TotalControl

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

Model predictive control theory and implementation for wind turbines
D3.8

Delivery date: 26.02.2021
Lead beneficiary: DNV
Dissemination level: Public

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No 725076.

Authors information:

Name	Organisation	Email
Martin Evans	DNV	martin.evans@dnv.com
Alan Wai Hou Lio	DTU	wali@dtu.dk

Acknowledgements/Contributions:

Name	Name	Name

Document information:

Version	Date	Prepared by	Reviewed by	Approved by
1	26.02.2021	Martin Evans, Alan Wai Hou Lio	Karl Merz	Ervin Bossanyi

Definitions/abbreviations

MPC	Model predictive control
PID	Proportional, integral, derivative
LQR	Linear quadratic regulator
LQG	Linear quadratic Gaussian
QP	Quadratic program
SISO	Single input, single output
MIMO	Multiple input, multiple output

Executive summary

Model predictive control (MPC) has been used in the process industry for decades. There are numerous more recent examples of it being successfully applied to faster acting plants, for example quadcopter drones. This has led academia and wind turbine manufacturers to consider controlling wind turbines with MPC. However, the wind turbine control problem poses challenges that MPC is not directly suited to, notably low observability of states, and very significant uncontrolled disturbances from turbulence. Previous attempts to apply MPC to wind turbine control have either simplified the problem by removing structural degrees of freedom, or reduced the range of operation down from cut-in/cut-out to a single wind speed with turbulence.

This paper gives a broad overview of MPC theory, then demonstrates some offline design steps that can be taken to tackle the problem in its entirety. The resulting control scheme allows the engineer to take an aeroelastic model of a wind turbine, and with almost no tuning, produce an MPC controller that works in the full wind speed range in a simulation with full realism. It concludes with a discussion of further considerations to make when moving the controller into field testing.

1 Introduction

Simple linear MPC is not sufficient for application to a problem as complex as wind turbine control. As rotor sizes grow, more structural modes with low damping appear, making application of MPC more problematic. This paper explains the underlying reasons why MPC can fail when applied to such a system, and what can be added in order to get it to run in a realistic simulation. We then perform a worked example of MPC design, demonstrate its effectiveness in such a simulation, and discuss what would need to happen to move the algorithm onto a real turbine.

The explanation of each aspect of the theory is intentionally brief, to allow a broad range of subjects to be covered. References are given to other works wherever possible to allow the reader to go more into depth if desired. When discussing the application of MPC to the wind turbine problem, we mean running it as the power production control algorithm in a Bladed simulation or similar, with all the available realism and modes turned on. It is also assumed that any linearisation is performed by Bladed, i.e. that no hand-derived models are used.

The fullness of the linear model and simulation is unusual in the research field of MPC for wind turbine control. We consider it important for the development of MPC as a viable control framework for wind turbines that the design process should be as simple and automated as possible, and that no concessions on the simulation settings should be made in order to achieve the desired results.

2 Linear MPC

We define a discrete linear time invariant (LTI) system as one whose state x at time step $k+1$ is based on its state at time step k and the control action u at time step k as follows:

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

In MPC, the next control action to be applied to the plant is optimised online, but so are all future control actions. This is what allows the constraints to be anticipated in advance. When performing this optimisation, we use the following notation: the control action i time steps into the future, which is subject to optimisation, is written $u_{i|k}$, where k is the current time step. After optimisation, $u_{0|k}$ is applied to the plant, which becomes u_k. All subsequent optimised

actions are re-calculated at the next time step. The (estimated) state at the current time step is written $x_{0|k}$ and future states that are dependent on the control actions being optimised are written $x_{i|k}$.

If we aim to regulate the state and action to zero over the infinite future horizon, and penalise these quantities according to a quadratic cost function, which is a reasonable choice because there are well known solutions to such a problem, then we can write that cost function to be minimised as follow:

$$J(\mathbf{u}_k) = \sum_{i=0}^{\infty} x_{i|k}^T Q x_{i|k} + u_{i|k}^T R u_{i|k} \tag{2}$$

Here, Q is a positive semidefinite matrix penalising states, R is a positive definite matrix penalising control actions and $\mathbf{u}_k$ is the infinite sequence of control actions subject to optimisation, starting with $u_{0|k}$. It would not be possible to optimise this if all future control actions are independent variables, since there are infinitely many of them. Instead, a commonly applied technique is the *dual mode* paradigm (see Rossiter et al. [1998]). Control actions $u_{0|k}$ to $u_{N-1|k}$ are optimised independently (Mode 1), whereas $u_{N|k}$ onward follow linear feedback (Mode 2). This reduces the degrees of freedom for Mode 2 to only $x_{N|k}$, and hence the entire cost of Mode 2 can be expressed in terms of only that variable and $\bar{Q}$, the solution to the Lyapunov equation.

The cost function in the dual mode paradigm is thereby:

$$J_d(\mathbf{u}_k) = x_{N|k}^T \bar{Q} x_{N|k} + \sum_{i=0}^{N-1} x_{i|k}^T Q x_{i|k} + u_{i|k}^T R u_{i|k} \tag{3}$$

Without constraints, N can be set arbitrarily small and the problem is equivalent to LQR (linear quadratic regulation, Kalman [1960]). Indeed, the LQR gain K is used for Mode 2 control actions. But the power of MPC is that it is an optimisation subject to constraints. We limit ourselves to linear constraints, which combined with the quadratic cost, creates a well known problem called a quadratic program (QP).

Such linear constraints can be written with two additional matrices, F, G as follows:

$$F x_{i|k} + G u_{i|k} \leq \mathbf{1} \qquad i = 0, \dots N-1 \tag{4}$$

Here, $\mathbf{1}$ is a vector of ones. These constraints only apply to Mode 1. To ensure all states and actions satisfy constraints in Mode 2, a *terminal set* is required, denoted X_f, which can be constructed according to e.g. Gilbert and Tan [1991]. The way the terminal set works is that it is invariant under Mode 2 dynamics, i.e.:

$$x_{N|k} \in X_f \implies x_{N+1|k} \in X_f \tag{5}$$

We can now state the dual mode linear MPC algorithm as:

$$\begin{aligned} &\text{Given } x_{0|k}, \\ &\qquad \mathbf{u}_k^* = \min_{\mathbf{u}_k} J_d(\mathbf{u}_k) \\ &\qquad \text{s.t. } F x_{i|k} + G u_{i|k} \leq \mathbf{1} \quad i = 0, \dots N-1 \\ &\qquad\qquad x_{N|k} \in X_f \end{aligned} \tag{6}$$

This assumes no time is required to perform the optimisation to obtain $\mathbf{u}_k^*$. Since online solution of a QP will probably take most of a time step to perform, a more correct formulation would have $u_{0|k}$ carried over from the previous optimisation rather than treating it as an optimisation variable. This way, the measurement vector is taken, the control action optimised previously, $u_{1|k-1}^*$ is immediately applied to the controller's output buffer, then the optimisation routine can start. For simplicity of notation, the remainder of this paper will ignore this implementation detail.

In Mode 1, each subsequent predicted state $x_{i|k}$ contains an increasing power of A. To improve numerical conditioning, the unconstrained optimal feedback K is applied autonomously to the plant in Mode 1, just like in Mode 2, and the optimisation variables are added to that feedback, as in Rossiter et al. [1998]. These new variables are denoted $c_{i|k}$ and the full Mode 1 sequence is accordingly:

$$\mathbf{c}_k = [c_{0|k} \dots c_{N-1|k}] \tag{7}$$

The Mode 1 dynamics can be re-written with $\Phi = A + BK$ as (for $i < N$):

$$\begin{aligned} x_{i+1|k} &= \Phi x_{i|k} + B c_{i|k} \\ u_{i|k} &= K x_{i|k} + c_{i|k} \end{aligned} \tag{8}$$

3 Recursive feasibility

As given in Rawlings and Muske [1993], if the constraints of (6) are satisfied then it is guaranteed that there will be a feasible solution to the same problem one time step later. This is called *recursive feasibility*. Essentially, while it might no longer be the optimal solution at time step $k+1$, the shifted values from the previous optimisation do at least satisfy the same constraints.

$$\begin{aligned} x_{i|k+1} &\leftarrow x_{i+1|k} \qquad i = 0 \dots N-1 \\ u_{i|k+1} &\leftarrow u_{i+1|k} \qquad i = 0 \dots N-2 \end{aligned} \tag{9}$$

From (5), $x_{N|k} \in X_f \Rightarrow x_{N|k+1} \in X_f$. All that remains is to check there is a feasible value for $u_{N-1|k+1}$, and indeed $u_{N-1|k+1} = u_{N|k} = K x_{N|k}$ is acceptable, due to the way the terminal set is defined and calculated.

Without recursive feasibility, the MPC algorithm running in an application can solve the QP one time step, then fail to find one the following step. A back-up algorithm would be required to provide control actions while the QP is infeasible, leading to system dynamics that are not known at design time. The following sections give some systems where recursive feasibility is not guaranteed without additional work done on the algorithm. The engineer is required to choose whether to invest in these or similar algorithm upgrades or to forego recursive feasibility.

4 Additive uncertainty

4.1 Problem

We now present a linear system with additive disturbance. Since the disturbance can be set to zero, we can re-state our system from (1) as:

$$x_{k+1} = Ax_k + Bu_k + Ew_k \tag{10}$$

Here, E is a matrix with number of columns equal to the number of independent sources of additive uncertainty. Vector w_k is the additive disturbance source at time step k. It is random, but we cannot treat it as white noise, since that would be unbounded and therefore impossible to guarantee any constraints in the future. Instead, we bound the value as $w_{\min} \le w_k \le w_{\max}$, element-wise.

