A brief introduction to Model Predictive Control

Morten Hovd
Engineering Cybernetics Department, NTNU

2 March 2004

1 Introduction

Model-based predictive control (MPC) has become the most popular advanced control technology in the chemical processing industries. There are many variants of MPC controllers, both in academia and in industry, but they all share the common trait that an explicitly formulated process model is used to predict and optimize future process behaviour. Most MPC controllers are able to account for constraints both in manipulated variables and states/controlled variables through the formulation of the optimization problem.

When formulating the optimization problem in MPC, it is important to ensure that it can be solved in the short time available (i.e., the sampling interval is an upper bound on the acceptable time for performing the calculations). For that reason, the optimization problem is typically cast into one of two standard forms:

- Linear programming (LP) formulation. In an LP formulation, both the objective function and the constraints are linear.
- Quadratic programming (QP) formulation. In a QP formulation, the objective function is quadratic, whereas the constraints have to be linear. In addition, to ensure that there exists a unique optimal solution that can be found quickly with effective optimization solvers, the QP problem must be convex[1].

LP formulation may sometimes be advantageous for very large optimization problems. However, a QP formulation generally leads to smoother control action and more intuitive effects of changes in the tuning parameters. The connection to 'traditional advanced control', i.e., linear quadratic (LQ) optimal control, is also much closer for a QP formulation than for an LP formulation. For these reasons, we will focus on a QP formulation in the following, and describe in some detail how a QP optimization problem in MPC may be formulated.

2 Formulation of a QP problem for MPC

A standard QP problem takes the form

$$\min_{v} \quad 0.5 v^T \widetilde{H} v + c^T v \tag{1}$$

subject to the constraints

$$Lv \leq b \tag{2}$$

[1] I.e. the Hessian matrix of the QP problem must be *positive definite on the subspace of the active constraints.*

Here v is the vector of free variables in the optimization, whereas $\widetilde{H}$ is the *Hessian matrix*, that was mentioned above, and which has to be positive definite. The vector c describes the linear part of the objective function, whereas the matrix L and the vector b describe the linear constraints. Some QP solvers allow the user to specify separate upper and lower bounds for v, whereas other solvers require such constraints to be included in L and b. For completeness, we will assume that these constraints have to be included in L and b.

The formulation of the MPC problem starts from a linear, *discrete-time* state-space model of the type

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \quad (3) \\ y_k &= Cx_k \quad (4) \end{aligned}$$

where the subscripts refer to the sampling instants. That is, subscript $k+1$ refers to the sample instant one sample interval after sample k. Note that for discrete time models used in control, there is normally no direct feed-through term, the measurement y_k does not depend on the input at time k, but it does depend on the input at time $k-1$ through the state x_k. The reason for the absence of direct feed-through is that normally the output is measured at time k before the new input at time k is computed and implemented.

In the same way as is common in control literature, the state x, input u and measurement y above should be interpreted as *deviation variables.* This means that they represent the deviations from some consistent set of of variables $\{x_L, u_L, y_L\}$ around which the model is obtained[2]. For a stable process, the set $\{x_L, u_L, y_L\}$ will typically represent a steady state - often the steady state we want to keep the process at. To illustrate, if in y_L represents a temperature of $330K$, a physical measurement of $331K$ corresponds to a devation variable $y = 1K$.

A typical optimization problem in MPC might take the form

$$\begin{aligned} \min_u \quad f(x,u) &= \sum_{i=0}^{n-1} \{(x_i - x_{ref,i})^T Q (x_i - x_{ref,i}) \quad (5) \\ &\quad + (u_i - u_{ref,i})^T P (u_i - u_{ref,i})^T \} \\ &\quad + (x_n - x_{ref,n})^T S (x_n - x_{ref,n}) \end{aligned}$$

[2]We do not here specify *how* the model is obtained, but typically it is either the result of identification experiments performed around the values $\{x_L, u_L, y_L\}$ or the result of linearizing and discretizing a non-linear, physical model around the values $\{x_L, u_L, y_L\}$.

subject to constraints

$$\begin{aligned} x_0 &= \text{given} \\ U_L &\leq u_i \leq U_U \quad \text{for } 0 \leq i \leq n-1 \\ Y_L &\leq Hx_i \leq Y_U \quad \text{for } 1 \leq i \leq n+j \end{aligned} \tag{6}$$

In the objective function Eq. (5) above, we penalize the deviation of the states x_i from some desired reference trajectory $x_{ref,i}$ and the deviation of the inputs u_i from some desired trajectory $u_{ref,i}$. These reference trajectories are assumed to be given to the MPC controller by some outside source. They may be constant or may also vary with time (subscript i), but they need to be consistent with the plant model, i.e.

$$x_{ref,i+1} = Ax_{ref,i} + Bu_{ref,i}.$$

The constraints on achievable inputs or acceptable states are usually not dependent on the reference trajectories, and therefore these reference trajectories do not appear in the constraint equations (6). Usually, the state constraints represent constraints on process measurements (giving $H = C$), but constraints on other combinations of states are also possible (including constraints on combinations of inputs and states).

In the following, this formulation of the optimization problem will be recast into the standard QP formulation in Eqs.(1) and (2), but first a number of remarks and explanations to the optimization problem formulation in Eqs.(5) to (6) are needed.

- In addition to the above constraints, it is naturally assumed that the process follows the model in Eqs. (3) and (4).
- Note that the state constraints above are imposed on a horizon *longer* than the control horizon n. The parameter j should be chosen such that if the constraints are feasible over the horizon $n+j$ they will remain feasible over an infinite horizon. This issue will be discussed further below.
- The matrices Q, P, and S are all assumed to be symmetric. P and S are assumed to be positive definite, whereas Q may be positive semi-definite.
- In many applications it may be more natural to put a weight (or cost) on the actual measurements rather than the states. This can easily be done by choosing $Q = C^T \widetilde{Q} C$, where $\widetilde{Q}$ is the weight on the measurements.
- One may also put constraints on the rate of change of the inputs, giving additional constraints on the form $\Delta U_L \leq u_i - u_{i-1} \leq \Delta U_U$.

- For the output constraints in Eq. (6) to be well defined, we must specify how the inputs u_i should behave on the interval $n \leq i \leq n+j-1$. Typical choices for this time interval are either that $u_i = u_{ref,i}$ or that $(u_i - u_{ref,i}) = K(x_i - x_{ref,i})$. The latter choice assumes that a (stabilizing) state feedback controller is used in this time interval. Note that this controller will never be used in practice (since the MPC calculations are re-computed at each sample instant), but it is needed to make the constraints well defined.

- If one assumes that $(u_i - u_{ref,i}) = K(x_i - x_{ref,i})$ for $n \leq i \leq n+j-1$, one should also include the input constraints in the problem formulation for the time interval $n \leq i \leq n+j-1$. These input constraints then effectively become state constraints for this time interval.

- Some MPC formulations use an objective function of the form $f(x,u) = \sum_{i=0}^{n_p}(x_i - x_{ref,i})^T Q(x_i - x_{ref,i}) + \sum_{i=0}^{n_u}(u_i - u_{ref,i})^T P(u_i - u_{ref,i})$, where $n_p > n_u$, and typically assume that $u_i = u_{ref,i}$ for $n_u < i < n_p$. Note that this corresponds to a particular choice for 'terminal state weight' S, since x_i for $n_u + 1 < i \leq n_p$ will then be given by x_{n_u+1} (and the process model).

