Merging Optimization and Control

Bjarne Foss and Tor Aksel N. Heirung

September 4, 2013

Contents

1 Introduction

This note is a supplement to the course TTK4135 — Optimization and Control. Key learning outcomes of this course are the ability to formulate appropriate engineering problems as a mathematical optimization problem, and analyze and solve such optimization problems. Important optimization problem classes are presented and optimality conditions are discussed. Further, algorithms for solving optimization problems are treated including some information on relevant software. This material is for the most part covered by the optimization textbook Nocedal and Wright (2006).

The course TTK4135 — Optimization and Control also covers Model Predictive Control (MPC). Instead of introducing an MPC textbook like Rawlings and Mayne (2009), Goodwin et al. (2004), Rossiter (2003), or Maciejowski (2002) we have written this note. The rationale is to provide a smooth transition from optimization to MPC via open loop optimal control. This approach is not taken in any available textbook on MPC.

The outline of this note is as follows. Section 2 repeats key concepts in optimization before Section 3 treats optimization of discrete time dynamic systems. This section shows how optimization based on static models easily can be extended to include dynamic time discrete models. This section introduces two important concepts, dynamic optimization and open loop optimization. This provides a foundation for the optimal control section, in which the MPC concept is introduced. Section 4 discusses linear MPC, which is extensively used in practice. Linear quadratic control is an interesting case, which is treated separately. This section also includes material on state estimation, nonlinear MPC and comments related to practical use of MPC.

2 Optimization

This section presents relevant background on optimization. It first classifies optimization problems and discusses some basic properties. Subsequently solution methods and computational issues are discussed.

2.1 Classes of optimization problems

Mathematical optimization problems include three main components, an *objective function*, *decision variables* and *constraints*. The objective function is a scalar function which describes a property that we want to minimize or maximize. The decision variables may be real variables, integers or binary variables, or belong to other spaces like function spaces. In this note the decision variables will mostly be limited to the *Euclidean space*, i.e., vectors of real variables. Constraints are normally divided into two types, *equality constraints* and *inequality constraints*. This background allows us to define an optimization problem, the *nonlinear program (NLP)*.

$$\min_{z \in \mathbb{R}^n} f(z) \tag{1a}$$

subject to

$$c_i(z) = 0, \quad i \in \mathcal{E} \tag{1b}$$
$$c_i(z) \geq 0, \quad i \in \mathcal{I} \tag{1c}$$

Some further explanation is required. f takes an n-dimensional vector and projects it onto the real axis. Further, the decision variables are defined on a Euclidean space since $z \in \mathbb{R}^n$ which excludes integer or binary decision variables. $\mathcal{E}$ and $\mathcal{I}$ are disjunct index sets, i.e., $\mathcal{E} \cap \mathcal{I} = \emptyset$, for the constraints. Due to the constraints the decision variables z may be selected from a subset of $\mathbb{R}^n$. This subset is called the *feasible region* or the *feasible set* and is defined by the constraints c_i. More precisely the feasible set is defined by

$$\Omega = \Big\{ z \in \mathbb{R}^n \,\big|\, \big(c_i(z) = 0,\ i \in \mathcal{E}\big) \wedge \big(c_i(z) \geq 0,\ i \in \mathcal{I}\big) \Big\} \tag{2}$$

In some cases the feasible set may be further limited by the domain of f itself. A function like $\sqrt{\cdot}$ is only defined for positive arguments. Such cases are, however, not considered in this note.

Example 1 (Optimization problem and feasible area)
Consider the optimization problem

$$\min_{z \in \mathbb{R}^2} \ f(z) = z_1^2 + 3z_2 \tag{3a}$$
$$\text{s.t.} \ \ c_1(z) = z_1^2 + z_2^2 - 1 = 0 \tag{3b}$$
$$c_2(z) = z_1 + z_2 \geq 0 \tag{3c}$$

Here, we have that $\mathcal{E} = \{1\}$ and $\mathcal{I} = \{2\}$. The feasible area for this problem is

$$\begin{aligned} \Omega &= \Big\{ z \in \mathbb{R}^2 \,\big|\, c_1(z) = 0 \wedge c_2(z) \geq 0 \Big\} \\ &= \Big\{ z \in \mathbb{R}^2 \,\big|\, z_1^2 + z_2^2 = 1 \wedge z_1 + z_2 \geq 0 \Big\} \end{aligned} \tag{4}$$

The feasible area Ω is a half circle (not a half-disc) with radius 1, marked with red in Figure 1. $\triangle$

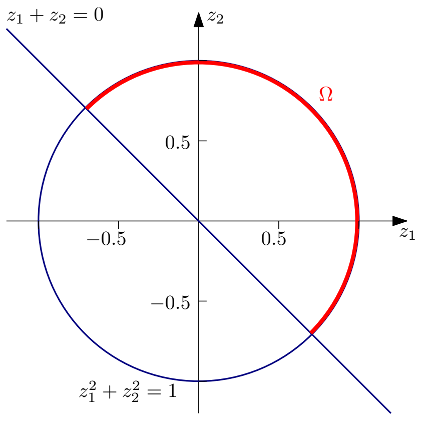


Figure 1: Feasible area (thick red curve) for Example 1.

The optimization problem in (1) is stated as a minimization problem. Some problems, where for instance the objective function represents profit, are clearly maximization problems. One should observe that any maximization problem can be translated into a minimization problem by observing that '$\max f(z)$' and '$\min -f(z)$' provide equal solutions apart from the opposite sign of the objective function value. The feasible set may be empty, i.e., $\Omega = \emptyset$. In this case there is no solution, or no *feasible points*, to the optimization problem in (1). $f(z)$ has to be bounded below for all feasible solutions meaning that $f(z) \neq -\infty$ for all $z \in \Omega$. This has no practical consequence since all reasonable engineering problems will be constrained such that the objective function remains bounded.

A point $z^* \in \Omega$ is the solution of (1) if

$$f(z^*) \leq f(z), \text{ for all } z \in \Omega$$

Such a point is a *global minimizer*. There may be several such minimizing points. In the case of only one single global minimizer the global minimizer is a *strict global minimizer*. A *local minimizer* z^* provides a minimum value within some neighborhood rather than the whole feasible region and is defined by

$$f(z^*) \leq f(z), \text{ for all } z \in \|z - z^*\| < \epsilon$$

where z can take values in a feasible open set about z^*. A *strict local minimizer* is defined by replacing $\leq$ with $<$ above.

A key question is how to identify a minimization point. Provided certain smoothness conditions and regularity conditions are satisfied the *Karush-Kuhn-Tucker (KKT) conditions* define necessary conditions for optimality. First, however, we need to define the *Lagrange function* and *active constraints*.

$$\mathcal{L}(z, \lambda) = f(z) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(z) \tag{5}$$

where λ_i are the *Lagrange multipliers*.

An active constraint is a constraint for which $c_i(z) = 0$, which implies that all equality constraints are active at a feasible point. An inequality constraint, however, may either be active or not active at a feasible point.

Since the KKT conditions is a key results in constrained optimization we state this result in some detail.

Theorem 1 (First order necessary conditions).
Assume that z^* is a local solution of (1) and that f and all c_i are differentiable and their derivatives are continuous. Further, assume that all the active constraint gradients are linearly independent at z^* (meaning that LICQ[1] *holds). Then there exists Lagrange multipliers λ_i^* for $i \in \mathcal{E} \cup \mathcal{I}$ such that the following conditions (called the KKT conditions) hold at (x^*, λ^*);*

$$\begin{aligned}
\nabla_z \mathcal{L}(z^*, \lambda^*) &= 0 && \text{(6a)}\\
c_i(z^*) &= 0, \quad i \in \mathcal{E} && \text{(6b)}\\
c_i(z^*) &\geq 0, \quad i \in \mathcal{I} && \text{(6c)}\\
\lambda_i^* &\geq 0, \quad i \in \mathcal{I} && \text{(6d)}\\
\lambda_i c_i(z^*) &= 0, \quad i \in \mathcal{I} && \text{(6e)}
\end{aligned}$$

The KKT conditions are necessary conditions for a local solution. Sufficient conditions, however, can be derived through second order conditions.

Theorem 2 (Second order sufficient conditions).
Suppose that for some feasible point $z^* \in \mathbb{R}^n$ there exists Lagrange multipliers λ^* such that the KKT conditions (6) *are satisfied, and that f and all c_i are twice differentiable and their derivatives are continuous. Assume also that*

$$\nabla_{zz} \mathcal{L}(z^*, \lambda^*) \succ 0 \tag{7}$$

Then z^* is a strict local solution of (1).[2]

$\nabla_{zz} \mathcal{L}(z^*, \lambda^*)$ needs only be positive in certain directions, more specifically in directions defined by the *critical cone*[3]. Hence, the theorem above can be relaxed by requiring $w^\top \nabla_{zz} \mathcal{L}(z^*, \lambda^*) w > 0$ where $w \in \mathcal{C}(x^*, \lambda^*)$. For further discussion on this we refer to Chapter 12 in Nocedal and Wright (2006).

A major divide in optimization is between *convex* and *nonconvex problems*. Convex problems hold the following key property.

A local minimizer of a convex optimization problem is also a global minimizer.

This implies that Theorem 1 provides necessary conditions and Theorem 2 sufficient conditions for a global solution for convex problems. It is therefore important to be able to identify convex problems.

An optimization problem is convex if it satisfies the following two conditions:

- *The objective function is a convex function.*

[1]LICQ stands for Linear Independence Constraint Qualification.

[2]The notation $\succ 0$ means a positive definite matrix. Similarly, $\succeq 0$, $\prec 0$, and $\preceq 0$ mean positive semidefinite, negative definite, and negative semidefinite, respectively.

[3]The critical cone is loosely speaking directions that do not violate the constraints locally, see also Chapter 12 in Nocedal and Wright (2006).

- *The feasible set Ω is a convex set.*

If a problem is *strictly convex* there will always only exist one (global) solution. For definitions of convex functions and convex sets we refer to Nocedal and Wright (2006).

We will now discuss some specific optimization problems defined by (1).

- If the objective function f is linear and all constraints c_i are linear, (1) is a *linear program (LP)* which is be defined by

  $$\min_{z \in \mathbb{R}^n} d^\top z \tag{8a}$$

  subject to

  $$c_i(z) = a_i^\top z - b_i = 0, \quad i \in \mathcal{E} \tag{8b}$$

  $$c_i(z) = a_i^\top z - b_i \geq 0, \quad i \in \mathcal{I} \tag{8c}$$

  Since all constraints are linear the feasible set is convex, and since f is linear it is also a convex function. Hence, LP problems are convex problems. LP problems are, however, not strictly convex since there may exist many solutions with equal optimal objective function values.

  (8) can be formulated in various ways. In particular all LP problems can be written on the standard form which is shown below, see e.g., Chapter 13 in Nocedal and Wright (2006).

  $$\min_{z \in \mathbb{R}^n} d'^\top z \tag{9a}$$

  subject to

  $$c_i'(z) = a_i'^\top z - b_i' = 0, \ i \in \mathcal{E}' \tag{9b}$$

  $$c_i'(z) = z_i \geq 0, \ i \in \mathcal{I}' \tag{9c}$$

  (8) and (9) are hence equivalent problems provided all vectors and sets are chosen appropriately. It may be noted that a constant term may be added to (8) or (9) without changing the solution except for the optimal objective function value. Hence, such a bias term is usually removed.

  It is worth mentioning that the KKT conditions are both necessary and sufficient conditions for the LP problem. Moreover, since the LP problem is convex the KKT conditions define a global solution.

- When the objective function f is a quadratic function and all constraints c_i are linear, (1) is a *quadratic program (QP)* which is defined by

  $$\min_{z \in \mathbb{R}^n} z^\top Q z + d^\top z \tag{10a}$$

  subject to

  $$c_i(z) = a_i^\top z - b_i = 0, \ i \in \mathcal{E} \tag{10b}$$

  $$c_i(z) = a_i^\top z - b_i \geq 0, \ i \in \mathcal{I} \tag{10c}$$

  Again all constraints are linear. Hence, the feasible set is convex. A constant term may be added to (10a). QP problems may be convex or nonconvex depending on the quadratic form $z^\top Q z$ and the constraints. Q is an n-dimensional symmetric matrix and the following statement holds.

– *If Q is positive semidefinite, i.e., $Q \succeq 0$, then (10) is a convex problem.*

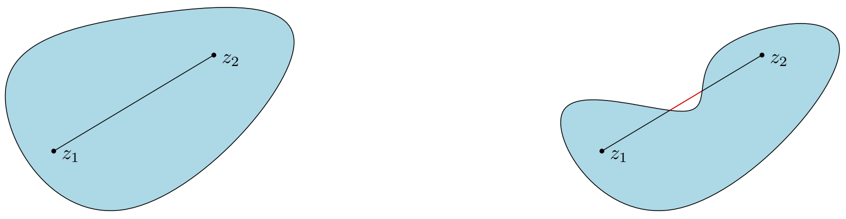
It may noted that $Q = 0$ is a positive semidefinite matrix. In this case (10) turns into a LP problem which indeed is a convex problem.
The $Q \succeq 0$ condition may be relaxed as follows, see also Chapter 16 in Nocedal and Wright (2006).

– *If $Z^\top QZ$ is positive semidefinite where Z spans the null space of the active constraints, then (10) is a convex problem.*

The KKT conditions are both necessary and sufficient conditions for a convex QP problem. Moreover, the KKT conditions define a global solution in this case.

- If the objective function f is a convex function, all the equality constraints are linear and $-c_i$ are convex functions for $i \in \mathcal{I}$, then (1) is a *convex programming* problem. Hence, the optimization problem is convex. It is useful to note that nonlinear equality constraints always give rise to nonconvex problems.
- If there are no constraints, i.e., both $\mathcal{E}$ and $\mathcal{I}$ are empty sets, (1) is an *unconstrained problem* meaning that $\Omega = \mathbb{R}^n$.
- If the objective function is some arbitrary nonlinear function or the constraints c_i are nonlinear functions, then (1) is a nonconvex problem.

Sketches of convex and nonconvex sets are shown in Figure 2 while convex and nonconvex functions are depicted in Figure 3. Note that the region above the graph of a convex function is a convex region, and that the region above the graph of a nonconvex function is a nonconvex region.



(a) A convex set. (b) A nonconvex set.

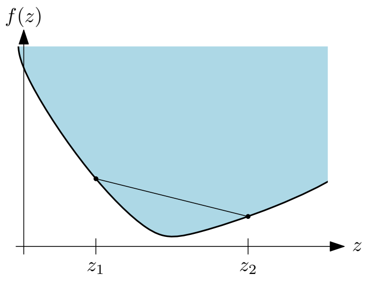
Figure 2: Comparison of a convex and a nonconvex set.

For completeness some comments on mixed integer formulations are included. A *mixed integer nonlinear program* (*MINLP*) is defined by

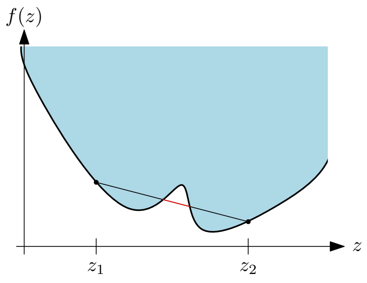
$$\min_{z\in\mathbb{R}^n, y\in\mathbb{Z}^q} f(z,y) \tag{11a}$$

subject to

$$c_i(z,y) = 0, \quad i \in \mathcal{E} \tag{11b}$$
$$c_i(z,y) \geq 0, \quad i \in \mathcal{I} \tag{11c}$$



(a) A convex function.



(b) A nonconvex function.

Figure 3: Comparison of a convex and a nonconvex function.

where $\mathbb{Z}^q$ defines a vector of integer variables. It is quite common to replace integer variables by binary variables, i.e., $y \in \{0,1\}^q$. By the definition of convex sets (11) is always a nonconvex problem since $\mathbb{Z}^q$ is a disconnected set. Integer variables are useful to describe discrete decisions such as a routing decision. One example is a stream which can be routed to one out of two pipelines and a second example are on-off valves, which are either fully open or closed. An important sub-class of problems are *mixed integer linear programs* (*MILP*) where the problem is convex except for the integer variables. Another important class of problems are *integer programs* (*IP*) with only discrete decision variables meaning that z is removed from (11) (Nemhauser and Wolsey, 1999).

2.2 Solution methods

Solution methods for optimization problems do in principle vary from explicit solution to complex iterative schemes. Explicit solutions are however only available for some simple problems. We will encounter one such relevant problem later, namely the *linear quadratic control (LQ)* problem. Otherwise the solution approach is always an iterative algorithm. There are different ways of categorizing such algorithms. We will start of by describing the structure of many such algorithms.

Algorithm 1 Iterative solution procedure

```
Given initial point $z_0$ and termination criteria
while termination criteria not satisfied do
    Compute the next iteration point
end while
```

Iterative algorithms need an initial value for z to start off. Some algorithms require a feasible initial point while this is no requirement for other algorithms. The well known *Simplex method* for solving LP problems and the active set method for solving QP problems require an initial feasible point while for instance the sequential quadratic programming (SQP) method does not require an initial feasible point.