Without an adjustment to the control law, this additive uncertainty would grow over the prediction horizon, because while for most probability distributions it is vanishingly unlikely that all disturbances are at the same extreme, we have no probabilistic framework so can only use the extreme bounds of the distribution.

To prevent such growth in uncertainty in the predicted states, we feed back information that will be made available to the controller as time proceeds through the prediction horizon. That

is to say, while at time step k we don't know the value of w_k, we do at time step $k+1$, or at least we will know Ew_k. So, as in Kouvaritakis et al. [2010], we decompose predicted states and inputs into the deterministic part $z_{i|k}$ and the uncertain part $e_{i|k}$:

$$\begin{aligned} x_{i|k} &= z_{i|k} + e_{i|k} \\ u_{i|k} &= K(z_{i|k} + e_{i|k}) + c_{i|k} \end{aligned} \tag{11}$$

The deterministic and uncertain parts evolve separately. The full expression for predicted states is:

$$\begin{aligned} x_{i+1|k} &= \Phi x_{i|k} + Bc_{i|k} + Ew_{i+k} \\ z_{i+1|k} + e_{i+1|k} &= \Phi(z_{i|k} + e_{i|k}) + Bc_{i|k} + Ew_{i+k} \end{aligned} \tag{12}$$

The above equation can be simply split out; the deterministic part contains optimisation variables and the uncertain part contains unknown disturbances.

$$\begin{aligned} z_{i+1|k} &= \Phi z_{i|k} + Bc_{i|k} \\ e_{i+1|k} &= \Phi e_{i|k} + Ew_{i+k} \end{aligned} \tag{13}$$

Since $e_{0|k} = 0$ and $e_{i|k}$ evolves independently of any variables other than the disturbance, which is bounded in a way that is independent of time, we can drop the k from the notation and refer to the uncertain part as e_i.

Re-writing constraints (4) with these terms, for $i = 0, \dots N-1$,

$$\begin{aligned} Fx_{i|k} + Gu_{i|k} &\leq \mathbf{1} \\ (F+GK)z_{i|k} + Gc_{i|k} &\leq \mathbf{1} - (F+GK)e_i \end{aligned} \tag{14}$$

4.2 Solution

Now, the left-hand side of (14) is deterministic and the right-hand side is completely free of initial conditions and optimisation variables. So we can pre-calculate the right hand side:

$$g_i = (F+GK)e_i \tag{15}$$

The vector g_i is called the tightening parameter. It is applied to the constraints for Mode 1, that only depend on i, the number of time steps into the future. The Mode 2 operation works the same as in the disturbance-free MPC algorithm, although there is an offline adjustment to the terminal set X_f required. Details of this change are omitted here – it only requires the same design process as X_f but with the N-th tightening parameter included, the one at the end of Mode 1.

With the above improvement to the MPC algorithm, called *robust* MPC, recursive feasibility is assured for systems with bounded additive uncertainty. It might be very conservative, however. For a disturbance with a triangular distribution for example, in the form $w_k = r_{1,k} - r_{2,k}$, where those two terms are drawn from a uniform distribution, the bounds are $|w_k| \leq 1$ but with a probability approaching zero at those bounds.

To summarise the solution, the following is an update to the algorithm in (6), where X_f and g_i are calculated offline.

$$\begin{aligned} &\text{Given } z_{0|k}, \\ &\quad \mathbf{c}_k^* = \min_{\mathbf{c}_k} J_d(\mathbf{c}_k) \\ &\text{s.t. } (F+GK)z_{i|k} + Gc_{i|k} \leq \mathbf{1} - g_i \quad i = 0, \dots N-1 \\ &\quad z_{N|k} \in X_f \end{aligned} \tag{16}$$

Numerous approaches have been made that allow future constraint violation with a prescribed probability, called *stochastic* MPC (see Kouvaritakis and Cannon [2016]). These algorithms are necessarily more computationally demanding than the robust case and since the purpose of this paper is to build a bridge between theory and application to wind turbine control, we will not further consider stochastic MPC. In any case, stochastic MPC does not avoid recursive feasibility considerations, because the predicted probability of violation of constraints is itself a constraint that must be met at every time step.

Robust MPC is no more computationally demanding at run time than nominal (disturbance-free) MPC, due to the separation constructed in (14). It is for the control engineer to decide whether robust MPC is sufficient, i.e. whether suitable bounds can be placed on the additive uncertainty without making the performance too conservative, or whether some treatment of the probabilities is worth the extra run time computation.

Robust MPC has been applied to the wind turbine problem in Koerber and King [2013] and in Mirzaei et al. [2012]. The former is not simulated with a full complexity turbine model. The latter is, but only at one wind speed, far above the transition region.

5 Multiplicative uncertainty

5.1 Problem

The system with additive uncertainty in (10) has known solution whereby the effect of the uncertainty is managed offline with tightening parameters. It does, however, assume the linearised system dynamics (A, B) are deterministic. This section explores the implication of allowing uncertainty in the linearisation. Our new system is defined as follows:

$$x_{k+1} = (A + a_k)x_k + (B + b_k)u_k \tag{17}$$

Here, a_k, b_k are uncertain time-varying matrices the same shapes as A, B. Note we assume no additive uncertainty for this discussion, for ease of notation. The techniques for additive and multiplicative uncertainty can be applied together if both types of uncertainty are present.

As before, the uncertain values in this system must be bounded, because otherwise there is a non-zero probability of violating the future constraints regardless of the control actions, rendering robustness impossible. For convenience, we introduce a new symbol Δ_k, which is simply the concatenation of the uncertain variables:

$$\Delta_k = [\ a_k \mid b_k\ ] \tag{18}$$

We don't know the value of Δ_k but we can bound it by a polytope $\mathbf{D}$ that contains the origin. There are two equivalent definitions of a polytope, both requiring a set of ρ vertices denoted $\Delta^{(1)}, \ldots, \Delta^{(\rho)}$. The definition first relates to points inside the set, which in words states that Δ is in $\mathbf{D}$ if it can be written as a weighted sum of the vertices where all the weights are positive and sum to unity:

$$\begin{aligned} \Delta \in \mathbf{D} \iff \exists\ d_1, \ldots, d_\rho \text{ s.t. } \Delta = \sum_{j=1}^{\rho} d_j \Delta^{(j)}, \\ d_j \geq 0,\ \sum_{j=1}^{\rho} d_j = 1 \end{aligned} \tag{19}$$

The second definition relates to points outside the set, which in words says Δ is not in $\mathbf{D}$ if there is a vector v and scalar α such that a plane normal to v can separate the vertices from Δ as follows:

$$\begin{aligned} \Delta \notin \mathbf{D} \iff \exists\ (v, \alpha) \text{ s.t. } v \cdot \mathrm{vec}(\Delta^{(j)}) \leq \alpha, \quad \forall j \\ v \cdot \mathrm{vec}(\Delta) > \alpha \end{aligned} \tag{20}$$

Here, vec is the vectorisation operator. The scalar α is the position along that normal vector v such that all the vertices of $\mathbf{D}$ are on one side of the plane and the point in question Δ is on the other side.

Defined as in the above description, $\mathbf{D}$ is also known as the *convex hull* of the vertices:

$$\mathbf{D} = \text{co}\left\{\Delta^{(j)} : \ j = 1, \ldots, \rho\right\} \tag{21}$$

5.2 Naive approach

The first approach to creating a robust MPC algorithm to handle multiplicative uncertainty might be to convert the extrema of that uncertainty into additive uncertainty and apply the methods of Section 4.2. Constraints (4) form a convex set – if that set $\mathbf{S}$ is also bounded then we could find the vertices that define it as a convex hull.

Denoting the number of such required points in the combined space of states and control actions as q, we can evaluate offline all ρq combinations of extrema from both convex hulls and then find additive uncertainty that covers all potential outcomes. The prediction dynamics are:

$$x_{i+1|k} = (A + a_{i+k})x_{i|k} + (B + b_{i+k})u_{i|k} \tag{22}$$

We want to find a value of E such that:

$$\begin{aligned} \{(A + a)x + (B + b)u : [\ a \mid b\ ] \in \mathbf{D}\} \subset \\ \{Ax + Bu + Ew : 0 \leq w \leq 1\}\ \forall (x, u) \in \mathbf{S} \end{aligned} \tag{23}$$

This is a process that can be performed offline and convert the multiplicative problem into an additive problem. However, $\mathbf{S}$ tends to be quite large and when combined with all possible values of Δ, this approach will yield a very conservative controller, if a terminal set can even be constructed with such large tightening parameters. Additionally, many applications will have systems with at least one state that must be allowed to vary without tight constraints. Applying very loose constraints to these states would make the vertices of $\mathbf{S}$ too far apart to be feasible in such an algorithm.

5.3 Geometric approach

In Evans et al. [2015], an approach is taken that is less conservative and also allows the state and actions to be constrained in an unbounded way. The method employed is *polytopic tubes*, a sequence of linear constraints placed on the uncertain part of the state prediction $e_{i|k}$ throughout Mode 1. The tubes are designed so that they contain the entire Mode 1 uncertainty in the prediction horizon, and the constraints on the state and actions are applied to the Minkowski sum of the deterministic part $z_{i|k}$ and the tube cross sections.