- It is common to introduce integral action in MPC controllers by using the input *changes* at time i as free variables in the optimization, rather than the input itself. This follows, since the actual inputs are obtained by integrating the changes in the input. This can be done within the same framework and model structure as above, using the model

$$\begin{aligned} \widetilde{x}_{k+1} &= \begin{bmatrix} x_{k+1} \\ u_k \end{bmatrix} = \widetilde{A}\widetilde{x}_k + \widetilde{B}\Delta u_k \\ y_k &= \widetilde{C}\widetilde{x}_k \end{aligned}$$

where $\Delta u_k = u_k - u_{k-1}$, and

$$\widetilde{A} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix}, \quad \widetilde{B} = \begin{bmatrix} B \\ I \end{bmatrix}, \quad \widetilde{C} = \begin{bmatrix} C & 0 \end{bmatrix}$$

- To have a stable closed-loop system, it is necessary to have at least as many feedback paths as integrators, i.e., one needs at least as many (independent) measurements as inputs. When the number of inputs exceeds the number of measurements, it is common to define 'ideal resting values' for some inputs. This essentially involves putting some inputs in the measurement vector, and defining setpoints for these.

In the following, we will recast the MPC optimization problem as a standard QP problem. We will assume that $u_i - u_{ref,n} = K(x_i - x_{ref,n})$ for $n \leq i \leq n+j-1$.

To start off, we stack the state references $x_{ref,i}$, input references $u_{ref,i}$, input deviations $v_i = u_i - u_{ref,i}$ and state deviations $\chi_i = x_i - x_{ref,i}$ in long (column) vectors x_{ref}, u_{ref}, v and χ_{dev} :

$$u_{ref} = \begin{bmatrix} u_{ref,0} \\ u_{ref,1} \\ \vdots \\ u_{ref,n-2} \\ u_{ref,n-1} \end{bmatrix}; \quad x_{ref} = \begin{bmatrix} x_{ref,1} \\ x_{ref,2} \\ \vdots \\ x_{ref,n-1} \\ x_{ref,n} \end{bmatrix};$$

$$v = \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix}; \quad \chi_{dev} = \begin{bmatrix} \chi_1 \\ \chi_2 \\ \vdots \\ \chi_{n-1} \\ \chi_n \end{bmatrix}$$

We will use the *superposition principle,* which states that the total effect of several inputs can be obtained simply by summing the effects of the individual inputs. The superposition principle is always valid for linear systems, but typically does not hold for non-linear systems. This allows us to first calculate the deviation from the desired state trajectory that would result, given the initial state x_0 and assuming that the nominal reference input u_{ref} is followed. This results in the trajectory of state deviations χ_0.

Repeated use of the model equation Eq. (3) then gives

$$\begin{aligned} \chi_0 &= \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^{n-1} \\ A^n \end{bmatrix} x_0 + \begin{bmatrix} B & 0 & \cdots & 0 & 0 \\ AB & B & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & 0 & 0 \\ A^{n-2}B & A^{n-3}B & \cdots & B & 0 \\ A^{n-1}B & A^{n-2}B & \cdots & AB & B \end{bmatrix} u_{ref} - x_{ref} \\ &= \widehat{A}x_0 + \widehat{B}u_{ref} - x_{ref} \end{aligned}$$

Thus, we have obtained a deviation from the desired state trajectory x_{ref}, which should be counteracted using deviations $v_i = u_i - u_{ref,i}$ from the nominal input trajectory. Note that χ_0 is independent from the deviation from the deviation from the input reference trajectory, i.e., independent of v, and may therefore be calculated prior to solving the MPC optimization. Similarly, the effect of the deviations from the nominal input trajectory on the states is given by $\chi_v = \widehat{B}v$ (which clearly does depend on the result of the MPC optimization), and we get

$$\chi_{dev} = \chi_0 + \chi_v \tag{7}$$

from the superposition principle.

Introducing the matrices

$$\widehat{Q} = \begin{bmatrix} Q & 0 & \cdots & 0 & 0 \\ 0 & Q & \ddots & \vdots & \vdots \\ 0 & 0 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & Q & 0 \\ 0 & 0 & \cdots & 0 & S \end{bmatrix}, \quad \widehat{P} = \begin{bmatrix} P & 0 & \cdots & 0 & 0 \\ 0 & P & \ddots & \vdots & \vdots \\ 0 & 0 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & P & 0 \\ 0 & 0 & \cdots & 0 & P \end{bmatrix} \tag{8}$$

the objective function can be written as

$$\begin{aligned} f(x,u) = f(\chi_{dev}, \chi_v, v) &= (x_0 - x_{ref,0})^T Q (x_0 - x_{ref,0}) + \\ &\quad (\chi_0 + \chi_v)^T \widehat{Q} (\chi_0 + \chi_v) + v^T \widehat{P} v \\ &= (x_0 - x_{ref,0})^T Q (x_0 - x_{ref,0}) + \chi_0^T \widehat{Q} \chi_0 + \\ &\quad 2\chi_0^T \widehat{Q} \chi_v + \chi_v^T \widehat{Q} \chi_v + v^T \widehat{P} v \end{aligned}$$

which should be minimized using the vector v as free variables.

Now, the terms $(x_0 - x_{ref,0})^T Q (x_o + x_{ref,0}) + \chi_0^T \widehat{Q} \chi_0$ will not be affected by the optimization, and may therefore be removed from the objective function. This is because we are primarily interested in finding the inputs that minimize the objective function, and not in the optimal value of the objective function. Thus, the objective function is in the form of a standard QP problem as defined in Eq. (1) if we define

$$\begin{aligned} \widetilde{H} &= \widehat{B}^T \widehat{Q} \widehat{B} + \widehat{P} \\ c^T &= \chi_0^T \widehat{Q} \widehat{B} \end{aligned} \tag{9}$$

It now remains to express the constraints in the MPC problem in the form of a standard QP problem. The lower and upper constraints on the manipulated variable from $0 \leq i \leq n-1$ simply become

$$Iv \geq \begin{bmatrix} U_L \\ \vdots \\ U_L \end{bmatrix} - u_{ref} \tag{10}$$

$$-Iv \geq -\begin{bmatrix} U_U \\ \vdots \\ U_U \end{bmatrix} + u_{ref} \tag{11}$$

