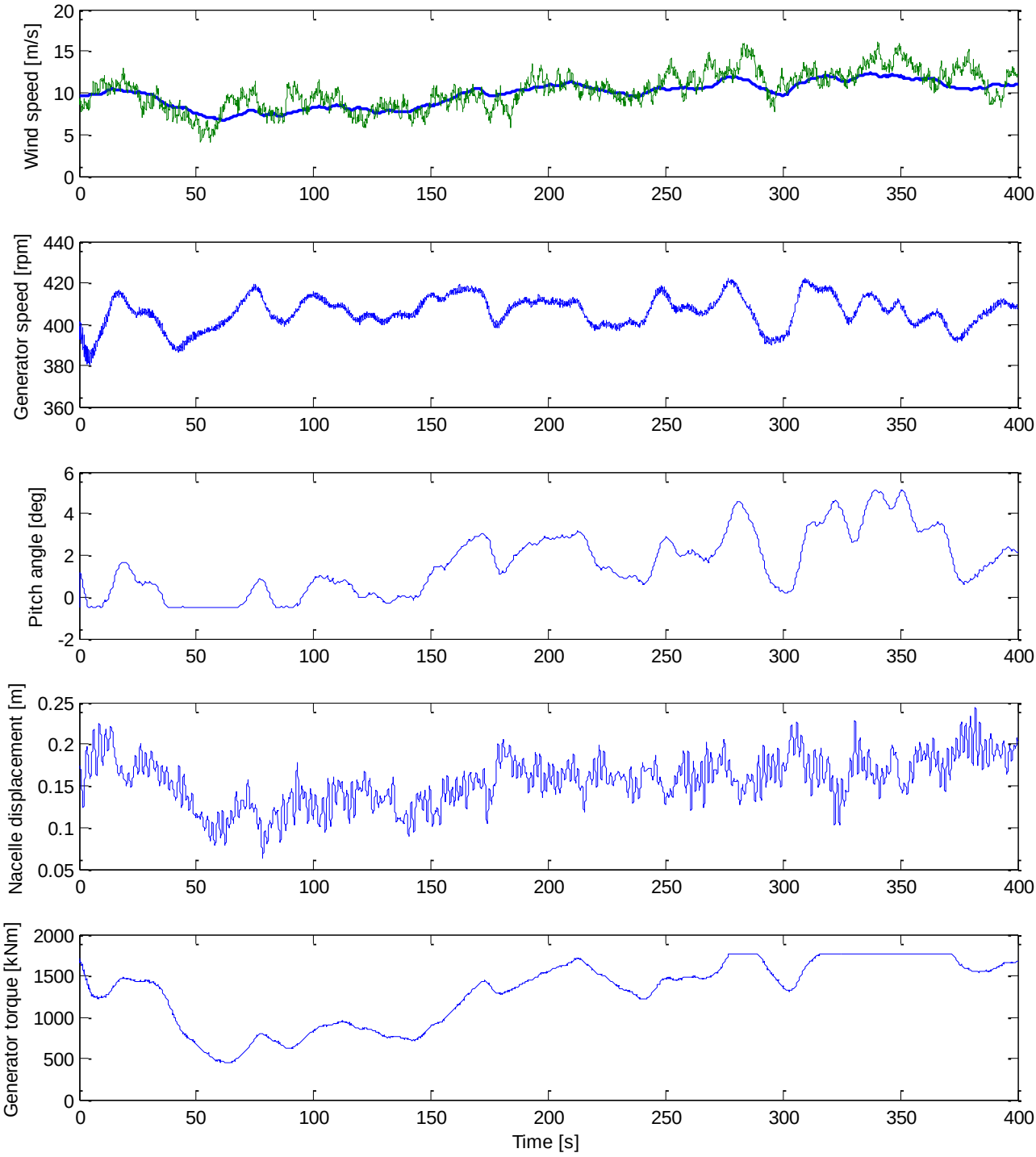

Figure 12: Results of a 400 s turbulence case showing: rotor average wind speed (blue), hub wind speed (green), generator speed, pitch angle, nacelle displacement fore-aft and generator torque. Pitch hits constraint at 40 s; torque hits constraint at 320 s; generator speed shows tight control in high tubulence intensity.

