CHAPTER 5

MODEL-BASED PREDICTIVE CONTROL

5.1 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

Advanced Chemical Process Control, First Edition.
By Morten Hovd Copyright © 2022 John Wiley & Sons, Inc.

ensure that there exists a unique optimal solution that can be found quickly with effective optimization solvers, the *Hessian matrix* in the objective function has to be *positive definite*[1].

LP problems can be solved more efficiently than QP problems, and an LP formulation may therefore 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.

So-called *explicit* MPC will not be considered here. However, the optimization formulations are the same in explicit and ordinary ('implicit') MPC, what differs is how and when the optimization problems are solved. While explicit MPC offers simpler on-line calculations and simpler on-line computer code, the off-line calculations can be very demanding, and current technology can therefore only manage problems of modest size.

5.2 Formulation of a QP problem for MPC

A standard QP problem takes the form

$$\min_{v} \quad 0.5 v^T \widetilde{H} v + c^T v \tag{5.1}$$

subject to the constraints

$$Lv \leq b \tag{5.2}$$

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 or semi-definite[2]. 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. Similarly, many QP solvers will allow entering equality constraints explicitly, but equality constraints may also be

[1]The Hessian matrix defines the quadratic term in the objective function, and is a symmetric matrix. Positive definiteness means that all eigenvalues are positive - for a monovariable optimization problem this implies that the coefficient for the quadratic term in the objective function is positive. If some eigenvalues are identically zero (but none negative), the matrix is called semi-definite.

[2]Semi-definiteness suffices for an optimum to exist (unless the constraints are infeasible), for the optimum to be unique the Hessian matrix should be positive definite *on the subspace orthogonal to the active constraints*

specified in (5.2), since $v = w$ is equivalent to the two inequality constraints $v \leq w$ and $-v \leq -w$.

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

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Ed_k \quad (5.3) \\ y_k &= Cx_k + Fd_k \quad (5.4) \end{aligned}$$

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. One may also argue that in most physically realistic system descriptions, inputs and disturbances affect the rate of change of states rather than the states themselves. To illustrate: mass transfer/flowrate disturbances affect the rate of accumulation of mass, heat transfer/temperature disturbances affect the rate of accumulation of energy, force disturbances affect acceleration (the rate of accumulation of momentum), etc.

In the same way as is common in control literature, the state x, input u, external disturbance d and measurement y above should be interpreted as *deviation variables.* This means that they represent the deviations from some consistent set of of variables $\{x_L, u_L, d_L, y_L\}$ around which the model is obtained[3]. For a continuous process, the set $\{x_L, u_L, d_L, y_L\}$ will typically represent a stationary point - often the stationary point we want to keep the process at.

A typical optimization problem in MPC might take the form

$$\begin{aligned} \min_u \quad f(x,u) \quad = \quad & \sum_{i=0}^{n-1} \{(x_i - x_{ref,i})^T Q (x_i - x_{ref,i}) \quad (5.5) \\ & + (u_i - u_{ref,i})^T P (u_i - u_{ref,i})^T \} \\ & + (x_n - x_{ref,n})^T S (x_n - x_{ref,n}) \end{aligned}$$

subject to constraints

$$\begin{aligned} x_0 &= \text{given} \\ M_i x_i + N_i u_i &\leq G_i \qquad \text{for } 0 \leq i \leq n-1 \\ M_n x_n &\leq G_n \quad (5.6) \end{aligned}$$

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

In the objective function Eq. (5.5) 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). 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 (5.6).

The constraints may be given for inputs u_i and states x_i independently or may be expressed in terms of combinations of inputs and states. At time $i = 0$, we cannot affect the states (since these are already given. The constraints at time $i = 0$ will therefore consider only the inputs. At time $i = 0$, an input constraint of the form $-U \leq u_0 \leq U$ can therefore be expressed using

$$M_0 = 0, \quad N_0 = \begin{bmatrix} I \\ -I \end{bmatrix}, \quad G_0 = \begin{bmatrix} U \\ -U \end{bmatrix}.$$

Typically, the constraints are constant with time, but a particular set of constraints is applied to the state vector at the end of the prediction horizon ($i = n$). These *terminal constraints* may be stricter than the constraints imposed over the rest of the prediction horizon. The purpose of this (possibly) stricter set of constraints is to ensure that the constraints can be fulfilled also for future MPC problems (as the prediction horizon 'moves forward'), and we will return to the determination of these terminal constraints later.

In the following, this formulation of the optimization problem will be recast into the standard QP formulation in Eqs.(5.1) and (5.2), but first a number of remarks and explanations to the optimization problem formulation in Eqs.(5.5) to (5.6) are needed.

- In addition to the above constraints, it is naturally assumed that the process follows the model in Eqs. (5.3) and (5.4).

- 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 $\Delta U_L \leq u_i - u_{i-1} \leq \Delta U_U$.

- The determination of the terminal constraints will require an assumption on how the inputs u_i will be used for $i \geq n$. Typical choices are either that $u_i = u_{ref,i}$, $u_i = u_{i-1}$, 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.

- Similarly, we must predict future values for disturbances. Good predictions may sometimes be available, due to e.g., knowledge about operation of upstream equipment. In the absence of such information, it is common to assume that the disturbance will keep its present (measured or estimated) value over the prediction horizon.
- If one assumes that $(u_i - u_{ref,i}) = K(x_i - x_{ref,i})$ for $n \leq i \leq n + j$, one should also include the input constraints in the problem formulation for the time interval $n \leq i \leq n + j$. These input constraints then effectively become state constraints for this time interval.
- Some MPC formulations use an objective function of the form

$$\begin{aligned} f(x,u) \quad &= \textstyle\sum_{i=0}^{n_p} (x_i - x_{ref,i})^T Q (x_i - x_{ref,i}) \\ &+ \textstyle\sum_{i=0}^{n_u} (u_i - u_{ref,i})^T P (u_i - u_{ref,i}), \end{aligned}$$

  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).

In the following, we will recast the MPC optimization problem as a standard QP problem. The state at the beginning of the prediction horizon is assumed to be given (either from measurements or from a state estimator), and cannot be affected by the outcome of the optimization in MPC. In the presentation below, the state x_0 will therefore be eliminated from the formulation (except where it denotes the initial state, and not a degree of freedom in the optimization). We will assume that $u_i - u_{ref,n} = K(x_i - x_{ref,n})$ for $i \geq n$. To start off, we stack the state references $x_{ref,i}$, input references $u_{ref,i}$, input deviations $v_i = u_i - u_{ref,i}$, state deviations $\chi_i = x_i - x_{ref,i}$, and predicted disturbances d_i in long (column) vectors x_{ref}, u_{ref}, v, χ_{dev}, and δ:

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

$$v = \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix}; \quad \chi = \begin{bmatrix} \chi_1 \\ \chi_2 \\ \vdots \\ \chi_{n-1} \\ \chi_n \end{bmatrix}; \quad \delta = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-2} \\ d_{n-1} \end{bmatrix}$$

Note that

$$\begin{bmatrix} u_0 \\ \vdots \\ u_{n-1} \end{bmatrix} = v + u_{ref}; \quad \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \chi + x_{ref}$$

We introduce the matrices

$$\widehat{Q} = \begin{bmatrix} Q & 0 & \cdots & 0 & 0 \\ 0 & Q & \ddots & \vdots & \vdots \\ 0 & 0 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & Q & 0 \\ 0 & 0 & \cdots & 0 & S \end{bmatrix}, \quad \widehat{P} = \begin{bmatrix} P & 0 & \cdots & 0 & 0 \\ 0 & P & \ddots & \vdots & \vdots \\ 0 & 0 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & P & 0 \\ 0 & 0 & \cdots & 0 & P \end{bmatrix} \tag{5.7}$$

$$\widehat{M} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ M_1 & 0 & \cdots & 0 & 0 \\ 0 & M_2 & \ddots & \vdots & \vdots \\ 0 & 0 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & M_{n-1} & 0 \\ 0 & 0 & \cdots & 0 & M_n \end{bmatrix} \tag{5.8}$$

$$\widehat{N} = \begin{bmatrix} N_0 & 0 & \cdots & 0 & 0 \\ 0 & N_1 & \ddots & \vdots & \vdots \\ 0 & 0 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & N_{n-2} & 0 \\ 0 & 0 & \cdots & 0 & N_{n-1} \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} \tag{5.9}$$

(5.10)

and the vector

$$\widehat{G} = \begin{bmatrix} G_0 \\ \vdots \\ G_n \end{bmatrix} \tag{5.11}$$

Next, three nominally equivalent formulations of the QP optimization problem in MPC will be described.

5.2.1 Future states as optimization variables

The optimization problem may be expressed as

$$\min_{\chi,v} \begin{bmatrix} \chi^T & v^T \end{bmatrix} \begin{bmatrix} \widehat{Q} & 0 \\ 0 & \widehat{P} \end{bmatrix} \begin{bmatrix} \chi \\ v \end{bmatrix} \tag{5.12}$$

Repeated use of the model equation results in

$$\begin{aligned} (\chi + x_{ref}) \;=\; & \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ A & 0 & 0 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & A & \ddots & 0 \\ 0 & \cdots & 0 & A & 0 \end{bmatrix} (\chi + x_{ref}) + \begin{bmatrix} A \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} x_0 \\ & + \begin{bmatrix} B & 0 & 0 & \cdots & 0 \\ 0 & B & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \cdots & B & 0 \\ 0 & \cdots & \cdots & 0 & B \end{bmatrix} (v + u_{ref}) + \begin{bmatrix} E & 0 & 0 & \cdots & 0 \\ 0 & E & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \cdots & E & 0 \\ 0 & \cdots & \cdots & 0 & E \end{bmatrix} \delta \end{aligned} \tag{5.13}$$

In addition, the inequality constraints (5.6) may be expressed as

$$\begin{bmatrix} \widehat{M} & \widehat{N} \end{bmatrix} \begin{bmatrix} \chi \\ v \end{bmatrix} \leq \begin{bmatrix} \widehat{G} - \widehat{M} x_{ref} - \widehat{N} u_{ref} \end{bmatrix} \tag{5.14}$$

The objective function (5.12), the (model) equality constraints (5.13), and the inequality constraints (5.14) together specify the MPC optimization problem on a standard form. Note that x_0, x_{ref}, u_{ref}, and δ all are assumed known to the QP solver. Whereas correct (or reasonable) values for x_0, x_{ref}, and u_{ref} will often easily be available, some prediction of future disturbances δ will have to be made. In the absence of any relevant information (e.g., measurements used for feedforward from disturbances), it is commonly assumed that the disturbance will remain constant at its present value.

In this case, the optimization variables are both the future plant inputs v and future plant states χ. The model equations (expressed as equality constraints) guarantee that the relationship between inputs and states are fulfilled, and the *de facto* maximum number of degrees of freedom in the optimization is given by the number of inputs times the prediction horizon n.

This formulation of the MPC problem results in a high number of optimization variables. On the other hand, the equations are easily formulated and the resulting matrices are highly structured and *sparse* (i.e., the contain many elements that are

identically zero). Provided the QP solver used can take advantage of the sparse structure, the high number of optimization variables need not imply an increased computational demand. Rao et al. [RWR98] study the formulation of efficient QP solvers for MPC problems with such structure.

5.2.2 Using the model equation to substitute for the plant states

When using a 'standard' QP solver for *dense* (i.e., non-sparse, ordinary) matrices, the computational demands generally increase rapidly with the number of optimization variables. In such cases it is generally advisable to formulate the MPC problem with as few degrees of freedom as possible in the optimization. To this end, the model equality constraints (5.13) are used to eliminate the future states from the problem formulation. To simplify notation, we express (5.13) as

$$I_A\left(\chi + x_{ref}\right) = A_0 x_0 + \tilde{B}\left(v + u_{ref}\right) + \tilde{E}\delta \tag{5.15}$$

Noting that I_A always is invertible, we solve for χ:

$$\chi = I_A^{-1}\left(A_0 x_0 + \tilde{B}(v + u_{ref}) + \tilde{E}\delta\right) - x_{ref} = \widehat{A}x_0 + \widehat{B}(v + u_{ref}) + \widehat{B}_d\delta - x_{ref} \tag{5.16}$$

Next we 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 split χ into to components, $\chi = \chi_{dev} + \chi_v$. Here χ_{dev} is 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, and that the predicted future disturbances are correct. Similarly, χ_v is the effect on the future state trajectory from the future deviations from the reference input. The model equations then give

$$\chi_{dev} = \widehat{A}x_0 + \widehat{B}u_{ref} + \widehat{B}_d\delta - x_{ref} \tag{5.17}$$

$$\chi_v = \widehat{B}v \tag{5.18}$$

Adding (5.17) and (5.18) we get (5.16).

The objective function can be written as

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

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

Thus, the objective function is in the form of a standard QP problem as defined in (5.1) and (5.2) if we define

$$\begin{aligned} \widetilde{H} &= \widehat{B}^T \widehat{Q} \widehat{B} + \widehat{P} \\ c^T &= \chi_{dev}^T \widehat{Q} \widehat{B} \end{aligned} \tag{5.19}$$

This formulation corresponds to dropping the constant term $\chi_{dev}^T \widehat{Q} \chi_{dev}$ from the objective function. This is allowable, since this term does not affect the location of the minimum of the objective function $f(x, u)$ in (5.5), *i.e.*, it only introduces an offset in the objective function, but does not affect the optimal input sequence that is the result of the optimization. It now remains to express the constraints in the MPC problem in the form of a standard QP problem. Using (5.17) and (5.18) to substitute the model equations into the inequality constraints, we obtain

$$(\widehat{M}\widehat{B} + \widehat{N})v \leq \widehat{G} - \widehat{M}(x_{ref} + \chi_{dev}) - \widehat{N}u_{ref} \tag{5.20}$$

5.2.3 Optimizing deviations from linear state feedback

The main reason for using model predictive control is usually it's ability to handle constraints. If constraints are not a problem, linear state feedback (using e.q. LQ-optimal control) would often be preferred. Indeed, if no constraints are active, many MPC formulations can be shown to be equivalent to linear state feedback. This has lead Rossiter [Ros03] to propose an MPC formulation where the degrees of freedom in the optimization are the deviations from linear state feedback that are necessary to adhere to the constraints. Thus, the input is parameterized as

$$u_i - u_{ref,i} = K(x_i - x_{ref,i}) + c_i \tag{5.21}$$

for some given state feedback controller K. Here c_i are the deviations from linear state feedback that are to be minimized, and it is assumed that $c_i = 0$ for $i \geq n$. We introduce the notation

$$\widehat{K} = \text{diag}\{K\}; \quad \widehat{c} = \left[c_0^T, c_1^T, \cdots, c_{n-1}^T\right]^T$$

Next, we use the model equations to express the future manipulated variables. When using (5.16) one needs to keep in mind that v starts from time $i = 0$, whereas χ starts from time $i = 1$. Thus we define

$$\begin{aligned} \widehat{A}' &= \text{the } n \text{ first blocks of rows of } \begin{bmatrix} I \\ \widehat{A} \end{bmatrix} \\ \widehat{B}' &= \text{the } n \text{ first blocks of rows of } \begin{bmatrix} 0 \\ \widehat{B} \end{bmatrix} \\ \widehat{B}_d' &= \text{the } n \text{ first blocks of rows of } \begin{bmatrix} 0 \\ \widehat{B}_d \end{bmatrix} \\ x_{ref}' &= \text{the } n \text{ first blocks of rows of } \begin{bmatrix} x_{ref,0} \\ x_{ref} \end{bmatrix} \end{aligned}$$

where the $'$ sign should not be confused with the transposition of the matrix. Future plant inputs may thus be expressed as

$$v = \widehat{K}\left(\widehat{A}'x_0 + \widehat{B}'(v + u_{ref}) + \widehat{B}'_d\delta - x'_{ref}\right) + \widehat{c} \tag{5.22}$$

Rearranging, we obtain

$$v = (I - \widehat{K}\widehat{B}')^{-1}\widehat{K}\left(\widehat{A}'x_0 + \widehat{B}'u_{ref} + \widehat{B}'_d\delta - x'_{ref}\right) + (I - \widehat{K}\widehat{B}')^{-1}\widehat{c} \tag{5.23}$$

where we note that $(I - \widehat{K}\widehat{B}')$ is always invertible since $\widehat{B}'$ have all non-zero elements below the main diagonal. It is trivial (but tedious if done by hand) to substitute (5.23) into (5.19) and (5.20) to obtain the corresponding standard QP formulation in terms of the new optimization variables $\widehat{c}$. It can be shown (*e.g.*, [Ims07]), that if the controller K is chosen as the LQ-optimal regulator according to (5.101) for the state weight Q and input weight P, then one ends up with $\tilde{H} = \mathrm{diag}(B^T SB + P)$ and $c = 0$ in (5.1), where S is the solution to the Riccati equation (5.101) that has to be solved to find the controller K.