Similarly, the constraints on the measurements/states for $1 \leq i \leq n$ become

$$
\begin{aligned}
&\begin{bmatrix} H & 0 & \cdots & \cdots & 0 \\ 0 & H & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & H & 0 \\ 0 & \cdots & \cdots & 0 & H \end{bmatrix} \chi_v \quad \geq \\
\begin{bmatrix} Y_L \\ \vdots \\ \vdots \\ \vdots \\ Y_L \end{bmatrix} - &\begin{bmatrix} H & 0 & \cdots & \cdots & 0 \\ 0 & H & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & H & 0 \\ 0 & \cdots & \cdots & 0 & H \end{bmatrix} \qquad (\chi_0 + x_{ref}) \\
- &\begin{bmatrix} H & 0 & \cdots & \cdots & 0 \\ 0 & H & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & H & 0 \\ 0 & \cdots & \cdots & 0 & H \end{bmatrix} \chi_v \quad \geq \\
- \begin{bmatrix} Y_U \\ \vdots \\ \vdots \\ \vdots \\ Y_U \end{bmatrix} + &\begin{bmatrix} H & 0 & \cdots & \cdots & 0 \\ 0 & H & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & H & 0 \\ 0 & \cdots & \cdots & 0 & H \end{bmatrix} \qquad (\chi_0 + x_{ref}) \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Updownarrow
\end{aligned}
$$

$$\begin{bmatrix} H & 0 & \cdots & \cdots & 0 \\ 0 & H & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & H & 0 \\ 0 & \cdots & \cdots & 0 & H \end{bmatrix} \widehat{B}v \geq \tag{12}$$

$$\begin{bmatrix} Y_L \\ \vdots \\ \vdots \\ \vdots \\ Y_L \end{bmatrix} - \begin{bmatrix} H & 0 & \cdots & \cdots & 0 \\ 0 & H & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & H & 0 \\ 0 & \cdots & \cdots & 0 & H \end{bmatrix} \qquad (\chi_0 + x_{ref})$$

$$- \begin{bmatrix} H & 0 & \cdots & \cdots & 0 \\ 0 & H & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & H & 0 \\ 0 & \cdots & \cdots & 0 & H \end{bmatrix} \widehat{B}v \geq \tag{13}$$

$$- \begin{bmatrix} Y_U \\ \vdots \\ \vdots \\ \vdots \\ Y_U \end{bmatrix} + \begin{bmatrix} H & 0 & \cdots & \cdots & 0 \\ 0 & H & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & H & 0 \\ 0 & \cdots & \cdots & 0 & H \end{bmatrix} \qquad (\chi_0 + x_{ref})$$

We found above that

$$\begin{aligned} x_n &= A^n x_0 + \left[\begin{array}{cccc} A^{n-1}B & A^{n-2}B & \cdots & AB \quad B \end{array} \right] (u_{ref} + v) \\ &= \chi_{0,n} + x_{ref,n} + \left[\begin{array}{cccc} A^{n-1}B & A^{n-2}B & \cdots & AB \quad B \end{array} \right] v \end{aligned}$$

The process model and the assumed control action $(u_i - u_{ref,n}) = K(x_i - x_{ref,n})$ for the time period $n \leq i \leq n+j-1$ gives, after trivial, but tedious manipulation

$$\begin{aligned} x_i &= (A+BK)^{i-n} x_n + \left\{ \sum_{j=0}^{i-n-1} (A+BK)^i \right\} B(u_{ref,n} - K x_{ref,n}) \\ u_i - u_{ref,n} &= K(x_i - x_{ref,n}) = K(A+BK)^{i-n} x_n \\ &\quad + K \left[\left\{ \sum_{k=0}^{i-n-1} (A+BK)^i \right\} B(u_{ref,n} - K x_{ref,n}) - x_{ref,n} \right] \end{aligned}$$

which combined with the above expression for x_n results in

$$\begin{bmatrix} K \\ K(A+BK)^1 \\ \vdots \\ K(A+BK)^{j-1} \\ K(A+BK)^j \end{bmatrix} \begin{bmatrix} A^{n-1}B & A^{n-2}B & \cdots & AB & B \end{bmatrix} v \geq \tag{14}$$

$$\begin{bmatrix} U_L \\ U_L \\ \vdots \\ U_L \\ U_L \end{bmatrix} - \begin{bmatrix} K \\ K(A+BK)^1 \\ \vdots \\ K(A+BK)^{j-1} \\ K(A+BK)^j \end{bmatrix} (\chi_{0,n} + x_{ref,n})$$

$$- \left\{ \begin{bmatrix} I \\ I \\ \vdots \\ I \\ I \end{bmatrix} + \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ I & 0 & \ddots & \vdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ I & \vdots & \ddots & 0 & 0 \\ I & I & \cdots & I & 0 \end{bmatrix} \begin{bmatrix} K \\ K(A+BK)^1 \\ \vdots \\ K(A+BK)^{j-1} \\ K(A+BK)^j \end{bmatrix} B \right\}$$

$$\times \begin{bmatrix} I & -K \end{bmatrix} \begin{bmatrix} u_{ref,n} \\ x_{ref,n} \end{bmatrix}$$

$$
-\begin{bmatrix} K \\ K(A+BK)^1 \\ \vdots \\ K(A+BK)^{j-1} \\ K(A+BK)^j \end{bmatrix} \begin{bmatrix} A^{n-1}B & A^{n-2}B & \cdots & AB & B \end{bmatrix} v \geq \tag{15}
$$

$$
-\begin{bmatrix} U_U \\ U_U \\ \vdots \\ U_U \\ U_U \end{bmatrix} + \begin{bmatrix} K \\ K(A+BK)^1 \\ \vdots \\ K(A+BK)^{j-1} \\ K(A+BK)^j \end{bmatrix} (\chi_{0,n} + \quad x_{ref,n})
$$

$$
+\left\{ \begin{bmatrix} I \\ I \\ \vdots \\ I \\ I \end{bmatrix} + \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ I & 0 & \ddots & \vdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ I & \vdots & \ddots & 0 & 0 \\ I & I & \cdots & I & 0 \end{bmatrix} \begin{bmatrix} K \\ K(A+BK)^1 \\ \vdots \\ K(A+BK)^{j-1} \\ K(A+BK)^j \end{bmatrix} B \right\} \times
$$

$$
\begin{bmatrix} I & -K \end{bmatrix} \begin{bmatrix} u_{ref,n} \\ x_{ref,n} \end{bmatrix}
$$

$$
\begin{bmatrix} H(A+BK) \\ H(A+BK)^2 \\ \vdots \\ H(A+BK)^{j-1} \\ H(A+BK)^j \end{bmatrix} \begin{bmatrix} A^{n-1}B & A^{n-2}B & \cdots & AB & B \end{bmatrix} v \geq \tag{16}
$$

$$
\begin{bmatrix} Y_L \\ Y_L \\ \vdots \\ Y_L \\ Y_L \end{bmatrix} - \begin{bmatrix} H(A+BK) \\ H(A+BK)^2 \\ \vdots \\ H(A+BK)^{j-1} \\ H(A+BK)^j \end{bmatrix} (\chi_{0,n} + x_{ref,n})
$$

$$-\begin{bmatrix} H(A+BK) \\ H(A+BK)^2 \\ \vdots \\ H(A+BK)^{j-1} \\ H(A+BK)^j \end{bmatrix} \begin{bmatrix} A^{n-1}B & A^{n-2}B & \cdots & AB & B \end{bmatrix} v \geq \tag{17}$$

$$-\begin{bmatrix} Y_U \\ Y_U \\ \vdots \\ Y_U \\ Y_U \end{bmatrix} + \begin{bmatrix} H(A+BK) \\ H(A+BK)^2 \\ \vdots \\ H(A+BK)^{j-1} \\ H(A+BK)^j \end{bmatrix} (\chi_{0,n} + x_{ref,n})$$

The overall set of constraints for the MPC problem is now obtained by simply stacking equations (10,11,12,13,14,15,16,17). All these constraint equations have a left hand side consisting of a matrix multiplied with the vector of free variables in the optimization, and a right hand side which is vector-valued (and which can be evaluated prior to the optimization), and are hence a set of linear constraints as in Eq. 2 . Note that the introduction of non-zero (and possibly time-varying) reference trajectories significantly complicate the expressions, in particular for the constraints in the period $n \leq i \leq n+j$.