As in (11) we decompose the dynamics:

$$\begin{aligned} x_{i+1|k} &= (A + a_{i+k})x_{i|k} + (B + b_{i+k})u_{i|k} \\ z_{i+1|k} + e_{i+1|k} &= (A + a_{i+k})(z_{i|k} + e_{i|k}) + (B + b_{i+k})(Kz_{i|k} + Ke_{i|k} + c_{i|k}) \end{aligned} \tag{24}$$

and again we can write the dynamics of each part separately:

$$\begin{aligned} z_{i+1|k} &= \Phi z_{i|k} + Bc_{i|k} \\ e_{i+1|k} &= (a_{i+k} + b_{i+k}K)z_{i|k} + (\Phi + a_{i+k} + b_{i+k}K)e_{i|k} + b_{i+k}c_{i|k} \end{aligned} \tag{25}$$

In this decomposition, the uncertain part *does* contain optimisation variables, so cannot be handled entirely offline. Furthermore, the vertices of $\mathbf{D}$ cannot be used to create constraints at

every prediction step, because the uncertainty is time-varying, and that would lead to ρ^N times the number of constraints as in the additive uncertainty algorithm.

The polytopic tube approach instead bounds $e_{i|k}$ by a set $\Pi_{i|k}$:

$$e_{i|k} \in \Pi_{i|k} = \{e : Ve \leq \alpha_{i|k}\} \tag{26}$$

Here, V is a constant real matrix, chosen at design time, and represents the normal vectors of a polytope that surrounds $e_{i|k}$. The vector $\alpha_{i|k} \geq 0$ is an additional optimisation variable to be assigned online. Using Bitsoris [1988], matrices are found offline that create linear constraints on $z_{i|k}$, $c_{i|k}$ and $\alpha_{i|k}$ to ensure that (4) is met, and similarly on $z_{N|k}$ and $\alpha_{N|k}$ to ensure that $x_{N|k} \in X_f$.

An illustration of a bounding polytope using this formulation is given in Figure 1. The set is bounded by a collection of half-planes (lines in 2D); each has a fixed orientation defined by a unit vector, which it is its normal vector. The distance of the nearest point on the bounding edge to the origin is the corresponding element of α.

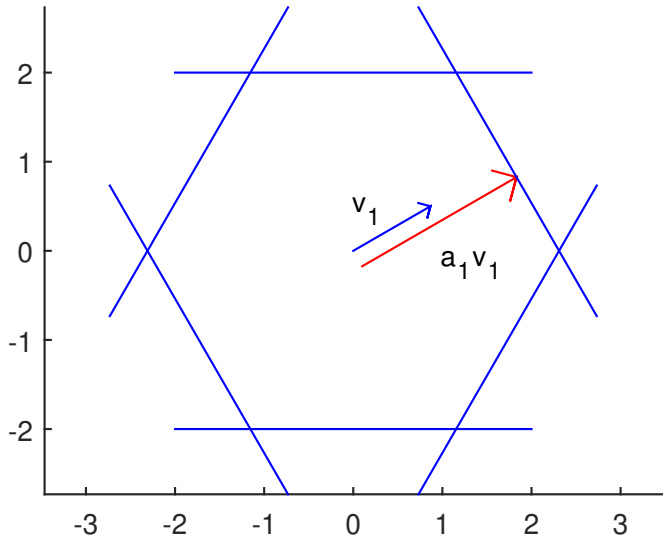

Figure 1: A polytope defined as in (26), useful for constraining uncertainty. A point e is in the set if $Ve \leq \alpha$, where $V = [v_1\ v_2\ \cdots\ v_6]^T$ and $\alpha = [a_1\ a_2\ \cdots\ a_6]^T$.

To summarise the solution, the following is an update to the algorithm in (16), where X_f is updated to have extra dimensions and where H is calculated offline according to Bitsoris [1988].

$$\begin{aligned} &\text{Given } z_{0|k}, \\ &\quad \mathbf{c}_k^* = \min_{\mathbf{c}_k, \mathbf{a}_k} J_d(\mathbf{c}_k) \\ &\text{s.t. } (F+GK)z_{i|k} + Gc_{i|k} + H\alpha_{i|k} \leq \mathbf{1} \quad i = 0, \ldots N-1 \\ &\quad (z_{N|k}, \alpha_{N|k}) \in X_f \end{aligned} \tag{27}$$

The optimisation variable for the scaling of the polytopic tube faces is $\mathbf{a}_k = [\alpha_{1|k} \ldots \alpha_{N|k}]$.

All of this is a substantial amount of design effort, not to mention the required redesign of the terminal set. Without handling multiplicative uncertainty, recursive feasibility cannot be guaranteed for this kind of system, and unfortunately, uncertainty on the plant dynamics is a natural consequence of linearisation of a real world system. So it is prudent to look at ways to cope without recursive feasibility.

6 Soft constraints

The aim of robust MPC is to ensure the constraints will never be violated, regardless of the random realisation of uncertainty in the system. However, it is shown above to require a great deal of design effort, which gets even more complicated when the reference shifts as in the wind turbine control problem.

The design of soft constraints is a method of avoiding infeasibility by introducing slack variables into the QP (see Scokaert and Rawlings [1999]). The constraints of (4) are modified as follows, for $i = 0, \dots N-1$:

$$\begin{aligned} Fx_{i|k} &\leq \mathbf{1} + \epsilon_{i|k} \\ Gu_{i|k} &\leq \mathbf{1} \end{aligned} \tag{28}$$

The cost function from (3) is now modified to penalise violations:

$$J_s(\mathbf{u}_k, \boldsymbol{\epsilon}_k) = x_{N|k}^T \bar{Q} x_{N|k} + \sum_{i=0}^{N-1} x_{i|k}^T Q x_{i|k} + u_{i|k}^T R u_{i|k} + \epsilon_{i|k}^T S \epsilon_{i|k} \tag{29}$$

Here $\boldsymbol{\epsilon}_k$ is the sequence of slack variables $\epsilon_{0|k}, \dots, \epsilon_{N-1|k}$ and S is a positive definite matrix penalising the slack variables. We want S to be large enough to prevent the optimiser favouring constraint violation over corrective control action. But very large values in S will lead to numerical ill-conditioning and/or high numbers of iterations of the solver per time step.

As explained in Kerrigan and Maciejowski [2000], it is not possible to pose an MPC algorithm as a QP and avoid the constraints being violated unnecessarily. In Zeilinger et al. [2011], the authors penalise the slack variables differently, which results in a second order cone program (SOCP) rather than QP. This type of problem can still be solved efficiently online.

For any real-world MPC design, the engineer must choose between robust, stochastic or soft-constrained MPC. See applications of soft constraints in wind turbine control in Henriksen et al. [2012], Lio et al. [2017]. Real systems have uncertainty and without one of these strategies, infeasibility is inevitable. The behaviour is no longer optimal (with respect to the original cost function) once the slack variables are active (non-zero), and depending on the choice of slack variable cost, the response as the system approaches the constraint could be far from desirable and hard to tune. Note, in closed-loop operation, the slack variables may be activated for states further in the prediction horizon even without the actual sequence of states x_k violating any constraints. For example $x_{9|k}$ might require a slack variable but nine time steps later the system has recovered so that $x_{0|k+9} = x_{k+9}$ is within constraints.

7 Multiple linearisations

If the linear system is calculated from a nonlinear system $x_{k+1} = f(x_k, u_k)$, the choice of the linearisation point will have an impact on the values of A, B. Writing the nonlinear dynamics as a Taylor series up to the first derivative, we have:

$$x_{k+1} = f(\bar{x}, \bar{u}) + A(\bar{x}, \bar{u})[x_k - \bar{x}] + B(\bar{x}, \bar{u})[u_k - \bar{u}] + \cdots \tag{30}$$

Now A, B are functions of the linearisation point. It is necessary to choose $\bar{x}, \bar{u}$ such that $f(\bar{x}, \bar{u}) = \bar{x}$, i.e. an equilibrium point. This is because the cost is applied to variations from that operating point and the infinite horizon cost must be finite, which in turn requires the system to have an expected cost per time step that approaches zero in Mode 2.

Higher order terms in the Taylor series are ignored by a linearisation, which contributes to the uncertainty in the system. Furthermore, as the system is disturbed from its equilibrium, even the first-order terms are no longer correct, adding more uncertainty.

While most states and inputs are regulated as close as possible to the linearisation point, the wind speed state, being uncontrollable, cannot be. This draws many other states and inputs away from the linearisation point in order to keep the system in equilibrium, such as pitch angle and rotor thrust.

Multiple linearisations is something classical control has handled in wind turbine control for decades. The pitch-speed control loop, for example, is *gain scheduled* so as to respond less aggressively at high wind speeds. Indeed Bladed provides linearisations at multiple wind speeds for this very purpose.

Designing an MPC controller at multiple linearisation points is a substantial challenge. In Kumar and Stol [2009] and in Soliman et al. [2011], multiple MPC models are created over a range of wind speeds. Multiple Kalman filters are maintained; multiple QPs are solved online. The control actions are mixed or switched based on a filtered wind speed estimate.

Due to the complexity involved in implementing multiple model MPC, it should be avoided unless necessary.