Algorithms also need termination criteria[4]. These will include one or several

[4]Also called stopping criteria.

of the following criteria: (i) a maximum allowed number of iterations, (ii) one or several progress metrics, and (iii) a characterization of the optimal point. A progress metric may be the change in objective function value, its gradient (∇f), or the gradient of the Lagrange function ($\nabla \mathcal{L}$) from one iteration to the next. Optimal points may characterized by $\|\nabla f\| < \epsilon$ or $\|\nabla \mathcal{L}\| < \epsilon$ where $\epsilon > 0$ is a small number.

Item 2 in Algorithm 1 is usually the most extensive point and there are a variety of methods. A large class of methods are gradient based methods in which the new iteration point uses gradient information to compute a new iterate according to the following scheme.

$$z_{k+1} = z_k + \alpha_k p_k \tag{12}$$

k denotes the iteration number, while $p_k \in \mathbb{R}^n$ is the search direction and $\alpha_k \in \mathbb{R}_+$ is a positive *line search* parameter. The search direction p_k depends on gradient information. Several alternatives exist, one alternative being the *steepest descent direction* in which $p_k = -\nabla f(z_k)$ and the *Newton direction* where $p_k = -(\nabla_{zz} f(z_k))^{-1} \nabla f(z_k)$. In the latter case $\nabla_{zz} f$ is the *Hessian* of f. A key point in all gradient based schemes is that p_k points in a descent direction for $f(z_k)$. Therefore the *directional derivative* should be negative, i.e., $\nabla f(z_k) p_k < 0$. It should be noted that both the derivative and Hessian information is required in Newton schemes. In practice, however, a Quasi-Newton method is usually applied which means that the Hessian matrix is approximated, i.e., $B_k \approx \nabla_{zz} f(z_k)$, and $p_k = -B_k^{-1} \nabla f(z_k)$. The calculation of B_k will be fast compared to the Hessian matrix since B_k uses gradient information only. Further, positive definiteness of B_k can be controlled. The latter is important since $B_k \succ 0$ implies that $-B_k^{-1} \nabla f(z_k)$ is a descent direction. A robust and efficient Quasi-Newton algorithm, which computes B_k^{-1}, is the *BFGS algorithm*.

Gradients can be computed using finite differencing and automatic differentiation. Finite differencing is based on Taylor's theorem and the gradient is estimated using perturbations. Approximation accuracy varies. A central difference scheme is for instance more accurate than forward differencing. The computation time for the former is, on the other hand, roughly the double of the latter. Accuracy also depends on the perturbation size. Automatic differentiation is an efficient way of computing derivatives. It basically adopts the chain rule by dividing complex functions into elementary operations. This approach is gaining popularity and is gradually implemented in optimization packages, see also Chapter 8 in Nocedal and Wright (2006) for more information.

Some methods exploit structural properties in order to compute the next iteration point. This is the case for the Simplex method, which uses the fact that it is sufficient to look for a solution in vertexes of the feasible polytope given by the linear constrains in LP problems. Hence, the iterates z_k jump from one feasible vertex to another until the KKT conditions, which are both necessary and sufficient conditions for a solutions, are met.

There are many *derivative free methods* (Conn et al., 2009), i.e., methods where no explicit gradient information is used in the search. These methods can be divided into two main groups, *pattern search methods* and *model based methods*. Examples of pattern search techniques are the Hooke Jeeves direct search (Hooke and Jeeves, 1961), the Nelder Mead algorithm (Conn et al., 2009), generalized pattern search (Torzcon, 1997) and the mesh adaptive direct search (Audet and Dennis JR., 2006). These methods are applicable to problems with a limited number of decisions variables, typically less than a few hundred optimization variables. They are, however, fairly easy to implement in a distributed computing environment, a measure that improves efficientcy. Model based methods approximate f in the vicinity of z_k by a simple, typically quadratic, model. These methods are called trust region derivative free methods, cf. Conn et al. (2009). An advantage of the trust region methods is that it is possible to establish stringent convergence proofs for these algorithms.

Derivative free methods are popular since they are easily understood and straightforward to implement. To elaborate further, evaluating the objective function will in some cases require extensive computations. This is for instance the case if the equality constraints are embedded in a simulator. One example, among many, is a simulator of the electricity grid in Southern Norway where the decision variables are capacitors placed in different locations of the grid. The objective would typically be to keep the voltage within upper and lower bounds by changing the capacitor load. In this case the simulator must be run once in order to compute the objective function once. Now, simulators do not usually supply gradients. Hence, one may be forced to use a derivative free method. To reiterate, a parallel implementation in this case may reduce runtime significantly.

3 Optimization of dynamic systems

Dynamic systems are characterized by changes over time which means that variables are functions of time. On the other hand, *static systems* are time independent. Dynamic systems are modeled in many different ways, for instance with differential equations, and there are different approaches to optimization of such systems. Two main categories, *quasi dynamic optimization* and *dynamic optimization*, are explained below.

- Quasi dynamic optimization: Optimize a dynamic system by repetitive optimization on a static model. The idea is that the dynamic changes can be compensated for by frequent reoptimizing on a static model. This approach works well for slowly varying systems and systems that are mostly in steady state.
- Dynamic optimization: Optimize on a dynamic model. In this case the solution will be a function of time, i.e., all decision variables will be functions of time. Dynamic optimization is necessary when dynamics plays a major role, which quite often is the case for systems with frequent changes in the operating conditions.

This note will focus on dynamic optimization. Before discussing this, however, we illustrate an example of quasi dynamic optimization.

Example 2 (Optimizing oil production)
Assume that 5 wells are producing according to the sketch in Figure 2. The well streams contain gas, oil and water. Further, the mass fraction of gas, oil and water in a well varies slowly with time. Hence, a well will produce according to

$$q_i(t) = a_{gi}(t)q_{gi}(t) + a_{oi}(t)q_{oi}(t) + a_{wi}(t)q_{wi}(t),\ i = 1, \dots, 5$$

where t defines time. $q_i(t)$ is the total rate from well i while $q_{gi}(t), q_{oi}(t), q_{wi}(t)$ are gas, oil and water rates, respectively. Since we use mass rates the mass fractions always add up to one, i.e., $a_{gi}(t) + a_{oi}(t) + a_{wi}(t) = 1$.

Assume that the goal is to maximize oil production while honoring operational constraints like gas processing and water processing capacities on a platform, and rate limits in each well. This may be formulated as an LP problem as in (8).

$$\min_{z \in \mathbb{R}^{20}} (0, 0, 1, 0, \dots, 0, 0, 1, 0)z \tag{13a}$$

subject to

$$q_i(t) - a_{gi}(t)q_{gi}(t) - a_{oi}(t)q_{oi}(t) - a_{wi}(t)q_{wi}(t) = 0, \qquad i = 1, \dots, 5 \tag{13b}$$
$$a_{gi}(t) + a_{oi}(t) + a_{wi}(t) = 1, \qquad i = 1, \dots, 5 \tag{13c}$$
$$q_i(t) \le q_i^{\max}, \qquad i = 1, \dots, 5 \tag{13d}$$
$$q_{g1}(t) + \cdots + q_{g5}(t) \le q_{\text{gas}}^{\max} \tag{13e}$$
$$q_{w1}(t) + \cdots + q_{w5}(t) \le q_{\text{water}}^{\max} \tag{13f}$$

where

$$z = (q_1, q_{g1}, q_{o1}, q_{w1}, \dots, q_5, q_{g5}, q_{o5}, q_{w5})^\top \tag{13g}$$

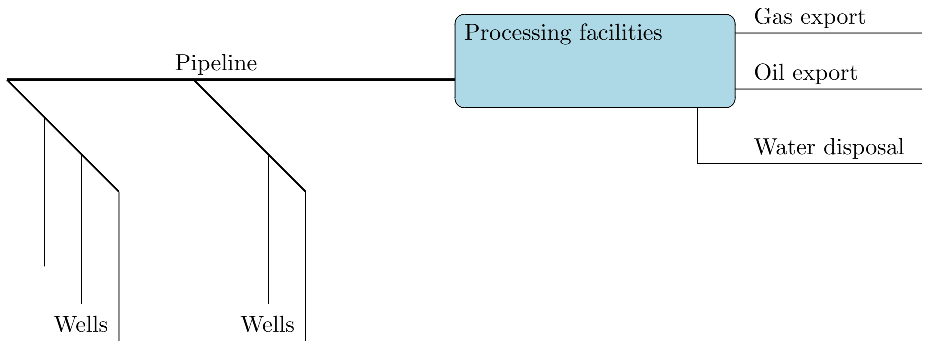


Figure 4: Sketch of oil production system for Example 2.

This is a simple LP problem, which is easily solved. One interesting question, however, is how to account for variations in time. One approach is to take a snapshot at a given time t_0 and solve (13) assuming static conditions, i.e, constant mass fractions. This solution is denoted by $z^*(t_0)$. After some time, i.e., $t_1 > t_0$, the mass fractions may have changed significantly, thus, $z^*(t_0)$ is no longer an (optimal) solution and it may even be infeasible. Hence, a new solution $z^*(t_1)$ is computed. A valid strategy is therefore to compute a new solution once the mass fractions have changed significantly[5].

A second approach is to is to treat the problem as a dynamical system. Let us assume that the flow dynamics associated with changes in flow rates are important. One operational example where this is important is during start up of a well, i.e., starting production after the well has been closed in for some time. Such a start up may take several hours. This behavior is not included in the description above since it requires a model with flow dynamics. Such a model may for instance be based on differential equations. $\triangle$

We will from now on only consider dynamic optimization and start by categorizing these problems by model type and control parametrization, i.e., how the control input is defined. Model formats are many and range from differential or differential algebraic (DAE) systems to transfer functions and further *state space models*. In this note we use state space models, see e.g., Chen (1999), since this is a fairly general class of system and such models are frequently used in dynamic optimization. Further, the presentation is limited to discrete time models since continuous time models require quite different solution methods.

3.1 Discrete time models

A discrete time system is sampled at discrete points in time. These sampling points are usually equidistant in time. Such a system can be described by a finite difference model in the following form

$$x_{t+1} = g(x_t, u_t) \tag{14a}$$

[5]The meaning of "significantly" in terms of a quantitative measure will have to be chosen in each individual case.

where system dimensions are given by

$$u_t \in \mathbb{R}^{n_u} \tag{14b}$$

$$x_t \in \mathbb{R}^{n_x} \tag{14c}$$

The function g maps an $(n_u + n_x)$-dimensional vector onto $\mathbb{R}^{n_x}$. The subscript t is a discrete time index. A slightly more general model is obtained if g is an explicit model in time, i.e., $g_t(x_t, u_t)$. The control input u_t is assumed to be piecewise constant, i.e., it is constant on the continuous time $[t, t+1\rangle$, and hence the change from u_t to u_{t+1} occurs in a stepwise manner at $t+1$. As opposed to the control inputs the states x_t are only defined at discrete points in time, and not in between these sampling points.

Note that subscript t is used above. It refers to discrete time, i.e., discrete points in time, as opposed to subscript k which is used as an iteration index, see e.g., (12).

The linear equivalent of (14) is obtained by replacing g with a linear function,

$$x_{t+1} = Ax_t + Bu_t \tag{15}$$

Equation (15) represents a *linear time invariant (LTI)* system since the matrices A and B are constant while the more general formulation below shows a *linear time variant (LTV)* since A and B are functions of time.

$$x_{t+1} = A_t x_t + B_t u_t \tag{16}$$

The LTV and LTI models are often used since nonlinear systems may be approximated by linear ones. The link between (14) and the linear models is established by the following approximation.

Assume that $\bar{x}_t, \bar{u}_t$ is a stationary point for g

$$x_{t+1} = \bar{x}_t = g(\bar{x}_t, \bar{u}_t) \tag{17}$$

and that g is at least one time differentiable. Further, define a perturbation about the stationary point by $\delta x_t, \delta u_t$. This situation may be approximated, or modeled, using a first-order Taylor series expansion.

$$\begin{aligned} x_{t+1} &= g(x_t, u_t) = g(\bar{x}_t + \delta x_t, \bar{u}_t + \delta u_t) \\ &\approx g(\bar{x}_t, \bar{u}_t) + \frac{\partial g(\bar{x}_t, \bar{u}_t)}{\partial x_t}\delta x_t + \frac{\partial g(\bar{x}_t, \bar{u}_t)}{\partial u_t}\delta u_t \end{aligned}$$

By defining a perturbation on the next state $x_{t+1} = \bar{x}_t + \delta x_{t+1}$ we obtain the following expression

$$x_{t+1} = \bar{x}_t + \delta x_{t+1} \approx g(\bar{x}_t, \bar{u}_t) + \frac{\partial g(\bar{x}_t, \bar{u}_t)}{\partial x_t}\delta x_t + \frac{\partial g(\bar{x}_t, \bar{u}_t)}{\partial u_t}\delta u_t$$

which reduces to

$$\delta x_{t+1} \approx \frac{\partial g(\bar{x}_t, \bar{u}_t)}{\partial x_t}\delta x_t + \frac{\partial g(\bar{x}_t, \bar{u}_t)}{\partial u_t}\delta u_t \tag{18}$$

This is equivalent to an LTV model if we replace the *Jacobian matrices*, i.e., $\frac{\partial g(\bar{x}_t, \bar{u}_t)}{\partial x_t}$ and $\frac{\partial g(\bar{x}_t, \bar{u}_t)}{\partial u_t}$, by A_t and B_t, and define δx_t as the states and δu_t as the control inputs. A few observations are however in order:

- If g is highly nonlinear the linear approximation (18) will be poor. Hence, it makes sense to use the nonlinear model (14) in such cases.
- If g is mildly nonlinear (18) may suffice.
- If g is mildly nonlinear and the the range in which x_t and u_t varies is small, an even coarser approximation than (18) may be used, namely an LTI model like in (15).

The discussion above may be repeated for non-stationary points by replacing (17) with $\bar{x}_{t+1} = g(\bar{x}_t, \bar{u}_t)$ and $x_{t+1} = \bar{x}_{t+1} + \delta x_{t+1}$.

3.2 Objective functions for discrete time systems

The dynamic optimization problem always stretches over time meaning that the objective function will include time. We therefore define the following objective function.

$$f(x_1, \ldots, x_N, u_0 \ldots, u_N) = \sum_{t=0}^{N} f_t(x_{t+1}, u_t) \tag{19}$$

This objective function will now be explained.

- The objective function is defined on a time horizon from $t = 0$ to $t = N$ where subscript t is the discrete time index. The time span from 0 to N is called the *prediction horizon*. It should be noted that it is straightforward to replace the prediction horizon by a time horizon which starts at time $t = j$ and continues until $t = j + N$. In this note we assume equidistant times, i.e., all the sampling times are equal.
- The number of decision variables $(x_1, \ldots, x_N, u_0 \ldots, u_N)$ increases linearly with the prediction horizon N.
- The objective function is structured such that it sums the contributions from each time step through $f_t(x_{t+1}, u_t)$.
- The initial state x_0 is not a decision variable since it is assumed to be given. It is either known exactly or it may be estimated. x_{t+1} is the state at the end of the control interval for u_t, i.e., at the end of the continuous time interval $[t, t+1\rangle$. This is why we pair x_{t+1}, u_t and therefore use $f_t(x_{t+1}, u_t)$ instead of $f_t(x_t, u_t)$.
- f_t may focus on different types of desired properties. Some examples are mentioned below.
  - Economic measures like revenue and cost may be maximized or minimized over the time horizon. One example is an objective function which mimimizes the energy consumption of an airplane or a ship on its journey between two destinations. Another example could be minimizing energy use in a building.
  - A system may need to follow a reference trajectory. Examples include a vessel that needs to follow a prescribed route, or a batch chemical reactor where the temperature profile over time is critical for product quality.

– A system should reach a given state at the end of the time horizon. An example is a rocket that should reach a certain altitude.

– Limit wear and tear on equipment while honoring operational constraints. An example is minimizing valve movements while maintaining required control performance.

There are problems which are not covered by (19), e.g., the minimal time case. An example of the latter is a fighter plane that should move from one location to another as fast as possible. In this case the prediction horizon N also becomes a decision variable.

3.3 Dynamic optimization with linear models

Given the options above a variety of dynamic optimization problems may be formulated. We start off with linear models and assume a quadratic objective function since this is a commonly used formulation.