There is a slight difference between the state constraint equations and the input constraint equations, in that Eqs. (10) and (11) include an input constraint at time zero (the present time), whereas the state constraint equations (Eqs. (12) and (13)) do not. This is because the state constraints cannot be enforced if they are violated at time zero, since the present state is unaffected by present and future inputs. Note also that Eqs. (14) and (15) covers the input constraints from time n until $n+j$, whereas Eqs. (16) and (17) covers the state constraints from time $n+1$ until $n+j$. The state constraints for time n is covered by Eqs. (12) and (13).

3 Step response models

In industrial practice, process models based on step response descriptions have been very successful. Whereas step response models have no theoretical advantages, they have the practical advantage of being easier to understand for engineers with little background in control theory.

With a soild understanding of the material presented above, the capable reader should have no particular problem in developing a similar MPC formulation based on a step response model. Descriptions of such formulations can

also be found in available publications, like Garcia and Morshedi's [4] original paper presenting "Quadratic Dynamic Matrix Control". Alternatively, step response models may also be expressed in state space form (with a larger number of states than would be necessary in a "minimal" state space model), see e.g. [6] for details.

The reader should beware that step-response models have "finite memory", and hence should only be used for asymptotically stable processes, that is, processes where the effect of old inputs vanish over time. Most industrially successful MPC controllers based on step response models are modified to handle also integrating processes, whereas truly unstable processes cannot be handled. Handling unstable processes using step response models would require more complex modifications to the controllers and model description, and would thereby remove the step response model's advantage of being easy to understand.

Partly due to these reasons, MPC controllers are seldom used on unstable processes. If the underlying process is unstable, it is usually first stabilised by some control loops, and the MPC controller uses the setpoint of these loops as "manipulated variables".

In academia, there is widespread resentment against step response models - and in particular against their use in MPC controllers. Although there are valid arguments supporting this resentment, these are usually of little practical importance for asymptotically stable processes - although in some cases the computational burden can be reduced by using a state space model instead.

Step response *identification* is another matter. A step input has Laplace transform $u(s) = \frac{k}{s}$, and hence excites the process primarily at low frequencies. The resulting model can therefore be expected to be good only for the slow dynamics (low frequencies). If medium to high bandwidth control is desired for an MPC application, one should make sure that any identification experiment excites the process over the whole desired bandwidth range for the controller.

4 Updating the process model

The MPC controller essentially controls the *process model*, by optimizing the use of the inputs in order to remove the predicted deviation from some desired state (or output) trajectory. Naturally, good control of the *true process* will only be obtained if the process model is able to predict the future behaviour of the true process with reasonable accuracy. Model errors and unknown disturbances must always be expected, and therefore it will be necessary to update the process model to maintain good quality predictions of the future process behaviour.

The most general way of doing this is through the use of a state estimator, typically a Kalman filter. The Kalman filter may also be modified to estimate unmeasured disturbances or model parameters that may vary with time. The

Kalman filter is described in advanced control engineering courses and in numerous textbooks, and will not be described further here.

The Kalman filter is, however, a tool that is valid primarily for *linear* problems, and may in some cases estimate state values that defy physical reason. For example, a Kalman filter may estimate a negative concentration of a chemical component in a process. In the rare cases where it is necessary to take physical constraints (and possibly non-linearities in the model) explicitly into account, it is possible to use an 'MPC-like', optimization-based approach to the estimation problem, resulting in what is known as 'moving horizon estimation'. To this author's knowledge, moving horizon estimation is not frequently used in industrial applications, and is to some extent still a research issue. However, it is an active research area. A recent overview can be found in Allgöwer et al. [1].

For asymptotically stable systems, a particularly simple model updating strategy is possible for MPC formulations that only use process inputs and measurements in the formulation (i.e., when unmeasured states do not appear in the objective function or in the constraints). In such cases, it would be natural to calculate the predicted *deviations from the desired output trajectory* (which may be called, say, ψ_{dev}), rather than the predicted deviations from the desired *state* trajectory χ_{dev}. Then, the model can be 'updated' by simply adding the present difference between process output and model output to the model's prediction of the future outputs. This is known as a 'bias update', and is widespread in industrial applications. Note, however, that the bias update

- is only appliccable to asymptotically stable systems, and may result in poor control performance for systems with very slow dynamics, and that
- it may be sensitive to measurement noise. If a measurement is noisy, one should attempt to reduce the noise (typically by a simple low-pass filter) before calculating the measurement bias.

5 Feedforward from disturbances

With MPC it is very simple to include feedforward from measured disturbances, provided one has a model of how the disturbances affect the states/outputs. Feedforward is naturally used to counteract the future effects of disturbances on the controlled variables (it is too late to correct the present value). Thus, feedforward in MPC only requires that the effect on disturbances on the controlled variables are taken into account when predicting the future state trajectory in the absence of any control action. Thus, in the formulation developed above, feedforward from disturbances results from taking the disturbances into account when calculating χ_{dev}.

The benefit obtained by using feedforward will (as always) depend on what bandwidth limitations there are in the system for feedback control. Furthermore, effective feedforward requires both the disturbance and process model to be reasonably accurate.

6 Feasibility and constraint handling

For any type of controller to be acceptable, it must be very reliable. For MPC controllers, there is a special type of problem with regards to *feasibility* of the constraints. An optimization problem is *infeasible* if there exists no exists no set of values for the free variables in the optimization for which all constraints are fulfilled. Problems with infeasibility may occur when using MPC controllers, for instance if the operating point is close to a constraint, and a large disturbance occurs. In such cases, it need not be possible to fulfill the constraint at all times. During startup of MPC controllers, one may also be far from the desired operating point, and in violation of some constraints. Naturally, it is important that the MPC controller should not 'give up' and terminate when faced with an infeasible optimization problem. Rather, it is desirable that the performance degradation is predictable and gradual as the constraint violations increase, and that the MPC controller should effectively move the process into an operating region where all constraints are feasible.

If the constraints are *inconsistent*, i.e., if there exists no operating point where the MPC optimization problem is feasible, then the problem formulation in meaningless, and the problem formulation has to be modified. Physical understanding of the process is usually sufficient to ensure that the constraints are consistent. A simple example of an inconsistent set of constraints is if the value of the minimum value constraint for a variable is higher than the value of the maximum value constraint.

Usually, the constraints on the inputs (manipulated variables) result from true, physical constraints that cannot be violated. For example, a valve cannot be more than 100% open. On the other hand, constraints on the states/outputs often represent operational desireables rather than fundamental operational constraints. State/output constraints may therefore often be violated for short periods of time (although possibly at the cost of producing off-spec products or increasing the need for maintenance). It is therefore common to modify the MPC optimization problem in such a way that output constraints may be violated if necessary. There are (at least) three approaches to doing this modification:

1. Remove the state/output constraints for a time interval in the near future. This is simple, but may allow for unnecessarily large constraint violations. Furthermore, it need not be simple to determine for how long a time interval the state/output constraints need to be removed - this may depend

on the operating point, the input constraints, and the assumed maximum magnitude of the disturbances.