8 State estimation

In all the above discussion, the state at time step k is assumed to be known. In reality, the controller only has access to a vector of measurements at each time step, denoted y_k, and the state must be estimated from the recent history of measurements and control actions. For this section we limit the complexity of the system to a single linearisation with additive uncertainty on both the dynamics and the measurement:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Ew_k \\ y_k &= Cx_k + Du_k + \nu_k \end{aligned} \tag{31}$$

Here, ν_k is the measurement uncertainty, or *noise* at time step k. We now outline the process of designing a Kalman filter, which is the optimal process to estimate the state under the prescribed circumstances. Firstly, the uncertainty is assumed to have zero mean and known covariance as follows:

$$\mathbb{E}(w_k w_k^T) = Q_e, \quad \mathbb{E}(\nu_k \nu_k^T) = R_e \tag{32}$$

where $\mathbb{E}$ is the expectation operator. Q_e, R_e are covariance matrices. Using standard techniques to find the solution L to the Riccati equation, we are able to update the estimated state for the *subsequent* time step $\hat{x}_{k+1}$, based on the current time step state estimate, measurement and control action. This estimation step is performed after the control action is calculated, and retained for one time step, since it requires a concurrent measurement and action.

$$\hat{x}_{k+1} = A\hat{x}_k + Bu_k + L(y_k - C\hat{x}_k - Du_k) \tag{33}$$

The Kalman filter is optimal in the mean square error sense. Likewise with the LQR gain. Indeed, in terms of optimality, the two calculations are the best we can do. Furthermore, due to the separation principle, K and L can be designed separately. But depending on the available measurements, some states can be estimated very poorly. Additionally, as shown in the famous paper by Doyle [1978], the closed-loop performance of a system subject to uncertainty, with a Kalman filter providing the state estimates, is not guaranteed to be acceptable, or even stable.

Therefore, after configuring the Kalman filter with the covariance matrices (these can be estimated from time series), it is important to close the loop with the linear plant and LQR gain and then inspect the closed-loop poles. Combining (31), (33) and $u_k = K\hat{x}_k$, ignoring uncertainty, and stacking the state and estimated state, we get:

$$\begin{bmatrix} x_{k+1} \\ \hat{x}_{k+1} \end{bmatrix} = \begin{bmatrix} A & BK \\ LC & A + BK - LC \end{bmatrix} \begin{bmatrix} x_k \\ \hat{x}_k \end{bmatrix} \tag{34}$$

If the eigenvalues of the matrix in (34) are close to the unit circle then the closed-loop system will have low damped modes and could be insufficiently robust to model-plant mismatch. Two helpful stages can be introduced into the design of the controller before the Kalman filter is constructed. Firstly, some system poles can be damped by classical SISO feedback, and secondly the linear model can be reduced in order, removing the states that are hard to estimate and instead filtering them out of the measurements. These two approaches are discussed next.

9 Incorporating classical feedback

As introduced in Section 2, a common technique used in MPC design is the dual-mode paradigm. In Mode 1 (from the current time up to the control horizon), the control actions are freely optimised. In Mode 2 (beyond the control horizon), the control law follows a linear feedback controller. The control action is then equal to LQR if the constraints are inactive. This section will discuss some benefits of incorporating a pre-determined linear controller in the MPC design.

9.1 Bringing certainty to nominal performance and robustness

The pre-determined linear controller can be designed separately from the MPC using common techniques that are familiar with the engineers, for example, PID, pole-placement and frequency-domain loop shaping. The engineers can then use the classical control techniques to analyse the closed-loop systems, for example, Bode diagram (gain/phase margins), Nyquist or pole-zero plot. By doing so, the engineers will have confidence in the closed-loop performance and robustness in the absence of constraint violations.

In Lio et al. [2017], an MPC design was built upon a pre-determined rotor speed regulation loop and an individual pitch controller. Both pre-determined controllers were designed using a loop-shaping technique in the frequency-domain. The pre-determined controllers were then integrated into the MPC design by solving inverse optimal control problem, where the weights in the cost function of the MPC need to satisfy a set of conditions. Therefore, the MPC control action is only active if a constraint violation is expected. Also, feed-forward information such as wind measurements from light detection and ranging (LIDAR) systems can be easily formulated into this modular MPC framework.

Another benefit of formulating the MPC upon an existing controller is in its transparency. The separate nature of the modular MPC framework offers transparency in the performance, where one can clearly see the additional benefits of constraint and feed-forward handling of the MPC on the pre-determined linear controller. In traditional MPC, the monolithic design formulated the feedback, feed-forward and constraints into one single optimisation problem, making it non-trivial to visualise the benefits of each element.

9.2 Making model reduction trivial

A pre-determined controller can be designed to control the system modes that are important or lightly damped. Subsequently, the engineers can easily remove any redundant modes using a model reduction technique without eliminating any important system modes. For example, the linear model obtained by linearising a nonlinear wind turbine typically consists of a high number of system states. It can be computationally expensive to use this high-order linear model in the MPC. Moreover, if the cost function is not carefully designed, the MPC control action might respond to some structural excitation unnecessarily and undesirably. Tuning a state observer for a high-order model is cumbersome and that might result in poor state estimation, thus leading to a poor MPC performance.

For these reasons, the linear model needs to be reduced in order. It is non-trivial to determine which system states are appropriate to eliminate. Simple model reduction method might work

if the system states are weakly coupled, where the removed states or the unmodelled dynamics have little or no influence to the closed-loop system. However, in wind turbines, some states are strongly coupled with each other. By simply ignoring some states in the linear model, the performance of the MPC is rather poor or even reaching instability.

A well-designed feedback controller can alter the dynamics of a system to its desired form, which makes model reduction easier. Given that the feedback controller is designed to shape the response of the closed-loop system, a subsequent model reduction can be applied to the closed-loop system with less important dynamic information lost. Compared to using model reduction on an open-loop system, this method ensures the critical system states are not removed and taken care by the pre-determined controller. Moreover, the nominal performance of the closed-loop system is guaranteed by the feedback controller.

10 Model reduction and augmentation

The computational complexity of solving a QP can be reduced by reducing the number of states in the prediction model. Furthermore, if some states are impossible to estimate with any reasonable accuracy from the measurements and actions, there is little benefit in retaining them in the model. Indeed, the closed-loop dynamics in (34) can be improved by removing states before designing the state estimator. Model reduction does however introduce further mismatch between the prediction model and the real world plant. If robust MPC is chosen for the design, this mismatch must be included in the uncertainty model.

There are many types of model reduction but one simple distinction is whether the states in the reduced model have any physical meaning (assuming they do in the original model). Most automatic reduction methods use some kind of projection of the state space onto a lower-dimensional space, in which individual states have no physical meaning. These methods aim to keep the input-output properties as close as possible to the full system. Note that the scaling of the inputs and outputs can have a significant impact here – if some input-output responses lose accuracy before others as the order of the reduced model is lowered, it might be due to this scaling.

After a new state space is defined by projection, e.g. proper orthogonal decomposition, the states with lowest input-to-output energy transfer are removed. To avoid loss of important dynamics in this process, under-damped system modes should be damped by SISO loop shaping as in Section 9. The input-to-output energy of these modes can then be safely discarded in the model reduction process with much lower model-plant mismatch.

It can also be helpful to add new states to the prediction model. If the linear model has a disturbance input of wind speed, the statistics of that disturbance do not fit the requirements of a Kalman filter, having a substantial correlation between subsequent time steps. In Mirzaei et al. [2012] and others, a wind model is created, which takes white noise as an input and produces a plausible spectrum of wind as an output. This model is augmented to the linear turbine model, creating at least one additional state, which the Kalman filter can estimate in real time. While higher order models have been investigated as a means of shaping the white noise into something resembling atmospheric turbulence, often a first order low pass filter is enough to get an MPC algorithm working. For example:

$$H_v(s) = \frac{1}{1 + \tau_v s} \tag{35}$$

Here, $H_v(s)$ is a transfer function in the s-domain, and τ_v is a time constant. Note that the additional state must not be penalised in the cost function, since it is uncontrollable. If the cost function contains the term $C^T Q_y C$ where Q_y is a positive semidefinite matrix penalising measurements, then it should be somehow arranged such that the states representing the persistent disturbance (in the above case only one state) are not somehow penalised, as this could lead to ill-conditioning.

11 Optimisation degrees of freedom

11.1 Motivation

If we assume no time is taken to solve the QP, then $u^*_{0|k}$ is applied as the control action at each time step. The asterisk denotes the value after the optimisation is complete at time step k. If one time step of computational time is required, as mentioned in Section 2, then $u^*_{1|k-1}$ is applied, but the discussion in this section is still valid. The reason the entire sequence $\mathbf{u}_k$ is calculated at each time step, rather than only the variables directly applied to the plant, is that doing so is what allows the optimiser to account for predicted future state constraint violations in the future, a feature that is not provided by anti-windup.

For an uncertain system, the further ahead in the prediction horizon, the more these variables are likely to change before being applied. This could be expressed mathematically as follows:

$$\mathbb{E}(\|u^*_{i+1|k} - u^*_{0|k+i+1}\|) > \mathbb{E}(\|u^*_{i|k} - u^*_{0|k+i}\|) \tag{36}$$

Therefore there is less importance on the accuracy of optimisation variables further ahead in time. It is also reasonable to assume that the higher frequency modes of the closed-loop system have significant energy for shorter time than the lower frequency modes. Given all this, can we reduce the computational complexity of the QP by not having strictly one degree of freedom per control action per time step?

11.2 Input blocking

Input blocking is the technique of forming blocks of optimisation variables so as to reduce the total number of degrees of freedom in the QP. One scheme to consider is to form a triangle, where the blocks grow in size, as follows:

$$\begin{aligned} &c_{0|k} \\ &c_{1|k} = c_{2|k} \\ &c_{3|k} = c_{4|k} = c_{5|k} \\ &\dots \end{aligned} \tag{37}$$

This way, n independent variables provide the $n(n+1)/2$ time steps of control actions (the triangle number). For example, at 50 Hz, 20 variables would only allow 0.4 s of Mode 1 predictions without input blocking, but with it we get 210/50=4.2 s. Note that $u_{1|k} \neq u_{2|k}$ because the LQR gain is still applied independently to each state in the prediction horizon. States cannot be blocked. If slack variables are in use, these can be blocked, but could add conservativeness.