Objective function (19) may now be written as

$$f(z) = \sum_{t=0}^{N} f_t(x_{t+1}, u_t) = \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^T Q_{t+1} x_{t+1} + d_{xt+1}^T x_{t+1} + \frac{1}{2} u_t^T R_t u_t + d_{ut}^T u_t$$

with $Q_t \succeq 0$ and $R_t \succeq 0$, which gives rise to the following QP problem.[6]

$$\min_{z \in \mathbb{R}^n} f(z) = \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^T Q_{t+1} x_{t+1} + d_{xt+1} x_{t+1} + \frac{1}{2} u_t^T R_t u_t + d_{ut} u_t \tag{20a}$$

subject to

$$x_{t+1} = A_t x_t + B_t u_t, \qquad t = 0, \dots, N-1 \tag{20b}$$
$$x_0 = \text{given} \tag{20c}$$
$$x^{\text{low}} \le x_t \le x^{\text{high}}, \qquad t = 1, \dots, N \tag{20d}$$
$$u^{\text{low}} \le u_t \le u^{\text{high}}, \qquad t = 0, \dots, N-1 \tag{20e}$$
$$-\Delta u^{\text{high}} \le \Delta u_t \le \Delta u^{\text{high}}, \qquad t = 0, \dots, N-1 \tag{20f}$$
$$Q_t \succeq 0 \qquad t = 1, \dots, N \tag{20g}$$
$$R_t \succeq 0 \qquad t = 0, \dots, N-1 \tag{20h}$$

where

$$\Delta u_t = u_t - u_{t-1} \tag{20i}$$
$$z^T = (x_1^T, \dots, x_N^T, u_0^T \dots, u_{N-1}^T) \tag{20j}$$
$$n = N \cdot (n_x + n_u) \tag{20k}$$

In the interest of simplifying the notation, we will not specify the set of times t for the constraints whenever a special case of (20) is stated later in this note.

Several comments are included to explain the formulation.

[6] It is common to include $\frac{1}{2}$ as a part of the quadratic terms in dynamic optimization. The reason is that some results, in particular linear quadratic control (LQ) which will be discussed later, uses this convention.

- The reason for using index $t+1$ for the states x_t is that the states of interest are $x_1, \dots, x_N$ since x_0 is fixed. This is different for the control input since u_0 defines the control on the time interval $[0,1\rangle$ and u_{N-1} the control on $[N-1, N\rangle$. Hence, we cover the whole prediction horizon.

- The discrete time model appears as an equality condition in the optimization problem. Further, we observe that it is repeated N times, once for each time step on the horizon. The reason why t runs only to $N-1$ is the fact that x_N is given by information at time $N-1$ since $x_N = A_{N-1}x_{N-1} + B_{N-1}u_{N-1}$.

- An initial value x_0, i.e., the state at the beginning of the prediction horizon, is required. It is given as an equality constraint meaning that x_0 is no free variable for optimization.

- Upper and lower bounds are placed on the states. These could be temperature or pressure limits, or for instance speed constraints. The constraints are repeated N times while no limit is set on x_0 since it is given by (20c). x^{low} and x^{high} are vectors and if no specific limits exists for certain states the corresponding elements of x^{low} and x^{high} may be set to $-\infty$ and $+\infty$, respectively.

- Constraint (20d) may well be generalized by placing limits on some controlled variable γ_t instead of the states, i.e., by replacing x_t with $\gamma_t = h(x_t)$ and constraining this function. An example of the latter may be limits on product quality, say viscosity of paint, which can be modeled as a function of the states which may include temperature, humidity and concentration of certain species. If h is a linear function then $\gamma_t = Hx_t$.

- The objective function (20a) may weight the controlled variable γ_t instead of x_t. For a linear model, $\gamma_t = Hx_t$, the objective function is then given by

$$\begin{aligned} f(z) &= \sum_{t=0}^{N-1} \frac{1}{2}\gamma_{t+1}^T Q_{t+1}\gamma_{t+1} + d_{\gamma t+1}^T \gamma_{t+1} + \frac{1}{2}u_t^T R_t u_t + d_{ut}^T u_t \\ &= \sum_{t=0}^{N-1} \frac{1}{2}x_{t+1}^T H^T Q_{t+1} H x_{t+1} + d_{\gamma t+1}^T H x_{t+1} + \frac{1}{2}u_t^T R_t u_t + d_{ut}^T u_t \end{aligned} \tag{21}$$

  where $H^T Q_{t+1} H$ replaces Q_{t+1} in (20a). In this case we typically choose $Q_t \succ 0$ since γ_t has been selected to inlude only important controlled variables.

- The objective function is of LTV type since Q_t and R_t are time varying. It is common to use an LTI objective where these matrices do not vary with time.

- The control input is limited by upper and lower bounds. An obvious example of this is a valve with a range from closed to fully open. The control input constraints run from 0 till $N-1$ since u_{N-1} defines the control input on the time horizon $t \in [N-1, N\rangle$.

- The change in the control input is restricted by (20f). This is very often an issue since for instance valves or motors do have limits on their dynamic performance, i.e., the speed with which the control input may change. Further, frequent and large changes in valve position may increase wear and tear and thereby reduce lifetime expectancy of such components.

- (20) is called en *open loop optimization problem* since it does not include feedback control. Feedback control is a central topic in Section 4.

Discrete time models usually have many more states than control inputs, i.e., $n_x \gg n_u$. Hence, the number of decision variables are essentially governed by the state dimension and the prediction horizon N. An example of a typical size could be $N = 20$, $n_x = 25$, $n_u = 4$ meaning that the number of decision variables is 580 since $n = N \cdot (n_x + n_u) = 20 \cdot (25 + 4) = 580$. According to (20b) and (20c), there are 525 equality constraints. Viewing this from a control viewpoint the future state trajectory $x_1, \dots, x_N$ is given by the initial state x_0 and the control trajectory $u_0, \dots, u_{N-1}$ through the discrete time model. Therefore the states may be eliminated by substituting $x_0, \dots, x_N$ in the objective function and the state constraints with $u_0, \dots, u_{N-1}$ using (20b) and (20c). At the end we are then left with $20 \times 4 = 80$ decision variables instead of 580. There is a penalty to be paid, however, since the matrices in the latter reduced space case are dense. The full space formulation will, however, be sparse since the *Karush-Kuhn-Tucker (KKT) matrix* is sparse[7]. All efficient solvers take advantage of this property. In this case, however, the reduced QP problem is probably the preferred choice due to the noticeable reduction in problem size. The reduced problem formulation is presented as the null space method in Chapter 16.2 in Nocedal and Wright (2006).

We have so far studied how a dynamic optimization problem can be converted into a QP problem. Two important questions still need attention. First, why is the quadratic objective function a reasonable option and, second, will (20) be a convex QP problem?

The quadratic objective function (20a) is a reasonable option and it is by far the most widely used objective function. Reasons for these are:

1. Economic measures like revenue or cost are often linear in one or several states and one or several control inputs, respectively. The typical structure of such terms are thus $d_{xt}x_t$ and $d_{ut}u_t$ where d_{xt} and d_{ut} include sales prices and the cost of input factors, respectively. This is a special case of (20a).

2. A system may need to follow a reference trajectory, one example being a vessel which has been issued a prescribed route. In this case the following term where $Q \succeq 0$ makes sense

$$\frac{1}{2}(x_t - x_t^{ref})^T Q (x_t - x_t^{ref}), \ Q \succeq 0 \tag{22a}$$

   This term can be rewritten as

$$\frac{1}{2}x_t^T Q x_t - (x_t^{ref})^T Q x_t + \frac{1}{2}(x_t^{ref})^T Q x_t^{ref} \tag{22b}$$

[7]In a sparse matrix the majority of elements are 0. The KKT matrix K is defined in Chapter 16.1 in Nocedal and Wright (2006).

Hence, in the tracking case the objective function term includes a quadratic term, a linear term and a constant. In some cases we replace x_t by the output variable γ_t, which was discussed in conjunction with (21), and which does not change the problem structure.

There is one common extension to (20a), which is to include the following term

$$\frac{1}{2}\Delta u_t^T R_{\Delta t} \Delta u_t$$

with $R_{\Delta t} \succ 0$. This penalizes control input moves[8] and thereby wear and tear on actuators like valves.

In dynamic optimization Q_t in (20) will almost always be diagonal and positive semidefinite as illustrated by item 2 above. The same holds for R_t and $R_{\Delta t}$.

The conclusion from the above discussion is that a convex QP problem, or alternatively an LP problem, is a useful formulation in dynamic optimization. Hence, the theory and solution methods discussed in Chapter 16 in Nocedal and Wright (2006), and in Chapter 13 for LP problems, do indeed provide an excellent platform for solving dynamic optimization problems.

An interesting observation is that the use of an LTI or LTV model in (20) is quite similar from a solution point of view. In both cases the problem is convex. The only difference is that the linear model calculations in (18) are more involved in the LTV case since $A_0, \ldots, A_{N-1}, B_0, \ldots, B_{N-1}$ are required instead of just A, B. This may be important if each Jacobian matrix computation is demanding.

3.4 Dynamic optimization with nonlinear discrete time models

In the former section the system dynamics were given by a linear dynamic model. Now a nonlinear dynamic model is introduced. This changes (20) to

$$\min_{z \in \mathbb{R}^n} f(z) = \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^T Q_t x_{t+1} + d_{xt+1}^T x_t + \frac{1}{2} u_t^T R_t u_t + d_{ut}^T u_t \tag{23a}$$

subject to

$$x_{t+1} = g(x_t, u_t) \tag{23b}$$
$$x_0 = \text{given} \tag{23c}$$
$$x^{\text{low}} \leq x_t \leq x^{\text{high}} \tag{23d}$$
$$u^{\text{low}} \leq u_t \leq u^{\text{high}} \tag{23e}$$
$$-\Delta u^{\text{high}} \leq \Delta u_t \leq \Delta u^{\text{high}} \tag{23f}$$
$$Q_t \succeq 0 \tag{23g}$$
$$R_t \succeq 0 \tag{23h}$$

[8]A control move is defined by $\Delta u_t = u_t - u_{t-1}$.

The comments that apply to (20) and were discussed above are equally relevant in this case. There is however one key difference and that is the nonlinear discrete time model which introduces nonlinear equality constraints. Hence, (20) is an NLP problem and thus also a nonconvex problem. This complicates matters since solution methods require much more runtime. Further, solution methods are less robust in the sense that a solution, if it exists, may not be found, and finally, there are few ways to determine the quality of a solution[9].

An NLP solver is needed to solve (23). There are two main classes of solver, *sequential quadratic programming (SQP)* and *interior point methods* as discussed in Chapter 18 and 19 in Nocedal and Wright (2006), respectively.

From a control viewpoint the future state trajectory $x_1, \dots, x_N$ is given by the initial state x_0 and the control trajectory $u_0, \dots, u_{N-1}$ through the nonlinear discrete time model. Therefore the states may be eliminated by substituting $x_0, \dots, x_N$ in the objective function and the state constraints with $u_0, \dots, u_{N-1}$ using (23b) and (23c). As in the linear case this reduces the number of decision variables significantly when $n_x \gg n_u$. There is, however, an additional penalty to be paid in the nonlinear case. The linear state constraints (23d) are nonlinear in the control inputs. As an example the state constraints at $t = 1$

$$x^{\text{low}} \leq x_1 \leq x^{\text{high}}$$

change to two nonlinear constraints in u_0. (Remember that x_0 is fixed.)

$$x^{\text{low}} \leq g(x_0, u_0) \leq x^{\text{high}}$$

Problem (23) may be generalized by using $f_t(x_t, u_t)$ instead of the quadratic term $x_t^T Q_t x_t + d_{xt} x_t + u_t^T R_t u_t + d_{ut} u_t$. As argued above the quadratic term covers most practical cases. Further, for linear models, $f_t(x_t, u_t)$ transform a convex QP problem, or an LP problem[10], into an NLP problem. Thus, this enforces a heavy penalty in terms of runtime and reliability. For nonlinear models the penalty by introducing $f_t(x_t, u_t)$ instead of the quadratic term will be less since (23) is an NLP problem in the first place.

3.5 Optimal open loop optimization examples

We continue this section with two examples of open loop optimization problems. The first example uses a *SISO system*[11] while the second discusses a *MIMO system*[12].

Example 3 (Finite horizon optimal open loop control for a SISO system) Consider the scalar dynamic discrete-time system

$$x_{t+1} = a x_t + b u_t \tag{24}$$

Our goal is to keep the state x_t as close to zero as possible while using a minimal amount of input. This can be formulated as minimizing some combination of

[9]The quality of a solution may for instance be measured by the duality gap.

[10]If $Q_t = 0$ and $R_t = 0$.

[11]A SISO system is a Single Input Single Output system. It has one input ($n_u = 1$) and one output, which in this case is the state variable. As discussed earlier the output may also be an auxiliary variable like $h(x_t)$ instead of the states.

[12]A MIMO system is a Multiple Input Multiple Output system. Thus, it has more than one input ($n_u > 1$) and more than one output.

qx_t^2 and ru_t^2 for all time instants t, where q and r are non-negative scalars that reflect how we prioritize the two objectives relative to each other. Assume we care more about keeping the state close to zero than about how much input we use; this would be the case if performance is critical and fuel is cheap. We could then choose values like

$$q = 4, \qquad r = 1 \tag{25}$$

for every value of t provided the values of x_t and u_t are similar[13]. If the time horizon for optimization is of length N, we can formulate the objective as minimization of

$$f(z) = \sum_{t=0}^{N-1} \frac{1}{2} q_{t+1} x_{t+1}^2 + \frac{1}{2} r_t u_t^2 \tag{26}$$

where

$$z = (x_1, \ldots, x_N, u_0, \ldots, u_{N-1})^\top \tag{27}$$

That is, z is a vector containing all variables. For the remainder of this example, we choose the rather short horizon length $N = 4$. We then have that

$$z = (x_1, x_2, x_3, x_4, u_0, u_1, u_2, u_3)^\top \tag{28}$$

With this definition of z, the objective function (26) can be written

$$f(z) = \frac{1}{2} z^T \underbrace{\begin{bmatrix} q_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & q_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & q_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & q_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & r_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & r_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & r_3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_4 \end{bmatrix}}_{G} z \tag{29}$$

where we call the big matrix G. If we write out the quadratic form,

$$f(z) = \frac{1}{2} z^T G z = \frac{1}{2}\big(q_1 x_1^2 + q_2 x_2^2 + q_3 x_3^2 + q_4 x_4^2 + r_0 u_0^2 + r_1 u_1^2 + r_2 u_2^2 + r_3 u_3^2\big) \tag{30}$$

it is clear that (26) and (29) are equivalent.

We now rewrite the state equation (24) as

$$- ax_t + x_{t+1} - bu_t = 0 \tag{31}$$

With this formulation, we can write the state equation for each time instant t:

$$-ax_0 + x_1 - bu_0 = 0 \tag{32a}$$
$$-ax_1 + x_2 - bu_1 = 0 \tag{32b}$$
$$-ax_2 + x_3 - bu_2 = 0 \tag{32c}$$
$$-ax_3 + x_4 - bu_3 = 0 \tag{32d}$$

[13]Similar values for x_t and u_t are obtained by scaling these variables.

This set of the equations can then be written in this fashion.

$$\begin{array}{ccccccccccc}
x_1 & & & & -bu_0 & & & & = & ax_0 \\
-ax_1 & +x_2 & & & & -bu_1 & & & = & 0 \\
& -ax_2 & +x_3 & & & & -bu_2 & & = & 0 \\
& & -ax_3 & +x_4 & & & & -bu_3 & = & 0
\end{array}$$

This suggests the matrix formulation

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & -b & 0 & 0 & 0 \\ -a & 1 & 0 & 0 & 0 & -b & 0 & 0 \\ 0 & -a & 1 & 0 & 0 & 0 & -b & 0 \\ 0 & 0 & -a & 1 & 0 & 0 & 0 & -b \end{bmatrix}}_{A_{\text{eq}}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ u_0 \\ u_1 \\ u_2 \\ u_3 \end{bmatrix}}_{z} = \underbrace{\begin{bmatrix} ax_0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{b_{\text{eq}}} \tag{33}$$

i.e., the set of system equations can be written as an equality constraint of the form

$$A_{\text{eq}} z = b_{\text{eq}} \tag{34}$$

Bounds on the states and controls are written

$$x^{\text{low}} \leq x_t \leq x^{\text{high}} \tag{35a}$$

$$u^{\text{low}} \leq u_t \leq u^{\text{high}} \tag{35b}$$

We can write these constraints in terms of z as

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ u_0 \\ u_1 \\ u_2 \\ u_3 \end{bmatrix} \geq \begin{bmatrix} x^{\text{low}} \\ x^{\text{low}} \\ x^{\text{low}} \\ x^{\text{low}} \\ u^{\text{low}} \\ u^{\text{low}} \\ u^{\text{low}} \\ u^{\text{low}} \end{bmatrix}, \quad - \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ u_0 \\ u_1 \\ u_2 \\ u_3 \end{bmatrix} \geq - \begin{bmatrix} x^{\text{high}} \\ x^{\text{high}} \\ x^{\text{high}} \\ x^{\text{high}} \\ u^{\text{high}} \\ u^{\text{high}} \\ u^{\text{high}} \\ u^{\text{high}} \end{bmatrix} \tag{36}$$

The problem is fairly easy to implement and solve in MATLAB using this formulation. For simplicity we let the system be stable and choose $a = 0.9$ and $b = 0.5$; we also set $x_0 = 4$, $x^{\text{low}} = -10$, $x^{\text{high}} = 10$, $u^{\text{low}} = -2$, $u^{\text{high}} = 2$, as well as the tuning parameters $q = 4$ and $r = 1$ for all t. The resulting optimal open loop state and control sequences are shown in Figure 5. Note that the first two control inputs are at the lower bound, and that the state does not reach zero at the end of the horizon. $\triangle$

The next example is more general than the previous one, and extends the formulation to MIMO systems.