This formulation embeds stabilizing feedback into the MPC formulation. Doing so can make the problem formulation numerically much better conditioned, especially for open loop unstable plants and long prediction horizons.

5.2.4 Constraints beyond the end of the prediction horizon

Fulfilling the constraints over the prediction horizon at a single timestep does not necessarily ensure that it will be possible to fulfill the constraints at the next timestep. Clearly, we would like a problem formulation such that if the problem is feasible initially, it will remain feasible for all future times, *i.e.*, we would like the MPC formulation to be *recursively feasible*. While (5.6) allows for constraints that vary along the prediction horizon, it also strongly suggests that a separate set of constraints is applied for the end of the prediction horizon. The set of allowable states specified by the constraint at the end of the prediction horizon is commonly termed the *terminal set*.

Clearly, if some linear, unconstrained controller $(u_i - u_{ref,i}) = K(x_i - x_{ref,i})$ is able to keep the state within the terminal set for all future time and for all possible disturbances, without violating any input constraints, then there will also exist a feasible solution to the MPC problem for all future times (provided the MPC problem is feasible initially).

Normally, the controller K used to calculate the terminal set will be the LQ controller for the same weights Q and P that are used in the MPC formulation. For time n to $n + j$, it is relatively straight forward to use the model equation (5.3) to express the states for $i > n$ and plant inputs for $i \geq n$ in terms of the predicted state x_n, predicted input and state references $u_{ref,i}$ and $x_{ref,i}$ and predicted future disturbances d_i. Thus, constraints in states and inputs for $i \geq n$ can be expressed as constraints on x_n.

Many of these state constraints at time n representing state or input constraints in the interval $n \leq i \leq n + j$ may be redundant. One would ideally like to remove

redundant constraints to ensure that the optimization problem is as small as possible. This can be done using the procedure described in Appendix 2. However, in applications where references ($u_{ref,i}$ and $x_{ref,i}$) or disturbances d_i vary, one will have to only remove constraints that are *always* redundant, i.e., constraints that are redundant for all conceivable values for $u_{ref,i}$, $x_{ref,i}$ and d_i, for $i \geq n$.

This is not as hopeless as it may seem. The constraints are linear, and this allows us to check for redundancy only at the extreme values of the variables. Furthermore, the prediction horizon is also commonly chosen sufficiently long for the plant to approach steady state, and thus it is reasonable to assume that $u_{ref,i} = u_{ref,n}$, $x_{ref,i} = x_{ref,n}$ and $d_i = d_n$, for $i \geq n$. This will reduce the number of variables that need to be considered.

Furthermore, if the control is supposed to remove offset at steady state, the references have to be consistent, i.e., at steady state (denoted by subscript ss), input $u_{ref,ss}$ and disturbance d_{ss} must result in the state $x_{ref,ss}$. Normally, one would consider d_{ss} and $x_{ref,ss}$ as independent variables, and $u_{ref,ss}$ as a dependent variable[4]. Thus, constraints should be checked for redundancy at all combinations of extreme values of d_{ss} and $x_{ref,ss}$.

Many control problems are formulated based on the assumption that the reference values for states and inputs are zero, and reference changes are implemented by 'shifting the origin' for the deviation variables. However, the constraints are typically independent of the references, and shifting the origin will result in a corresponding shift in the constraints (relative to the origin). Thus, shifting the origin does not remove the problem of variable references when constraints have to be considered.

5.2.5 Finding the terminal constraint set

As explained above, we want the predicted state at $i = n$ to lie within a set within which the state feedback controller does not violate any constraints, and the state feedback controller should be able to keep the state within that set for all $i > n$. Ideally, we would like to identify the largest such set in the state space, since this leads to the largest feasible region for a given prediction horizon n.

This set is known as the *maximal output admissible set*, often denoted $\mathcal{O}_\infty$. The properties and the determination of $\mathcal{O}_\infty$ are studied by Gilbert and Tan [GT91]. We will assume that the state constraints when considering only the timestep at the end of the prediction horizon in isolation, together with the state constraints implied by the input constraints when using state feedback, constitute a closed and bounded polyhedron in the state space, and that the origin is in the interior of this polyhedron. Operation arbitrarily far from the origin is of no practical interest, and if the assumption above is not fulfilled it is therefore possible to add very lax state constraints to fulfill the assumption. This allows us to use the results of [GT91] for rather straight forward determination of the terminal constraint set.

[4]Calculating consistent steady state references for given disturbances is the task of the *target calculation*, addressed in Section 5.8

Let $\mathcal{O}_t$ denote the set in the state space for which the constraints are feasible over t time steps using the state feedback controller. Obviously, $\mathcal{O}_\infty \subseteq \mathcal{O}_{t+1} \subseteq \mathcal{O}_t$. We will use the following results from [GT91]:

R1 $\mathcal{O}_\infty$ is closed and bounded (and is convex due to the linearity of the constraints).

R2 $\mathcal{O}_\infty$ is *finitely determined* if $\mathcal{O}_\infty = \mathcal{O}_t$ for finite t. For the cases studied here, $\mathcal{O}_\infty$ is finitely determined by construction.

R3 If $\mathcal{O}_t = \mathcal{O}_{t+1}$ then $\mathcal{O}_\infty = \mathcal{O}_t$.

This leads to the following algorithm for determination of $\mathcal{O}_\infty$:
Algorithm 1. Maximal Output Admissible Set.

1. Set $t = 0$, and let $\mathcal{O}_0$ be described by the constraints that apply for timestep $k = n$. The constraints considered should be both the state constraints, and the constraints on the states implied by the input constraints, due to the use of the state feedback controller. That is, let

$$M_{n'} x_n + N_{n'} u_n \leq G_{n'}$$

represent the state and input constraints we wish to impose at the end of the prediction horizon - without consideration of any additional constraints included in the terminal constraint set in order to ensure recursive feasibility. Substituting for the state feedback controller, this can be expressed as

$$(M_{n'} + N_{n'} K) x_n \leq G_{n'} + M_{n'} x_{ref} + N_{n'} K u_{ref}$$

As noted above, if this set of constraints in the state space does not constitute a bounded set, we can add additional lax state constraints to make the set bounded.

2. Increment the time index t, and express the constraints at time t in terms of x_n, using the system model (5.3) and the equation for the state feedback controller.

3. Remove any redundant constraints for time t. If all constraints for time index t are redundant, $\mathcal{O}_{t-1} = \mathcal{O}_t$, and hence $\mathcal{O}_\infty = \mathcal{O}_{t-1}$. Stop. Otherwise, augment the set of constraints describing $\mathcal{O}_{t-1}$ by the non-redundant constraints for time t to define $\mathcal{O}_t$. Go to Step 2.

Due to R2 above, this algorithm will terminate in finite time for the problems considered here. Checking for redundancy of constraints is also straight forward for linear systems subject to linear inequality constraints, as explained in Appendix 2.

For problems where references or disturbances may vary, it is necessary to verify the redundancy of the constraints for all combinations of extreme values of these variables, as explained in the preceding subsection. The determination of $\mathcal{O}_\infty$ for systems with disturbances has been addressed in [KG95].

5.2.6 Feasible region and prediction horizon

It was explained above that in order to guarantee closed loop stability, we will want the state at time x_n to lie within the maximal output admissible set $\mathcal{O}_\infty$. The feasible region for an MPC controller is therefore the set of states from which the state can be brought to $\mathcal{O}_\infty$ in n steps, without violating any constraints. The feasible region for a given n and given $\mathcal{O}_\infty$ can be found using Fourier-Motzkin elimination (Appendix 1), as noted in [KM00]. However, the Fourier-Motzkin procedure produces a number of redundant constraints which subsequently has to be removed. To minimize this problem, it is recommended to start from a prediction horizon $n = 0$ (i.e., the feasible region $= \mathcal{O}_\infty$) and gradually increment the prediction horizon, and remove redundant constraints along the way. Efficient calculation of feasible sets for MPC is furter described in [SOH11].

5.3 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 solid 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 [GM86a] 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. [HLM93] 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. Indeed, the MPC formulation in

(5.6) and (5.6) can easily be modified such that the step response coefficients appear in the intermediate calculations, by formulating the MPC problem using the outputs y_k and the change of inputs $\Delta u_k = u_k - u_{k-1}$ instead of the state x_k and the input u_k. The step response coefficients will then occur when relating the outputs y to the change in inputs Δu.

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.

Also, a step response model typically contains many more than the minimum number of parameters necessary to describe the plant dynamics. The direct identification of step response coefficients is therefore not recommended. It would be better to identify a minimal order state space model (or possibly a low order transfer function model), which usually contains fewer parameters and whose identification is therefore likely to be less sensitive to measurement noise. The identified state space model may later be converted to a step response model if this is required either in the MPC formulation or for discussing plant behavior with operators.

5.4 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 design of state estimators or observers is itself a vast area, and is the subject of numerous books. Furthermore, this is an area that has seen a lot of interesting developments recently. No attempt will therefore be made at giving a comprehensive treatment of this subject. Instead, a short description of techniques that are particularly relevant for MPC applications will be given - but readers are certainly encouraged to obtain more thorough insight elsewhere.

5.4.1 Bias update

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 applicable to asymptotically stable systems, and may result in poor control performance for systems with slow disturbance 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.

Note that the bias update is a simplistic way of estimating a disturbance at the output, see Section 5.5.3.

5.4.2 Kalman filter and Extended Kalman Filters

The Kalman filter is probably the model updating technique of choice for the 'purist', as it is 'optimal' in the sense of minimizing the variance of the estimation error for linear systems subject to Gaussian noise[5].

[5]The use of 'inverted commas' around *purist* and *optimal* should not be interpreted as any disregard of control theory. One should, however, keep in mind that most real-life systems are not linear, and that Gaussian noise cannot capture all the observed differences between model predictions and actual observations. Despite these reservations, the Kalman filter has proven valuable in numerous applications.

In order to present the Kalman filter equations, some nomenclature must be introduced:

$\hat{x}_{k\|k-n}$	The n step ahead prediction of the state at time $k-n$.
$\hat{x}_{k\|k-1}$	The 1 step ahead prediction of the state at time $k-1$, i.e., the best estimate of the state at time k using information available up to and including time $k-1$ (also known as the *a priori* estimate).
$\hat{x}_{k\|k}$	The estimate of the state at time k, accounting for information available up to and including time k (also known as the *a posteriori* estimate).
w_k	State excitation noise at time k, assumed to be normally distributed with zero mean, and to have no correlation between values at different times k.
v_k	Measurement noise at time k, also assumed to be normally distributed with zero mean, without correlation in time.
W	Variance of the state exitation noise w.
V	Variance of the measurement noise v.
$\Pi_{k\|k-n}$	Variance in the state estimate for time k, when accounting for information up to and including time $k-n$.
$\Pi_{k\|k}$	Variance in the state estimate for time k, when accounting for information up to and including time k.
$\Pi_0 = \Pi_{0\|0}$	Variance in initial state estimate (given or estimated).

The state excitation noise and measurement noise are included in the plant model as follows:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Ew_k \\ y_k &= Cx_k + v_k \end{aligned} \tag{5.24}$$

The Kalman filter equations are then given by (see, e.g., [AEBH69]):

$$\hat{x}_{k+1|k} = A\hat{x}_{k|k} + Bu_k \tag{5.25}$$

$$\Pi_{k+1|k} = A\Pi_{k|k}A^T + EWE^T \tag{5.26}$$

$$\Pi_{k+1|k+1} = \Pi_{k+1|k} - \Pi_{k+1|k}C^T(C\Pi_{k+1|k}C^T + V)^{-1}C\Pi_{k+1|k} \tag{5.27}$$

$$\tag{5.28}$$

When the measurement y_{k+1} is obtained, this is used to update the state estimate:

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K_{k+1}(y_{k+1} - C\hat{x}_{k+1|k}) \tag{5.29}$$

where K_{k+1} is the Kalman filter gain at time $k+1$, and is given by

$$K_{k+1} = \Pi_{k+1|k+1}C^TV^{-1} \tag{5.30}$$

From (5.26) we see that the uncertainty (represented by the variance of the state estimate) in stable states reduces with time, and that the uncertainty for unstable

states increase. Similarly, the same equation tells us that the state excitation noise increases uncertainty. Equation (5.27) shows that the uncertainty is reduced by taking new measurements, but the reduction in uncertainty is small if the measurement noise is large. All this does of course agree with intuition.

Provided some technical assumptions are met (like detectability - all unstable states must show up in the measurements), the variances will converge to steady values, which may be found by setting $\Pi_{k+1|k} = \Pi_{k|k-1}$ and $\Pi_{k+1|k+1} = \Pi_{k|k}$. Equations (5.26, 5.27) then give

$$\Pi_{k+1|k} = A\Pi_{k+1|k}A^T + A\Pi_{k+1|k}C^T(V + C\Pi_{k+1|k}C^T)^{-1}C\Pi_{k+1|k}A^T + EWE^T \tag{5.31}$$

and the corresponding steady state value of $\Pi_{k|k}$ can be found from (5.27), and the steady state Kalman gain from (5.30).