2. To solve a separate optimization problem prior to the main optimization in the MPC calculations, if the MPC problem is infeasible. This initial optimization minimizes some measure of how much the output/state constraints need to be moved in order to produce a feasible optimization problem. The initial optimization problem is usually a LP problem, which can be solved very efficiently.

3. Introducing *penalty functions* in the optimization problem. This involves modifying the constraints by introducing additional variables such that the constraints are always feasible for sufficiently large values for the additional variables. Such modified constraints are termed *soft constraints*. At the same time, the objective function is modified, by introducing a term that penalizes the magnitude of the constraint violations. The additional variables introduced to ensure feasibility of the constraints then become additional free variables in the optimization. Thus, feasibility is ensured by increasing the size of the optimization problem.

The two latter approaches are both rigorous ways of handling the feasibility problem. Approach 3 has a lot of flexibility in the design of the penalty function. One may ensure that the constraints are violated according to a strict list of priorites, i.e., that a given constraint will only be violated when it is impossible to obtain feasibility by increasing the constraint violations for less important constraints. Alternatively, one may distribute the constraint violations among several constraints. Although several different penalty functions may be used, depending on how the magnitude of the constraint violations are measured, two properties are desireable:

- That the QP problem in the optimization problem can still be solved efficiently. This implies that the Hessian matrix for the modified problem should be positive definite, i.e., that there should be some cost on the *square* of the magnitude of the constraint violations.

- That the penalty functions are *exact*, which means that no constraint violations are allowed if the original problem is feasible. This is usually obtained by putting a sufficiently large weight on the magnitude of the constraint violations (i.e., the linear term) in the objective function.

The use of penalty functions is described in standard textbooks on optimization (e.g. [3]), and is discussed in the context of MPC in e.g. [2, 9, 5].

Feasibility at steady state is discussed in more detail in the section on 'Target calculation' below. The techniques used there closely resemble those that are

applied to the dynamic optimization problem in MPC, with the simplification that only steady state is addressed i.e., there is no prediction horizon involved and the variation in constraint violations over the prediction horizon is not an issue. Thus, only the techniques of points 2 and 3 above are relevant for target calculation.

In addition to the problem with feasibility, hard output constraints may also destabilize an otherwise stable system controlled by an MPC controller, see [11]. Although this phenomenon probably is quite rare, it can easily be removed by using a soft constraint formulation for the output constraints [2]. The following section will discuss closed loop stability with MPC controllers in a more general context.

7 Closed loop stability with MPC controllers

The objective function in Eq. (5) closely resembles that of discrete-time Linear Quadratic (LQ) - optimal control, which in this context (since in many MPC applications the states will not be directly measured) may be seen as the control subproblem of LQG-optimal control. Infinite horizon LQ-optimal control is known to result in a stable closed loop system. However, we get additional requirements for ensuring stability whenever not all states are measured. It is well known that stability of the overall system requires the system to be detectable[3]. The requirement for detectability carries over to MPC, note however that the requirement for detectability does not only imply that unstable modes must be detectable from the physical measurements (i.e., that (C, A) is detectable), but also that the unstable modes must affect the objective function, i.e., $(Q^{1/2}, A)$ must be detectable.

With the stabilizability and detectability requirements fulfilled, a *finite horizon* LQG-optimal controller is stable provided the weight on the 'terminal state', S, is sufficiently large. How large S needs to be is not immediately obvious, but it is quite straight forward to calculate an S that is sufficiently large. In the MPC context, this can be done by designing a stabilizing state feedback controller K (typically, one would choose the infinite horizon LQG-optimal controller, obtained by solving a Riccati equation), and then calculate the S that gives the same contribution to the objective function that would be obtained by using the controller K, and summing the terms $(x_i - x_{ref,n})^T Q(x_i - x_{ref,n})$ from $i = n$ to infinity. Since the controller K results in an asymptotically stable system, this sum is finite, and hence S is finite. The value of S can be obtained by solving a discrete Lyapunov equation

[3]Stabilizability is a weaker requirement than the traditional state controllability requirement, since a system is stabilizable if and only if all unstable modes are controllable, i.e., a system can be stabilizable even if some stable modes are uncontrollable. Similarly, a system is detectable if all unstable modes are observable.

$$S - (A + BK)^T S(A + BK) = Q$$

With a sufficiently large S, obtained as described above, the remaining requirement for obtaining closed loop stability is that constraints that are feasible over the horizon $n \leq i \leq n + j$ will remain feasible over an infinite horizon (assuming no new disturbances enter). Rawlings and Muske [8] have shown how to calculate a sufficiently large j. First arrange all state constraints for $n \leq i \leq n+j$ (including input constraints that effectively become state constraints through the assumption that a state feedback controller is used) on the form

$$\widetilde{H} x_i \leq \widetilde{h}$$

and diagonalize the autotransition matrix $A + BK$:

$$A + BK = T \Lambda T^{-1}$$

where Λ is a diagonal matrix (which may have complex-valued elements if $A + BK$ contains oscillatory modes). Then, a sufficiently large value of j can be calculated from

$$\begin{aligned} a &= \frac{\widetilde{h}_{\min}}{\overline{\sigma}(\widetilde{H})\gamma(T)\,\|x_n\|} \\ j &= \max\left\{0, \frac{\ln a}{\ln \lambda_{\max}}\right\} \end{aligned}$$

where $\widetilde{h}_{\min}$ is the smallest element in h, $\overline{\sigma}(\widetilde{H})$ is the maximum singular value of $\widetilde{H}$, $\gamma(T)$ is the condition number of T (ratio of largest to smallest singular value), $\|x_n\| = (x_n^T x_n)^{1/2}$, and $\lambda_{\max}$ is the magnitude of the largest element in Λ, i.e., the largest eigenvalue of $A + BK$.

The problem with the above estimate of j is that it depends on x_n, which can only be predicted when the result of the optimization is available. If the optimization is performed with too small a value for j, one will then have to re-calculate the optimization with an increased j, in order to be able to give any strict stability guarantee. In practice, a constant value for j based on simulations will be used.

The above results on how to find values for S and j to guarantee stability, are not very useful if, e.g., a step response model is used, since the values of the states are then unavailable. Step response-based MPC controllers therefore do not have a terminal state weight S, but rather extend the prediction of the outputs further into the future than the time horizon over which the inputs are optimized (corresponding to $n_p > n_u$ in the comments following Eq. (6). Although a sufficiently large prediction horizon n_p compared to the "input horizon"

n_u will result in a stable closed loop system (the open loop system is assumed asymptotically stable, since a step response model is used), there is no known way of calculating the required n_p. Tuning of step-response based MPC controllers therefore typically rely heavily on simulation. Nevertheless, the industrial success of step response-based MPC controllers show that controller tuning is not a major obstacle in implementations.