Example 4 (Finite horizon optimal open loop control for a MIMO system) Consider the MIMO system

$$x_{t+1} = A_t x_t + B_t u_t \tag{37}$$

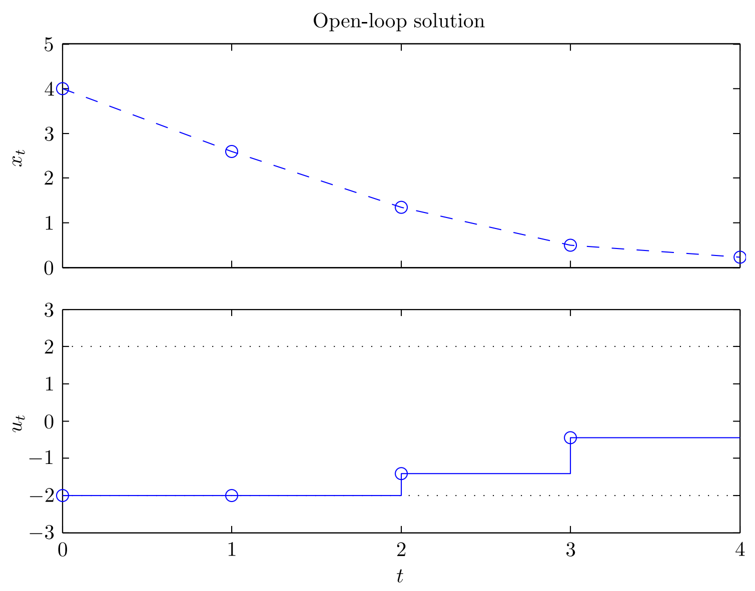


Figure 5: Open loop optimal state trajectory and control sequence for Example 3.

where $x_t \in \mathbb{R}^2$ and $u_t \in \mathbb{R}^2$. Again, we wish to keep both states at zero with minimal use of input. This can be formulated as minimizing some combination of $x_t^\top Q x_t$ and $u_t^\top R u_t$ for all time instants t, where Q and R are matrices that reflect how we prioritize the different objectives relative to each other. We have here assumed that Q and R do not vary with time. If there are two states and two control inputs and we for instance care more about keeping the first state of x_t small than we care about keeping the second state of x_t small, we could use

$$Q = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \tag{38a}$$

If we care very little about how large the inputs are, we could choose

$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \tag{38b}$$

If the time horizon for optimization is of length N, we then formulate the objective as minimization of

$$f(z) = \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^\top Q x_{t+1} + \frac{1}{2} u_t^\top R u_t \tag{39}$$

where z now is

$$z = (x_1^\top, \dots, x_N^\top, u_0^\top, \dots, u_{N-1}^\top)^\top \tag{40}$$

Note that here, x_1 is the state vector x at $t = 1$. We now choose a very short horizon length, $N = 2$, and write the objective as

$$f(z) = \frac{1}{2} x_1^\top Q x_1 + \frac{1}{2} u_0^\top R u_0 + \frac{1}{2} x_2^\top Q x_2 + \frac{1}{2} u_1^\top R u_1 \tag{41}$$

where

$$z = (x_1^\top, x_2^\top, u_0^\top, u_1^\top)^\top \tag{42}$$

Since $n_x = 2$, $n_u = 2$, and $N = 2$, z contains 8 variables. The objective can be written in the matrix form $f(z) = \frac{1}{2} z^\top G z$ with

$$G = \begin{bmatrix} Q_1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & \ddots & Q_N & \ddots & & \vdots \\ \vdots & & \ddots & R_0 & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & R_{N-1} \end{bmatrix} \tag{43}$$

Note that this kind of matrix is called *block diagonal.* For our example, with the weight matrices in (38), we get

$$G = \begin{bmatrix} Q_1 & 0 & 0 & 0 \\ 0 & Q_2 & 0 & 0 \\ 0 & 0 & R_0 & 0 \\ 0 & 0 & 0 & R_1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 \end{bmatrix} \tag{44}$$

For some arbitrary horizon length N, we can write the state equations as the equality constraint $A_{\text{eq}} z = b_{\text{eq}}$ with

$$A_{\text{eq}} = \left[\begin{array}{ccccc|ccccc} I & 0 & \cdots & & \cdots\ 0 & -B_0 & 0 & \cdots & \cdots & 0 \\ -A_1 & I & \ddots & & \vdots & 0 & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -A_N & I & 0 & \cdots & \cdots & 0 & -B_N \end{array} \right] \tag{45a}$$

and

$$b_{\text{eq}} = \begin{bmatrix} A_0 x_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{45b}$$

For our short horizon ($N = 2$), we then get

$$A_{\text{eq}} = \left[\begin{array}{cc|cc} I & 0 & -B_0 & 0 \\ -A_1 & I & 0 & -B_1 \end{array} \right], \quad \text{and} \quad b_{\text{eq}} = \begin{bmatrix} A_0 x_0 \\ 0 \end{bmatrix} \tag{46}$$

Regardless of the number of states, control inputs, and the horizon length, this formulation is fairly simple to implement in MATLAB. △

4 Optimal control

In this chapter we attempt to bridge the gap between dynamic optimization and control. This will be done by first presenting a concept which merges feedback control with dynamic optimization; *Model Predictive Control (MPC)*. Second, *linear MPC* in which a convex QP problem is the core component is reviewed. Third, the linear MPC problem is relaxed by removing inequality constraints. This is known as the *linear quadratic (LQ)* controller where the solution is given by a linear controller. Fourth, *nonlinear MPC (NMPC)*, which uses a nonlinear dynamic model, is briefly treated before this section ends with some additional comments.

4.1 Model predictive control

In Section 3.3 and 3.4 we presented dynamic optimization where a discrete time model is optimized on a time horizon from $t = 0$ till $t = N$. This is an *open loop optimization problem* since there is no feedback present in the solution. In other words the solution is computed at time $t = 0$ and this solution is used throughout the prediction horizon. There is an alternative to this, *closed loop optimization*, in which the optimal solution is recomputed at every time step t to include feedback control. Mayne et al. (2000) formulate the MPC principle as

> Model predictive control is a form of control in which the current control action is obtained by solving, at *each* sampling instant, a finite horizon open loop optimal control problem, using the current state of the plant as the initial state; the optimization yields an optimal control sequence and the first control in this sequence is applied to the plant.

A basic MPC algorithm is given below.

Algorithm 2 State feedback MPC procedure

```
for $t = 0, 1, 2, \dots$ do
    Get the current state $x_t$.
    Solve a dynamic optimization problem on the prediction horizon from $t$ to
    $t + N$ with $x_t$ as the initial condition.
    Apply the first control move $u_t$ from the solution above.
end for
```

The dynamic optimization problem referenced in the algorithm may be the convex QP problem (20) or the nonconvex NLP problem (23).

One remark is in order. We will always assume that the constraints, which in (20) and (23) are defined on the time intervals $0, \dots, N-1$ and $1, \dots, N$, are aligned with t meaning that they are shifted to $t, \dots, t+N-1$ and $t+1, \dots, t+N$ at time t.

The concept is illustrated in Fig. 6 and it essentially solves a similar optimization problem over and over again at each time step. Hence, MPC uses a *moving horizon*[14] approach in which the prediction horizon changes from $t, \dots, t+N$ to $t+1, \dots, t+N+1$ from one time step to the next.

[14]The name receding horizon is also used.

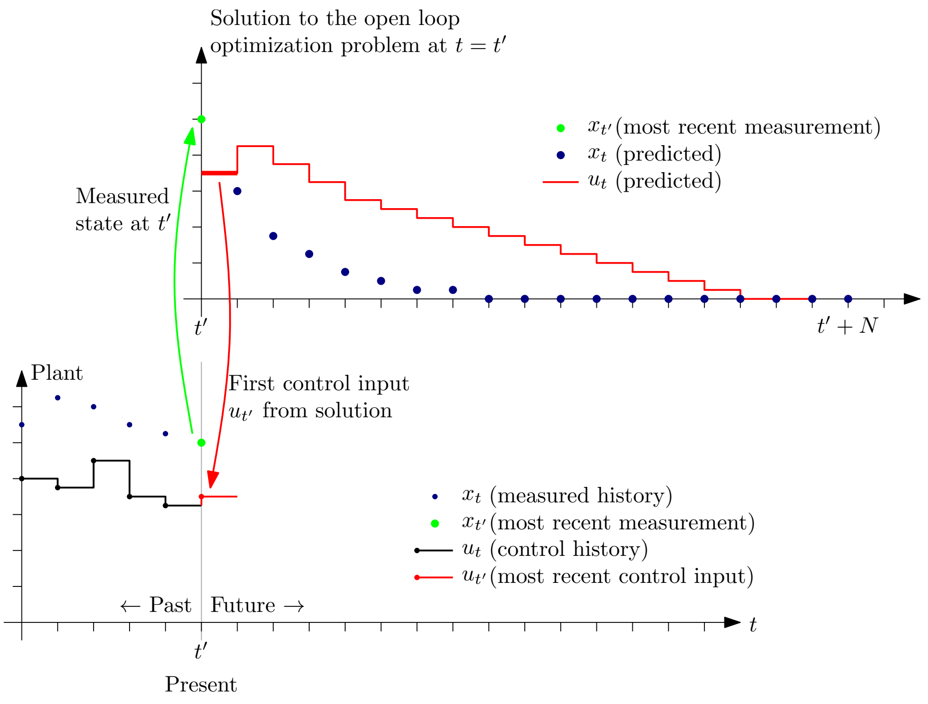

Solution to the open loop optimization problem at $t = t'$
$x_{t'}$ (most recent measurement)
x_t (predicted)
u_t (predicted)
Measured state at t'
t'
$t' + N$
Plant
First control input $u_{t'}$ from solution
x_t (measured history)
$x_{t'}$ (most recent measurement)
u_t (control history)
$u_{t'}$ (most recent control input)
← Past
Future →
t
t'
Present

Figure 6: Illustration of the MPC principle.

One obvious question is "What is the advantage of MPC vs. the open loop solution?" The brief answer is that *MPC couples open loop optimization with feedback control* in the following way: At each time step MPC requires a new solution of the dynamic optimization problem. This solution z_t^* contains the future states $x_{t+1}^*, \ldots, x_{t+N}^*$. Recalling (20c) or (23c) the optimization problem at $t + 1$ requires an initial value, i.e., x_{t+1}. A key question is how to select this initial value. One option is to use the prediction x_{t+1}^* computed at t. However, this prediction does not account for errors in the discrete time model and disturbances that occur on the time interval between t and $t + 1$. Hence, a better option is to compute a state estimate $\hat{x}_{t+1}$ which relies on the latest available measurements and use this instead of x_{t+1}^*. State estimation extends Algorithm 2 as follows.

Algorithm 3 Output feedback MPC procedure

```
for $t = 0, 1, 2, \ldots$ do
    Compute an estimate of the current state $\hat{x}_t$ based on the measured data up until time $t$.
    Solve a dynamic optimization problem on the prediction horizon from $t$ to $t + N$ with $\hat{x}_t$ as the initial condition.
    Apply the first control move $u_t$ from the solution above.
end for
```

Before moving on we reiterate the difference between Algorithm 2 and Algorithm 3. Algorithm 2 requires an exact measure of the state at each time step and is for this reason called *state feedback MPC*. Algorithm 3 relies on a state estimate which uses available measurements, i.e., available output data, and is for this reason denoted *output feedback MPC*.

To clarify, the term measured data in Algorithm 3 includes both the control input u_t and measured output. The latter will be defined in conjunction with state estimation (55).

Several books are available on MPC. Arguably the most comprehensive reference is Rawlings and Mayne (2009).

4.2 Linear MPC

Linear MPC applies the MPC concept to problems with quadratic objective functions and linear constraints. Hence, a convex QP problem, like (20), is solved at each time step. We make two changes to this QP problem before stating the algorithm. First, it runs for all times, i.e., $t = 0, 1, \ldots$, and second, it includes control input changes, i.e., Δu_t, in the objective function since this is common in MPC.

$$\min_{z \in \mathbb{R}^n} f(z) = \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^T Q_{t+1} x_{t+1} + d_{xt+1} x_{t+1} + \frac{1}{2} u_t^T R_t u_t + d_{ut} u_t + \frac{1}{2} \Delta u_t^T R_{\Delta t} u_t \tag{47a}$$

subject to

$$x_{t+1} = A_t x_t + B_t u_t \tag{47b}$$
$$x_0 = \text{given} \tag{47c}$$
$$x^{\text{low}} \leq x_t \leq x^{\text{high}} \tag{47d}$$
$$u^{\text{low}} \leq u_t \leq u^{\text{high}} \tag{47e}$$
$$-\Delta u^{\text{high}} \leq \Delta u_t \leq \Delta u^{\text{high}} \tag{47f}$$
$$Q_t \succeq 0 \tag{47g}$$
$$R_t \succeq 0 \tag{47h}$$
$$R_{\Delta t} \succeq 0 \tag{47i}$$

z and Δu_t is defined as in (20).

The state feedback linear MPC algorithm is given by Algorithm 4.

Algorithm 4 Linear MPC with state feedback

```
for $t = 0, 1, 2, \ldots$ do
    Get the current state $x_t$.
    Solve the convex QP problem (20) on the prediction horizon from $t$ to $t+N$
    with $x_t$ as the initial condition.
    Apply the first control move $u_t$ from the solution above.
end for
```

We will now discuss this algorithm starting with a simple example, and address the non zero reference case as in (22) later.

Example 5 (Comparing MPC with an open loop strategy)
Consider the unstable scalar system

$$x_{t+1} = ax_t + bu_t \tag{48}$$

with $a = 1.2$, and $b = 0.5$, and objective function

$$f(z) = \sum_{t=0}^{N-1} \frac{1}{2}qx_{t+1}^2 + \frac{1}{2}u_t^2 \tag{49}$$

with $N = 4$, $q = 1$, and $r = 4$. Figure 7 shows all open loop solutions with a prediction horizon length of $N = 4$ in a simulation of 15 time instants. The open loop solution at $t = 0$ is shown with one filled blue dot at $t = 1$ and three unfilled blue dots at $t = 2, 3, 4$. The same holds for the control input. The same coloring scheme is repeated for each time step. $\triangle$

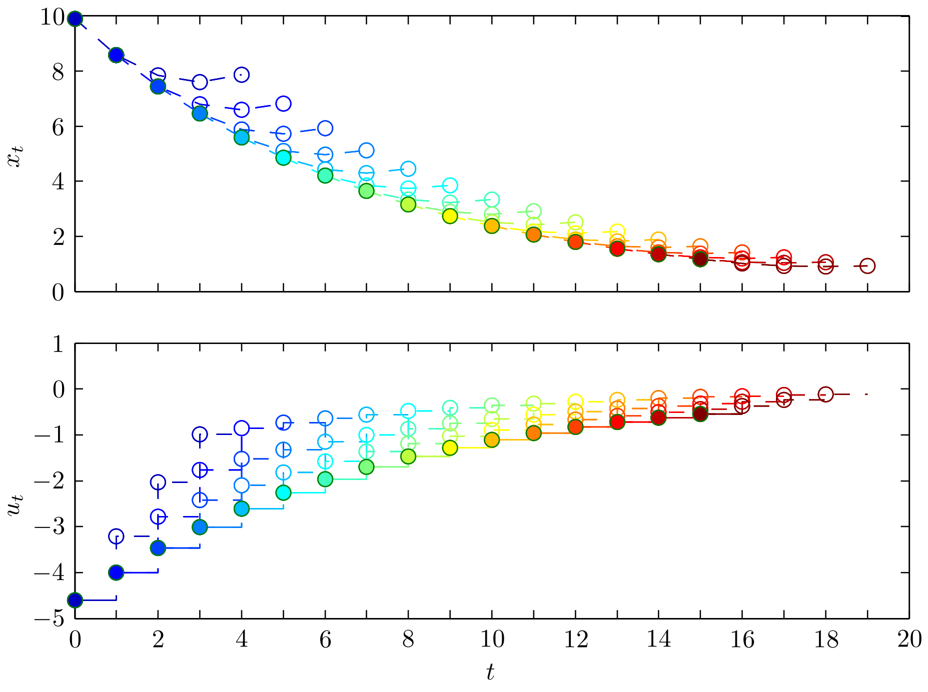


Figure 7: Simulation of the system (48) with MPC using the objective function (49).