Recursive feasibility is not possible with this form of input blocking, even for a system with no uncertainty. Recall that to guarantee recursive feasibility, a set of optimisation variables must exist that satisfy the constraints, if the constraints were satisfied in the previous time step. When all time steps have independent variables, this can be shown by setting $c_{i|k+1} = c_{i+1|k}$ up to the final step of Mode 1, where $c_{N-1|k+1} = 0$. Attempting this procedure with input blocking is clearly not possible since $c_{1|k+1} = c_{2|k+1}$ would require $c_{2|k} = c_{3|k}$. For further discussion on this subject see Cagienard et al. [2007].

11.3 Exponential basis functions

An alternative to input blocking for reducing the computational complexity of MPC is the use of *exponential basis functions*. The following scheme is recursively feasible and also has the benefit of removing the need for a terminal set.

A basis function in this context is a time series that is defined for the infinite future, a set of which are mutually orthogonal. By weighting these functions, much like in a model reduction,

the time series of control actions can be defined over the infinite horizon with a finite (small) number n_L of independent variables. Exponential basis functions have the crucial property that each time step in each basis function is a linear combination of the preceding values. Such a basis set for example is the *Laguerre* functions.

The engineer chooses how many Laguerre functions to use, n_L. The first time step in all of these functions is denoted ℓ_0, which is a vector with n_L elements. Each subsequent time step can be calculated offline as follows:

$$\ell_{i+1} = A_L \ell_i \tag{38}$$

The values of ℓ_0 and the lower triangular matrix A_L are given in Wang [2004]. The time series created with such a method have the useful property of the rate of change decreasing over time, and all functions tending to zero after some initial oscillations. An example set of Laguerre functions is shown in Figure 2. Choice of n_L allows the engineer to trade off the level of complexity, and thereby optimality, of the functions applied in the MPC algorithm, against the number of optimisation variables, and thereby the computational cost at run time.

The vector $c_{i|k}$ is now a linear sum of ℓ_i for each control action as follows:

$$c_{i|k} = C_k \ell_i \tag{39}$$

Matrix C_k, which has n_L columns, contains all the optimisation variables, and recall that ℓ_i is a constant. Now, due to (38), we know that $C_k A_L$ will be feasible at time step $k+1$, since:

$$c_{i|k+1} = C_k A_L \ell_i = C_k \ell_{i+1} \tag{40}$$

which is $c_{i+1|k}$, which was feasible at time step k.

The basis functions extend to infinity in time, so the cost function does not need two modes. Instead, a similar technique to the construction of the terminal cost $\bar{Q}$ is employed, creating a quadratic cost function in terms of C_k directly. The new cost matrix is calculated by solving a semidefinite program (SDP) offline. Furthermore, the techniques for making such a scheme robust against additive and multiplicative uncertainty can both be adapted and applied.

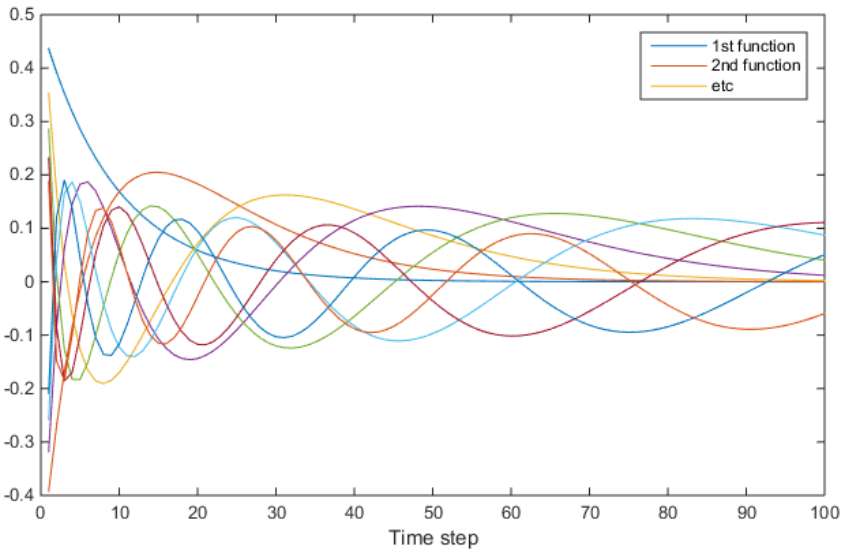

Figure 2: The first nine Laguerre functions with a decay parameter of 0.9.

12 Offset tracking

Cost functions such as (2) assume the control actions, states and measurements are all regulated to zero. If we use the linearisation point $\bar{u}, \bar{x}, \bar{y}$ as in (30) as the offset, we can simply subtract $\bar{y}$ from the measurements as they are taken, and add $\bar{u}$ to the action before it is applied. However, this is not appropriate in the wind turbine control problem since the wind speed will vary away from that operating point.

The simple solution, as in Muske and Badgwell [2002] and many others, is to have a look-up table from the persistent disturbance to the offsets, which then vary online. If, as in our case, the persistent disturbance is not directly known, the estimated state from the Kalman filter can be used. If we use a disturbance model for the wind speed, as in Section 10, one of the estimated states is wind speed. If we extract the estimated wind speed from the estimated state vector with a matrix C_V, we get our time varying offsets as follows:

$$\begin{aligned} \bar{u}_k &= u_{ss}(C_V \hat{x}_k) \\ \bar{x}_k &= x_{ss}(C_V \hat{x}_k) \\ \bar{y}_k &= y_{ss}(C_V \hat{x}_k) \end{aligned} \tag{41}$$

Here, u_{ss}, x_{ss}, y_{ss} are look-up tables for steady state operation at different wind speeds. Bladed provides these values in its exported linear model, if a range of wind speeds are used for linearisation. Alternatively, they can be manually selected for specific operational modes, for example curtailment.

13 Measurement filters

The linear model does not capture dynamics associated with wind speed variation around the rotor, which result in 3P disturbances, i.e. forces with a dominant frequency three times the rotor frequency. Since we do not wish the actuators (pitch and torque) to respond at these frequencies, a measurement filter is required. For 3P, some kind of notch filter is recommended. Potentially a 1P and 6P notch could also be beneficial, as well as high frequency measurement noise attenuation, if required.

Such filters bring the benefit of shielding the MPC algorithm to unmodelled disturbances, but they introduce a new challenge, namely phase shift. In classical loop shaping control design processes, filters are included as part of the controller for this very reason. It's no different in MPC; indeed the steps required are automatic.

Just as in (35), we can design the filter as a simple SISO block and augment to the MIMO plant. Before the linear model is reduced, and at the same time as SISO control feedback is added, the measurements have the required filters connected in series. Now the predictions in the MPC formulation will account for the filtering that will be applied at run time.

14 Worked example

14.1 Model

We now demonstrate the practical steps to take a Bladed model and run a simulation with MPC as the controller. The turbine model used is the demo model that comes with Bladed, which is a 2 MW three bladed upwind turbine with a traditional steel tower, rotor diameter of 80 m, rated generator speed of 1500 RPM and a gearbox ratio of 83. Linearisations are performed at 2 m/s steps from 5 m/s to 25 m/s. The single linearisation used for the prediction model is

the one at 13 m/s but the others are used to find steady state values for the offset tracking. The controller runs at 20 Hz.

Linear models exported from Bladed to Matlab come with the inputs:

- Wind speed
- Pitch angle demand
- Generator torque demand

The outputs (measurements) can be chosen in the Bladed settings; in this work we selected:

- Measured generator speed
- Measured generator torque
- Blade 1 pitch angle
- Blade 1 pitch rate
- Nacelle fore-aft velocity
- Nacelle fore-aft acceleration

14.2 Filters and feedback

Since the pitch rate and fore-aft velocity cannot be directly measured in the real world, a filter is set up as in Section 13, from pitch angle to pitch rate and from acceleration to velocity, as follows:

$$\frac{s}{s/10+1}, \qquad \frac{1}{s+1} \tag{42}$$

The 3P filter applied to the generator speed and nacelle acceleration measurement channels is a notch at ω_3, which is three times the rated rotor speed in radians per second.

$$\frac{\left(\frac{s}{\omega_3}\right)^2 + 2\zeta_n \frac{s}{\omega_3} + 1}{\left(\frac{s}{\omega_3}\right)^2 + 2\zeta_d \frac{s}{\omega_3} + 1} \tag{43}$$

where $\zeta_n = 0.05$ and $\zeta_n = 0.3$. The impact of the notch filter on the measured generator speed is shown in Figure 3.

The next step is to dampen the edgewise and drive train modes with torque-speed feedback. This is a simple bandpass filter from filtered generator speed to torque demand. The loop is closed like this and the feedback control is saved for use at run time. In this example, the torque-speed feedback is:

$$\frac{400s}{s+2.8} \tag{44}$$

The filter in (44) is actually a high-pass, but since the highest frequency mode we want to dampen is around 4 Hz and the Nyquist limit of the controller is 10 Hz, there is no need for an extra pole to bring down high frequency gain. The effect of the SISO feedback on this example is shown in Figure 4, where structural modes between 10 and 30 rad/s are substantially damped.