8 Target calculation

It is common for MPC controllers to perform a 'target calculation' prior to the main optimization described above. The purpose of this target calculation is to determine consistent steady-state values for the state references $x_{ref,\infty}$ and input references $u_{ref,\infty}$. Most MPC implementation have infrequently changing setpoints, and will use target values that are constant throughout the prediction horizon, i.e. $x_{ref,i} = x_{ref,\infty} \forall i$ and $u_{ref,i} = u_{ref,\infty} \forall i$. This covers industrial practice in the majority of installations, but will not be applicable to some problems, e.g. batch processes or cyclically operated plants. We will also use a linear plant model, which is also common industrial practice. Extending the following to non-linear plant models should in principle not be difficult for the competent reader. However, performing the target calculation at each timestep means that one should be concerned with being able to do the calculations quickly and reliably, and using linear models makes it much simpler to acertain that is actually the case.

One prerequisite for offset-free control is that the minimum value of the objective function is at the desired targets, and to ensure that one desires that

$$(I - A)x_{ref,\infty} = Bu_{ref,\infty} + Ed_\infty \quad (18)$$

$$\widehat{y}_{ref,\infty} = \widehat{C}x_{ref,\infty} + \widehat{F}d_\infty \quad (19)$$

Here $\widehat{y}_{ref,\infty}$ is the desired steady state value of some measurements, and the $\widehat{\ }$ is used to emphasise that the steady state targets may in general be chosen based on other measurements than what are used in the regular control (in the MPC setting, the regular measurements y are used at each time step to update the model). The variable vector d_∞ is the expected/predicted/estimated steady state value of the disturbances affecting the process. In the (rare) unconstrained case, and with as many inputs u as controlled outputs $\widehat{y}$, the state and input targets can be found from a simple matrix inversion

$$\begin{bmatrix} x_{ref,\infty} \\ u_{ref,\infty} \end{bmatrix} = \begin{bmatrix} -(I-A) & B \\ \widehat{C} & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 & -E \\ I & -\widehat{F} \end{bmatrix} \begin{bmatrix} \widehat{y}_{ref,\infty} \\ d_\infty \end{bmatrix} \tag{20}$$

$$= M^{-1} \begin{bmatrix} 0 & -E \\ I & -\widehat{F} \end{bmatrix} \begin{bmatrix} \widehat{y}_{ref,\infty} \\ d_\infty \end{bmatrix} \tag{21}$$

Clearly, for the targets $x_{ref,\infty}$ and $u_{ref,\infty}$ to be well defined, the matrix M above needs to be of full rank.

Many factors may make it impossible to obtain the targets by the simple calculations above:

- There may be more inputs than outputs.
- There may be more controlled variables than inputs.
- In addition to desired values for the controlled variables, one may wish to keep the inputs close to specific values.
- Achieving the desired values for the controlled variables may be impossible (or otherwise unacceptable) due to constraints.

When such problems of concern (and if they are not, there is probably little reason to use MPC in the first place), the target calculations are performed by solving an optimization problem or a series of such problems. In the following, we will use the subscript $_d$ to denote *desired* values of controlled variables $\widehat{y}$ and inputs u, whereas the subscript $_{ref}$ will still refer to the reference values or targets used in the MPC calculations. The desired values are set by operators or higher level plant optimization, whereas the MPC targets are the result of the target calculation.

The most straightforward formulation will cast the target calculation as a QP problem:

$$\min_{x_{ref,\infty}, u_{ref,\infty}} \quad (\widehat{y}_d - \widehat{C}x_{ref,\infty} - \widehat{F}d_\infty)^T Q(\widehat{y}_d - \widehat{C}x_{ref,\infty} - \widehat{F}d_\infty) \tag{22}$$
$$+(u_d - u_{ref,\infty})^T W(u_d - u_{ref,\infty})$$

subject to given values for y_d, u_d and d_∞ , the model equations Eq. (18) and the relevant maximum and minimum value constraints on $x_{ref,\infty}$ and $u_{ref,\infty}$. The matrix Q is assumed to be positive definite. A positive definite W will in general result in offset in the controlled variables even in cases when the desired values $\widehat{y}_d$ can be achieved. The matrix W may therefore be chosen to be positive

semi-definite. Muske [7] shows how to specify a semi-definite W which does not introduce offset in the controlled variables. Note, however, that when there are more inputs than controlled variables, the number of inputs without any weight in the optimization problem must not exceed the number of controlled variables. Also, in many cases there may be reasons for keeping the inputs close to a specified value, and in such cases the inputs concerned should be given a weight in the optimization problem above. Ideally, the target values should comply with the same maximum and minimum value constraints as that of the MPC problem, c.f. Eq. (6), but there may also be other constraints, e.g. on elements of $\widehat{y}$ that are not present in y. Let us assume that all such constraints can be described by the inequality

$$\widehat{H} \begin{bmatrix} x_{ref,\infty} \\ u_{ref,\infty} \end{bmatrix} \geq \widehat{b} \tag{23}$$

Difficulties will arise whenever there is no feasible region in which the constraints of Eq. (18) and Eq. (23) can all be fulfilled. This is indeed often the case when operating in a highly constrained region (which is the major advantage of MPC), but may also result from operators specifying overly stringent constraints. For *any* sort of control to be feasible in such a case, it becomes necessary to relax some of the constraints. It should be obvious that the process model Eq. (18) cannot be relaxed, since it is given by the physics of the problem at hand. Likewise, most input constraints are hard constraints that cannot be relaxed, such as actuator limitations. On the other hand, many state or output constraints represent operational desireables rather than physical necessities, and violation of such constraints may be possible without putting the safety of the plant in jeopardy. Allowing violations in selected constraints can be achieved by introducing additional variables into the optimisation problem. Thus, instead of Eq. (22) we get

$$\begin{aligned} \min_{x_{ref,\infty}, u_{ref,\infty}, p} \quad & (\widehat{y}_d - \widehat{C} x_{ref,\infty} - \widehat{F} d_\infty)^T Q (\widehat{y}_d - \widehat{C} x_{ref,\infty} - \widehat{F} d_\infty) \\ & + (u_d - u_{ref,\infty})^T W (u_d - u_{ref,\infty}) + l^T p + p^T Z p \end{aligned} \tag{24}$$

where l is a vector of positive constraint violation costs and Z is positive definite. The vector p gives the magnitude of the constraint violations. The model equations in Eq. (18) are assumed to hold as before, whereas the constraints in Eq. (23) are modified to

$$\begin{aligned} \widehat{H} \begin{bmatrix} x_{ref,\infty} \\ u_{ref,\infty} \end{bmatrix} + \widehat{L} p &\geq \widehat{b} \\ p &\geq 0 \end{aligned} \tag{25}$$

The matrix $\widehat{L}$ determines which constraints are relaxed. Its elements will take the values 0 or 1, with exactly one element equal to 1 for each column, and at most one element equal to 1 for each row. If a row of $\widehat{L}$ contains an element equal to 1, this means that the corresponding constraint may be relaxed.

For a sufficiently large l, the optimal solution to Eq. (24) is also the optimal solution to Eq. (22), provided a feasible solution for Eq. (22) exists.