There are some important observations to be made from this idealized example where the state is assumed to be known and where there are no model errors[15], i.e., the model used in the MPC algorithm equals the model which drives the real system.

The open loop solution computed at $t = 1$, i.e., for $t = 2, 3, 4, 5$, clearly differs from the open loop solution calculated at $t = 0$, both for the state x_t

[15]Model errors occur when the actual system differs from the MPC model. This is always the case in a practical application.

and the control input u_t. This is seen by for instance comparing the filled and unfilled blue dots at $t = 2$. The same also holds for all later time steps. Hence, the open loop solution and the MPC solution, which are marked as filled dots, differ even in this case with no uncertainty in the discrete time model or noisy measurements.

4.2.1 Ensuring feasibility at all times

Returning to Algorithm 4 a relevant question is: "Will there always exist a feasible point?" The answer is clearly no and a simple example illustrates why.

Example 6 (Infeasible solution)
Imagine a vehicle that moves along one axis and therefore has two states, position and velocity. It is controlled by Algorithm 4. Further, there are linear control input constraints as well as linear state constraints, which must be satisfied at each time step, see (47d) and (47e). The control input constraints could refer to a power limit while the state constraints may refer to limits in allowable vehicle position. Now, imagine a severe disturbance occurring between time step $t-1$ and t, which moves the vehicle beyond the state constraint limits. In this case no feasible point may exist at t since the state constraints at $t+1$, and possibly at consecutive time steps, may be violated for all feasible control inputs. $\triangle$

The example shows that a feasible point may not exist and that a control input may not available. This is an unacceptable situation. To avoid this we soften the state constraints in (47) by using *slack variables.* This is shown below where the time index t runs indefinitely as in Algorithm 4.

$$\begin{aligned}\min_{z\in\mathbb{R}^n} f(z) = \sum_{t=0}^{N-1} &\frac{1}{2}x_{t+1}^T Q_t x_{t+1} + d_{xt+1}^T x_{t+1} + \frac{1}{2}u_t^T R_t u_t + d_{ut}^T u_t \\ &+ \frac{1}{2}\Delta u_t^T R_{\Delta t} u_t + \rho^T \epsilon + \frac{1}{2}\epsilon^T S \epsilon \qquad (50a)\end{aligned}$$

subject to

$$\begin{aligned} x_{t+1} &= A_t x_t + B_t u_t && (50b)\\ x_0 &= \text{given} && (50c)\\ x^{\text{low}} - \epsilon \le x_t &\le x^{\text{high}} + \epsilon && (50d)\\ u^{\text{low}} \le u_t &\le u^{\text{high}} && (50e)\\ -\Delta u^{\text{high}} \le \Delta u_t &\le \Delta u^{\text{high}} && (50f)\end{aligned}$$

where

$$\begin{aligned} &\epsilon \in \mathbb{R}^{n_x} \ge 0\\ &\rho \in \mathbb{R}^{n_x} \ge 0\\ &S \in \operatorname{diag}\{s_1, \dots, s_{n_x}\},\ s_i \ge 0,\ i = \{1, \dots, n_x\}\end{aligned}$$

The slack variable ϵ and the tuning parameters ρ and $s_1, \cdots, s_{n_x}$ are defined such that all their elements are positive. Two terms have been added to the original QP problem, $\rho^T\epsilon$ and $\frac{1}{2}\epsilon^T S\epsilon$. These are both positive terms, hence there is a desire to drive these terms to zero. More precisely the slack variables should be nonzero only if the corresponding constraints are violated. This corresponds to the definition of an *exact penalty function* in Chapter 15.4 in Nocedal and Wright (2006). The function (50) will be an exact penalty function in this sense provided the ρ elements are big enough. Again referring to Chapter 15.4 in Nocedal and Wright (2006) the quadratic term $\frac{1}{2}\epsilon^T S\epsilon$ alone can never guarantee an exact penalty function, no matter how large the elements $s_1, \dots, s_{n_x}$ are. In practice an application will either use the linear term or the quadratic term.

The slack variable ϵ may be time dependent by for instance selecting different variables for the first part of the horizon and the latter part, respectively. The reason is to prevent unnecessary constraint violation during the latter prediction horizon since constraint violation tends to appear early on the prediction horizon. Such an approach increases the number of slack variables and hence computation time.

The output constraints may be widened during the first time steps of the prediction horizon, or remove these output constraints altogether, in order to limit the use of slack variables.

4.2.2 Stability of linear MPC

A second relevant question for Algorithm 4 is: "Will the linear MPC controller always be stable?", or which assumptions must be made in order to guarantee stability[16] of linear state feedback MPC? The somewhat surprising answer is at stability cannot be guaranteed even if a feasible solution exists for all $t = 0, 1, \dots$. A simple example illustrates this.

Example 7 (Stability of linear MPC)
Assume the following objective function

$$f(x_1, u_0) = \frac{1}{2}(x_1^2 + ru_0^2), \; r > 0 \tag{51a}$$

subject to

$$x_{t+1} = 1.2x_t + u_t \tag{51b}$$
$$x_0 = 1 \tag{51c}$$

There are no inequality constraints and the optimal control input at any time step t, i.e., u_t, can therefore be written as an explicit function of x_t as follows:

$$u_t = -\frac{1.2}{r+1}x_t \tag{51d}$$

By inserting (51d) into (51b) we obtain the closed loop system dynamics.

$$x_{t+1} = \left(1.2 - \frac{1.2}{r+1}\right)x_t = \alpha x_t \tag{52}$$

The open loop system (51b) has an eigenvalue 1.2 and is therefore an open loop unstable system. The closed loop eigenvalue in (52) exceeds 1, i.e., $\alpha > 1$, if $r < 5$, meaning that the closed loop system is unstable for these choices of r. △

[16]By stability we mean asymptotic stability, see e.g., Khalil (2002)

This simple example illustrates the fact that an MPC controller may give rise to an unstable closed loop system, even though a feasible point exists at every time step. This rather confusing fact was an important reason why stability analysis of MPC lagged behind practical applications for many years. Early references of practical MPC applications dates back to the late 70's and early 80's, see e.g. Richalet et al. (1978) and Cutler and Ramaker (1980), while the first sound stability proof was published in the early 90's. An excellent review of these topics is given in Rawlings and Muske (1993).

It is beyond the scope of this note to discuss stability in detail. We will however include some theoretical as well as practical comments.

The example above shows that feasibility of MPC is no guarantee for closed loop stability. Stability can, however, be proven by reformulating the MPC problem. There are several approaches. We will discuss a reformulation where the following equality constraint is added to (47).

$$x_N = 0 \tag{53}$$

In this case Algorithm 4 gives a stable closed loop solution provided a feasible solution is available at all times and that there are no model errors. Since there are no model errors, i.e., the MPC model and the actual system are identical, this is a *nominal stability* result. Adding (53), however, may cause infeasibility problems, in particular for short prediction horizons. An alternative to (53) is to increase the weight on the final state constraint, i.e., Q_N, which has a similar effect to the equality constraint $x_N = 0$. We refer to Rawlings and Mayne (2009) for more theory on stability of linear MPC with state feedback. If we use Algorithm 4 with output feedback instead of state feedback, stability proofs are very involved and of limited use for practical algorithms. The key problem comes from the interaction between the dynamics of the estimation loop and the control loop. Some results are available, see e.g., Imsland et al. (2003). We will return with some comments on stability analysis in the later Section 4.4.3.

If we return to Example 7 stability is easily achieved by increasing the prediction horizon. As a general rule the prediction horizon must be at least as long as the *dominant dynamics*[17] [18]. In practice stability is no major issue provided the MPC controller is designed and tuned according to sound principles. This includes the choice of model and the length of prediction horizon. In addition MPC is seldom used to control unstable processes as will be discussed later.

4.2.3 Output feedback

As discussed several times state feedback is no realistic option. Output feedback is needed meaning that Algorithm 4 changes to Algorithm 5.

The state estimate $\hat{x}_t$ is needed. In linear MPC different estimators are used depending on the model type. Since we focus on state space models there are two main estimator classes, *Kalman filter based estimators* and *Moving horizon estimators (MHE).* The celebrated Kalman filter was first proposed by Kalman (1960) and has been widely used since.

[17]The dominant dynamics is a qualitative term which may be defined by the longest time constant of a stable system.

[18]Example 7 is an unstable system. In this case the time constant is not defined. However, the unstable eigenvalue gives an indication of the minimum prediction horizon, which in this case is about 5 time steps.

Algorithm 5 Linear MPC with output feedback

```
for $t = 0, 1, 2, \dots$ do
    Compute an estimate of the current state $\hat{x}_t$ based on the measured data up until time $t$.
    Solve the convex QP problem (20) on the prediction horizon from $t$ to $t+N$ with $\hat{x}_t$ as the initial condition.
    Apply the first control move $u_t$ from the solution above.
end for
```

The structure of output feedback linear MPC is shown in Figure 8 and the estimator equations are given by

$$\hat{x}_{t+1} = A\hat{x}_t + Bu_t + K_F(y_t - \hat{y}_t) \tag{54a}$$

$$\hat{y}_t = C\hat{x}_t \tag{54b}$$

$$\hat{x}_0 = \text{given} \tag{54c}$$

where K_F is the *Kalman gain* and C is the *measurement matrix*, i.e., data y_t is measured at each sample. We assume a linear measurement model

$$y_t = Cx_t \tag{55}$$

The real system is defined within the dashed lines marked "System" while the estimator is marked by the "Estimator" dashed lines. The estimator receives measurements y_t and control signals u_t, and computes estimated states $\hat{x}_t$ that are used by the controller. The system in the figure is an LTI system. In the case of an LTV system the appropriate choice is a time varying Kalman gain[19]. The controller solves the convex QP problem (20).

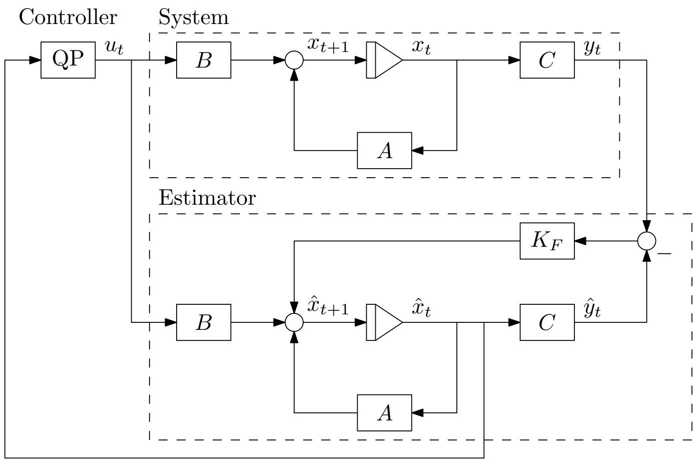


Figure 8: The structure of an output feedback linear MPC.

The Kalman filter complicates tuning of the MPC controller since the Kalman filter itself needs tuning. A rule of thumb is to make the estimator dynamics significantly faster than the linear MPC feedback loop to limit interaction between

[19]Even in the LTI case a time varying Kalman gain can be applied by using the finite horizon formulation instead of the infinite (stationary) Kalman filter solution. This is, however, rarely done.

the estimator and the control loop. Brown and Hwang (2012) is a comprehensive textbook on the Kalman filter.

As a remark the Kalman filter is based on a stochastic model description by adding noise terms to the dynamic model equation as well as the measurement model

$$x_{t+1} = Ax_t + Bu_t + v_t \tag{56a}$$
$$y_t = Cx_t + w_t \tag{56b}$$

where v_t and w_t are process noise and measurement noise terms, respectively, and the noise covariances of these terms determine the Kalman gain K_F. Hence, these covariances are used to tune the filter gain. If for instance the process noise covariance is much larger than the measurement noise covariance we rely a lot on the measurements, which in Kalman filter terms imply a large Kalman filter gain. In the opposite case the Kalman filter gain will be small. Kalman filter theory is described in many books, see for instance Brown and Hwang (2012).

MHE is an estimator type that computes $\hat{x}_t$ by solving an optimization problem (Rao et al., 2003). It uses recent data, i.e. data at times $\{t-M,\dots,t\}$, to estimate the current state x_t. Hence, MHE uses data on a time horizon backwards in time as opposed to MPC, which computes the control based on predictions forward in time. An advantage of MHE is that it easily accomodates state constraints. Examples of the latter are bounds on velocity, position, temperature and pressure. The MHE optimization problem is a convex QP problem in linear MPC. Hence, it can be solved by the same algorithm as the MPC problem on the forward horizon.

4.2.4 Reference tracking and integral action

MPC should handle different types of control problems like disturbance rejection and setpoint tracking. The linear MPC formulation (47) may be regarded as a basic formulation and extensions are therefore needed to account for some control problems. In the following two key problems will be discussed; setpoint tracking and integral action.

A system may need to follow a reference trajectory, i.e., we may track some output variable, e.g., the position of a vessel or pressure in a reactor. This situation is covered by (47) as discussed in conjunction with (22)[20]. To be more explicit, however, assume that the goal is to track a variable γ_t, which depends linearly on the states, i.e., $\gamma_t = Hx_t$. γ_t is usually called a controlled variable[21]. A suitable objective function, derived from (47), may be

$$f(z) = \sum_{t=0}^{N-1} \frac{1}{2}(\gamma_t - \gamma_t^{ref})^T Q(\gamma_t - \gamma_t^{ref}) + \frac{1}{2}\Delta u_t^T R_\Delta u_t \tag{57}$$

An important question is whether the reference trajectory is feasible. To simplify the discusion let us assume a fixed setpoint $\gamma_t^{ref} = \gamma^{ref}$, an LTI system and subsequently pose the question: Is the stationary point feasible? This can be analyzed as follows:

[20]If $d_{xt} = 0$ and $d_{ut} = 0$ the origin is the optimal solution and hence the setpoint is 0 in this case.

[21]The term controlled variable, or CV, is discussed later in conjunction with practical aspects, see Section 4.6.5.

A stationary point (x^s, γ^s, u^s) is found by setting $x_{t+1} = x_t$ in the system model.

$$\begin{aligned} x^s &= Ax^s + Bu^s \\ \gamma^s &= Hx^s \end{aligned}$$

Since the control input u_t is constrained by lower and upper bound as in (47e), there may not exist a stationary state x^s such that $\gamma^{ref} = \gamma^s$. This can easily be checked since the model and constraints are all linear. A simple example illustrates this.

Example 8 (Feasibility of a stationary point)
Assume the following stable linear system:

$$\begin{aligned} x_{t+1} &= \begin{bmatrix} 0.8 & 0.4 \\ -0.1 & 0.9 \end{bmatrix} x_t + \begin{bmatrix} 1.0 & 0.5 \\ 0.0 & 2.0 \end{bmatrix} u_t \\ \gamma_t &= \begin{bmatrix} 1.0 & -1.0 \end{bmatrix} x_t \\ 0 &\le u_t \le 1 \end{aligned}$$

A stationary point is given by

$$\begin{aligned} x^s &= \begin{bmatrix} 1.67 & 14.17 \\ -1.67 & 5.83 \end{bmatrix} u^s \\ \gamma^s &= \begin{bmatrix} 3.33 & 8.33 \end{bmatrix} u^s \end{aligned}$$

Any reference $\gamma^{ref} < 0$ or $\gamma^{ref} > 11.66$ will be infeasible since $0 \le u_t \le 1$. $\triangle$

It is useful to note that the number of controlled variables usually is smaller than the number of states, hence, there may exist several x^s that satisfy $\gamma^{ref} = \gamma^s$. If the dimension of the control input u_t is larger than the dimension of the output γ_t, there may likewise exist several stationary controls u^s that satisfy $\gamma^{ref} = \gamma^s$. In this case we may specify preferred values for some of the control inputs.

$$f(z) = \sum_{t=0}^{N-1} \frac{1}{2}(\gamma_t - \gamma^{ref})^T Q(\gamma_t - \gamma^{ref}) \\ + (u_t - u^{ref})^T R(u_t - u^{ref}) + \frac{1}{2}\Delta u_t^T R_\Delta u_t \quad (58)$$

u^{ref} defines preferred control input values and $R \succeq 0$. It may be noted that this objective is covered by (47a). We will subsequently explore the extra degrees of freedom by returning to the example above.

Example 9 (Feasibility of a stationary point — revisited)
Assume a feasible reference output $\gamma^{ref} = 2.0$ in the previous example. This may be realized by an (infinite) number of control input combinations. Three of these options are $u^s = \begin{bmatrix} 0.00 \\ 0.24 \end{bmatrix}$ or $u^s = \begin{bmatrix} 0.17 \\ 0.17 \end{bmatrix}$ or $u^s = \begin{bmatrix} 0.60 \\ 0.00 \end{bmatrix}$. In this case the second choice will usually be preferred since both control inputs then operate away from their limits, i.e., none of the control input constraints are active. Hence, in the case of a dynamic disturbance both control inputs can move in both directions to compensate for the disturbance. $\triangle$

The selection of reference outputs is not treated herein, apart from a discussion in the later Section 4.6.1 where it is treated in the context of the control hierarchy.