The wind speed transfer function is augmented onto the model next, then we reduce the model. In this example, the model is reduced from 69 states to 10. Figure 5 shows the Bode plot from wind noise (the input to the wind speed transfer function), pitch angle demand and torque demand to filtered generator speed. It shows the full model, the reduced model and a model where the reduction is less severe, namely to 15 states. Note that the model reduction process

is naive, and clearly shows evidence of favouring modes that have a higher output amplitude. This is not ideal for our purposes, because the input-output gain of the model is dictated by the scaling factors, which were primarily for numerical conditioning. More advanced model reduction methods exist than this, which should be tested in future work.

Since MPC is a time domain control scheme, it is helpful to visualise the impact of the model reduction on the step response. For clarity of presentation, only the pitch demand input is shown, with its response in all six measured signals. Figure 6 shows these responses with both the full model and reduced model.

The Kalman filter is designed next, based on the reduced order model. Of the three inputs (wind noise, pitch demand, torque demand), the wind noise is assumed uncertain with a known variance, while the other two are configured to be known exactly. Of the six measurements, all are assumed uncertain, due to measurement noise.

To test the Kalman filter, the full model is excited by white noise for the wind noise input, and random steps for the pitch and torque demand inputs. The resulting measurements are recorded over a 200 second simulation. These measurements and the known steps for pitch and torque are fed into the Kalman filter and the states of the reduced order model are estimated. From that estimated state history, the estimated wind speed is back-calculated using C_V, and compared to the output of the wind speed transfer function. The results are shown in Figure 7, where a small lag and smoothing is evident. The wind speed estimate is used in the run time operation to calculate the steady state set points, while the state estimates are an input to the MPC optimisation.

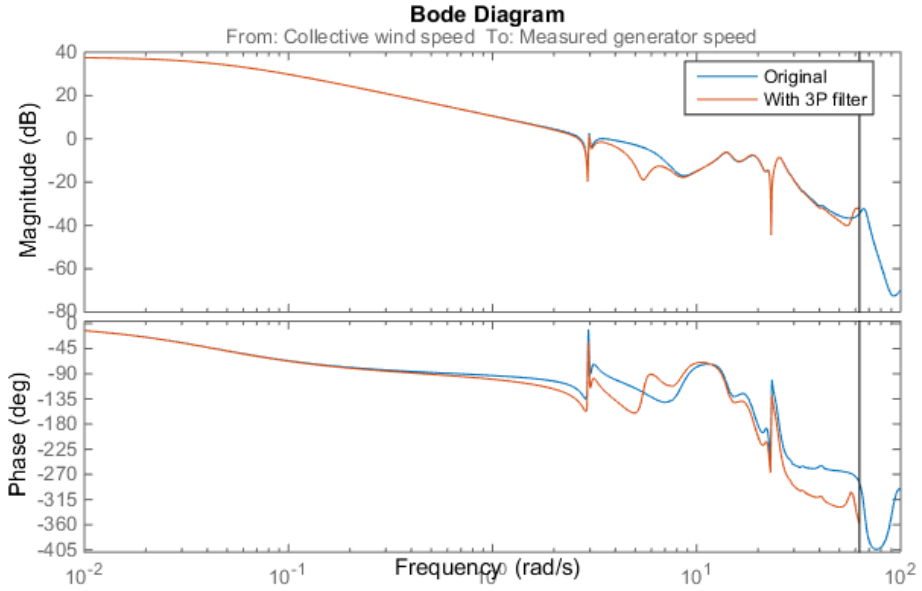

Figure 3: Bode diagram from wind speed to measured generator speed, showing original plant (blue) and with a 3P filter (red).

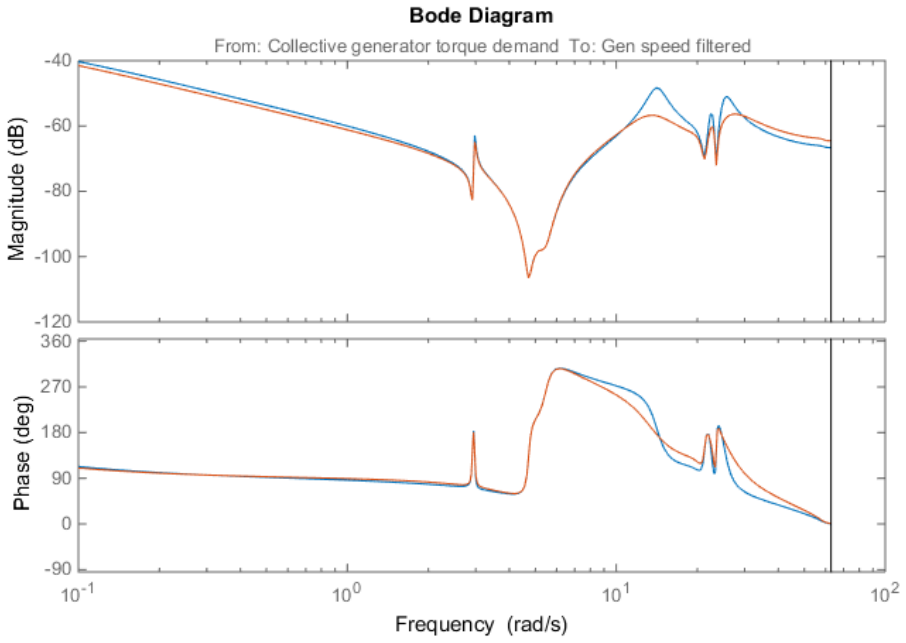

Figure 4: Bode diagram from torque demand to generator speed, showing plant with measurement filters (blue) and with torque-speed feedback applied (red).

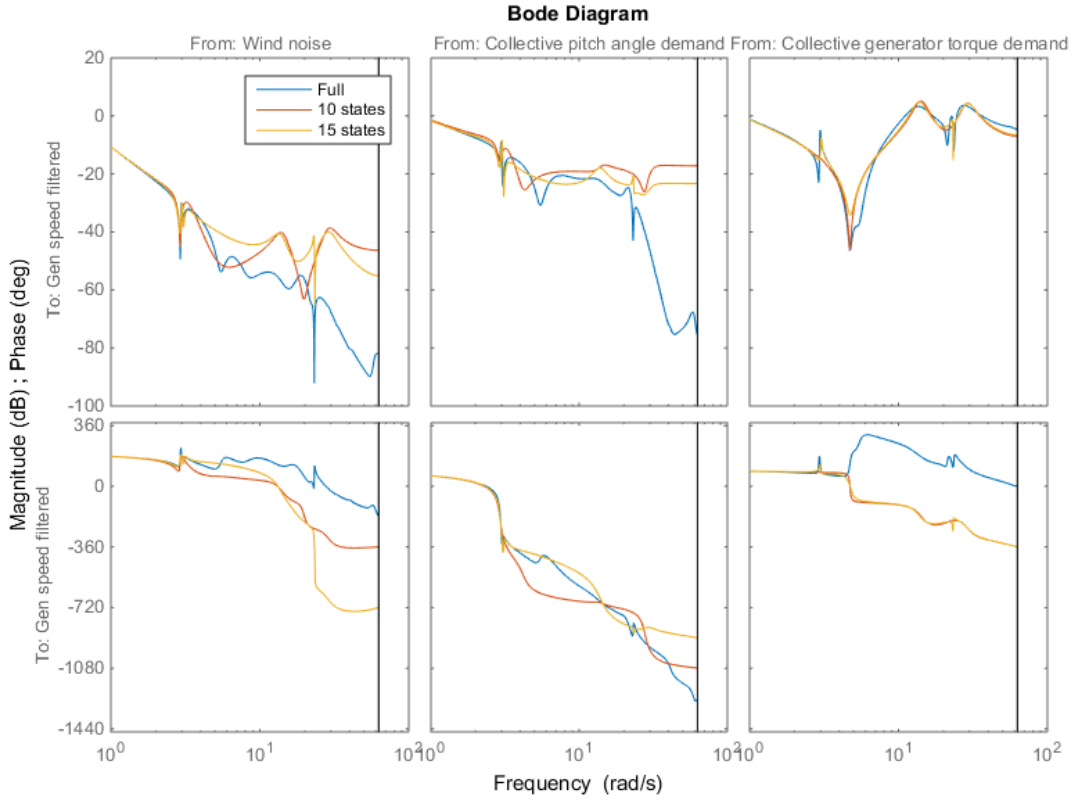

Figure 5: Bode diagram from wind noise, pitch demand and torque demand to filtered generator speed, showing full model (blue), model reduced to 10 states (red) and model reduced to 15 states (yellow).

14.3 Optimisation

The control actions are optimisation variables defined by a set of Laguerre functions. In this example we use the first nine functions, which are shown in Figure 2. The time step is 50 ms, the horizon is 100 time steps and the decay parameter is 0.9. To illustrate the ability of these Laguerre basis functions to approximate useful prediction horizons, a constrained optimisation is performed with the optimisation variables defined as weights for the first nine Laguerre functions. A target time series is defined, and the optimal approximation to that target is found, using only the weights, subject to a linear constraint. The results are shown in Figure 8. This illustrates the capability of exponential basis functions, since the optimisation problem has only nine degrees of freedom, yet is able to respect the constraints over a horizon of 100 time steps.

The constraints used in turbine simulation are maximum and minimum pitch angle, pitch rate and torque. All constraints are applied to the predicted measurements, i.e.

$$\begin{aligned} A_c y_{i|k} &\leq b_c \\ y_{i|k} &= C x_{i|k} + D u_{i|k} \end{aligned} \tag{45}$$

Although this is a different format from (4), there is an equivalence, and this format is easier to understand, since

$$A_c = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}, \tag{46}$$

which clearly shows maximum (elements of 1) and minimum (elements of -1) constraints and the columns correspond to the measurements in the prediction model.