The target calculation formulation in Eqs. (24 - 25) will distribute the constraint violations between the different relaxable constraints. If one instead wishes to enforce a strict priority among the constraints, so that a given constraint is violated only if feasibility cannot be achieved even with arbitrarily large constraint violations in the less important constraints, this may be achieved by solving a series of LP problems, followed by a QP problem for the target calculation. The following algorithm may be used:

1. Simple inspection at the design stage will often ensure that the non-relaxable constraints are always feasible. If not, it may be necessary to check that there exists a feasible solution to the problem when only considering the non-relaxable constraints. Set $\widehat{H}_r$ to the rows of $\widehat{H}$ corresponding to the non-relaxable constraints, and $\widehat{b}_r$ to the corresponding elements of $\widehat{b}$. Set c_r to $\left[\begin{array}{ccccc} 0 & 0 & 1 & \cdots & 1 \end{array}\right]^T$ where the leading zeros should be interpreted as zero vectors of dimensions corresponding to the dimensions of the state and input vectors, respectively. Solve the LP problem

$$\min_{x_{ref,\infty}, u_{ref,\infty}, p} c_r^T \left[\begin{array}{c} x_{ref,\infty} \\ u_{ref,\infty} \\ p \end{array}\right]$$

subject to the constraints

$$\begin{aligned} \left[\begin{array}{cc} \widehat{H}_r & I \end{array}\right] \left[\begin{array}{c} x_{ref,\infty} \\ u_{ref,\infty} \\ p \end{array}\right] &\geq \widehat{b}_r \\ p &\geq 0 \end{aligned}$$

If the optimal value for this LP problem is larger than 0, the non-relaxable constraints are infeasible, which would indicate serious mistakes in the constraint specifications or abnormally large disturbances (the latter of which could affect $\widehat{b}_r$). Proceed to the next step in the algorithm if the non-relaxable constraints are feasible, if not, there is reason to activate an alarm to get operator attention.

2. Add the most important of the remaining relaxable constraints and find the minimum constraint violation in that constraint only which results in a feasible solution. This is done by adding the corresponding row of $\widehat{H}$ and $\widehat{b}$ to $\widehat{H}_r$ and $\widehat{b}_r$, respectively, using a scalar 'dummy variable' p, and setting c_r to $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$The zeros in c_r are still zero vectors of appropriate dimension, whereas the 1 is scalar. The LP problem to solve at this stage becomes

$$\min_{x_{ref,\infty}, u_{ref,\infty}, p} c_r^T \begin{bmatrix} x_{ref,\infty} \\ u_{ref,\infty} \\ p \end{bmatrix}$$

subject to the constraints

$$\begin{bmatrix} & & 0 \\ & \widehat{H}_r & \vdots \\ & & 0 \\ & & 1 \end{bmatrix} \begin{bmatrix} x_{ref,\infty} \\ u_{ref,\infty} \\ p \end{bmatrix} \geq \widehat{b}_r$$

$$p \geq 0$$

3. Move the contribution of the dummy variable p into $\widehat{b}_r$. That is, set $\widehat{b}_r \leftarrow \widehat{b}_r + \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix}^T p$. If there are more relaxable constraints, go to point 2 above.

4. When all constraints are accounted for, and a feasible solution is known to exist, solve the QP problem for target calculation with modified constraints.

Instead of solving a series of LP problems, the solution may be found by solving a single LP problem [10]. However, the required LP problem is quite complex to design. Although this design problem is solved off-line, it will need to be modified whenever the constraint specifications change. At the time of writing, no reliable software is known to exist for solving this LP design problem.

9 Robustness of MPC controllers

The main advantage of MPC controllers lie in their ability to handle constraints. On the other hand, they may be sensitive to errors in the process model. There have been tales about processes which are controlled by MPC controllers when prices are high (and it is important to operate close to the process' maximum

throughput), but are controlled by simple single-loop controller when prices are low (and production is lower, leading to no active constraints). The potential robustness problems are most easily understood for cases when no constraints are active, i.e., when we can study the objective function in Eq. (1) with H and c from Eq. (9). We then want to minimize

$$f(v) = v^T(\widehat{B}^T\widehat{Q}\widehat{B} + \widehat{P})v + \chi_{dev}^T\widehat{A}^T\widehat{Q}\widehat{B}v$$

with respect to v. The solution to this minimization can be found analytically, since no constraints are assumed to be active. We get[4]

$$v = -(\widehat{B}^T\widehat{Q}\widehat{B} + \widehat{P})^{-1}\widehat{B}^T\widehat{Q}\widehat{A}\chi_{dev}$$

Clearly, if the model contains errors, this will result in errors in $\widehat{B}$ and $\widehat{A}$, and hence the calculated trajectory of input moves, v, will be different from what is obtained with a perfect model. If the Hessian matrix $\widehat{B}^T\widehat{Q}\widehat{B} + \widehat{P}$ is *ill-conditioned*[5], the problem is particularly severe, since a small error in the Hessian can then result in a large error in its inverse. For a physical motivation for problems with ill-conditioning consider the following scenario:

- The controller detect an offset from the reference in a direction for which the process gain is low.
- To remove this offset, the controller calculates that a large process input is needed in the low gain input direction.
- Due to the model errors, this large input actually slightly "misses" the low gain input direction of the true process.
- The fraction of the input that misses the low gain direction, will instead excite some high gain direction of the process, causing a large change in the corresponding output direction.

Now, there are two ways of reducing the condition number of $\widehat{B}^T\widehat{Q}\widehat{B} + \widehat{P}$:

1. Scaling inputs and states in the process model, thereby changing $\widehat{B}$.
2. Modifying the tuning matrices $\widehat{Q}$ and $\widehat{P}$.

Scaling inputs and states (or outputs, if the objective function uses outputs instead of states) is essentially the same as changing the units in which we measure

[4]Note that $\widehat{Q} = \widehat{Q}^T$, and that the assumptions on Q, S and P ensures that $(\widehat{B}^T\widehat{Q}\widehat{B} + \widehat{P})$ is of full rank, and hence invertible.

[5]A matrix is ill-conditioned if the ratio of the largest singular value to the smallest singular value is large. This ratio is called the *condition number*.

these variables. In some cases this is sufficient, but some processes have inherent ill-conditioning that cannot be removed by scaling.

In theory, one may use non-zero values for all elements in the tuning matrices $\widehat{Q}$ and $\widehat{P}$, with the only restriction that $\widehat{Q}$ should be positive semi-definite[6] and $\widehat{P}$ should be positive definite (and hence both should be symmetric). However, little is known on how to make full use of this freedom in designing $\widehat{Q}$ and $\widehat{P}$, and in practice they are obtained from Q, P and S as shown in Eq. (8), and typically Q and P are diagonal. It is common to try to reduce the ill-conditioning of the Hessian matrix by multiplying all elements of $\widehat{P}$ by the same factor. If this factor is sufficiently large, the condition number of the Hessian matrix will approach that of P - which can be chosen to have condition number 1 if desired. However, increasing all elements of $\widehat{P}$ means that the control will become slower in all output directions, also in directions which are not particularly sensitive to model uncertainty.