Integral action is an important feature of PID controllers and provides the means to compensate for disturbances with a bias component. In other words we want some controlled variables, which are measured, to converge to a constant reference value despite a constant disturbance[22].

We will now discuss how integral action can be embedded in an MPC controller and show one way of doing this. We impose integral action on the controlled variables γ_t, meaning that we want γ_t to approach a constant reference γ^{ref} in spite of a constant disturbance. The online measurements are as earlier defined by the output vector y_t, and we make the common assumption that integral action is imposed on all or some of the variables that are measured. Viewing the output model and the model for the controlled variables, respectively,

$$\gamma_t = Hx_t \tag{59a}$$

$$y_t = Cx_t \tag{59b}$$

H will contain a subset of the rows of C.

The basis for the approach taken here is to extend the dynamic model with a disturbance model,

$$\begin{bmatrix} x_{t+1} \\ d_{t+1} \end{bmatrix} = \begin{bmatrix} A & A_d \\ 0 & I \end{bmatrix} \begin{bmatrix} x_t \\ d_t \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_t$$

$$y_t = \begin{bmatrix} C & C_d \end{bmatrix} \begin{bmatrix} x_t \\ d_t \end{bmatrix}$$

and the idea is to accurately estimate the disturbance. We observe that the disturbance model is an integrating model. The augmented model, including the states and disturbances, are estimated using a state estimator as discussed in Section 4.2.3. Taking a stochastic viewpoint, as briefly commented upon in that section, noise is added to the dynamic model and the output model, and thus the state estimator updates an augmented state vector $(x_t, d_t)^T$ as shown below.

$$\begin{bmatrix} \hat{x}_{t+1} \\ \hat{d}_{t+1} \end{bmatrix} = \begin{bmatrix} A & A_d \\ 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}_t \\ \hat{d}_t \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u_t + \begin{bmatrix} K_F \\ K_d \end{bmatrix} (y_t - \hat{y}_t) \tag{60a}$$

$$\hat{y}_t = \begin{bmatrix} C & C_d \end{bmatrix} \begin{bmatrix} \hat{x}_t \\ \hat{d}_t \end{bmatrix} \tag{60b}$$

An estimate of the controlled variable is then given by

$$\hat{\gamma}_t = H\hat{x}_t \tag{61}$$

and integral action is obtained by replacing the measured control variable γ_t with its estimate $\hat{\gamma}_t$ given by (60) and (61) in an appropriate objective function. A typical objective is given below.

$$f(z) = \sum_{t=0}^{N-1} \frac{1}{2}(\hat{\gamma}_t - \gamma^{ref})^T Q(\hat{\gamma}_t - \gamma^{ref}) + \frac{1}{2}\Delta u_t^T R_\Delta u_t \tag{62}$$

[22]One may seek zero offset for a time varying reference signal. This is, however, beyond the scope of this exposition.

It may be useful to compare (62) with (57). There are two differences. First, we use a constant reference in (62) instead of a time varying reference since we focus on the constant reference case. Second, the measured control variable is replaced by its estimate to compensate for the bias component.

We use a simple example to explain the importance of the noise model in the state estimator.

Example 10 (Integral action and noise model)
Assume a first order model, which is open loop stable.

$$\begin{aligned} x_{t+1} &= ax_t + bu_t \\ y_t &= x_t + n_t \\ |a| &< 1 \\ \gamma_t &= x_t \end{aligned}$$

n_t is the measument noise and the controlled variable equals the state. We assume constant noise $n_t = \bar{n}$ and constant control $u_t = \bar{u}$. The stationary solution is given by

$$\begin{aligned} \bar{x} &= \frac{b}{1-a}\bar{u} \\ \bar{y} &= \bar{x} + \bar{n} = \frac{b}{1-a}\bar{u} + \bar{n} \\ \bar{\gamma} &= \bar{x} = \frac{b}{1-a}\bar{u} \end{aligned}$$

We will now test two different estimators starting with the simplest option.

$$\begin{aligned} \hat{x}_{t+1} &= a\hat{x}_t + bu_t + k(y_t - \hat{y}_t) \\ \hat{y}_t &= \hat{x}_t \end{aligned}$$

The stationary value of the state estimate is given by

$$\begin{aligned} \hat{\bar{x}} &= a\hat{\bar{x}} + b\bar{u} + k(\bar{y} - \hat{\bar{y}}) \\ \hat{\bar{y}} &= \hat{\bar{x}} \\ &\Downarrow \\ \hat{\bar{x}} &= \frac{b}{1-a}\bar{u} + \frac{k}{1-a+k}\bar{n} \\ \hat{\bar{\gamma}} &= \frac{b}{1-a}\bar{u} + \frac{k}{1-a+k}\bar{n} \end{aligned}$$

and we immediately observe that $\hat{\bar{\gamma}}$ differ $\bar{\gamma}$ from when $\bar{n} \neq 0$. Thus, there will be a bias in the state estimate and therefore a bias in the controlled variable γ.

The estimator will subsequently be extended with a noise model

$$\begin{aligned} \hat{x}_{t+1} &= a\hat{x}_t + bu_t + k(y_t - \hat{y}_t) \\ \hat{d}_{t+1} &= \hat{d}_t + k_d(y_t - \hat{y}_t) \\ \hat{y}_t &= \hat{x}_t + \hat{d}_t \end{aligned}$$

where $\hat{d}_t$ is an estimate of the measurement noise. In this case the stationary value of the state estimate is given by

$$
\begin{aligned}
&\hat{\bar{x}} = a\hat{\bar{x}} + b\bar{u} + k(\bar{y} - \hat{\bar{y}}) \\
&\hat{\bar{d}} = \hat{\bar{d}} + k_d(\bar{y} - \hat{\bar{y}}) \\
&\hat{\bar{y}} = \hat{\bar{x}} + \hat{\bar{d}} \\
&\quad \Downarrow \\
&\hat{\bar{x}} = \frac{b}{1-a}\bar{u} + \frac{k}{1-a+k}(\bar{n} - \hat{\bar{d}}) \\
&\hat{\bar{d}} = \hat{\bar{d}} + k_d(\bar{y} - \hat{\bar{y}}) \\
&\quad \Downarrow \\
&\hat{\bar{x}} = \frac{b}{1-a}\bar{u} + \frac{k}{1-a+k}(\bar{n} - \hat{\bar{d}}) \\
&\hat{\bar{d}} = \hat{\bar{d}} + \frac{k_d b}{1-a}\bar{u} + k_d\bar{n} - k_d\hat{\bar{x}} - k_d\hat{\bar{d}} \\
&\quad \Downarrow \\
&\hat{\bar{x}} = \frac{b}{1-a}\bar{u} + \frac{k}{1-a+k}(\bar{n} - \hat{\bar{d}}) \\
&\hat{\bar{d}} = \bar{n} \\
&\quad \Downarrow \\
&\hat{\bar{\gamma}} = \hat{\bar{x}} = \frac{b}{1-a}\bar{u}
\end{aligned}
$$

Hence, the constant disturbance does not affect the estimate of the controlled variable and it is thus an accurate estimate. If we apply objective function (62) and replace γ_k with its estimate, the optimal solution is $\hat{\gamma}_k = \gamma^{ref}$ and $\Delta u_t = 0$ provided the inequality constraints allow for this solution. Since there is no bias in the stationary estimate, integral action is guaranteed. $\triangle$

Integral action on γ_t will not always be feasible. First, the dimension of γ_t, i.e., the number of variables with integral action, cannot exceed the number of control inputs. Second, if some of the control inputs saturate, some degrees of freedom are lost. One may thus loose integral action on some of the control variables. Useful material on tracking and integral action can be found in Rawlings and Mayne (2009).

4.3 Linear Quadratic control

In this section we remove the inequality constraints altogether. First, the finite horizon LQ problem is presented and analyzed. Thereafter it is compared to the linear MPC case. Second, the infinite horizon LQ problem is discussed. This problem is subsequently discussed in the context of output feedback.

4.3.1 Finite horizon LQ control

We now remove all inequalities from (47) and pose the following problem.

$$\min_{z \in \mathbb{R}^n} f(z) = \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^T Q_{t+1} x_{t+1} + \frac{1}{2} u_t^T R_t u_t \tag{63a}$$

subject to

$$x_{t+1} = A_t x_t + B_t u_t \tag{63b}$$

$$x_0 = \text{given} \tag{63c}$$

where

$$z^T = (x_1^T, \dots, x_N^T, u_0^T \dots, u_{N-1}^T) \tag{63d}$$

$$n = N \cdot (n_x + n_u) \tag{63e}$$

The linear terms and the quadratic Δu_t term have also been removed from the objective function in addition to the inequality constraints. This will be commented later. (63) is usually called the LQ problem. Referring to Chapter 16.1 in Nocedal and Wright (2006) we should expect an explicit solution in this case as opposed to linear MPC which includes inequality constraints. We will now show that the solution of the LQ problem can be written in closed form. Moreover, this closed form is given as a linear state feedback controller, i.e., $u_t = K_t x_t$.

Theorem 3 (LQ control and the Riccati equation).
The solution of (63) with $Q_t \succeq 0$ and $R_t \succ 0$ is given by

$$u_t = -K_t x_t \tag{64a}$$

where the feedback gain matrix is derived by

$$K_t = R_t^{-1} B_t^T P_{t+1} (I + B_t R_t^{-1} B_t^T P_{t+1})^{-1} A_t, \qquad t = 0, \dots, N-1 \tag{64b}$$

$$P_t = Q_t + A_t^T P_{t+1} (I + B_t R_t^{-1} B_t^T P_{t+1})^{-1} A_t, \qquad t = 0, \dots, N-1 \tag{64c}$$

$$P_N = Q_N \tag{64d}$$

Proof. Part 1: KKT conditions

Define the Lagrange function

$$\mathcal{L}(z, \lambda_1, .., \lambda_N) = \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^T Q_{t+1} x_{t+1} + \frac{1}{2} u_t^T R_t u_t - \sum_{t=0}^{N-1} \lambda_{t+1}^T (x_{t+1} - A_t x_t - B_t u_t)$$

The KKT conditions, see e.g., Chapter 12 in Nocedal and Wright (2006), are given by[23]

$$\frac{\partial}{\partial u_t} \mathcal{L}(z, \lambda_1, .., \lambda_N) = R_t u_t + B_t^T \lambda_{t+1} = 0, \qquad t = 0, \dots, N-1 \tag{65a}$$

$$\frac{\partial}{\partial x_t} \mathcal{L}(z, \lambda_1, .., \lambda_N) = Q_t x_t - \lambda_t + A_t^T \lambda_{t+1} = 0, \qquad t = 1, \dots, N-1 \tag{65b}$$

$$\frac{\partial}{\partial x_N} \mathcal{L}(z, \lambda_1, .., \lambda_N) = Q_N x_N - \lambda_N = 0 \tag{65c}$$

[23]We skip the definition of the index set, e.g., $t = 0, \dots, N-1$, many places below to simplify notation.

and (63b) and (63c).[24]

Since R_t is positive definite, it is also invertible. Then, (65a) gives

$$u_t = -R_t^{-1}B_t^T\lambda_{t+1} \tag{66}$$

We use (65c) to postulate that the Lagrange multipliers λ_t depend linearly on the states x_t.

$$\lambda_t = P_t x_t \tag{67}$$

This gives

$$P_N = Q_N \tag{68}$$

We substitute (66) and (67) into (63b)

$$x_{t+1} = A_t x_t + B_t(-R_t^{-1}B_t^T P_{t+1}x_{t+1})$$

and, solving for x_{t+1}, we obtain

$$x_{t+1} = (I + B_t R_t^{-1} B_t^T P_{t+1})^{-1} A_t x_t \tag{69}$$

Then, we substitute (67) into (65b)

$$Q_t x_t - \lambda_t + A_t^T \lambda_{t+1} = Q_t x_t - P_t x_t + A_t^T P_{t+1} x_{t+1} = 0 \tag{70}$$

Now x_{t+1} can be removed from (70) by using (69)

$$Q_t x_t - P_t x_t + A_t^T P_{t+1}(I + B_t R_t^{-1} B_t^T P_{t+1})^{-1} A_t x_t = 0$$

Since x_t is a common factor, we can write

$$(Q_t - P_t + A_t^T P_{t+1}(I + B_t R_t^{-1} B_t^T P_{t+1})^{-1} A_t) x_t = 0$$

This equality must hold for any x_t, in particular it must hold for $x_t \neq 0$. This gives

$$Q_t - P_t + A_t^T P_{t+1}(I + B_t R_t^{-1} B_t^T P_{t+1})^{-1} A_t = 0,$$

which we recognize as (64c).

Now, (66), (67) and (69) give

$$\begin{aligned} u_t &= -R_t^{-1}B_t^T P_{t+1} x_{t+1} \\ &= -R_t^{-1}B_t^T P_{t+1}(I + B_t R_t^{-1} B_t^T P_{t+1})^{-1} A_t x_t \\ &= -K_t x_t \end{aligned}$$

This shows that (64) satisfies the KKT conditions.

Part 2: Second order conditions

[24]Equation (65b) can be difficult to apprehend Hence, it may be useful to derive it from scratch for a simple example.

Note that the states $x_1, \dots, x_N$ can be written:

$$\begin{aligned}
\chi = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} &= \begin{bmatrix} A_0 \\ A_1 A_0 \\ \vdots \\ A_{N-2} \cdot \ldots \cdot A_0 \\ A_{N-1} \cdot \ldots \cdot A_0 \end{bmatrix} x_0 \\
&+ \begin{bmatrix} B_0 & 0 & \cdots & 0 & 0 \\ A_1 B_0 & B_1 & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & 0 & 0 \\ A_{N-2} \cdot \ldots \cdot A_1 B_0 & A_{N-2} \cdot \ldots \cdot A_2 B_1 & \cdots & B_{N-2} & 0 \\ A_{N-1} \cdot \ldots \cdot A_1 B_0 & A_{N-1} \cdot \ldots \cdot A_2 B_1 & \cdots & A_{N-1} B_{N-2} & B_{N-1} \end{bmatrix} v \\
&= \widehat{A} x_0 + \widehat{B} v
\end{aligned} \tag{71}$$

where

$$v^T = (u_0^T, .., u_{N-1}^T)$$

We define the following matrices

$$\widehat{Q}_0 = \begin{bmatrix} Q_1 & 0 & \cdots & 0 & 0 \\ 0 & Q_2 & \ddots & \vdots & \vdots \\ 0 & 0 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & Q_{N-1} & 0 \\ 0 & 0 & \cdots & 0 & Q_N \end{bmatrix} \tag{72a}$$

$$\widehat{R}_0 = \begin{bmatrix} R_0 & 0 & \cdots & 0 & 0 \\ 0 & R_1 & \ddots & \vdots & \vdots \\ 0 & 0 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & R_{N-2} & 0 \\ 0 & 0 & \cdots & 0 & R_{N-1} \end{bmatrix} \tag{72b}$$

By using (71) and (72) the objective function (63a) may be written as a function of v instead of z, meaning that the states $x_1, \dots, x_N$ are eliminated through the equality constraints.

$$\begin{aligned}
f(z) = g(v) &= \frac{1}{2}\chi^T \widehat{Q}_0 \chi + \frac{1}{2} v^T \widehat{R}_0 v + \frac{1}{2} x_0^T Q_0 x_0 \\
&= \frac{1}{2}(\widehat{A} x_0 + \widehat{B} v)^T \widehat{Q}_0 (\widehat{A} x_0 + \widehat{B} v) + \frac{1}{2} v^T \widehat{R}_0 v + \frac{1}{2} x_0^T Q_0 x_0 \\
&= \frac{1}{2} v^T (\widehat{R}_0 + \widehat{B}^T \widehat{Q}_0 \widehat{B}) v + x_0^T \widehat{A}^T \widehat{Q}_0 \widehat{B} v + \frac{1}{2} x_0^T (Q_0 + \widehat{A}^T \widehat{Q}_0 \widehat{A}) x_0.
\end{aligned}$$

Thus, minimizing $g(v)$ without equality constraints is equal to the original LQ problem. Note that $\widehat{R}_0 + \widehat{B}^T \widehat{Q}_0 \widehat{B}$ is positive definite since $\widehat{R}_0$ is positive definite[25] and $\widehat{B}^T\widehat{Q}_0\widehat{B}$ is positive semidefinite[26]. Hence, $g(v)$ is a strictly convex function and second order conditions are therefore satisfied.[27]

Part 3: Summing up

We choose to use Theorem 12.6 in Nocedal and Wright (2006) to show sufficiency.

a) (64) satisfies the KKT conditions.

b) In Part 2 of the proof we showed that $g(v)$ is a strictly convex function and (12.65) in Nocedal and Wright (2006) will hence be satisfied.

c) The LQ problem is a strictly convex problem; hence the solution is a unique global solution.[28]

$\square$

There are several interesting features related to the LQ solution. We note that the solution can be formulated as a state feedback controller as shown in Figure 9. The controller is a linear time varying (LTV) controller. It is important to note that *the gain matrix K_k can be computed independently of the states.* It only depends on the system matrices (A_t, B_t) and the weighting matrices in the objective function (Q_t, R_t). It can hence be computed and stored prior to the actual use of the solution.