The cost function is a simple quadratic in $y_{i|k}$, where the engineer can tune the weights for each output. The constraints and cost function combined are input into the YALMIP (see Löfberg [2004]) toolbox, with the solver selected as SEDUMI.

14.4 Results

The full linear model was run as a plant in MATLAB with white noise (scaled up by a factor of 4) as the input to the wind speed model and the combination of torque-speed feedback and MPC as the pitch and torque demands. The results are shown in Figure 9. The wind speed varies very dramatically due to the large scale on the wind noise input.

The results illustrate many of the important features of this implementation of MPC. Dashed lines show the steady state offset for generator speed, pitch angle and torque. These are based on a look-up table exported from BLADED, with the independent variable being the wind speed estimated from the Kalman filter, also shown in the figure.

Due to the large turbulence, the simulation contains two periods of below-rated operation and an example at 65 s of a gust, which the pitch responds to rapidly but without exciting the tower. Since this simulation is based on a linear model, it does not have realism such as tower shadow, wind shear, unsteady aerodynamics, etc. For this reason, we coupled the MATLAB code that runs the optimisation to a full BLADED simulation and the results are very similar.

15 Other considerations in implementation

The primary consideration when moving from simulation to real world is clearly safety. There must be a backup control law in the event of the QP being infeasible, or any other solver-related

run time concern. When switching from MPC to the backup law, the controller must be capable of doing so in a way that does not cause a structural or power train shock, a feature sometimes called *bumpless* transfer.

Supervisory control (start up and shut down) must be addressed carefully, since the present MPC algorithm only handles normal production. It is common for wind turbine supervisory control to modify the set points of the closed-loop control law in order to move the turbine between operating states. If this approach is used with MPC, the solver must be able to handle operation far from the linearisation point.

Computational demand, i.e. the time taken to reach a solution to the QP each time step, is important too, since failing to solve the problem in the allocated time is as serious as not being able to solve it. Some QP solvers start each solution process inside the constraints, with the intention that if the time runs out, the iteration can stop and the solution satisfies the constraints if not being the absolute optimal point. This is not a free lunch, since the closed-loop dynamics will have been tested in simulation with no such sub-optimality. Merely satisfying the constraints does not guarantee the dynamic behaviour will be acceptable.

Finally, there is the offline design process to consider. Will future control engineers understand the code? Does the solver used in simulation match the solver used on the turbine? When there is an infeasibility in simulation, is it possible to find the root cause? Some solvers give quite unhelpful error messages such as 'Infeasible' or 'Nan'. It is much harder to debug a 20-dimensional (if using Laguerre) or even 200-dimensional (if using one optimisation variable per time step) QP than a SISO frequency domain problem. Perfectly designed tools and training exist for SISO loop shaping, while it will be many years before MPC catches up.

An important question is whether the MPC can do what the traditional methods cannot do. So far the wind turbine control problem can typically be treated as a few separate decoupled problems, one for each loop. Actuator constraints and safe operational limits can be handled by simple anti-windup or bespoke logic. As rotors grow larger and as floating platforms grow in popularity, addressing the coupling between different turbine modes becomes more urgent. By then, there will be inevitably a need for using a advanced and well-developed MIMO method. MPC could then be the prefect candidate for this kind of problem.

16 Conclusions

In TotalControl deliverable D3.4, some benefits and challenges of using MPC for wind turbine control were identified. In particular, computational complexity and reliability of the solution to the quadratic program were raised as priorities for further development. This deliverable, D3.8, has targeted those priorities and has brought MPC closer to readiness for use in the real world.

This deliverable forms an essential bridge between theory and practice. It tackles some key contributing factors to the computational complexity challenge and gives a worked example that shows promising results. However, substantial work remains to implement the presented algorithm on a wind turbine control system such as the Levenmouth Demonstration Turbine which forms part of the TotalControl project. Either the MPC algorithm must be re-coded and optimised sufficiently to allow it to run on the turbine's PLC, substantially reducing its processor and memory requirements, or it must be run on a separate computer, with a communication link to the PLC. Either way, the practicalities are beyond the scope and budget available in TotalControl.

Furthermore, a new agreement on tuning objectives for MPC will be required, and a bumpless backup control law would need to be developed for any time steps where the QP cannot be solved. This could be a fruitful direction for future work but is also beyond the TotalControl project scope.

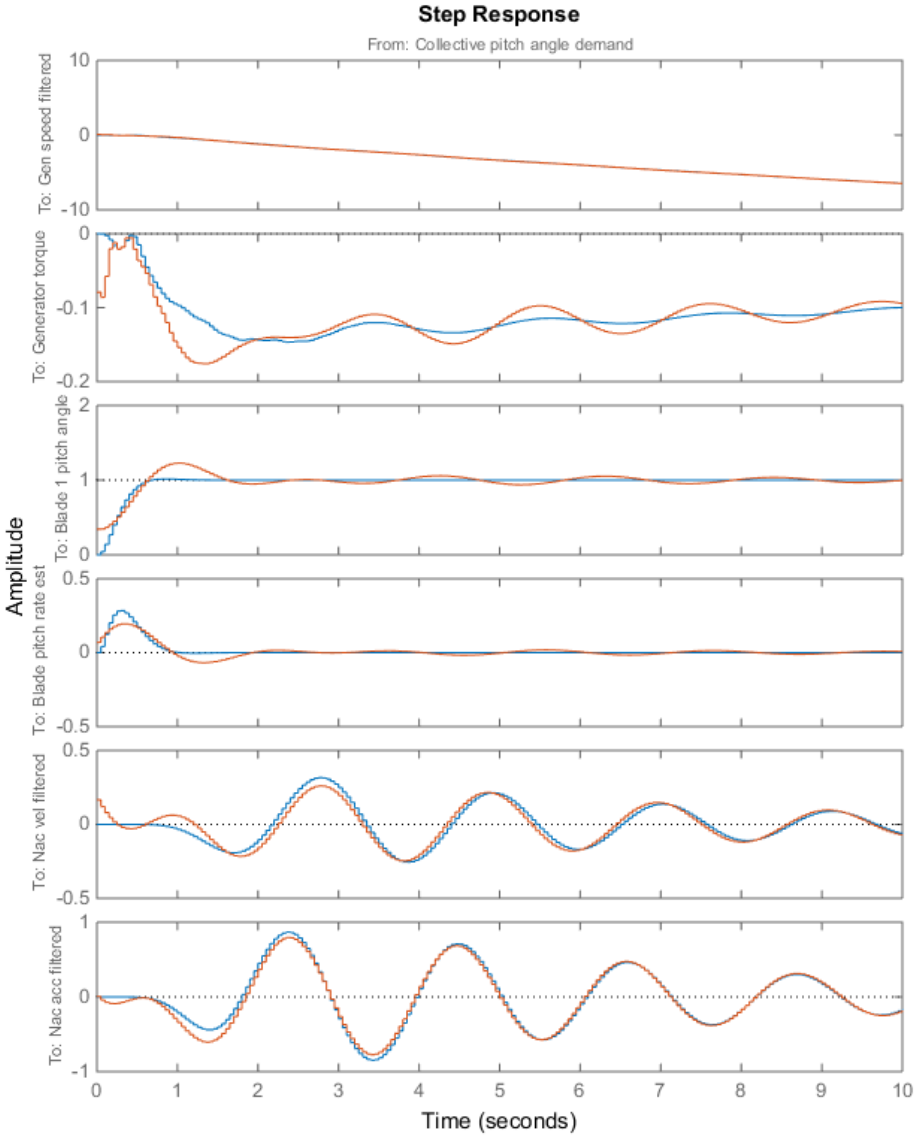

Figure 6: Step response of pitch demand to all filtered measurements for full model (blue) and model reduced to 10 states (red). Note the full model has had inputs and outputs scaled so that important input-output pairs have approximately unity DC gain. This is for numerical conditioning.

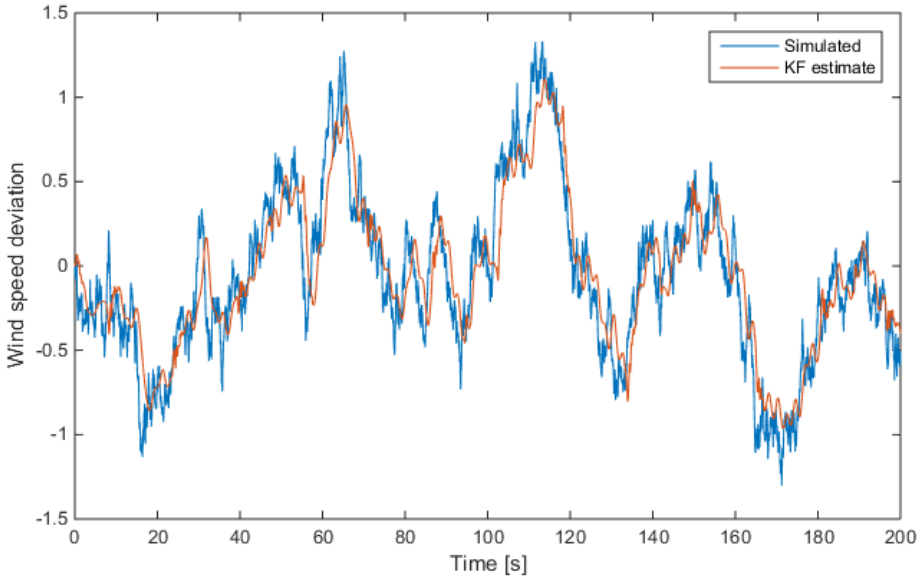

Figure 7: Simulated wind speed time series (blue) with wind speed estimate from Kalman filter (red). All signals have been scaled so have no units.