Although it is natural to assume that the state estimates are more uncertain initially, it is quite common to ignore the transient behaviour described by (5.26, 5.27), and only use the steady state solution to the Kalman filter. Software for calculating the steady state Kalman filter is readily available, (5.31) is cumbersome and difficult to solve by hand for systems with more than one state.

5.4.2.1 *Augmenting a disturbance description*

In many applications, assuming disturbances (represented by the state excitation noise, w), to be a sequence of zero mean, normally distributed, independent impulses (a 'white noise' description) is a poor representation of how disturbances actually enter the system. Often, there is strong temporal correlation in how disturbances affect system states and outputs. Such disturbances may be modelled by augmenting states representing the slow disturbance dynamics to the plant model. A good representation of disturbance dynamics is often a crucial element in achieving good closed-loop control performance.

A Kalman filter using an augmented disturbance description is often termed an Augmented Kalman Filter (AKF). In section 5.5.3 an example of a state space model, augmented with integrating states to represent disturbances both at the plant inlet and at the plant outlet, is shown in (5.94 - 5.96). When augmenting the model with integrating states, it is important that the augmented model is detectable. This point is further elaborated in section 5.5.3.

5.4.2.2 *The Extended Kalman Filter*

The Extended Kalman Filter (EKF) is an extension of the Kalman filter to non-linear systems. Although this extension seems quite natural and sensible, it is nevertheless somewhat ad hoc.

We start from a non-linear plant model, with additive measurement noise:

$$x_{k+1} = f(x_k, u_k, w_k) \tag{5.32}$$

$$y_k = h(x_k, u_k) + v_k \tag{5.33}$$

Equation (5.32) is used directly (assuming $w_k = 0$) to calculate $\hat{x}_{k+1|k}$. Similarly, (5.33) is used to calculate $\hat{y}_{k+1|k}$ (using $v_{k+1} = 0$, $\hat{x}_{k+1|k}$, and u_k)[6]. The value of $\hat{y}_{k+1|k}$ then enters instead of $C\hat{x}_{k+1|k}$ in (5.29). On the other hand, the propagation of estimate variances (5.26, 5.27) and calculation of the Kalman filter gain (5.30) are done with local linearizations of the nonlinear model. Thus, we use:

$$A_k = \left.\frac{\partial f}{\partial x}\right|_{w_k=0,\hat{x}_{k|k},u_k} \tag{5.34}$$

$$B_k = \left.\frac{\partial f}{\partial u}\right|_{w_k=0,\hat{x}_{k|k},u_k} \tag{5.35}$$

$$E_k = \left.\frac{\partial f}{\partial w}\right|_{w_k=0,\hat{x}_{k|k},u_k} \tag{5.36}$$

$$C_{k+1} = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_{k+1|k},u_k} \tag{5.37}$$

The EKF is commonly used for state estimation for nonlinear plants, and often performs well if the linearization is a fairly accurate approximation to the non-linear system over a single time step.

5.4.2.3 The Iterated Extended Kalman Filter The Iterated Extended Kalman Filter (IEKF) is and attempt to enhance the ability of the EKF to handle non-linearity. We note that C_{k+1} in (5.37) is obtained by linearizing around the *a priori* state estimate $\hat{x}_{k+1|k}$. If the system is strongly non-linear, the resulting value of C_{k+1} may therefore be inaccurate, and a more accurate linearized measurement equation may be obtained by linearizing around the *a posteriori* estimate $\hat{x}_{k+1|k+1}$ - once that estimate is available. Further iterations would allow forther improvements in state estimation accuracy.

To present the IEKF, we will need a second subscript on several of the matrices in the EKF formulation, as well as the *a posteriori* state estimate. This second subscript represents the iteration number (at time $k + 1$), with iteration number 0 representing the initial EKF calculations. Thus, from the initial EKF calculations we have $\hat{x}_{k+1|k,0} = h(\hat{x}_{k|k,N}, u_k)$ and $\Pi_{k+1|k}$, where N is the number of iterations of the IEKF at each timestep. For iteration i at time $k + 1$ the IEKF calculations then

[6]Normally, the state estimate is updated *before* a new input is calculated, and therefore the input which is applied when the measurement y_{k+1} is obtained is actually u_k (assuming that a 'zero order hold' is used).

proceed as follows:

$$
\begin{aligned}
C_{k+1,i} &= \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_{k+1|k+1,i-1},u_k} && (5.38)\\
K_{k+1|i} &= \Pi_{k+1|k} C_{k+1,i}^T \left(C_{k+1,i}\Pi_{k+1|k}C_{k+1,i}^T + V\right)^{-1} && (5.39)\\
\Pi_{k+1|k+1,i} &= (I - K_{k+1,i}C_{k+1,i})\Pi_{k+1|k} && (5.40)\\
\hat{x}_{k+1|k+1,i} &= \hat{x}_{k+1|k} && (5.41)\\
&+ K_{k+1,i}\left[y_{k+1} - h(\hat{x}_{k+1|k+1,i-1}, u_k) - C_{k+1,i}(\hat{x}_{k+1|k} - \hat{x}_{k+1|k+1,i-1})\right]
\end{aligned}
$$

The calculations proceed a predetermined number of iterations, or terminate when the change in state estimate between subsequent iterations is sufficiently small. At that point, after N iterations, one specifies

$$
\begin{aligned}
\hat{x}_{k+1|k+1} &= \hat{x}_{k+1|k+1,N}\\
\Pi_{k+1|k+1} &= \Pi_{k+1|k+1,N}
\end{aligned}
$$

which allows initiating the IEKF at time $k+2$ using the ordinary EKF. Although it is no general rule, it is often found that most of the improvement in the state estimate is achieved with a low number of iterations in the IEKF - often only one iteration after the EKF calculations.

5.4.3 Unscented Kalman filter

The Unscented Kalman Filter is a more recent modification to the Kalman filter, to better handle nonlinear models. The UKF avoids using a local linearization of the nonlinear model, but instead uses the model directly to propagate state estimates and (approximations of) probability distributions forward in time. Although not many industrial applications are reported, it seem that the UKF compares well with the more common EKF, in particular when the nonlinearities are pronounced. The presentation of the UKF in this note is based on Simon [Sim06], who gives an accessible introduction to both traditional state estimation and more recent developments in the area, and includes extensive references to the state estimation literature.

For simplicity of presentation, we assume that both the state excitation noise w and the measurement noise v enter the equations linearly, i.e.

$$
\begin{aligned}
x_{k+1} &= f(x_k, u_k) + w_k && (5.42)\\
y_k &= h(x_k) + v_k && (5.43)
\end{aligned}
$$

The noises w and v are both assumed to be zero mean, normally distributed, with known covariances W and V, respectively. Let n denote the number of states in the model (the dimension of the state vector x).

The UKF is initialized with known (or assumed) initial values for the mean value of the state $x_{0|0}$ and the state covariance $\Pi_{0|0}$.

The UKF then proceeds as follows:

- *Propagate the mean state estimate from time $k-1$ to time k.* Instead of propagating the mean value $\hat{x}_{k-1|k-1}$ directly through the system dynamics, $2n$ perturbed state values are perturbed, to better capture how the system non-linearity affects the mean.

  1. Select the perturbed states as follows:

$$\hat{x}^{(i)}_{k-1} = \hat{x}_{k-1|k-1} + \tilde{x}^{(i)} \qquad i = 1, \ldots, 2n \tag{5.44}$$

$$\tilde{x}^{(i)} = \left(\sqrt{n\Pi_{k-1|k-1}}\right)^T_i \qquad i = 1, \ldots, n \tag{5.45}$$

$$\tilde{x}^{(n+i)} = -\left(\sqrt{n\Pi_{k-1|k-1}}\right)^T_i \qquad i = 1, \ldots, n \tag{5.46}$$

     where $\left(\sqrt{n\Pi}\right)_i$ denotes the i'th row of the matrix square root of $n\Pi$, defined such that $(\sqrt{n\Pi})^T(\sqrt{n\Pi}) = n\Pi$. The matrix square root may be calculated by the Matlab functions `sqrtm` or `chol`[7]. These perturbed state values $\hat{x}^{(i)}$ are often termed *sigma points*.

  2. Propagate each sigma point through the system dynamics:

$$\hat{x}^{(i)}_{k|k-1} = f(\hat{x}^{(i)}_{k-1}, u_{k-1}) \tag{5.47}$$

  3. Combine the points $\hat{x}^{(i)}_{k|k-1}$ to obtain the *a priori* state estimate:

$$x_{k|k-1} = \frac{1}{2n}\sum_{i=1}^{2n} \hat{x}^{(i)}_{k|k-1} \tag{5.48}$$

- *Calculate the a priori state covariance estimate:*

$$\Pi_{k|k-1} = \frac{1}{2n}\sum_{i=1}^{2n}\left(\hat{x}^{(i)}_{k|k-1} - x_{k|k-1}\right)\left(\hat{x}^{(i)}_{k|k-1} - x_{k|k-1}\right)^T + W \tag{5.49}$$

- *Implement the measurement equation.*

  1. Determine new sigma points around x_{k_k-1}:[8]

$$\hat{x}^{(i)}_{k} = x_{k|k-1} + \tilde{x}^{(i)} \qquad i = 1, \ldots, 2n \tag{5.50}$$

$$\tilde{x}^{(i)} = \left(\sqrt{n\Pi_{k|k-1}}\right)^T_i \qquad i = 1, \ldots, n \tag{5.51}$$

$$\tilde{x}^{(n+i)} = -\left(\sqrt{n\Pi_{k|k-1}}\right)^T_i \qquad i = 1, \ldots, n \tag{5.52}$$

[7] The matrix square root is not uniquely defined (even for the positive definite covariance matrices considered here), and the two functions may therefore give different results. This is thought to be of little consequence here.

[8] This step may be omitted, and the propagated sigma points $\hat{x}^{(i)}_{k|k-1}$ calculated above used instead, if reducing the computational load is essential.

2. Pass each of the new sigma sigma points through the measurement equation:

$$\hat{y}_k^{(i)} = h(\hat{x}_k^{(i)}) \tag{5.53}$$

3. Calculate the predicted measurement at time k:

$$\hat{y}_k = \frac{1}{2n}\sum_{i=1}^{2n} \hat{y}_k^{(i)} \tag{5.54}$$

- *Estimate the measurement covariance:*

$$\Pi_{y,k} = \frac{1}{2n}\sum_{i=1}^{2n} \left(\hat{y}_k^{(i)} - \hat{y}_k\right)\left(\hat{y}_k^{(i)} - \hat{y}_k\right)^T + V \tag{5.55}$$

- *Estimate the cross covariance between the state estimate $\hat{x}_{k|k-1}$ and the measurement estimate $\hat{y}_k$:*

$$\Pi_{xy,k} = \frac{1}{2n}\sum_{i=1}^{2n} \left(\hat{x}_{k|k-1}^{(i)} - x_{k|k-1}\right)\left(\hat{y}_k^{(i)} - \hat{y}_k\right)^T \tag{5.56}$$

The *a posteriori* state estimate and covariance are now obtained from

$$\begin{aligned} K_k &= \Pi_{xy,k}\Pi_{y,k}^{-1} && (5.57)\\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k(y_k - \hat{y}_k) && (5.58)\\ \Pi_{k|k} &= \Pi_{k|k-1} - K_k\Pi_{y,k}K_k^T && (5.59) \end{aligned}$$

Remark: Note that the UKF applies also to time-varying systems. Both the system dynamics $f(\cdot)$, the measurement equation $h(\cdot)$ and the noise covariances W and V may be time varying (as long as they are known).

Many feedback systems are characterized by continuous-time system dynamics and discrete-time control and estimation. For such systems, so-called 'hybrid' EKFs have been developed, see, e.g., Simon [Sim06]. Note that for the UKF the functions $f(\cdot)$ and $h(\cdot)$ need not be explicitly given as discrete-time functions - they may just as well result (implicitly) from the integration of ordinary differential equations. It is therefore rather straight forward to apply the UKF also to continuous-time systems with discrete-time estimation and control.

In some cases the state excitation noise and the measurement noise may enter non-linearly, i.e., we have

$$\begin{aligned} x_{k+1} &= f(x_k, u_k, w_k) && (5.60)\\ y_k &= h(x_k, v_k) && (5.61) \end{aligned}$$

In such cases, the state vector x can be augmented with the noise vectors, giving

$$x_k^a = \begin{bmatrix} x_k \\ w_k \\ v_k \end{bmatrix} \tag{5.62}$$

$$\hat{x}_{0|0}^a = \begin{bmatrix} \hat{x}_{0|0} \\ 0 \\ 0 \end{bmatrix} \tag{5.63}$$

$$\Pi_{0|0}^a = \begin{bmatrix} \Pi_{0|0} & 0 & 0 \\ 0 & W & 0 \\ 0 & 0 & V \end{bmatrix} \tag{5.64}$$

Thus the UKF procedure described above can be used. Note, however, that the state excitation noise w_k and the measurement noise v_k are now accounted for when calculating the sigma points. Therefore W should not be added when calculating the *a priori* covariance estimate nor should V be added when calculating the measurement covariance.

The IEKF and UKF are both modifications of the (E)KF for the purpose of improved handling of nonlinearity. Another such modification is the second-order EKF (see, e.g., [Sim06]). This author is not aware of systematic comparisons of performance and computational requirements for these state estimation methods. Clearly, the UKF can be computationally rather demanding, if propagating the sigma points through (5.42) is demanding. This can occur, e.g., if $f(\cdot)$ in (5.42) results implicitly from the integration of a high order, stiff continuous-time model. However, for such problems, the rigorous propagation of the covariance matrix Π for a hybrid (continuous - discrete) EKF is also likely to be demanding.

5.4.4 Receding Horizon Estimation

Receding Horizon Estimation (RHE, a.k.a. Moving Horizon Estimation, MHE) is inspired by the success of MPC in control problems where constraints are important.

There are also many estimation problems where knowledge about the plant is easily formulated as constraints, and where such constraints will improve on the plant knowledge that is captured by the model alone. An opinion commonly held in academia seems to be that a sufficiently detailed plant model will capture all relevant constraints. Whereas this may be true (and often relatively straight forward to capture) in a simulation model, the way models are used in estimation may often destroy such model features. Two examples:

- The EKF uses a local linearization of the plant model, and the state update may easily result in infeasible state estimates.
- When propagating probability distributions for the UKF, the states are perturbed. These perturbed states may be infeasible.

There does exist approaches for ensuring feasible state estimates both for the EKF and the UKF, usually involving the 'projection' of the state estimate onto a feasible region of the state space. However, it may be better to embed the knowledge about what state estimates are possible directly into the state estimation. In such cases, RHE seems to be an obvious choice.