(64c) is the well known *discrete Riccati equation.* It is a discrete time nonlinear matrix equation. A special feature of this equation is the fact that the boundary condition is given at the end of the optimization horizon as shown in (64d). This implies that the sequence of matrices $(P_1, P_2, ..., P_{N-1})$ is computed by iterating (64c) backwards in time. Note that these matrices are necessary to obtain the gain matrices $(K_0, K_1, ..., K_{N-1})$ as shown in (64b).

The matrices $(P_1, P_2, ..., P_N)$ are known as the Riccati matrices. They are symmetric positive semidefinite matrices. To ensure that this property is retained during computation (64d) it is usually substituted by some equivalent equation with improved numerical properties. One example is

$$P_t = Q_t + A_t^T(P_{t+1}^{-1} + B_t R_t^{-1} B_t^T)^{-1} A_t.$$

In the event of an LTI system and constant weight matrices (64) is slightly simplified.

$$K_t = R^{-1}B^T P_{t+1}(I + BR^{-1}B^T P_{t+1})^{-1}A, \qquad t = 0, \dots, N-1 \tag{73a}$$
$$P_t = Q + A^T P_{t+1}(I + BR^{-1}B^T P_{t+1})^{-1}A, \qquad t = 0, \dots, N-1 \tag{73b}$$
$$P_N = Q \tag{73c}$$

It is important to note that the feedback gain (K_t) is time varying even though the system and weighting matrices are time invariant.

[25]Remember that R_t is assumed to be positive definite.

[26]The reason is that $\widehat{Q}_0$ is positive semidefinite and hence $\widehat{B}^T\widehat{Q}_0\widehat{B}$ will be positive semidefinite

[27]Theorem 12.6 in Nocedal and Wright (2006) will for instance be satisfied.

[28]As opposed to a local solution as in Theorem 12.6 in Nocedal and Wright (2006).

The tuning parameters of the LQ controller are the weighting matrices (Q_t, R_t). To make an analogy, in a SISO PI controller we adjust the gain and integral time as opposed to the weighting matrices in a LQ controller. As a side remark we note that the LQ controller is a MIMO controller since the dimensions of the control input and states usually are greater than one.

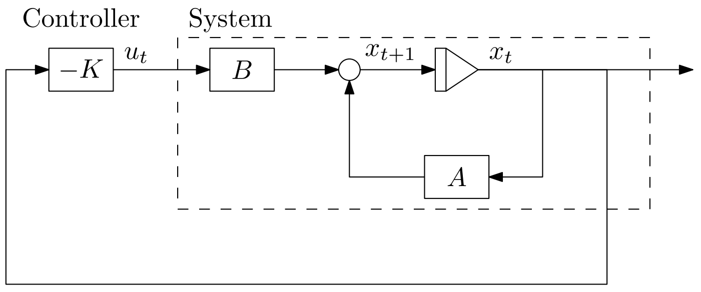


Figure 9: Solution of the LQ control problem, i.e., with state feedback.

A deeper understanding of how to select weighting matrices can only be achieved through experience. Some insight can be obtained by inspection of the equations. In particular increased values of the (diagonal) elements of R_t reduces the gain of the LQ controller. This becomes quite clear by noting that $K_t = R_t^{-1}B_t^T P_{t+1}(I + B_t R_t^{-1} B_t^T P_{t+1})^{-1}A_t \approx R_t^{-1}B_t^T P_{t+1}A_t$ when R_t is large. To provide some additional insight we present a simple example similar to an example used earlier.

Example 11 (LQ control and tuning)
Assume a simple first order system

$$x_{t+1} = 1.2x_t + u_t \tag{74a}$$

$$x_0 = 1 \tag{74b}$$

and an objective function where $N = 11$.

$$f(z) = \sum_{t=0}^{10} \frac{1}{2}x_{t+1}^2 + \frac{1}{2}ru_t^2, \quad r > 0 \tag{74c}$$

The Riccati equation is given by

$$\begin{aligned} P_t &= 1 + 1.2P_{t+1}(1 + r^{-1} \cdot P_{t+1})^{-1}1.2 \\ &= 1 + \frac{1.44\, r\, P_{t+1}}{P_{t+1} + r}, \qquad t = 0, \dots, 10 \end{aligned} \tag{75a}$$

$$P_{11} = 1 \tag{75b}$$

and the gains are computed by

$$K_t = r^{-1}P_{t+1}(1 + r^{-1}P_{t+1})^{-1}1.2 = 1.2\frac{P_{t+1}}{P_{t+1} + r}, \qquad t = 0, \dots, 10 \tag{75c}$$

The controller may be written as

$$u_t = -K_t x_t, \qquad t = 0, \dots, 10 \tag{75d}$$

Figure 10 shows the Riccati matrix, which is a scalar in this example, and the controller gain. The controller gain is clearly time varying even though the system and weight matrices are time invariant. Further, the controller gain increases with a reduced value of r. Figure 11 shows the state x_t and the control input u_t. It is quite apparent from this figure that the bandwidth of the controller increases with a decrease in r. $\triangle$

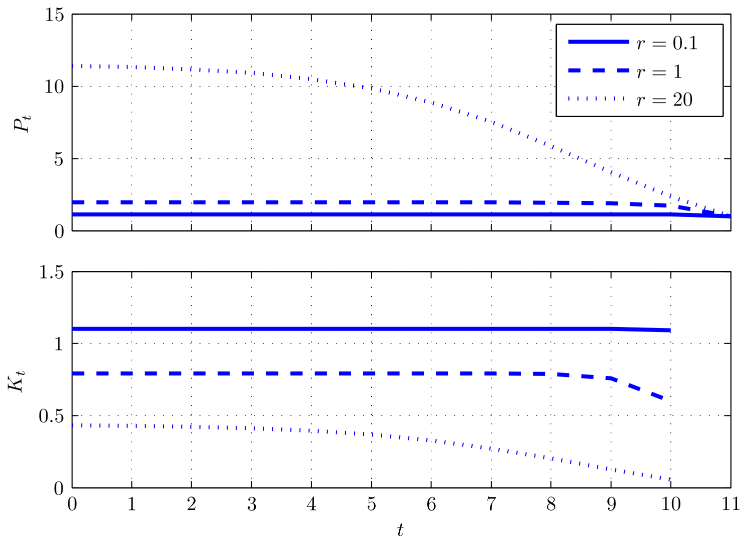


Figure 10: The figure shows P_t and the controller gain K_t. Solid line: $r = 0.1$. Dashed line: $r = 1$. Dotted line: $r = 20$.

As in Section 3.3 we may include a reference trajectory by including a term like $(x_{t+1} - x_{t+1}^{ref})^T Q_{t+1}(x_{t+1} - x_{t+1}^{ref})$ in the objective function. This will however extend the optimal solution with a feedforward term from the reference[29].

4.3.2 Moving horizon LQ control

Linear MPC, and finite horizon LQ control as discussed above, do have significant similarities. By removing the inequality constraints from linear MPC we find a simple closed form solution for the LQ problem. Hence, the computation at each time step is reduced to a simple matrix multiplication instead of solving a QP problem. This is observed by comparing (64a) with Algorithm 2, i.e. state feedback MPC.

The LQ controller is the solution of an open loop optimization problem even though the solution appears as a state feedback controller. The structure of the solution, however, accommodates a closed loop solution in a straightforward way. We just repeat the solution on a moving horizon. In this case K_t will be constant for all t for an LTI system with constant weight matrices, cf. (73), as opposed to the open loop solution in which K_t varies from one time step to the next.

[29]The solution will in this case actually require knowledge of the future reference trajectory at a given time t, see e.g., Anderson and Moore (1990).

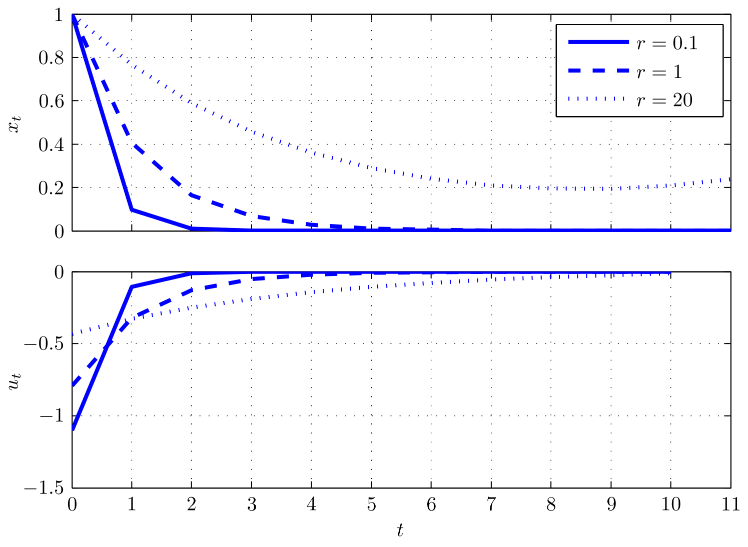


Figure 11: The figure shows the state x_t and the control input u_t. Solid line: $r = 0.1$. Dashed line: $r = 1$. Dotted line: $r = 20$.

In the output feedback case (64a) is replaced by

$$u_t = -K_t \hat{x}_t \tag{76}$$

where $\hat{x}_t$ is the state estimate. The algorithm is shown in Algorithm 3.

Algorithm 6 Output feedback moving horizon LQ

```
for $t = 0, 1, 2, \dots$ do
    Compute an estimate of the current state $\hat{x}_t$ based on the data up until time $t$.
    Compute and apply the control $u_t = K_t \hat{x}_t$. $K_t$ will be constant for all $t$ for a LTI system and an objective function with constant weight matrices.
end for
```

4.4 Infinite horizon LQ control

In the above section the optimal controller was an LTV controller. We will now develop an even simpler controller by seeking a controller with a constant feedback gain, that is $u_t = -Kx_t$. We start with the state feedback case.

4.4.1 State feedback infinite horizon LQ control

To guide the search we observe from Figure 10 that the gain seems to settle to some stationary value after some iterations.[30] This is a general property of the Riccati equation provided the system satisfies certain conditions to be defined later. By increasing the horizon sufficiently it should therefore be possible to obtain an optimal controller with a constant gain matrix within any time interval

[30]Remember that the Riccati matrix and thereby the gain is computed backwards in time.

of interest provided A, B, Q and R do not vary with time. This is the basis for the infinite horizon LQ controller where we optimize on an infinite time horizon, i.e., $N \to \infty$. The problem is formulated as follows:

$$\min_{z} f^{\infty}(z) = \sum_{t=0}^{\infty} \frac{1}{2} x_{t+1}^T Q x_{t+1} + \frac{1}{2} u_t^T R u_t \tag{77a}$$

subject to equality constraints

$$x_{t+1} = Ax_t + Bu_t \tag{77b}$$
$$x_0 = \text{given} \tag{77c}$$

and

$$Q \succeq 0 \tag{77d}$$
$$R \succ 0 \tag{77e}$$

where system dimensions are given by

$$u_t \in \mathbb{R}^{n_u} \tag{77f}$$
$$x_t \in \mathbb{R}^{n_x} \tag{77g}$$
$$z^T = (u_0^T, \dots, u_\infty^T, x_1^T, \dots, x_\infty^T) \tag{77h}$$

There are some issues that need clarification before presenting the solution.

- Problem (77) is an infinite dimensional QP problem since $n \to \infty$ in (77a). Hence, it cannot be solved using a conventional method since the *KKT matrix* is infinetely large and hence not well defined[31].
- The objective function must be bounded above, i.e., $f^{\infty}(z) < \infty$, for some feasible z. Inspection of the objective function shows that this implies that $u_t \to 0$ when $t \to \infty$. Otherwise $f^{\infty}(z) \to \infty$.

If the system (77b) is *stabilizable*[32] there will always exist a well defined solution of (77). Stabilizability is a weaker form of *controllability*[33]. It means that at least all the unstable modes of a system must be controllable. The relationship between controllability and stabilizability is therefore as follows:

System (A,B) is controllable $\Rightarrow$ System (A,B) is stabilizable

We may now state a formal result on the existence of a solution to (77a)–(77b).

Theorem 4.
Let (77b) be stabilizable. Then there exists a static solution to (73b), i.e., $P_t \to \overline{P}$ when $t \to -\infty$. This solution is equal to the unique positive semidefinite solution of the algebraic Riccati equation (78c).

[31]See e.g. Chapter 16 in Nocedal and Wright (2006) for a definition of the KKT matrix.
[32]System (A,B) is stabilizable if all the uncontrollable modes are asymptotically stable.
[33]System (A,B) is controllable if for any initial state x_0 and any final state x_N, there exists a finite number of inputs $u_0, \dots, u_{N-1}$ to transfer x_0 to x_N.

Proof. See Chapter 2 in Lewis (1986). □

Provided the above theorem is satisfied the infinite LQ controller is given as follows:

Theorem 5.
The solution of (77) is given by

$$u_t = -Kx_t \quad \textit{for } 0 \leq k \leq \infty \tag{78a}$$

where the feedback gain matrix is derived by

$$K = R^{-1}B^T P(I + BR^{-1}B^T P)^{-1}A \tag{78b}$$
$$P = Q + A^T P(I + BR^{-1}B^T P)^{-1}A \tag{78c}$$
$$P = P^T \succeq 0 \tag{78d}$$

Proof. See Chapter 2 in Lewis (1986). □

The infinite horizon LQ controller $u_t = -Kx_t$ is often called the *Linear Quadratic Regulator* or just *LQR*.

It should be noted that the algebraic Riccati-equation (78c) is a quadratic equation in the unknown matrix P. Thus, there may exist several solutions to this equation. Only one solution, however, will be positive semidefinite and therefore satisfy (78d).

Returning to Example 11 the Riccati equation was equal to

$$P_t = 1 + \frac{1.44\, r\, P_{t+1}}{P_{t+1} + r}$$

The algebraic version, i.e., (78c) is obtained by choosing $P_t = P_{t+1} = P$.

$$P^2 + (r - 1 - 1.2^2 r)P - r = 0$$

This equation has two solutions; one is positive and the other is negative. Hence, the positive solution is chosen.

A key difference between the finite and infinite LQ problem is the fact that the infinite LQ problem is defined on an infinite time horizon. Hence, stability becomes an important issue. We therefore formulate the following question: *Under which conditions will the closed loop system*

$$x_{t+1} = Ax_t + Bu_t = Ax_t - BKx_t = (A - BK)x_t \tag{79}$$

be stable? By stability we mean asymptotic stability, which implies that the states $x_t \to 0$, and thereby $u_t \to 0$, when $t \to \infty$.[34] The answer is given by the following theorem.

Theorem 6.
Given problem (77). Let system (A, B)[35] be stabilizable and (A, D)[36] detectable. D is defined by $Q = D^T D$. Then the closed loop system given by the optimal solution is asymptotically stable.

[34]The system $x_{t+1} = Ax_t$ is asymptotically stable if the absolute value of the eigenvalues of A are less than 1, i.e., $|\lambda_i(A)| < 1$ for all $1 \leq i \leq n_x$, where $\lambda_i(A)$ is the ith eigenvalue of A.

[35]$x_{t+1} = Ax_t + Bu_t$

[36]The system $x_{t+1} = Ax_t$ or $x_{t+1} = Ax_t + Bu_t$ is observed through the output $y_t = Dx_t$

Proof. See Chapter 2 in Lewis (1986). □

This theorem deserves some comments.

- *Detectability*[37] is a milder form of *observability*[38]. This implies that an observable system always is detectable. The opposite is however not necessarily true.
- Since Q is positive semidefinite it will always be possible to find a D defined by $Q = D^T D$. As an example if $Q = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \succeq 0$ we choose $D = \begin{bmatrix} 0 & 1 \end{bmatrix}$.[39]

The two conditions in Theorem 6 do have an intuitive interpretation. The stabilizabity condition tells us that we must be able to influence all unstable modes. A system with unstable modes that cannot be influenced by the control input will namely remain unstable for all control designs. The objective function must be sensitive to all the unstable modes to satisfy the detectability condition. In the opposite case a state may go to infinity without influencing the value of the objective function. Hence, unboundedness of such a state cannot be guaranteed in this situation.

Example 12 (Stability on an infinite horizon LQ controller)
To elaborate on Theorem 6 we study the following system

$$x_{t+1} = \begin{bmatrix} 1.2 & 0 \\ 0 & 0.8 \end{bmatrix} x_t + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} u_t, \quad x_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{80}$$

with the weight matrices

$$Q = I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \; R = 1 \tag{81}$$

It may be noted that (80) is an open loop unstable system since one of the eigenvalues is outside the unit circle.