CONCLUSIONS

Replacing the entire controller in a wind turbine simulation is a major challenge. Not only does the present work achieve that, but it does so without a single manual tuning step and still results in satisfactory performance. MPC has the potential to improve wind turbine control because it is a framework that suits design through intent: the user is only exposed to parameters that have relevant meanings, i.e. the relative penalties for each measurement and the constraints they wish to impose. The computations involved to produce actuator demands are hidden and automatic.

This report gives the theory behind successful MPC and some applied techniques necessary for the particular challenge of a wind turbine: a nonlinear, multi-body flexible structure, subject to persistent stochastic disturbance, with only two control actuators, all subject to constraints.

The results show what can be achieved with this framework, but they do not purport to have improved upon classical control yet. MPC for wind turbines has some way to go, with decades of developments in classical control to catch up on. Much progress has been made in academia to add elements to the framework that handle particular needs for wind, but it is a daunting task, with much less explanation than is required to build something from the ground up. On the other side, turbine designs have a library of tools and techniques that make MPC seem a distant land. The author of the present work hopes to provide something of a bridge between the two.

Suggested future work includes:

- Export the QP to C++ for faster run times in Bladed and the option to execute in Bladed Batch
- Native MPC handling of azimuth depended disturbances, e.g. 3P filters
- Variable generator speed range for lower wind speeds, e.g. $K\omega^2$, rather than constant speed
- Start-ups and shutdowns
- Pitch rate, pitch acceleration and torque rate constraints

REFERENCES

[Evans 2014] Evans, M.A. *Multiplicative robust and stochastic MPC with application to wind turbine control*. PhD Thesis, Oxford University.

[Gilbert 1991] Gilbert, E. and Tan, K. "Linear systems with state and control constraints: The theory and application of maximal output admissible sets". *IEEE Trans. Automatic Control*, 36(9):1008-1020

[Kalman 1960] Kalman, R. "Contributions to the theory of optimal control". *Bol. Soc. Mat. Mexicana*, 5(2):102-119.

[Kumar 2009] Kumar, A. and Stol, K. "Scheduled model predictive control of a wind turbine". *47th AIAA Aerospace Sciences Meeting*.

[Lio 2014] Lio, W.H., Rossiter, J.A. and Jones, B.L. "A review on applications of model predictive control to wind turbines." *IEEE UKACC International Conference on Control*, 673-678.

[Lio 2017] Lio, W.H., Jones, B.L. and Rossiter, J.A. "Preview predictive control layer design based upon known wind turbine blade-pitch controllers". *Wind Energy*, 20(7), 1207–1226.

[Liu 2018] Liu, Y., Patton, R.J. and Shi, S. "Wind Turbine Load Mitigation Using MPC with Gaussian Wind Speed Prediction". *UKACC 12th International Conference on Control*.

[Löfberg 2004] Löfberg, J. "YALMIP: A toolbox for modelling and optimization in MATLAB." *CACSD Conference*, Taipei.

[Mayne 2000] Mayne, D., Rawlings, J., Rao, C. and Scokaert, P. "Constrained model predictive control: Stability and optimality". *Automatica*, 36(6):789-814.

[Rawlings 1993] Rawlings, J. and Muske, K. "The stability of constrained receding horizon control". *IEEE Trans on Automatic Control*, 38(10):1512-1516.

[Rossiter 1998] Rossiter, J., Kouvaritakis, B., and Rice, M. "A numerically robust state space approach to stable predictive control strategies". *Automatica*, 34(1):65-73.

[Selvam 2007] Selvam, K. *Individual Pitch Control for large scale wind turbines: multivariable control approach*. ECN Report, ECN-E-07-053.

[Sturm 1999] Sturm, J.F. "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones". *Optimization Methods and Software* 11-12:625-653.

[Sznaier 1987] Sznaier, M. and Damborg, M. "Suboptimal control of linear systems with state and control inequality constraints". *IEEE Conference on Decision and Control*, 26:761-762.

[Wright 2004] Wright, A.D. *Modern Control Design for Flexible Wind Turbines*. NREL Report, NREL/TP-500-35816.