If the above ways of reducing the condition number of the Hessian matrix are insufficient or unacceptable, one may instead modify the process model such that the controller "does not see" offsets in the low gain directions. Inherent ill-conditioning (which cannot be removed by scaling) is typically caused by physical phenomena which make it difficult to change the outputs in the low gain direction. Fortunately, this means that disturbances will also often have a low gain in the same output direction. It may therefore be acceptable to ignore control offsets in the low gain output directions. In terms of the MPC formulation above, the controller can be forced to ignore the low gain directions by modifying $\widehat{B}$ by setting the small singular values of $\widehat{B}$ to zero. This is known as *singular value tresholding*, since we remove all singular values of $\widehat{B}$ that is smaller than some treshold. If we term this modified matrix $\widehat{B}$ for $\widehat{B}_m$, we find that the trajectory of input moves calculated by the (unconstrained) MPC optimization now becomes

$$v = -(\widehat{B}_m^T \widehat{Q} \widehat{B}_m + \widehat{P})^{-1} \widehat{B}_m^T \widehat{Q} \widehat{A} \chi_{dev} = -(\widehat{B}_m^T \widehat{Q} \widehat{B}_m + \widehat{P})^{-1} \chi_m$$

Note that the conditioning of the Hessian matrix is not improved by setting the small singular values of $\widehat{B}$ to zero, but the vector χ_m does not show any control offset in the corresponding output directions, and hence the vector v will contain no input moves in the corresponding input directions.

Singular value tresholding is effective in improving robustness to model errors, but it clearly causes nominal control performance (the performance one would get if the model is perfect) to deteriorate, since the controller ignores control offsets in some output directions. Removing too many singular values from $\widehat{B}$ will result in unacceptable control performance.

[6]The lower right diagonal block of $\widehat{Q}$, corresponding to the terminal state weight S, should be strictly positive definite (and sufficiently large).

10 Using rigorous process models in MPC

Most chemical processes are inherently nonlinear. In some cases, rigorous dynamical models based on physical and chemical relationships are available, and the process engineers may wish to use such a model in an MPC controller. This would for instance have the advantage of automatically updating the model when the process is moved from one operating point to another.

However, to optimize directly on the rigorous model is not straight forward. The non-linearity of the model typically results in optimization problems that are non-convex. Optimization of non-convex problems is typically *a lot* more time consuming than optimization of convex problems and the time required to find a solution can vary dramatically with changing operating point or initial states. This means that direct optimization of non-linear models is usually ill-suited for online applications like MPC. Furthermore, it is often the case that the most important 'non-linearities' in the true system are the constraints, which are handled effectively by MPC.

This does not mean that rigorous models cannot be utilized by MPC controllers, but it means that one can make only partial use of such models. The idea is to utilize these models to the extent that the available time permits. One may then approximate the true optimization problem by a modified, convex problem, or a series of such problems.

Predict using the rigorous model. The simplest way of (partially) accounting for non-linearity in the process model, is to calculate the deviation from the desired state (or output) trajectory from a rigorous, non-linear model, whereas the other parts of the optimization formulation uses a linearized model. In this way, the calculated input trajectory v will to some extent account for the non-linearities.

Line search If greater accuracy is needed, one may do a line search using the non-linear model to optimize what multiple of v should be implemented, i.e., perform a search to optimize (while taking the constraints into account)

$$\min_{\alpha} f(x,u) = \min_{\alpha} f(x_0, u_{ref} + \alpha v) \tag{26}$$

where α is a positive real scalar. Such line searches are a standard part of most non-linear optimization methods, and are covered in many textbooks on optimization e.g. in [3]. When performing the minimization in Eq. (26) above, the full non-linear model is used to calculate future states from $(x_0, u_{ref} + \alpha v)$.

Iterative optimization. Even with the optimal value of α, one probably has not found the optimal solution to the original non-linear optimization problem. Still better solutions may be found by an iterative procedure, where the

predicted deviation from the desired state trajectory x_{ref} is found using the best available estimate of the future input trajectory. That is, for iteration number k, use the model to calculate the resulting vector $\chi_{dev,k}$ when the input trajectory $u_{ref} + v_t$ is applied, where $v_t = \sum_{l=0}^{k-1} v_l$, and minimize

$$\min_{v_k} f(v) = (v_t + v_k)^T (\widehat{B}^T \widehat{Q} \widehat{B} + \widehat{P})(v_t + v_k) + \chi_{dev,k}^T \widehat{A}^T \widehat{Q} \widehat{B}(v_t + v_k)$$

subject to constraints that should be modified similarly. It is also assumed that a line search is performed between each iteration. The iterations are initialized by setting $v_0 = 0$, and are performed until v_k approaches zero, or until the available time for calculations is used up. The iterative procedure outlined above need not converge to a globally optimal solution for the original problem, it may end up in a local minimum. Furthermore, there is no guarantee that this is a particularly efficient way of solving the original optimization problem (in terms of the non-linear model). It does, however, have the advantage of quickly finding reasonable, and hopefully feasible, input sequences. Whenever this is the case, even if the optimization has to terminate before the optimization has converged, a 'good' input has been calculated and is available for implementation on the process.

Linearize around a trajectory. If the operating conditions change significantly over the time horizon (n) in the MPC controller, the linearized model may be a reasonable approximation to the true process behaviour for only a part of the time horizon. This problem is relatively rare when constant reference values are used, but may be relevant when moving from one operating point to another. It is then possible to linearize the process around the predicted process trajectory ($x_{ref} + \chi_0$) rather than around a constant state. One then gets a time-varying (but still linear) model, i.e., a "new model" for each time interval into the future. Conceptually, linearizing around a trajectory does not add much complexity compared to linearizing around a constant state, but it does add significantly to the notational complexity that is necessary in the mathematical formulation of the optimization problem. Furthermore, analytical representations of the linearized models are typically not available, and the linearization has to be performed by numerically perturbing the process around the predicted process trajectory. This can clearly add significantly to the computational burden. Linearizing around a trajectory can be combined with iterative optimization as outlined above - which would further add to the computational burden.

References

[1] F. Allgöwer, T. A. Badgewell, J. S. Qin, J. B. Rawlings, and S. J. Wright. Nonlinear model predictive control and moving horizon estimation - an in-

troductory overview. In *Advances in Control. Highlights of the ECC'99*. Springer, 1999.

[2] N. M. C. de Olivieira and L. T. Biegler. Constraint handling and stability properties of model-predictive control. *AIChE Journal*, 40(7):1138–1155, July 1994.

[3] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 2nd edition, 1987.

[4] C. E. Garcia and A. M. Morshedi. Quadratic programming solution of dynamic matrix control (QDMC). *Chem. Eng. Commun.*, pages 73–87, 1986.

[5] M. Hovd and R. D. Braatz. Handling state and output constraints in MPC using time-dependent weights. In *Proceedings of the American Control Conference*, 2001.

[6] M. Hovd, J. H. Lee, and M. Morari. Truncated step response models for model predictive control. *J. Proc. Cont.*, 3(2):67–73, 1993.

[7] K. R. Muske. Steady-state target optimization in linear model predictive control. In *Proc. American Control Conference*, pages 3597–3601, 1997.

[8] J. B. Rawlings and K. R. Muske. The stability of constrained receding horizon control. *IEEE Transactions on Automatic Control*, 38(10):1512–1516, 1993.

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

[10] J. Vada. *Prioritized Infeasibility Handling in Linear Model Predictive Control: Optimality and Efficiency*. PhD thesis, Engineering Cybernetics Department, NTNU, 2000.

[11] E. Zafiriou and A. L. Marchal. Stability of SISO quadratic dynamic matrix control with hard output constraints. *AIChE Journal*, 37(10):1550–1560, October 1991.