We assume as before that the state excitation noise and measurement noise are zero mean, independent and normally distributed, with covariances W and V, respectively. We also assume that an estimate $\hat{x}_0$ of the initial state is available, with a known covariance Π_0.

At time k, a natural formulation of the state estimation problem would then be to solve

$$\min_{\tilde{\mathbf{x}},\mathbf{w},\mathbf{v}} \left((\hat{x}_0 - \tilde{x}_0)^T \Pi_0^{-1} (\hat{x}_0 - \tilde{x}_0) + \sum_{i=1}^{k} v_i^T V^{-1} v_i + w_{i-1}^T W^{-1} w_{i-1} \right) \tag{5.65}$$

subject to constraints

$$\begin{aligned}
\hat{x}_0 &\quad \text{given} \\
y_i &\quad \text{given}; \qquad i = 1, \cdots, k \\
u_i &\quad \text{given}; \qquad i = 0, \cdots, k-1 \\
y_i &= C\tilde{x}_i + v_i; \qquad i = 1, \cdots, k && (5.66) \\
\tilde{x}_{i+1} &= A\tilde{x}_i + Bu_i + Ew_i; \qquad i = 0, \cdots, k-1 && (5.67) \\
X_L \leq \tilde{x}_i &\leq X_U; \qquad i = 0, \cdots, k && (5.68)
\end{aligned}$$

Here

$$\begin{aligned}
\tilde{\mathbf{x}} &= \begin{bmatrix} \tilde{x}_0^T & \cdots & \tilde{x}_k^T \end{bmatrix}^T \\
\mathbf{w} &= \begin{bmatrix} w_0^T & \cdots & w_{k-1}^T \end{bmatrix}^T \\
\mathbf{v} &= \begin{bmatrix} v_1^T & \cdots & v_k^T \end{bmatrix}^T
\end{aligned} \tag{5.69}$$

One may also put constraints explicitly on w_i and v_i. Note, however, that when both w_i and v_i (or $\tilde{x}_i$ and v_i) are constrained, outliers in measurements, etc., may result in an infeasible optimization problem.

The optimization problem above is called a 'Full Information' problem, at each step in time it accounts for all the information that is available at that time. Converting the Full Information problem to a standard QP should not be difficult, following the lines of what was done above for the MPC formulation. However, one problem is apparent: the size of the optimization problem grows without bounds as time progresses. The typical way of handling this problem, is to consider only a 'window' in the recent past in the optimization problem. The information available from the time

before the start of the current window is accounted for by a weight on the given state estimate at the beginning of the window.

Below, the problem formulation for a fixed window length of N timesteps is presented. Following what is conventional in the literature, the time indices on the variables are changed to reflect 'standing at time $t = k$ and looking backwards in time', rather than 'standing at time $t = 0$ and looking forward in time'. This gives the following problem formulation:

$$\begin{aligned} \min_{\tilde{\mathbf{x}},\mathbf{w},\mathbf{v}} \quad & (\hat{x}_{k-N} - \tilde{x}_{k-N})^T S(\hat{x}_{k-N} - \tilde{x}_{k-N}) \\ + \quad & \sum_{i=1}^{N} \left(v_{k-N+i}^T V^{-1} v_{k-N+i} + w_{k-N-1+i}^T W^{-1} w_{k-N-1+i} \right) \end{aligned} \tag{5.70}$$

subject to constraints

$$\hat{x}_{k-N} \quad \text{given}$$

$$y_{k-N+i} \quad \text{given}; \quad i = 1, \cdots, N$$

$$u_{k-N+i} \quad \text{given}; \quad i = 0, \cdots, N-1$$

$$y_{k-N+i} = C\tilde{x}_{k-N+i} + v_{k-N+i}; \quad i = 1, \cdots, N \tag{5.71}$$

$$\tilde{x}_{k-N+i+1} = A\tilde{x}_{k-N+i} + Bu_{k-N+i} + Ew_{k-N+i}; \quad i = 0, \cdots, N \tag{5.72}$$

$$X_L \leq \tilde{x}_{k-N+i} \leq X_U; \quad i = 0, \cdots, N \tag{5.73}$$

Clearly the definitions of $\tilde{\mathbf{x}}$, $\mathbf{w}$ and $\mathbf{v}$ need to be modified:

$$\tilde{\mathbf{x}} = \begin{bmatrix} \tilde{x}_{k-N}^T & \cdots & \tilde{x}_k^T \end{bmatrix}^T \tag{5.74}$$

$$\mathbf{w} = \begin{bmatrix} w_{k-N}^T & \cdots & w_{k-1}^T \end{bmatrix}^T \tag{5.75}$$

$$\mathbf{v} = \begin{bmatrix} v_{k-N+1}^T & \cdots & v_k^T \end{bmatrix}^T \tag{5.76}$$

$$\tag{5.77}$$

Note that

- The problem formulation above reflects the situation where, at each timestep, the state estimation is performed after receiving new measurements, before the MPC calculations are performed. Thus, the MPC calculations are performed with the *a posteriori* state estimate as a initial condition. To reduce computational delay before a new manipulate variable is available, one may instead in the MPC use the *a priori* state estimate as the initial condition - and at each timestep perform the MPC calculations before the state estimation. This may be particularly relevant for some nonlinear MPC problems, where the model at each timestep is linearized around a predicted future state and input trajectory. Using the *a priori* state estimate in the MPC allows the linearization and

subsequent problem formulation to be performed 'at the end of the previous timestep' rather than before solving the MPC optimization 'at the start of the new timestep'.

- In the problem formulation above, the effect of v_0 is assumed accounted for in the estimate $\hat{x}_0$, and v_0 therefore does not enter the optimization problem.
- Likewise, no effect of w_k can be observed before time $k+1$, and w_k therefore does not enter the optimization problem.
- In the MPC formulation, the state constraints represent undesirable operating conditions. In the estimation formulation, however, the state constraints represent *impossible* (or highly improbable) operating conditions - typically constraints such as *'the concentration of any chemical component cannot be negative'*. That is, the state constraints typically are not the same in MPC and RHE. If an operating condition is undesirable, it is important to get away from that operating condition as quickly as possible. Therefore, the RHE must be able to detect such an operating condition - and the state constraint introduced in the MPC to avoid the undesirable operating condition therefore should not be included in the RHE.

5.4.4.1 The arrival cost

In the RHE formulation above, the term

$$(\hat{x}_{k-N} - \tilde{x}_{k-N})^T S(\hat{x}_{k-N} - \tilde{x}_{k-N})$$

accounts for the information that has been available about the system *before the start of the estimation window*. This term is often called the *arrival cost*. The ideal arrival cost would make the fixed window length problem in (5.70-5.73) identical to the Full Information problem in (5.65-5.68). In general, we are not able to determine such an arrival cost. The exception is the linear, unconstrained case, where the Kalman filter can provide us with the arrival cost. However, the arrival cost also depends on how information is passed between subsequent timesteps of the RHE, which will be further explained in the next two subsections.

5.4.4.2 The filtering formulation of RHE

In the filtering formulation of the RHE, we use the estimate

$$\hat{x}_{k-N} = \tilde{x}_{k-N|x-N} \tag{5.78}$$

That is, $\hat{x}_{k-N}$ is the (*a posteriori*) estimate obtained the first time the time instant $k-N$ was included in the estimation window, and is based only on information available at time $k-N$. With this formulation, we use

$$S = \Pi^{-1}_{k-N|k-N} \tag{5.79}$$

where $\Pi_{k-N|k-N}$ is the *a posteriori* estimate covariance at time $k-N$, as obtained by a (possibly extended) Kalman Filter.

5.4.4.3 The smoothing formulation of RHE In the smoothing formulation, we use instead the most recent estimate of x_{k-N}. Thus, at time k we use

$$\hat{x}_{k-N} = \tilde{x}_{k-N|k-1} \tag{5.80}$$

This means that the estimate $\hat{x}_{k-N}$ is 'smoothed' (and improved) using measurements obtained *after* time $k-N$. In this case, it has been shown (see, e.g., [RRL01]), that the arrival cost should consist of two terms. The first term represents the uncertainty (covariance) of the estimate $\hat{x}_{k-N}$, represented by $\Pi_{k-N|k-1}$. The second term is added to prevent the information in $y_{k-N}, \cdots, y_{k-1}$ to be used twice (both in $\hat{x}_{k-N}$ and in the RHE calculations at time k).

To calculate the second term, we need the covariance of the estimate of the measurement sequence $Y_{N-1} = \begin{bmatrix} y_{k-N+1}^T & \cdots & y_{k-1}^T \end{bmatrix}^T$, given x_{k-N}.

Manipulating the model equations, we get

$$\begin{aligned} Y_{N-1} &= \begin{bmatrix} CA \\ CA^2 \\ \vdots \\ CA^{N-2} \\ CA^{N-1} \end{bmatrix} x_{k-N} \\ &+ \begin{bmatrix} 0 & 0 & \cdots & 0 & CB \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & CB & \cdots & CA^{N-3}B \\ 0 & CB & \cdots & CA^{N-3}B & CA^{N-2}B \end{bmatrix} \begin{bmatrix} u_{k-1} \\ u_{k-2} \\ \vdots \\ u_{k-N+1} \\ u_{k-N} \end{bmatrix} \\ &+ \begin{bmatrix} 0 & 0 & \cdots & 0 & CE \\ 0 & 0 & \ddots & CE & CAE \\ \vdots & \ddots & \cdots & \ddots & \vdots \\ 0 & CE & \cdots & CA^{N-3}E & CA^{N-2}E \end{bmatrix} \begin{bmatrix} w_{k-1} \\ w_{k-2} \\ \vdots \\ w_{k-N+1} \\ w_{k-N} \end{bmatrix} \\ &+ \begin{bmatrix} 0 & I_{(N-1)\cdot n_y \times (N-1)\cdot n_y} \end{bmatrix} \begin{bmatrix} v_k \\ v_{k-1} \\ \vdots \\ v_{k-N+1} \end{bmatrix} \end{aligned} \tag{5.81}$$

Noting that Y_{N-1} is independent of v_k, this may be reformulated as

$$Y_{N-1} - \mathcal{O}_{N-1} x_{k-N} - \tilde{B}\mathbf{u} = \tilde{E}\mathbf{w} + \tilde{I}\mathbf{v}_{k-1} \tag{5.82}$$

where

$$\tilde{I} = \left[I_{(N-1)\cdot n_y \times (N-1)\cdot n_y} \quad 0_{(N-1)n_y \times \cdot n_y} \right]; \quad \mathbf{v}_{k-1} = \begin{bmatrix} v_{k-1} \\ \vdots \\ v_{k-N} \end{bmatrix}$$

The variance of the left hand side of (5.82) (for a given x_{k-N}) and fixed $\mathbf{u}$ is therefore

$$S_2^{-1} = \tilde{E}\tilde{W}\tilde{E}^T + \tilde{I}\tilde{V}\tilde{I}^T \tag{5.83}$$

where $\tilde{W} = \mathrm{diag}\{W\}$ and $\tilde{V} = \mathrm{diag}\{V\}$.

The above expression clarifies some ambiguities in the expression in [RRL01]. Next, we need to account for the fact that the inputs u, although known, depend on the noises w and v. To express this dependency, we need to account for feedback in both control and estimation. An explicit formulation of MPC and RHE would enable accounting for the constraints active at each timestep. However, the explicit solution often is not available, and the formulation would become both complex and time-varying. Instead, we will assume that a 'conventional' QP-based MPC formulation is in use, which when constraints are not active corresponds to (an easily computable) LQ-optimal controller K. Similarly, the state estimation will be represented by the steady-state Kalman filter gain L.

The plant model, together with the (unconstrained) control and estimation, then yields

$$\begin{aligned} x_{k+1|k+1} &= Ax_{k|k} + Bu_k + L(y_k - Cx_{k|k}) + w_k \\ &= (A + BK)x_{k|k} + Lv_k + w_k \end{aligned} \tag{5.84}$$

Starting from a given value of x_{k-N}, we then obtain

$$\begin{aligned} u_{k-N} &= Kx_{k-N} \\ u_{k-N+1} &= K(A+BK)x_{k-N} + KLv_k + Kw_k \\ u_{k-N+2} &= K(A+BK)^2 x_{k-N} + K(A+BK)Lv_k + KLv_{k+1} + K(A+BK)w_k + Kw_{k+1} \\ u_{k-N+i} &= K(A+BK)^i x_{k-N} \\ &+ K\begin{bmatrix} I & (A+BK) & \cdots & (A+BK)^{i-1} \end{bmatrix} \begin{bmatrix} Lv_{k-N+i} + w_{k-N+i} \\ \vdots \\ Lv_{k-N} + w_{k-N} \end{bmatrix} \end{aligned} \tag{5.85}$$

Thus, we get

$$\begin{aligned}
\begin{bmatrix} u_{k-1} \\ u_{k-2} \\ \vdots \\ u_{k-N} \end{bmatrix} &= \tilde{K} \begin{bmatrix} (A+BK)^{N-1} \\ (A+BK)^{N-2} \\ \vdots \\ (A+BK) \\ I \end{bmatrix} x_{k-N} \\
&+ \tilde{K} \begin{bmatrix} I & (A+BK) & \cdots & (A+BK)^{N-2} & (A+BK)^{N-1} \\ 0 & I & (A+BK) & \cdots & (A+BK)^{N-2} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & I & (A+BK) \\ 0 & 0 & \cdots & 0 & I \end{bmatrix} \\
&\times \begin{bmatrix} Lv_{k-1} + w_{k-1} \\ \vdots \\ Lv_{k-N} + w_{k-N} \end{bmatrix} \\
&= \tilde{K} A_K x_{k-N} + \tilde{K} B_K \mathbf{w} + \tilde{K} B_K \tilde{L} \mathbf{v}_{k-1} \qquad (5.86)
\end{aligned}$$

Substituting (5.86) into (5.82) we obtain

$$Y_{N-1} - \left(\mathcal{O}_{N-1} + \tilde{K} A_K \right) x_{k-N} = \left(\tilde{E} + \tilde{K} B_K \right) \mathbf{w} + \left(\tilde{I} + \tilde{K} B_K \tilde{L} \right) \mathbf{v}_{k-1} \qquad (5.87)$$

This corresponds to the arrival cost

$$\begin{aligned}
\Gamma(\tilde{x}_{k-N}) &= (\hat{x}_{k-N} - \tilde{x}_{k-N})^T S_1 (\hat{x}_{k-N} - \tilde{x}_{k-N}) \qquad (5.88) \\
&- \left(Y_{N-1} - \left(\mathcal{O}_{N-1} + \tilde{K} A_K \right) \tilde{x}_{k-N} \right)^T S_2 \left(Y_{N-1} - \left(\mathcal{O}_{N-1} + \tilde{K} A_K \right) \tilde{x}_{k-N} \right)
\end{aligned}$$

with

$$\begin{aligned}
S_1^{-1} &= \Pi_{k-N|k-1} \qquad (5.89) \\
S_2^{-1} &= \left(\tilde{E} + \tilde{K} B_K \right) \tilde{W} \left(\tilde{E} + \tilde{K} B_K \right)^T \\
&+ \left(\tilde{I} + \tilde{K} B_K \tilde{L} \right) \tilde{V} \left(\tilde{I} + \tilde{K} B_K \tilde{L} \right)^T \qquad (5.90)
\end{aligned}$$

In order to account for how the input u depends on the noises w and v, the expression for S_2 in (5.90) should be used instead of the expression in (5.83). Note, however, that the expression in (5.90) is also based on rather idealized assumptions. To obtain the smoothed covariance $\Pi_{k-N|k-1}$ we must first propagate the Kalman filter covariances *forward* in time to obtain $\Pi_{k-N+i|k-N+i}$ and $\Pi_{k-N+i|k-N+i-1}$.