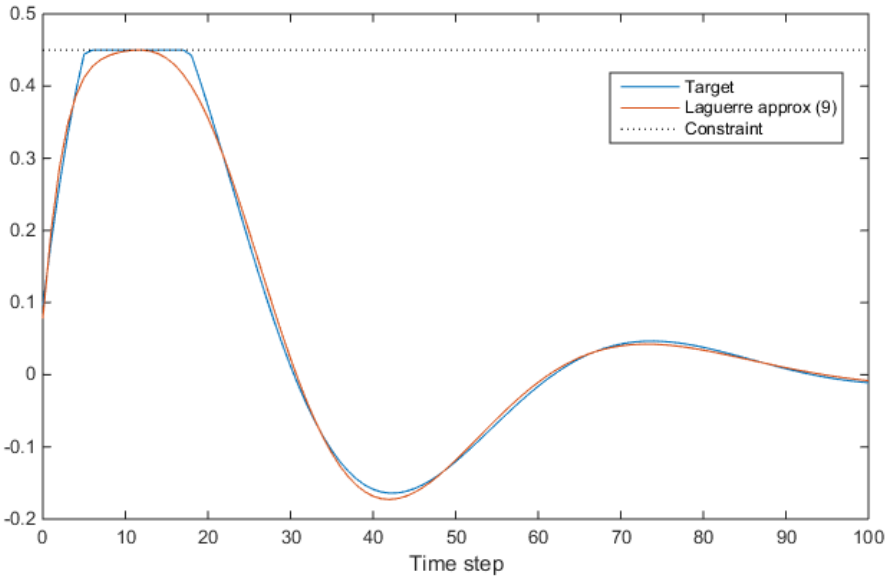

Figure 8: A target time series (blue) with the closest possible (in the least squares sense) approximation (red) using only nine optimisation variables as weights for Laguerre functions, subject to a constraint (dotted).

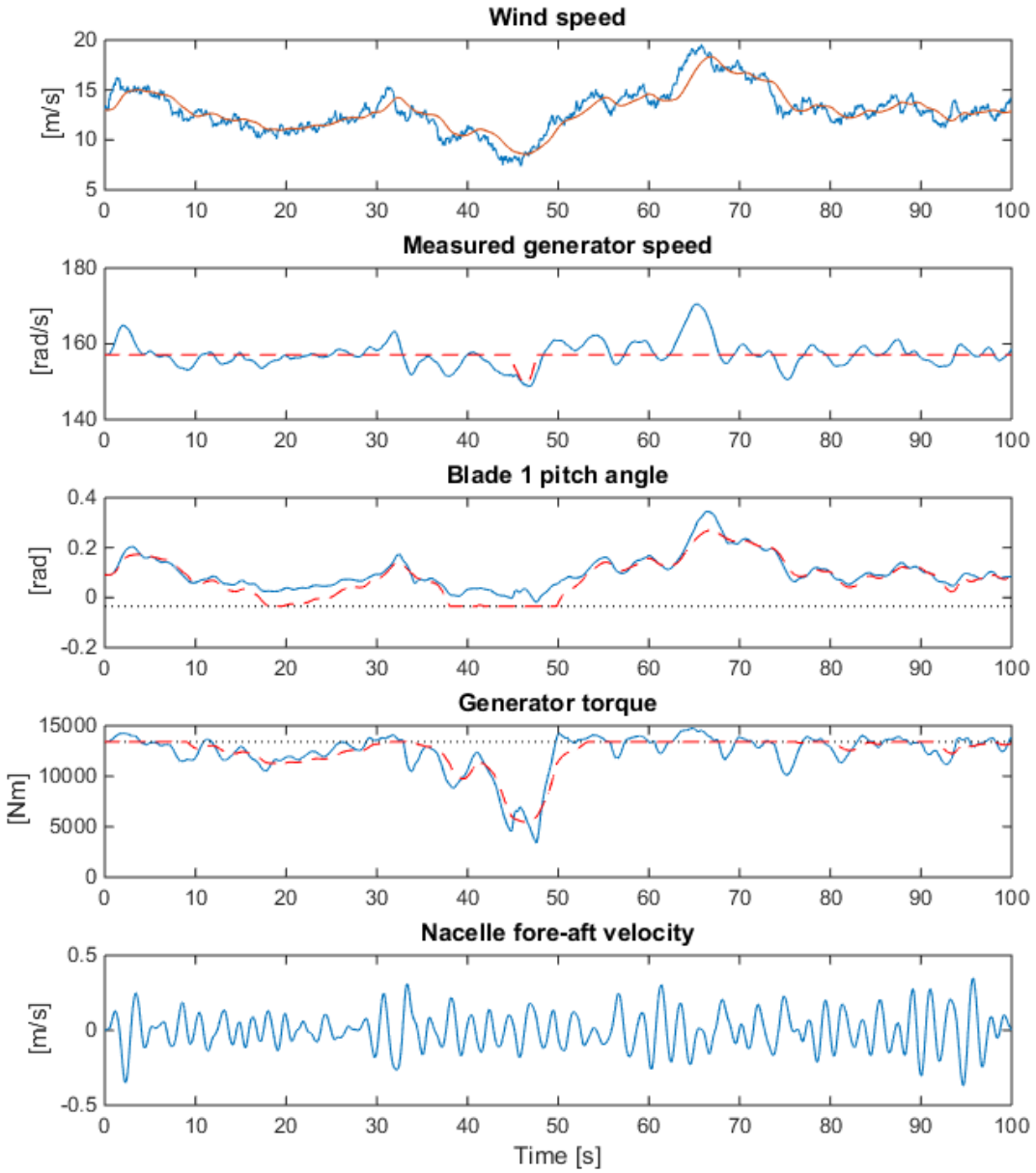

Figure 9: An example simulation with true wind speed and raw measured signals (blue), estimated wind speed (red solid) and steady state offsets based on that (red dashed). Pitch and torque constraints are shown as black dotted lines.

References

G. Bitsoris. On the positive invariance of polyhedral sets for discrete-time systems. *Systems & Control Letters*, 11(3):243–248, Sept. 1988. ISSN 0167-6911.

R. Cagienard, P. Grieder, E. C. Kerrigan, and M. Morari. Move blocking strategies in receding horizon control. *Journal of Process Control*, 17(6):563–570, 2007.

J. C. Doyle. Guaranteed margins for LQG regulators. *IEEE Transactions on automatic Control*, 23(4):756–757, 1978.

M. A. Evans, M. Cannon, and B. Kouvaritakis. Robust MPC tower damping for variable speed wind turbines. *IEEE Transactions on Control Systems Technology*, 23(1):290–296, 2015. doi: 10.1109/TCST.2014.2310513.

E. Gilbert and K. Tan. Linear systems with state and control constraints: the theory and application of maximal output admissible sets. *IEEE Transactions on Automatic Control*, 36 (9):1008–1020, 1991.

L. Henriksen, M. Hansen, and N. Poulsen. Wind turbine control with constraint handling: a model predictive control approach. *IET Control Theory & Applications*, 6(11):1722, 2012. ISSN 17518644.

R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35, 1960.

E. C. Kerrigan and J. M. Maciejowski. Soft constraints and exact penalty functions in model predictive control. In *Proc. UKACC International Conference (Control)*, 2000.

A. Koerber and R. King. Combined feedback–feedforward control of wind turbines using state-constrained model predictive control. *IEEE Transactions on Control Systems Technology*, 21 (4):1117–1128, 2013.

B. Kouvaritakis and M. Cannon. *Model Predictive Control: Classical, Robust and Stochastic*. Advanced Textbooks in Control and Signal Processing. Springer International Publishing, 2016. ISBN 978-3-319-24851-6.

B. Kouvaritakis, M. Cannon, S. V. Raković, and Q. Cheng. Explicit use of probabilistic distributions in linear predictive control. *Automatica*, 46(10):1719–1724, 2010.

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

W. H. Lio, B. L. Jones, and J. A. Rossiter. Preview predictive control layer design based upon known wind turbine blade-pitch controllers. *Wind Energy*, 20(7):1207–1226, 2017. ISSN 10954244. doi: 10.1002/we.2090.

J. Löfberg. YALMIP : A toolbox for modeling and optimization in MATLAB. In *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

M. Mirzaei, N. Poulsen, and H. Niemann. Robust model predictive control of a wind turbine. In *American Control Conference*, pages 4393–4398, 2012.

K. R. Muske and T. A. Badgwell. Disturbance modeling for offset-free linear model predictive control. *Journal of Process Control*, 12(5):617–632, 2002.

J. B. Rawlings and K. R. Muske. The Stability of Constrained Receding Horizon Control. *IEEE Transactions on Automatic Control*, 38(10):1512–1516, 1993.

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

P. O. Scokaert and J. B. Rawlings. Feasibility issues in linear model predictive control. *AIChE Journal*, 45(8):1649–1659, 1999.

M. Soliman, O. Malik, and D. Westwick. Multiple model predictive control for wind turbines with doubly fed induction generators. *IEEE Transactions on Sustainable Energy*, 2(3):215–225, 2011. ISSN 1949-3029.

L. Wang. Discrete model predictive controller design using Laguerre functions. *Journal of Process Control*, 14(2):131–142, 2004.

M. Zeilinger, C. Jones, and M. Morari. Robust stability properties of soft constrained MPC. In *49th IEEE Conference on Decision and Control*, pages 5276 – 5282, 01 2011. doi: 10.1109/CDC.2010.5717488.