The smoothed covariance is then obtained by propagating *backwards* from $k-1$ to $k-N$ using the following relationships [AEBH69]:

$$\Pi_{T-i|T} = \Pi_{T-i|T-i} - Z_{T-i}(\Pi_{T-i+1|T-i} - \Pi_{T-i+1|T})Z_{T-i}^T \quad (5.91)$$

$$Z_{T-i} = \Pi_{T-i|T-i}A^T\Pi_{T-i+1|T-i}^{-1} \quad (5.92)$$

starting with $\Pi_{k-1|k-1}$ and $T = k-1$.

5.4.5 Concluding comments on state estimation

It is clearly impossible to cover all relevant formulations of state estimators in a chapter of this book. Other relevant and interesting estimator types include

- The second order EKF[Sim06], mentioned briefly above.
- The particle filter [Sim06]. This is essentially a Monte Carlo approach to state estimation, and may be particularly relevant for systems where the probability density function of the state estimate is multi-modal. For such systems it is clearly misleading to represent the state estimation accuracy using the state estimate covariance only.
- The Ensemble Kalman Filter (EnKF), [Eve94, Eve03], a modification of the Kalman filter for applications to systems of very high order, such as meteorological models and petroleum reservoir models.

In addition, there is also a large area of observer design for deterministic systems.

Another area that has not been addressed, is the practical implementation of the state estimators, both for computational efficiency and robustness. For all of these topics, the reader is referred to more specialized literature. The book by Simon [Sim06] is proposed as a good place to look for information and references to other works on many of these issues.

5.5 Disturbance handling and offset-free control

In most control applications, the ability to handle disturbances is important. In addition, differences between the model and the actual plant will lead to erroneous prediction, and hence to steady state offset - which is often considered unacceptable in process control.

Disturbances that can be measured directly, and whose effect on the controlled variables are known, can be handled by feedforward, which is easily included in MPC. This is addressed briefly in the next subsection.

5.5.1 Feedforward from measured 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. Feedforward from measured disturbances is included in the MPC formulation above, through the term $\hat{B}_d\delta$ in (5.16), where δ represents the *present* and *future* disturbances. If no other information is available, it is usually assumed that the future disturbances are equal to the present disturbance. Control performance will of course be affected by the accuracy of this assumption. In some cases information from upstream units, or knowledge of planned production changes, can provide better information about future disturbances.

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.

5.5.2 Requirements for offset-free control

In order to achieve offset-free control, we require that

1. The model predictions are accurate at steady state, and
2. the desired steady state must correspond to the minimum of the objective function.

The first point is normally achieved by including integral action in the model updating functionality[9]. The second point is achieved by augmenting the plant model with integrators either at the input or at the output of the plant (or combinations thereof), or by recalculating state and input references to correspond to the desired output values and the measured or estimated disturbances. Recalculating state and input references is addressed in Section 5.8 below. Note, however, that one cannot have offset-free control in more outputs than the number of inputs, if there are more outputs than inputs one must therefore put a zero weight on the offset in some of the outputs.

5.5.3 Disturbance estimation and offset-free control

If offset-free control is desired, it is necessary to account for differences between the model and the actual plant. This can be done by estimating *unmeasured* disturbances affecting the plant. The 'bias update' is a simple way of doing this, but it is often desired to be able to account for more general disturbance dynamics. This is done by augmenting the plant model with additional states $d_{i,k}$ representing disturbances entering at the plant inputs, and $d_{o,k}$ representing disturbances entering at the plant

[9]It will hopefully become evident below that the bias update described in Section 5.4.1 is a special case of estimating a disturbance at the plant output.

output. The material in this section is based on [MB02], where a more complete description may be found.

After augmenting the model with states representing disturbances at the input and/or output, the state space model becomes

$$\begin{aligned} \tilde{x}_{k+1} &= \tilde{A}_k \tilde{x}_k + \tilde{B} u_k + \tilde{E} d_k \\ y_k &= \tilde{C} \tilde{x}_k + F d_k \end{aligned} \tag{5.93}$$

Here d_k represent *measured* disturbances, whereas the estimated disturbances are included in the augmented state vector $\tilde{x}$. The augmented state vector and the correspondingly modified state space matrices are given by

$$\tilde{x} = \begin{bmatrix} x \\ d_i \\ d_o \end{bmatrix} \tag{5.94}$$

$$\tilde{A} = \begin{bmatrix} A & E_i & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}; \qquad \tilde{B} = \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix}; \qquad \tilde{E} = \begin{bmatrix} E \\ 0 \\ 0 \end{bmatrix} \tag{5.95}$$

$$\tilde{C} = \begin{bmatrix} C & 0 & C_{do} \end{bmatrix} \tag{5.96}$$

Here the matrix E_i determines how the estimated input disturbances affect the (ordinary) model states. This model can be used for state estimation, using, e.g., a Kalman filter or Receding Horizon Estimation. Muske and Badgwell [MB02] show that it is always possible to choose E_i and C_{do} such that the augmented state space model is detectable, provided

- the original model $\{C, A\}$ is detectable, and
- the number of estimated disturbance states (the sum of the number of elements in d_i and d_o) is no larger than the number of independent measurements used for estimation.

Naturally, the matrices E_i and C_{do}, as well as the dimensions of d_i and d_o, should be chosen to reflect the observed disturbance dynamics as well as possible. However, unfortunate choices for E_i and C_{do} may make the augmented model undetectable.

If $\{C, A\}$ is detectable, detectability of the overall system is determined by

$$\mathrm{R}ank \begin{bmatrix} (I - A) & -E_i & 0 \\ C & 0 & C_{do} \end{bmatrix}$$

which should equal the number of states in the *augmented* model. From this is derived a few conditions for detectability of the augmented system:

- The augmented system $\{\tilde{C}, \tilde{A}\}$ is not detectable if E_i and/or C_{do} are not full column rank.

- The augmented system $\{\tilde{C}, \tilde{A}\}$ is not detectable if the number of disturbance states exceeds the number of linearly independent outputs.
- The augmented system $\{\tilde{C}, \tilde{A}\}$ is not detectable if the range of E_i contains an unobservable mode of $\{C, A\}$.
- The augmented system $\{\tilde{C}, \tilde{A}\}$ is not detectable if the range of C_{do} contains the output space spanned by an integrating mode of A.

If a detectable augmented state-space model is used for estimation, it is straight forward to use a model of the type (5.94-5.96) in the MPC formulation. while the output disturbances d_o affect the output directly without affecting the states. In addition, one must ensure that the state references x_{ref} and manipulated variable references u_{ref} are consistent at steady state with the steady state (measured and estimated) disturbances and the input and output targets specified by higher levels of the operational hierarchy. This is further addressed in the section on *Target calculation* below. If the references are consistent at steady state, and the system is stable in closed loop, the disturbance estimation scheme described above will result in offset-free control at steady state[10].

A potential problem with estimating disturbances at the plant output is that it can result in poor control performance. It implicitly assumes that the effects of disturbances is modelled well as steps in the measured output. In many applications, disturbances show dynamics over a significant timescale - typically the same timescale as for the manipulated variables. That is, disturbances often enter at the plant inputs rather than at the plant outputs. Good performance for MPC requires the effects of disturbances to be modelled well. For disturbances entering at the plant inputs and a plant with slow dynamics, erroenously estimating the disturbances at the output will lead to poor performance in the face of disturbances.

Note that the augmented disturbance states are *uncontrollable*, and hence it does not make sense to assign a weight to them in the MPC formulation. Similarly, the model used for calculating the terminal weight must be a minimal model (without the augmented disturbance states), as otherwise the Riccati equation will have no solution.

Example 5.5.1 (Effect of estimating disturbances at the input or output.) *We will here investigate the effects of estimating disturbances at the input or output of the plant. This is done by comparing the results of with identical tuning parameters for the MPC, and identical noise variances for the Kalman filter. The system under study is the 5-state distillation column model in [SP05]. The system has 2 inputs, 2 outputs, and 2 physical disturbances. These disturbances act through the states of the plant (*i.e.*, at the plant input). Furthermore, the system dynamics are quite slow.*

- *In Case 1 the disturbances are estimated at the output of the plant.*

[10]The integrating disturbance models ensure that there is no steady state error in the predicted outputs, while consistent target calculation ensures that the minimum of the MPC objective function corresponds to inputs/states that achieve the specified output references.

- *In Case 2 the disturbances are estimated at the input of the plant.*

Both simulations start from steady state, and at time $t = 50$ *a disturbance* $d = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$ *hits the plant. The results are given in deviation variables. In Case 1, the Kalman filter doesn't estimate the actual disturbances (since the disturbances are modelled entering the wrong place in the plant). Instead, the Kalman filter estimates output disturbances that has the same effect on the measurements as the disturbances actually hitting the plant. However, the output disturbances are modelled without any dynamics, so the Kalman filter has no way of knowing that the effects of the disturbance will continue to increase in the future, and at each timestep the MPC assumes that the effect of the disturbances will not change in the future. The net effect on the closed loop control are modest offsets in the measurements that are removed very slowly. The actual measurements as well as the prior and posterior estimates of the outputs are shown in Fig. 5.1. The output disturbances estimated by the Kalman filter are given in Fig. 5.2. It is clear that these disturbance estimates develop slowly.*

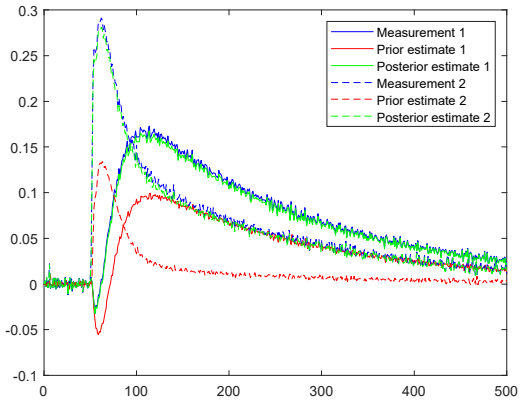

Figure 5.1: Measured and estimated outputs in Case 1.

In Case 2, the disturbances are estimated where they actually enter the plant, and the MPC therefore has a model of the true dynamics for the effects on the controlled outputs. The MPC is therefore quickly able to compensate for the disturbances, as shown in Fig. 5.3. The disturbance estimates also quickly converge to the true values, as seen in Fig. 5.4.

An alternative to re-calculating state and input references is to augment the plant model with integrators at either the plant input or the plant output. This is explained next.

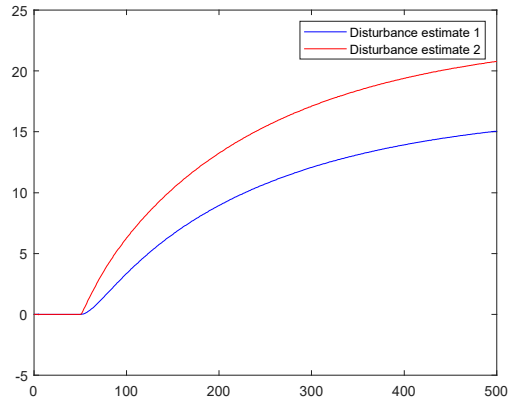

Figure 5.2: Estimated output disturbances in Case 1.

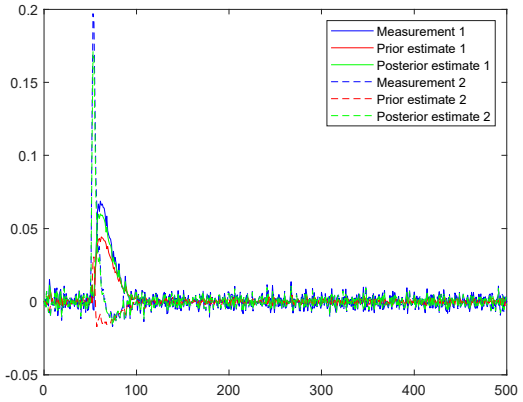

Figure 5.3: Measured and estimated outputs in Case 2.

5.5.4 Augmenting the model with integrators at the plant input

Augmenting the model with integrators at the plant input is a common way to introduce integral action in MPC controllers. This essentially corresponds to 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 in Section 5.2, using the model

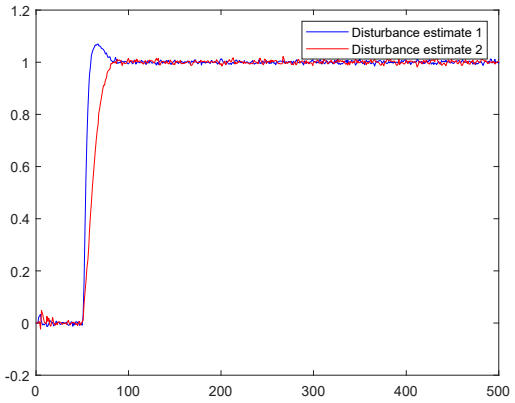

Figure 5.4: Estimated output disturbances in Case 2.

$$\begin{aligned} \widetilde{x}_{k+1} &= \begin{bmatrix} x_{k+1} \\ u_k \end{bmatrix} = \widetilde{A}\widetilde{x}_k + \widetilde{B}\Delta u_k + \begin{bmatrix} E \\ 0 \end{bmatrix} d_k \\ y_k &= \widetilde{C}\widetilde{x}_k \end{aligned}$$

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

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

When combined with a model updating strategy that gives zero model error at steady state (such as the ones described above), this way of including integral action into the MPC formulation will guarantee zero steady state offset.

To have a stable closed-loop system, it is necessary to have at least as many feedback paths as integrators. When integral action is included in the way described above, this means that 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 copying some inputs into the measurement vector, and defining setpoints for these.

Note that for the augmented model (with inputs Δu), the references for these inputs should always be zero. Furthermore, the augmented states representing the actual inputs u should have a zero weight in the MPC formulation - unless an *ideal resting value* is defined for that specific input.

Conversely, when there are more outputs n_y than inputs n_u, the offset free control will only be achieved in n_u linear combinations of outputs, with offset remaining in $n_y - n_u$ linear combinations of outputs. That is, in this situation, we are likely

to achieve offset-free control in no individual output[11]. To handle this situation, it is better to augment the integrators at n_u (or fewer) outputs for which offset-free control is required.

5.5.5 Augmenting the model with integrators at the plant output

We here assume that we have n_y outputs and n_u inputs, with $n_u < n_y$, and that we require offset-free control in n_u (or fewer) outputs. We achieve this by augmenting the model with n_u (or fewer) integrators at the output of the plant, as described next. Start from a model of the form specified in (5.94-5.96), and assume that the measurements are ordered so that the outputs for which offset-free control is required appear first in the measurement vector. The following state space model will perform a (discrete-time) integration of the leading inputs, while all the inputs are passed through unchanged in the second partition of the output vector

$$\left[\begin{array}{c|c} A_I & B_I \\ \hline C_I & D_I \end{array}\right] = \left[\begin{array}{c|c} I & \begin{bmatrix} I & 0 \end{bmatrix} \\ \hline \begin{bmatrix} I \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ I & 0 \\ 0 & I \end{bmatrix} \end{array}\right] \tag{5.97}$$

where the dimensions of the identity matrices are hopefully obvious from context. The series connection with the model in (5.94-5.96), after introducing the desired output variable y_{do} for the outputs where offset-free control is required, the state equation of the augmented model becomes

$$\hat{x}_{k+1} = \begin{bmatrix} A_I & B_I\tilde{C} \\ 0 & \tilde{A} \end{bmatrix} \hat{x}_k + \begin{bmatrix} 0 \\ \tilde{B} \end{bmatrix} u_k + \begin{bmatrix} 0 \\ \tilde{E} \end{bmatrix} d_k - \begin{bmatrix} I \\ 0 \end{bmatrix} y_{do} \tag{5.98}$$

$$\hat{y}_k = \begin{bmatrix} C_I & D_I\tilde{C} \end{bmatrix} \hat{x}_k \tag{5.99}$$

where $\hat{x} = \begin{bmatrix} x_I & \tilde{x} \end{bmatrix}^T$, and $\hat{y}$ contain the integrated outputs on top and the physical measurements in the bottom. The reader should keep in mind that d_k refers to the *measured* disturbances (if any), the estimated disturbances are included as components of $\tilde{x}$.

Clearly, the desired output value y_{do} should be relative to the same origin for the deviation variables as used for the states, and with y_{do} entered this way, the state reference for the augmented states can be set to zero.

Example 5.5.2 (MPC for a Fluid Catalytic Cracker model.) *This example studies MPC for the Fluid Catalytic Cracking (FCC) model given in [HS93b]. The model*

[11] Careful design of the state weight Q to have zero weight in state directions corresponding to measurements for which we do *not* require offset-free control may also give offset-free control in the required outputs. However, this puts very high demands on the accuracy of the state space model.

has two states, two inputs (air flowrate and catalyst flowrate), and three outputs (riser outlet temperature, cyclone temperature, and regenerator temperature). There are four different disturbances defined in the model, entering partly at the plant input and partly at the plant outputs. However, it can be concluded that:

- *Although accurate modeling of how the disturbances enter the plant generally lead to better estimation performance, in this case there are four different disturbances and three measurements. A maximum of three disturbance states can therefore be augmented to the plant model, and it will therefore be impossible to model (all) the disturbances accurately.*
- *With only two states, no more than two disturbance states can be augmented at the plant input.*

So, in this case, the matrix E_i in (5.95) is chosen as the 2×2 identity matrix, and the third disturbance state is augmented at the cyclone temperature output.

The riser outlet and regenerator temperatures are considered the most important for plant yield, while for the cyclone temperature is is primarily important to stay under some maximal bound (not further addressed in this example). Therefore, since there can only be accurate control of two outputs with two inputs for control, it is chosen to augment integrators at the riser outlet and regenerator temperatures.

In the simulation, an unmeasured disturbance (a change in the coke production rate) enters at time $t = 50$. At $t = 500$ the setpoints for both the riser outlet temperature and the regenerator temperature are increased by $1K$. The results are shown in Fig. 5.5 and Fig. 5.6. Both disturbance rejection and reference following is found to be adequate for the two temperatures with augmented integrators, while the cyclone temperature drifts (in the absence of any active constraint). Observe that acceptable control performance is obtained, even though the MPC does not use the correct disturbance model (for reasons explained above).

5.5.6 MPC and integrator resetting

When augmenting the model with disturbances at the input or output to model how unmeasured disturbances enter the system, these augmented states will be uncontrollable, and therefore should have zero weight in the MPC objective function. On the other hand, the exact value of the disturbances are in general not known, and the corresponding states therefore need to be included in the state estimation. If the state augmentation is done correctly, the states will always be observable.

The situation is different for the augmented states at the input or output of the plant, that are augmented in order to provide integral action. These states exist only in the computer where the MPC is implemented, and there is therefore no uncertainty with regards to their correct values. Hence, these states need not be included in the state estimation, see also Section 6.3.6. However, this does not mean that these augmented states cannot be subject to optimization. The state values are available in the computer, and may therefore be set to any desirable value - in contrast to the states representing what goes on in the 'physical world', for which the measured or

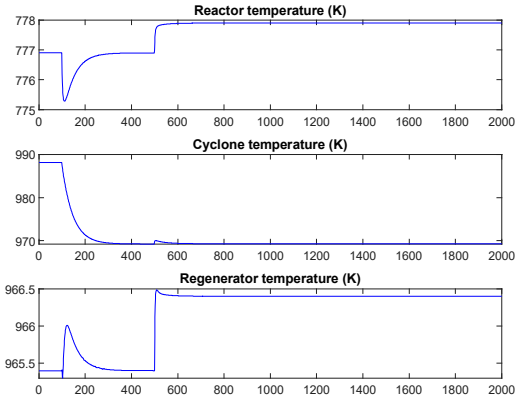

Figure 5.5: MPC for FCC model: resulting temperatures.

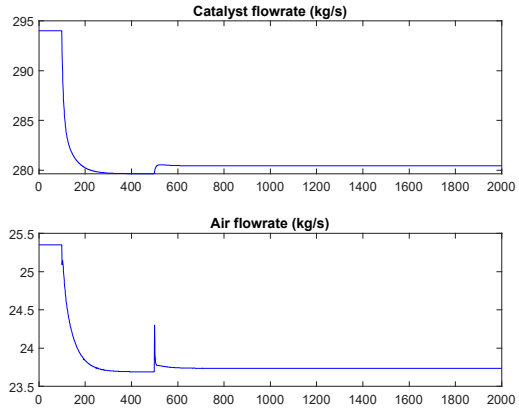

Figure 5.6: MPC for FCC model: input usage.

estimated values cannot be changed. In the following discussion we will assume that the states augmented to provide integral action are at the plant output, but a similar opportunity exists for optimizing the value of integrating state(s) at the plant input.

We see from (5.13) that the initial states x_0 enter linearly in the constraint equations. By now it should therefore be straight forward for the dilligent reader to split the initial state vector into to components

$$x_0 = \begin{bmatrix} x_{0p} \\ x_{0i} \end{bmatrix}$$

where x_{0p} represent the 'physical' states and x_{0i} represent states that are augmented to provide integral action, and add x_{0i} to the vector of variables that are optimized over. One may then add a term $(x_{0,i} - x_{0,i}^-)^T R(x_{0,i} - x_{0,i}^-)$ to the objective function (where $R \geq 0$). Here $x_{0,i}$ refers to the value of the integrating state that the MPC optimizes over, while $x_{0,i}^-$ is the calculated value prior to optimization. The state values x_{0p} that are provided to the MPC optimization problem should be in deviation variables relative to the current desired operating point. That is, in (5.15), the term $A_0 x_0$ is replaced by

$$A_{0p}(x_{0p} - x_{ref}) + A_{0i}x_{0i}$$

where $A_0 = [A_{0p}\ A_{0i}]$ and x_{ref} is the desired value of the physical states for the current value of disturbances and the output reference y_{ref}.

Such integrator resetting has been proposed by several authors, both in the context of MPC, and for more classical control with augmented states to provide integral action. Wada [Wad15] addresses this in the context of MPC, and cites previous publications going back to the 1950's. Here we will illustrate on a simplified example from [Wad15].

Example 5.5.3 (MPC with integrator resetting) *Consider the simple one-state system*

$$\begin{aligned} x_{k+1} &= 1.025x_k + 0.05u_k \\ y_k &= x_k \end{aligned}$$

We augment this model with an integrator at the output, giving

$$A = \begin{bmatrix} 1.025 & 0 \\ 1 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0.05 \\ 0 \end{bmatrix}$$

Note that the physical state is measured directly, so there is no need for state estimation. Also, since this simple example has no constraints, and $y_k = x_k$ (for the physical state), we have $x_{ref} = y_{ref}$. The MPC is designed using the A and B matrices above, which corresponds to updating the augmented state according to

$$x_{i,k+1}^- = x_{i,k} + x_{p_k} - y_{ref,k}$$

The simulation is done with a prediction horizon of 10,

$$Q = \begin{bmatrix} 2 & 0 \\ 0 & 5 \end{bmatrix}, \quad P = 1,$$

and the corresponding terminal state weight S. For the case with integrator resetting, $R = 0.01$ is used.

A few comments are in order

- *Figure 5.7 shows that the response is quicker and without overshoot when integrator resetting is used.*

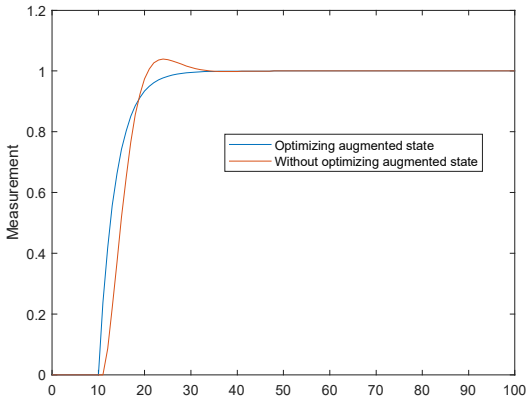

Figure 5.7: Effect of integrator resetting on measured output.

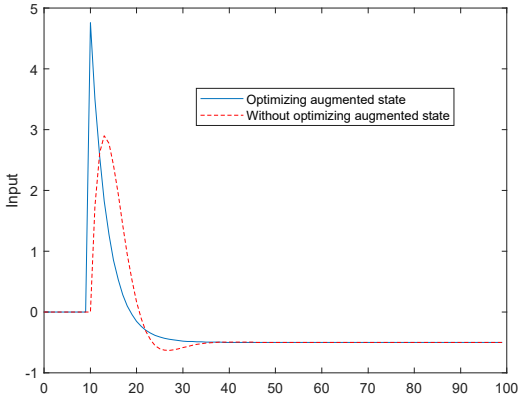

Figure 5.8: Input usage with and without integrator resetting.

- *From Fig. 5.8 we see that the input usage increases with integrator resetting. This increase (and also the performance improvement for the output) will depend on the size of R.*

- *If input constraints (or constraints on the change in input) are included, the effect of integrator resetting on the input must be accounted for. Such constraints limiting input usage may also reduce the benefit of integrator resetting.*

5.6 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 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 desirables 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. 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 con-

straints 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 priorities, 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. The importance of this point is dependent on the time available for calculation and the QP solver to be used - many QP solvers will handle semi-definite Hessian matrices well.

- 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. [Fle87]), and is discussed in the context of MPC in e.g. [dOB94, SR99, HB01]. A computational approach for designing penalty functions ensuring exact soft constraints is described in [HS14].

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 [ZM91]. Although this phenomenon probably is quite rare, it can easily be removed by using a soft constraint formulation for the output constraints [dOB94]. The following section will discuss closed loop stability with MPC controllers in a more general context.

5.7 Closed loop stability with MPC controllers

The objective function in Eq. (5.5) closely resembles that of discrete-time Linear Quadratic (LQ) - optimal control. For stabilizable and detectable[12] systems, infinite horizon LQ-optimal control is known to result in a stable closed loop system. Note 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* LQ-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 , 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

$$S - (A + BK)^T S(A + BK) = Q + K^T PK \tag{5.100}$$

Note that if one chooses to use the infinite horizon LQ-optimal controller, solving the Riccati equation gives both the controller K and the terminal state weight S:

$$S = A^T SA + Q - A^T SB(P + B^T SB)^{-1} B^T SA \tag{5.101}$$

and the corresponding controller is given by

$$K = -(B^T SB + P)^{-1} B^T SA$$

With a sufficiently large S, obtained as described above, the remaining requirement for obtaining closed loop stability is that constraints can be fulfilled over the infinite horizon. For the appropriately determined terminal constraint set, obtained as described above, this condition is fulfilled provided the constraints are feasible initially.

The above results on how the terminal cost and the terminal constraint set 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

[12] 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.

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. (5.6). 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.

5.8 Target calculation

It is common for MPC controllers 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 $u_{ref,\infty}$, taking account of any measured or estimated disturbances affecting the system. Many continuous plants operate around a constant desired steady state, and if integrators are augmented to the inputs and/or outputs to ensure offset-free control in the face of changing disturbances (c.f. Section 5.5.2), in may suffice to do the target calculation infrequently. Without such augmented states, target calculation must be performed regularly if offset-free control is required. Clearly, batch processes or cyclic processes should not maintain a constant steady state, and for such processes the targets over the prediction horizon will have to be obtained by other means not addressed here.

We will 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 ascertain that will actually be the case.

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

$$(I - A)x_{ref,\infty} = Bu_{ref,\infty} + \tilde{E}\tilde{d}_\infty \tag{5.102}$$

$$y_{ref} = Cx_{ref,\infty} + \tilde{F}\tilde{d}_\infty \tag{5.103}$$

Here y_{ref} is the desired steady state value of some variables, the desired values of which are determined by higher levels in the operational hierarchy[13].

[13]In general, the higher levels of the operational hierarchy may specify targets in terms of different measurements than those that are used for control/estimation at the supervisory control level. In such cases, the relationships between the variables used for supervisory control (including estimated output disturbances) and the variables for which targets are specified, will need to be modelled.

The disturbance variable vector $\tilde{d}_\infty$ is the expected/predicted/estimated steady state value of *all* disturbances affecting the process, i.e., it should contain the steady state values of measured disturbances d, estimated input disturbances d_i, and estimated output disturbances d_o. Thus,

$$\begin{aligned}\tilde{d}_\infty &= \begin{bmatrix} d_\infty \\ d_{i,\infty} \\ d_{o,\infty} \end{bmatrix} \\ \tilde{E} &= \begin{bmatrix} E & E_i & 0 \end{bmatrix} \\ \tilde{F} &= \begin{bmatrix} F & 0 & C_{do} \end{bmatrix}\end{aligned}$$

In the (rare) unconstrained case, and with as many inputs u as controlled outputs y, the state and input targets can be found from a simple matrix inversion

$$\begin{bmatrix} x_{ref,\infty} \\ u_{ref,\infty} \end{bmatrix} = \begin{bmatrix} -(I-A) & B \\ C & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 & -\tilde{E} \\ I & -\tilde{F} \end{bmatrix} \begin{bmatrix} y_{ref,\infty} \\ \tilde{d}_\infty \end{bmatrix} \tag{5.104}$$

$$= M^{-1} \begin{bmatrix} 0 & -\tilde{E} \\ I & -\tilde{F} \end{bmatrix} \begin{bmatrix} y_{ref,\infty} \\ \tilde{d}_\infty \end{bmatrix} \tag{5.105}$$

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 outputs 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 are 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 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 straight forward formulation will cast the target calculation as a QP problem:

$$\min_{x_{ref,\infty},u_{ref,\infty}} \quad \left(y_d - Cx_{ref,\infty} - \tilde{F}\tilde{d}_\infty\right)^T Q \left(y_d - Cx_{ref,\infty} - \tilde{F}\tilde{d}_\infty\right) \tag{5.106}$$

$$+(u_d - u_{ref,\infty})^T W (u_d - u_{ref,\infty}) \tag{5.107}$$

subject to given values for y_d, u_d and d_∞, the model equations Eq. (5.102) 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 [Mus97] 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. (5.6), but there may also be other constraints. Let us assume that all such constraints can be described by the inequality

$$\widehat{H} \begin{bmatrix} x_{ref,\infty} \\ u_{ref,\infty} \end{bmatrix} \geq \widehat{b} \tag{5.108}$$

Difficulties will arise whenever there is no feasible region in which the constraints of Eq. (5.102) and Eq. (5.108) 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. (5.102) 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 desirables 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. (5.106) we get

$$\min_{x_{ref,\infty},u_{ref,\infty},p} (y_d - Cx_{ref,\infty} - \tilde{F}\tilde{d}_\infty)^T Q (y_d - Cx_{ref,\infty} - \tilde{F}\tilde{d}_\infty)$$

$$+ (u_d - u_{ref,\infty})^T W (u_d - u_{ref,\infty}) + l^T p + p^T Z p \tag{5.109}$$

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. (5.102) are assumed to hold as before, whereas the constraints in Eq. (5.108) are modified to

$$\begin{aligned} \widehat{H}\begin{bmatrix} x_{ref,\infty} \\ u_{ref,\infty} \end{bmatrix} + \widehat{L}p &\geq \widehat{b} \\ p &\geq 0 \end{aligned} \tag{5.110}$$

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. (5.109) is also the optimal solution to Eq. (5.106), provided a feasible solution for Eq. (5.106) exists.

The target calculation formulation in (5.109 - 5.110) 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 [14], 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 $\begin{bmatrix} 0 & 0 & 1 & \cdots & 1 \end{bmatrix}^T$,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

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

subject to the constraints

[14] A series of QP problems may sometimes be preferable, if one wishes to distribute constraint violations between several constraints of the same importance. Using a QP formulation only affects the criterion functions of the following optimization problems, not the constraints.

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

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 [Vad00]. 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.

5.9 Speeding up MPC calculations

MPC is a computationally demanding controller type, which can limit its applicability for systems requiring high sampling rates. Techniques for speeding up MPC calculations may therefore be useful. Considering the wide range of MPC formulations and application areas, it is not surprising that there are many different approaches to speeding up the MPC calculations, such as

- Using an *explicit* MPC formulation [BMDP02, TJB03], where all possible optimization problems are solved at the design state. The result is a set of affine state feedback controllers, each of which is valid in a specific region of the state space (depending on what constraints are active in that region). The online calculations are reduced to a simple table search to identify the affine state feedback controller and the calculation of the corresponding manipulated variable value[15]. Currently, the size of MPC problems that can be handled using an explicit formulation is somewhat limited, due to very demanding offline computations and a large memory requirement for storing the resulting solution table.
- In MPC for nonlinear systems, one may use special tools for the efficient calculation of gradients. These are not very relevant for MPC for linear systems.
- One may use optimization solvers that utilize the structure of the problem to speed up calculations, see, *e.g.*,[RWR98]. These techniques are most applicable when both states and manipulated variables are kept as optimization variables, as the resulting QP problem is highly structured with sparse matrices.

Here, we will instead focus on techniques that can be applied using standard, off-the-shelf optimization solvers.

5.9.1 Warm-starting the optimization

Many optimization solvers will find the solution more quickly if one can provide a good guess at what the solution will be[16], so that the optimization solver can start its search close to the actual solution. In MPC, the solution from the last timestep is easily used to find such a good guess at the solution at the next timestep. The solution at time k provides the predicted sequence of manipulated variables

$$\mathbf{u}_k = \begin{bmatrix} u_{0,k} \\ u_{1,k} \\ \vdots \\ u_{N-1,k} \end{bmatrix}.$$

[15] An affine state feedback controller is of the form $u = Kx + k$, *i.e.*, with a constant term in addition to the linear state feedback.

[16] *Interior point solvers* will typically not benefit much from warm starting.

Assuming that a terminal controller $u_{N+i} = Kx_{N+i}, \quad i = 0, 1, \ldots$ is used in the MPC formulation, a reasonable guess for the solution at time $k+1$ would be obtained by simply removing the first input in the sequence $\mathbf{u}_k$ (the input that has already been applied), and adding the feedback from the predicted state at time $k + N$ at the end of the sequence, giving

$$\tilde{\mathbf{u}}_{k+1} = \begin{bmatrix} u_{1,k} \\ u_{2,k} \\ \vdots \\ u_{N-1,k} \\ Kx_{k+N|k} \end{bmatrix}.$$

Clearly, this idea is easily generalized to formulations where states are included among the optimization variables, or where the deviation from linear feedback is used as optimization variables.

The main problem with the warm start is initialization: what to do when starting up the MPC and we don't have a previous solution? One may then

- Require the plant to be at some known, 'calm' state when starting the MPC, for which a reasonable initial guess is available. For large plants, bringing it to such a known, 'calm' state may be very difficult (in particular without the help of an advanced controller), but there are many smaller control problems where this approach is reasonable.

- Terminate the optimization solver prematurely if the optimum is not found in the time available, and instead use a sub-optimal intermediate solution. It will then depend on the particular solver whether such an intermediate solution is even feasible. If the infeasibility occurs some timesteps into the future, one may be lucky and be 'saved' by improved optimization solutions at later timesteps. However, it is only when a feasible solution is found at the first timestep that we can guarantee feasibility at later timesteps and closed loop stability.

- Allow the optimization at the first timestep to take longer time. This will mean that the calculated manipulated variable will be implemented later than what was assumed in the design of the MPC. This could again jeopardize feasibility at later timesteps and thereby also closed loop stability.

Clearly, none of these approaches to initialization of the warm start is entirely satisfactory, but it will be highly case dependent how severe the indicated problems are.

5.9.2 Input blocking

The time to find a solution for most optimization problems is highly dependent on the number of degrees of freedom - more degrees of freedom requires longer solution times. For convex quadratic programming (which is most often used in MPC), the

solution time may grow linearly with the number of degrees of freedom for specially tailored QP solvers, but may grow with the cube of the number of degrees of freedom if a 'naive' approach to solving the QP is used [RWR98]. In either case, fewer degrees of freedom leads to faster QP solution, which is the motivation for *input blocking*.

Input blocking means that in the optimization formulation the input (manipulated variable) is held constant over several timesteps (while state constraints are usually imposed at each timestep). Thus, the number of degrees of freedom is reduced. The same approach may also be used when optimizing the deviation from linear state feedback - in this case it is the deviation from linear state feedback that is kept constant over several timesteps (this is sometimes called *offset blocking*).

Note that it is only in the optimization formulation that the input is kept constant over several timesteps. In operation the first element of the optimal input vector is implemented at each timestep, and the input may therefore change at every timestep.

In input blocking, it is common to have short 'blocks' in the near future, and gradually longer blocks far into the prediction horizon. Apparently, the reason for this is the expectation that most of the control action will be taken in the near future, while less aggressive control moves are made in the fare future, when it is expected that the state is approaching the desired operating conditions.

Input blocking (and offset blocking) ruins guarantees for both recursive feasibility and stability. This may or may not be a practical problem. Often, a blocking scheme that appears to work is found using simulation studies - but there are no guarantees that the same scheme will work with different disturbances or initial conditions than what was used in simulations.

In [CGKM07] Moving Window Blocking (MWB) is proposed. The key point in MWB is that the length of the last block varies from timestep to timestep, such that recursive feasibility and stability can be guaranteed, see the original publication for details.

5.9.3 Enlarging the terminal region

Recall the crucial role of the terminal region in proving stability of MPC. The terminal region is positively invariant for the system when using the terminal controller $u_{k+N+i} = Kx_{k+N+i}, \quad i \geq 0$, and no constraints are violated when using this terminal controller inside the terminal region. The feasible region is the region from which it is possible to reach the terminal region within the prediction horizon, while adhering to the state and input constraints. Obviously, it is important that the feasible region for the MPC is large enough to cover the states that are likely to occur in plant operation. A longer prediction horizon will thus typically result in a larger feasible region, since more timesteps can be used to reach the terminal region. However, a longer prediction horizon will mean more degrees of freedom in the optimization, and hence longer calculation times for the MPC. Conversely, if the terminal region is enlarged, the required feasible region may be obtained with a shorter prediction horizon. A method for enlarging the terminal region, due to Limon et al. [LAaC08], will be described next. The method is focused on enlarging the terminal set for refer-

ences that remain constant for a significant time. Hence, the MPC objective in (5.5) is re-stated for constant references

$$\begin{aligned} \min_{u} \quad f(x,u) \quad = \quad & \sum_{i=0}^{n-1} \{(x_i - x_{ref})^T Q (x_i - x_{ref}) \\ & + (u_i - u_{ref})^T P (u_i - u_{ref})^T \} \\ & + (x_n - x_{ref})^T S (x_n - x_{ref}) \end{aligned} \tag{5.111}$$

The expression of the corresponding constraints in (5.6) are modified by explicitly including the model equations, and explicitly stating that the terminal state should lie inside the maximal output admissible set $\mathcal{O}_\infty$:

$$\begin{aligned} x_0 &= \text{given} \\ x_{i+1} &= Ax_i + Bu_i \\ M_i x_i + N_i u_i &\leq G_i \quad \text{for } 0 \leq i \leq n-1 \\ x_n &\in \mathcal{O}_\infty \end{aligned} \tag{5.112}$$

In [LAaC08], the terminal set is enlarged by introducing a feasible steady state x_s and corresponding feasible input u_s. The MPC objective does not weigh the deviation from x_{ref} and u_{ref}, but rather the deviation from x_s and u_s. Then, an extra term is introduced in the objective function to penalize the difference between x_{ref} and x_s:

$$\begin{aligned} \min_{u,u_s,x_s} \quad f(x,u) \quad = \quad & \sum_{i=0}^{n-1} \{(x_i - x_s)^T Q (x_i - x_s) \\ & + (u_i - u_s)^T P (u_i - u_s)^T \} \\ & + (x_n - x_s)^T S (x_n - x_s) + V_T (x_{ref} - x_s) \end{aligned} \tag{5.113}$$

where V_T is some positive definite function of $(x_{ref} - x_s)$. A simple choice (which allows the optimization problem to be solved as a standard QP problem) would be a quadratic function $(x_{ref} - x_s)^T T (x_{ref} - x_s)$ for some positive definite matrix T. For simplicity, we assume that the state constraints are time invariant, and the corresponding constraints are

$$\begin{aligned} x_0 &= \text{given} \\ x_{i+1} &= Ax_i + Bu_i \\ x_s &= Ax_s + Bu_s \\ M_i x_i + N_i u_i &\leq G_i \quad \text{for } 0 \leq i \leq n-1 \\ x_n &\in \mathcal{O}^s_\infty, \;\; x_s \in \mathcal{O}^s_\infty \end{aligned} \tag{5.114}$$

The third equality constraint above ensures that x_s and u_s are a steady state with the corresponding input, while the two next inequalities ensure that x_s and u_s are

feasible. The terminal set is changed to $\mathcal{O}_\infty^s$, which is larger than $\mathcal{O}_\infty$ because the terminal controller uses x_s as a setpoint, i.e.,

$$u_{k+n+i} = K(x_{k+n+i} - x_s) + u_s; \quad i \geq 0. \tag{5.115}$$

To calculate the set $\mathcal{O}_\infty^s$, let us first parameterize the subspace that x_s and u_s must lie in to correspond to a steady state. From the model equations, this is expressed as[17]

$$\begin{bmatrix}(A-I) & B\end{bmatrix} \begin{bmatrix}x_s \\ u_s\end{bmatrix} = W \begin{bmatrix}x_s \\ u_s\end{bmatrix} = \begin{bmatrix}0\end{bmatrix} \tag{5.116}$$

Thus, x_s and u_s must lie in the right null-space of the matrix W. The matrix W has dimensions $n_x \times (n_x + n_u)$. Denote the rank of W as n_r. A convenient way of obtaining a basis for the right null-space of W is to perform a singular value decomposition of $W = U\Sigma V^H$. Let M be the last $n_x + n_u - n_r$ columns of the input singular vector matrix V. Thus, if

$$\begin{bmatrix}x_s \\ u_s\end{bmatrix} = M\theta \tag{5.117}$$

where θ is a vector of length $n_x + n_u - n_r$, the steady state condition (5.116) is fulfilled for all values of θ. Considering constant references, we might therefore describe the closed loop system behaviour after the end of the prediction horizon in the enlarged space of (x, θ), resulting in

$$\begin{bmatrix}x \\ \theta\end{bmatrix}_{k+1} = \begin{bmatrix}A+BK & BL \\ 0 & I\end{bmatrix} \begin{bmatrix}x \\ \theta\end{bmatrix}_k = A_W \begin{bmatrix}x_s \\ \theta\end{bmatrix}_k \tag{5.118}$$

where $L = \begin{bmatrix}-K & I\end{bmatrix} M$. Describe the state and input constraints as $H_x x \leq h_x, \quad H_u u \leq h_u$ Define the convex polyhedral set Γ_λ, given by the inequalities

$$\begin{bmatrix}H_x & 0 \\ 0 & H_u\end{bmatrix} \begin{bmatrix}I & 0 \\ K & L\end{bmatrix} A_w^i \begin{bmatrix}x \\ \theta\end{bmatrix} \leq \begin{bmatrix}h_x \\ h_u\end{bmatrix}; \quad i = 0, 1, ... \tag{5.119}$$

$$\begin{bmatrix}H_x & 0 \\ 0 & H_u\end{bmatrix} M\theta \leq \lambda \begin{bmatrix}h_x \\ h_u\end{bmatrix} \tag{5.120}$$

The set Γ_λ can be calculated as follows:

1. Define P_0 as the polyhedron consisting of inequalities (5.119) for $i = 0$ and (5.120).

[17] In [LAaC08] the authors also include the option of having x_s and u_s fulfilling some additional 'target output value'. This would add additional line(s) in (5.116). However, adding such output targets removes degrees of freedom for maximizing the terminal set, and is therefore not included here.

2. Increment i, and define S the polyhedron consisting of inequalities (5.119) for the new value of i.

3. Define the polyhedron P_1 as the intersection of polyhedra P_0 and S.

4. If $P_1 = P_0$, set $\Gamma_\lambda = P_1$ and terminate. Else set $P_0 = P_1$ and go to step 2.

These operations on polyhedra are simple to perform using appropriate software such as the MPT toolbox for Matlab. The largest possible invariant set is found for $\lambda = 1$, but unfortunately the procedure described above is not guaranteed to terminate in this case. Instead, we have to choose $\lambda = 1 - \epsilon$ for some small $\epsilon > 0$, and calculate the corresponding Γ_λ (in this case the calculations are guaranteed to terminate). The resulting Γ_λ will be an inner approximation to the maximal invariant set in (x, θ)-space, but for small values of ϵ the difference from Γ_1 will not be significant. The enlarged terminal set for use in MPC is found by the projection of Γ_λ onto x, i.e.,

$$\mathcal{O}_\infty^s = \mathtt{proj}_x \Gamma_\lambda$$

This projection may be conveniently done using software such as the MPT toolbox for Matlab - or using the Fourier-Motzkin elimination described in Appendix A.

5.10 Robustness of MPC controllers

The term *robustness*, when used about an MPC controller, can refer to several different aspects of controller functionality:

- Robust *feasibility*. The MPC optimization problem should remain recursively feasible in the face of model errors and disturbances.

- Robust *stability*. The system should remain closed loop stable in the face of model errors and disturbances. While for linear systems closed loop stability can be investigated without taking constraints into account, we remember from Section 4.9.4 that for constrained open loop unstable systems that disturbances can indeed affect closed loop stability.

- Robust *performance*. Beyond requiring robust stability, we would like the control performance to degrade 'gracefully' in the face of model errors and disturbances.

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 are numerous works addressing each of the issues above, and a comprehensive coverage would require an entire monograph. Of most interest here is robust stability, as this is the minimum requirement for the controller to be practically useful. A number of approaches have been proposed in the literature, including

- *Optimizing the worst-case system response*, as proposed by, *e.g.*, [ZM93]. This approach generally leads to very difficult *min max* optimization problems.

- *Optimization over feedback policies*, *i.e.*, optimizing over the control law instead of optimizing the input directly, as proposed by *e.g.*, [KBM96]. This also easily leads to complex online optimization formulations.

- *Robust MPC using tubes e.g.*, [LCRM04], where the MPC essentially only addresses the control of the nominal system, while an auxiliary controller ensures robustness by keeping the true system in a 'narrow tube' around the nominal system. Since the states of the true system are not necessarily the same as the states of the nominal system, and the auxiliary controller will use part of the range of movement for the inputs, the constraints on both states and inputs need to be tightened for the MPC of the nominal system. While the online computational requirements for this approach may be less demanding than the other two approaches, the design of the robust auxiliary controller goes beyond the scope of this book.

A more comprehensive presentation of robust MPC, with a focus on tube-based MPC, can be found in [RM09]. Here we will instead focus on simpler industrial approaches to robustness, while noting that in many practical cases the inherent robustness of feedback will provide the required robust stability and performance. A first step towards accounting for the robustness of feedback in the MPC formulation is this to optimize over the deviation from linear state feedback, as described above.

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. (5.1) with H and c from Eq. (5.19). We then want to minimize

$$f(v) = 0.5v^T(\widehat{B}^T\widehat{Q}\widehat{B} + \widehat{P})v + \chi_{dev}^T\widehat{A}^T\widehat{Q}\widehat{B}v$$

with respect to v. The solution to this minimization can be found analytically, since no constraints are assumed to be active. We get[18]

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

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*[19], 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:

[18]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.

[19]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*.

- The controller detects 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 approaches to 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 $\widehat{P}$.

Note that these two approaches are related, since changing $\widehat{B}$ by scaling inputs and states will lead to changed tuning (changed closed loop performance - also for the nominal model) if $\widehat{Q}$ and $\widehat{P}$ are kept unchanged.

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 these variables. In some cases this 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[20] 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. (5.7), 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

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

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 thresholding*, since we remove all singular values of $\widehat{B}$ that is smaller than some threshold. 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 thresholding 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. Also, one should take care not to remove control offsets in directions corresponding to open loop unstable modes. For this reason, singular value tresholding is usually applied only for open loop stable plants (or plants that have been stabilized by some lower-level controllers).

5.11 Using rigorous process models in MPC

Most 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.

To optimize directly on the rigorous model is not straight forward. Nevertheless, over recent years there have been significant advances in many aspects of relevance to this, including formulation of the optimization problem *per se*, efficient computation of derivatives, and preparing for efficient computations in the interval between sample times. For a comprehensive presentation of these issues the reader is referred to [RMD17].

The presentation below will only scratch the surface of how to use rigorous nonlinear models in MPC, with a focus on approaches that are relatively simple extensions of standard MPC for linear systems.

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)

$$\min_{\alpha} f(x,u) = \min_{\alpha} f(x_0, u_{ref} + \alpha v) \tag{5.121}$$

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 [Fle87]. When performing the minimization in Eq. (5.121) 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 the optimization converges, 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 particularily 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. 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 prediction horizon of 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 $(x_{ref} + \chi_{dev})$ 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, unless automatic differentiation routines are used, finding the required linearized models may itself be computationally burdensome. Linearizing around a trajectory can be combined with iterative optimization and line search as outlined above - which would further add to the computational burden.

5.12 Misconceptions, clarifications, and challenges

We wrap of the chapter on MPC by trying to clarify a few misconceptions that appear to be common, and then briefly discuss some challenges.

5.12.1 Misconceptions

5.12.1.1 MPC is not good for performance This clearly is not true in general - as in the constraint-free case MPC can be designed to be equivalent to LQG control (LQ control + Kalman filter). Although LQG is claimed not to have found much application in the process industries, it clearly can provide good performance. The idea that MPC does not provide good performance probably comes from the fact that the closed loop performance does not come from the MPC alone. We saw in section 5.5.3 the strong effect of the model update (disturbance estimation) on the control performance. It appears that the 'bias update' (equivalent to estimating disturbances at the plant output) is the most common model updating method used in industry. With slow disturbance dynamics the bias update will lead to slow disturbance rejection. This is a likely cause for the misconception that MPC does not provide good performance.

5.12.1.2 MPC requires very accurate models This is only partly true. The requirements for model accuracy increase as required closed loop bandwidth increases. If slow control is acceptable, one can often get away with first order + deadtime models, also for plants with more complex dynamics at higher frequencies. Before returning to academia, this author was involved in an MPC project for a crude oil distillation column [HMM97]. All responses were modelled as first order + deadtime, even though simple step tests showed damped oscillatory responses in some variables (clearly requiring at least two states for an accurate model). Nevertheless, the MPC drastically improved operation of the column, with standard deviation in controlled outputs being reduced by factors up to 24. It is quite likely that more accurate modelling and fine tuning could have further significantly reduced standard deviation in the controlled variables - but it is much less clear whether the improvement would justify the effort required. Anyway, this illustrates that acceptable control can be obtained with MPC using a quite inaccurate model, provided a realistic tuning is used.

5.12.1.3 MPC cannot prioritize input usage or constraint violations Sometimes it is desired to make full usage of a 'cheap' input before using an 'expensive' input. It is sometimes claimed this cannot be achieved with MPC. This is true only

for MPC with quadratic objective functions - which, admittedly, is the focus in this book. However, using an LP formulation (with a piecewise affine objective function), a strict priority of inputs can be achieved, as shown by [Dan21]. Similar effects can be achieved with (otherwise) quadratic objective functions, by using soft constraints on 'expensive' inputs, and having sufficiently large linear weights on the constraint violations in the objective function.

Similarly, it may sometimes be desirable to accept arbitrarily large violations of some constraints, and accept violations of other constraints only when feasibility of the constraint cannot be achieved with any magnitude of violation for the less important constraints. It is shown in [Vad00] that this can be achieved by solving an LP problem prior to solving the main MPC optimization problem, where the solution to LP problem provides the optimal relaxation (taking account of constraint priorities) of the constraints.

5.12.2 Challenges

Model Predictive Control has such wide scope that there will always be a number of challenges for research. Below, the more important challenges with respect to industrial application - as seen by the author - are briefly discussed.

5.12.2.1 Obtaining a plant model Obtaining a plant model is typically one of the more demanding tasks in an MPC project. This will typically involve experiments on the plant - which seldom is very popular with plant operating personnel. Note that a dynamic model is required, so operational data from only steady state operating conditions will not suffice - and one would be lucky to find data with sufficient excitation in all inputs in data historians.

One may choose to develop an entirely empirical model, based on experimental data alone, or to develop a (more) rigorous model from first principles modelling. Such rigorous dynamical models will often require substantial expertise and competent personnel. While competent personnel will also be required for empirical modelling, developing such models can often be quicker than going the 'rigorous route'. Rigorous models will also require plant experiments to determine parameter values that are unknown or uncertain, but often the required amount of experimentation will be less than for empirical modelling.

5.12.2.2 Maintenance Plants change with time - due to a number of factors including equipment wear, fouling, catalyst activity degradation, replacement of equipment, plant revamps, etc. In addition, product specifications and operational costs may change. The accumulation of such factors will sooner or later result in need for maintenance of the MPC controller - and such maintenance typically require higher competence than retuning PID loops. Many plants will require external assistance for such maintenance tasks.

If the model used by the MPC is based on a first principles model (that may be linearized before use in the MPC), such maintenance may be targeted to the modelling of what has changed in the plant. With empirical models, however, such targeted

maintenance is more difficult, and it may well be necessary to perform extensive new plant experiments to develop an updated model.

5.12.2.3 Capturing the desired behavior in the MPC design Capturing the desired behavior in the MPC design requires detailed knowledge about both the plant and MPC design. Whereas the need for detailed knowledge of the plant may be obvious, it appears to be a more common misconception that MPC is a 'one size fits all' type of product. It is not uncommon to see MPC designs that are not well suited to the problem at hand due to unfortunate constraint formulations, inability to achieve offset-free control, unfortunate objective function design - in addition to MPCs suffering from poor models or poorly maintained models. MPC is indeed a very flexible control design paradigm - but taking full advantage of all this flexibility does require quite a broad set of competences.

EXERCISES

5.1

a) Why is model updating necessary in MPC (usually at every timestep)?
b) What is meant by 'bias update'?
c) What advantages, limitations and disadvantages are associated with the use of 'bias update'?

5.2 Given a linear system

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

and the MPC formulation

$$\min_{u_0,\ldots,u_{n-1}} \sum_{i=0}^{n-1} x_{i+1}^T Q x_{i+1} + u_i^T R u_i$$

where the initial state x_0 is assumed given, and the linear model above is included as equality constraints in the formulation. In this problem, constraints other than the linear model need not be considered (as they are addressed in the exercise below). The MPC is implemented in the familiar receding horizon fashion, i.e., at each timestep $u_0, \ldots, u_{(n-1)}$ are calculated, but only the first element (u_0) is implemented on the plant, and the entire sequence of inputs is re-calculated in the next timestep.

a) Can MPC (with the formulation above, or with some alternative formulation) stabilize an open-loop unstable system?
b) What conditions would you put on the weights Q and R, and what modifications would you make in the objective function above, in order to guarantee closed loop stability?

5.3 Given an MPC problem for a system with the linear dynamics

$$x_{i+1} = Ax_i + Bu_i$$

where the following constraints should be fulfilled at every timestep:

$$Mx_i \leq m \quad (1)$$
$$Wu_i \leq w \quad (2)$$

In order to make an MPC problem recursively feasible , it is common to assume that unconstrained linear state feedback is applied after the end of the prediction horizon:

$$u_i = Kx_i, \text{ for } i \geq N$$

where N is the length of the prediction horizon and the controller K is known. You can assume that the system with the controller K is asymptotically stable provided input constraints are inactive. To ensure that the linear state feedback does not violate constraints at times beyond the prediction horizon, one must therefore express input constraints for times $i \geq N$, and state constraints for times $i > N$ in terms of x_N, the predicted state at N timesteps in the future.

a) Take a single state constraint $M_k x_{N+1} \leq m_k$, and express this in terms of the predicted state x_N.
b) Assuming all state and input constraints defined in (1) and (2) above are fulfilled for time $i = N$, explain how you can formulate a test to find out whether the additional constraint in a) is redundant.
c) Explain how repeated use of steps similar to a) and b) above leads to determining a terminal set, also known as the maximal output admissible set.

5.4 Given two different cost functions for MPC:

$$\min_{x,u} \sum_{i=0}^{N-1} (x_i^T Q x_i + u_i^T R u_i) + x_N^T S x_N \quad \text{(I)}$$
$$\min_{x,u} \sum_{i=0}^{N_p} x_i^T Q x_i + \sum_{i=0}^{N_u} u_i^T R u_i \quad \text{(II)}$$

In addition a process model is specified:

$$x_{i+1} = Ax_i + Bu_i$$

and you can assume that all relevant constraints in manipulated and controlled variables are defined, and that x_0 is available (either measured directly or from a good state estimator). For cost function (II) we can assume that $N_p > N_u$.

a) What more must be defined for the MPC problem using cost function (II) to be well posed?
b) Show or explain that the cost function (II) can always be expressed using (I).
c) A third cost function for MPC is given by

$$\min_{y,x,\Delta u} \sum_{i=0}^{N_p} y_i^T P y_i + \sum_{i=0}^{N_u} \Delta u_i^T R \Delta u_i \quad \text{(III)}$$

where $\Delta u_i = u_i - u_{i-1}$, and in addition to the process model, constraints, and x_0, the following measurement equation is given

$$y_i = Cx_i,$$

together with the manipulated variable $u_(- 1)$ used in the last timestep.

- Consider the case with two manipulated variables and three controlled variables. Will the cost function in c) enable removing steady state offset in all controlled variables?
- Consider the case with two controlled variables and three manipulated variables. Will the cost function in c) enable removing steady state offset in all controlled variables? Do you see any other problems with the formulation in c)?

5.5

An optimization problem for MPC may be formulated as

$$\begin{aligned} &\min_u \ u^T H u + x_0^T F u \\ &\text{s.t.} \\ &Gu \leq W + Mx_0 \end{aligned}$$

a) It is important that the MPC controller provides an input at each time step, and for that the optimization problem needs to be feasible. Indicate how you would modify the optimization problem formulation above, using a penalty function, to ensure that the problem is always feasible. You may (if necessary) assume that all constraints can be relaxed.

b) What are the two most common penalty function types used in MPC? What are their respective advantages and disadvantages?

c) Use of penalty functions make hard constraints soft (the term soft indicating that the constraint may be violated). What is an exact soft constraint?