If $b_1 \neq 0$ and $b_2 \neq 0$ the system (A, B) is stabilizable. It is even controllable. Further, (A, D) is detectable, and also observable since $Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and thereby $D = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Therefore the infinite LQ controller will stabilize the system, i.e., the eigenvalues of $(A - BK)$ will lie inside the unit circle.

The system will not be stabilizable if $b_1 = 0$ since the first state will grow independently of the control action, i.e., $x_{1,t} \to \pm\infty$ as $t \to \infty$ provided $x_0 \neq 0$. As a side remark one may note that the system is stabilizable, but not controllable, if $b_1 \neq 0$ and $b_2 = 0$.

The objective function will not be influenced by the first state $x_{1,t}$ if Q is changed to $Q = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$ since system (A, D), where $D = \begin{bmatrix} 0 & 1 \end{bmatrix}$, will not

[37]System (A,D) is detectable if all the unobservable modes are asymptotically stable.

[38] System (A,D) is observable if the state x_N can be determined from the system model, its inputs and outputs for a finite number of steps.

[39]Since rank$(Q) = 1$ in this case, D will be a 1×2 matrix. Hence, a 2×2 matrix is no valid decomposition as opposed to (81) where rank$(Q) = 2$.

be detectable. In the latter case D is a 2×2 matrix. If $Q = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$ then $D = \begin{bmatrix} 1 & 0 \end{bmatrix}$ and (A, D) is detectable. The system is, however, not observable in this case.

$\triangle$

A further analysis of the closed loop system, provided Theorem 6 is satisfied, can be of interest. The infinite horizon LQ controller has excellent stability margins. Safonov and Athans (1977) showed that the controller has a 60 degree *phase margin* and 6 dB *gain margin*. This result depends on one critical assumption, however, a state feedback controller. As argued in the MPC section the states must be estimated and thus an output feedback algorithm, similar to Algorithm 3, will in practice be the only realistic option. In this case nominal stability can be proven, although without any stability margin. This was shown in an elegant paper by Doyle (1978). The combination of an LQ controller and a Kalman filter is usually denoted a *linear quadratic Gaussian (LQG) controller* since the Kalman filter initially was based on stochastic linear models with Gaussian white noise.

4.4.2 Output feedback infinite horizon LQ control

To gain further insight on LQ control we analyze the output feedback case in an LTI setting. We combine the system (77b),(77c) with a measurement model $y_t = Cx_t$, the output feedback LQ controller $u_t = -K\hat{x}_t$, and the Kalman filter (54).

$$x_{t+1} = Ax_t + Bu_t \tag{82a}$$
$$y_t = Cx_t \tag{82b}$$
$$u_t = -K\hat{x}_t \tag{82c}$$
$$x_0 = \text{given} \tag{82d}$$
$$\hat{x}_{t+1} = A\hat{x}_t + Bu_t + K_F(y_t - \hat{y}_t) \tag{82e}$$
$$\hat{y}_t = C\hat{x}_t \tag{82f}$$
$$\hat{x}_0 = \text{given} \tag{82g}$$

These equations can be rewritten in a compact form

$$\xi_{t+1} = \begin{bmatrix} x_{t+1} \\ \tilde{x}_{t+1} \end{bmatrix} = \begin{bmatrix} A - BK & BK \\ 0 & A - K_F C \end{bmatrix} \xi_t \tag{83}$$
$$\tilde{x}_t = x_t - \hat{x}_t$$
$$\xi_0 = \text{given}$$

where the dimension of the augmented state is given by

$$\xi_t \in \mathbb{R}^{2 \cdot n_x}$$

and $\tilde{x}_t$ refers to the error in the state estimates. The overall structure of the LQG controller is shown in Figure 12.

Several comments and observations can be made from this.

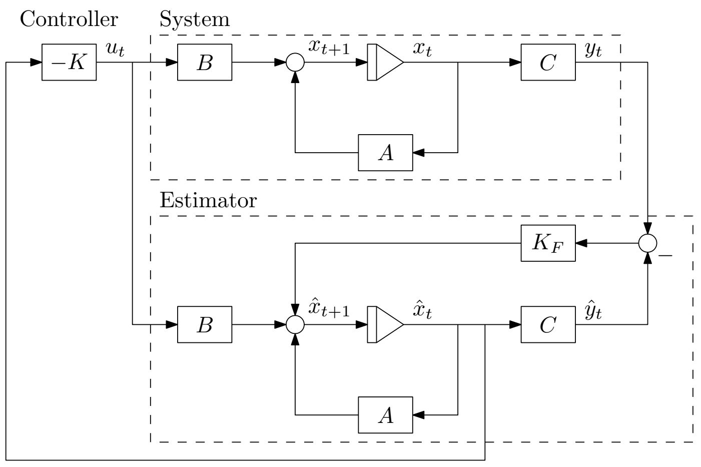


Figure 12: Structure of the LQG controller, i.e., output feedback LQ control.

- The dimension of the system and the state estimator is $2 \cdot n_x$, i.e., double the size of the original system. This reason is that the estimator introduces dynamics. To repeat, in the state feedback case the closed loop system has n_x eigenvalues while in the output feedback case the closed loop system has $2 \cdot n_x$ eigenvalues.
- The dynamics of the matrix in (83) are given by the eigenvalues of $A-BK$ and $A-K_FC$, respectively, since there is a 0 matrix in the lower left corner. Hence the eigenvalues are given by the dynamics of state feedback LQ control, i.e., $A-BK$, and the estimator dynamics $A-K_FC$. This simplifies tuning since the estimator and controller can be tuned separately. As discussed earlier the estimator dynamics should be significantly faster than the LQ controller dynamics to limit interaction between these loops.
- (83) defines the system dynamics in an idealized case since the estimator model equals the system model. In practice there will always be model errors. Model errors corrupt the matrix structure by for instance introducing non negative terms in the lower left matrix. Nevertheless, separation of dynamics according to (83) provide guidelines for tuning the estimator and controller provided the model is of a reasonable quality.
- The equation $\xi_0 =$ given deserves a comment. The latter n_x elements of this vector defines the initial values for the errors in the state estimates.

4.4.3 Stability of linear MPC with infinite horizon LQ control

Based on the infinite horizon LQ stability result in Theorem 6 we continue the stability discussion of linear MPC in Section 4.2.2. We first simplify the linear MPC formulation (47) by removing the linear terms and the Δu_t term from the objective function, and select constant weight matrices and an LTI system. Further, we assume state feedback. Finally, (47) is changed by extending the prediction horizon to infinity.

$$\min_z f(z) = \sum_{t=0}^{\infty} \frac{1}{2} x_{t+1}^T Q x_{t+1} + \frac{1}{2} u_t^T R u_t \tag{84a}$$

subject to

$$x_{t+1} = Ax_t + Bu_t \tag{84b}$$
$$x_0 = \text{given} \tag{84c}$$
$$x^{\text{low}} \leq x_t \leq x^{\text{high}} \tag{84d}$$
$$u^{\text{low}} \leq u_t \leq u^{\text{high}} \tag{84e}$$

where

$$Q \succeq 0 \tag{84f}$$
$$R \succ 0 \tag{84g}$$
$$z^T = (u_0^T, \ldots, u_\infty^T, x_1^T, \ldots, x_\infty^T) \tag{84h}$$

We first note that the origin is the only stationary point for the system. Otherwise the objective function will be unbounded.

This optimization problem may be split into two parts by dividing the prediction horizon into two parts: $\{1, \ldots, N-1\}$ and $\{N, N+1, \ldots\}$ (Muske and Rawlings, 1993; Chmielewski and Manousiouthakis, 1996; Scokaert and Rawlings, 1998). The idea is to choose N such that the problem is unconstrained on the latter infinite horizon $\{N, N+1 \ldots\}$. Provided the system is controllable and N is large enough the system can be driven to a state $x_N \in X'$. X' is a set from where the latter optimal trajectory is unconstrained (Gilbert and Tan, 1991). This means that none of the state or control input constraints are active. The optimal objective function value on the latter horizon, usually referred to as the *value function*, is given then by

$$f_N^\infty(z^*) = \frac{1}{2} x_N^T P x_N \tag{85a}$$

where

$$P = (A - BK)^T P (A - BK) + K^T R K + Q \tag{85b}$$

K is the infinite horizon feedback gain, cf. (78b). The infinite horizon problem (84) may now be reformulated into a *finite dimensional QP problem.*

$$\min_z f(z) = \sum_{t=0}^{N-1} \frac{1}{2} x_{t+1}^T Q x_{t+1} + \frac{1}{2} u_t^T R u_t + f_N^\infty(z^*) \tag{86a}$$

subject to

$$x_{t+1} = Ax_t + Bu_t \tag{86b}$$
$$x_0 = \text{given} \tag{86c}$$
$$x^{\text{low}} \leq x_t \leq x^{\text{high}} \tag{86d}$$
$$u^{\text{low}} \leq u_t \leq u^{\text{high}} \tag{86e}$$

where

$$z^T = (u_0^T, \dots, u_{N-1}^T, x_1^T, \dots, x_N^T) \tag{86f}$$

Provided the problem is feasible at all times stability is guaranteed, more specifically the origin is asymptotically stable (Rawlings and Muske, 1993). This is proved by showing that the value function for the whole horizon is a *Lyapunov function* and that this function decays towards the origin.

The proof requires a choice for N, which is difficult, if not impossible, in practice. A proof like this, and similar stability proofs, should therefore be used to provide insight into problems and solutions rather than precisely define an MPC algorithm.

4.5 Nonlinear MPC

Similar to the open loop optimization case the extension from linear MPC to nonlinear MPC is in principle straightforward. We rewrite (47) similar to (23).

$$\min_{z\in\mathbb{R}^n} f(z) = \sum_{t=0}^{N-1} \frac{1}{2}x_{t+1}^T Q_{t+1} x_{t+1} + d_{xt+1}x_{t+1} + \frac{1}{2}u_t^T R_t u_t + d_{ut}u_t + \frac{1}{2}\Delta u_t^T R_{\Delta t} u_t \tag{87a}$$

subject to

$$x_{t+1} = g(x_t, u_t) \tag{87b}$$
$$x_0 = \text{given} \tag{87c}$$
$$x^{\text{low}} \le x_t \le x^{\text{high}} \tag{87d}$$
$$u^{\text{low}} \le u_t \le u^{\text{high}} \tag{87e}$$
$$-\Delta u^{\text{high}} \le \Delta u_t \le \Delta u^{\text{high}} \tag{87f}$$
$$Q_t \succeq 0 \tag{87g}$$
$$R_t \succeq 0 \tag{87h}$$
$$R_{\Delta t} \succeq 0 \tag{87i}$$

The state feedback linear MPC algorithm is given by Algorithm 7. In the output feedback case an estimate of the state is needed, i.e., $\hat{x}_t$.

Algorithm 7 Nonlinear MPC with state feedback

```
for $t = 0, 1, 2, \dots$ do
    Get the current state $x_t$.
    Solve the optimization problem (87) on the prediction horizon from $t$ to $t + N$ with $x_t$ as the initial condition.
    Apply the first control move $u_t$ from the solution above.
end for
```

It is beyond the scope of this note to discuss NMPC in any detail. We will however provide a few comments.

The key difference, and in fact the only difference between (47) and (87), is the nonlinear model (87b). This turns the convex QP problem for linear MPC into a nonlinear and nonconvex problem. This complicates solution procedures significantly as discussed in conjunction with dynamic optimization of nonlinear systems in (23).

In the output feedback case state estimation plays a key role. Estimators for nonlinear dynamic models tend to be more complex than for linear models and an exact solution of the nonlinear state estimation problem is in fact intractable. Hence, approximations must be made. The Kalman filter, as discussed in Section 4.2.3, is also used for nonlinear systems. However, alternatives that address nonlinear behavior directly and thereby improve performance have been developed. The *extended Kalman filter (EKF)* was proposed in order to apply the Kalman filter to nonlinear spacecraft navigation problems (Bellantoni and Dodge, 1967) and is probably the most used method in applied nonlinear state estimation. The EKF is based on a linearization of a nonlinear model at each time step. Hence, it uses the time varying Jacobian matrices, cf. (18), to compute a new filter matrix at each time step. Another approach, which approximates the nonlinear filter solution more accurately, is the *unscented Kalman filter (UKF)*, see Julier and Uhlmann (2004). The MHE, which was discussed in Section 4.2.3, is also applicable to nonlinear systems.

4.6 Comments

There are many aspects of dynamic optimization and MPC, some of which have been covered in the earlier sections. Subsequently we comment on some important topics to provide additional insight.

4.6.1 The control hierarchy and MPC

A control system is often structured according to layers as shown in Figure 13. The controlled system provides realtime data to the control system through sensors and a communication channel to controllers in a *regulatory control layer*. These are conventional controllers, often *PID controllers*[40], and they are used to control properties like pressure, flow rate, temperature, power, voltage, speed and heading depending on the application. These controllers are embedded in a *distributed control system (DCS)*. The regulatory control layer needs setpoints, for instance for flow rates, temperatures, power, voltages, position and heading. This is supplied by the *advanced process control (APC) layer* where the use of MPC is spreading rapidly (Qin and Badgwell, 2003). Hence, MPC controls a process through a regulatory layer and may communicate with the DCS through a network based on for instance the *OPC communication protocol*. This solution does not necessarily satisfy industry requirements for reliability and availability. Therefore, safety critical controllers are placed in the regulatory layer. This is the case for unstable processes, like an exothermic reactor, where the stabilizing controller will lie in the regulatory layer.

The upper scheduling and optimization layer may include production plans of different products based on some manual analysis technique or as an output from an optimization application. Such applications are usually based on static models and the oil optimization case, cf. example 2, is an example of this.

The control hierarchy is valid for many sectors and two examples are given below.

[40]A PID controller will often only activate its proportional and integral action.

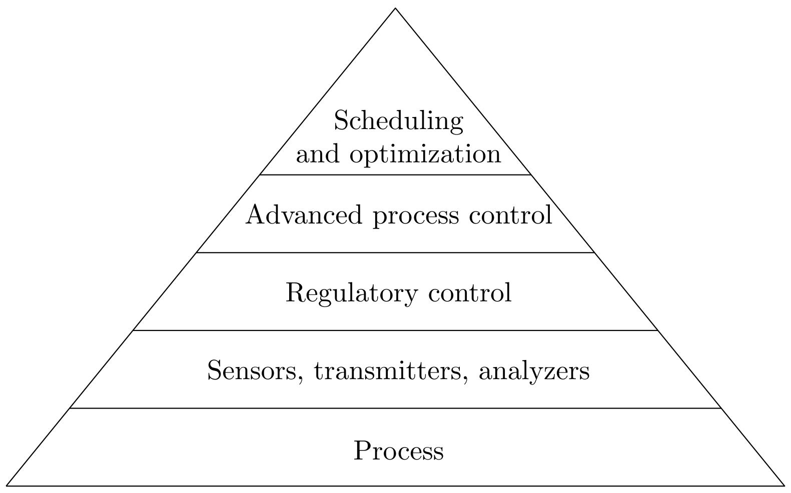


Figure 13: Typical control hierarchy.

- In refineries an optimization application based on static models may be used to schedule production for a whole refinery. Such an application solves a large LP or NP problem where models of the complete refinery are included. The APC layer will include linear MPC applications for controlling key parts of the refinery like distillation columns. The APC layer also includes simpler control strategies, e.g., heuristic based control logic. The solution from the upper optimization problem provides some of the setpoints and constraints that are used in the APC layer. The regulatory control layer includes a large number of PID controller for controlling basic variables like flow rates, pressures, temperature, levels and concentrations where the setpoints are supplied by the controllers in the APC layer.
- A ship or an airplane may use an optimization application to decide on a route from A to B. This information is transferred to an MPC application. Hence, the MPC controller may be used to keep the ship or plane close to the initial trajectory, or it may even change the trajectory based on recent information. The latter may for instance save time or fuel. The regulatory control layer will include basic controllers, e.g. for heading control.

4.6.2 MPC performance

An important reason for the industrial success of MPC is improved performance through increased disturbance rejection and better tracking performance. Hence, MPC helps systems comply with tighter performance specifications from worldwide competition and increasing constraints from environmental and safety considerations, and thereby improves economy. We will now elaborate further on this.

A well tuned MPC application usually improves control as illustrated in Figure 14. In this case, after the MPC controller is switched on, control performance improves since disturbance rejection increases significantly. Now, given an operational limit, which must be satisfied, a constraint supplied by the upper optimization layer may be increased due to improved control as shown towards the end of the time interval. This is often called “squeeze and shift”. The operational limit may for instance be temperature in an exothermic reactor, oil production rate, or the distance between a vessel and an oil platform. In all these cases improved MPC performance may improve the economics of the reactor, the oil production system or vessel operations. In terms of the control hierarchy in Figure 13 this discussion is related to the two upper layers where the MPC controller receives some of its inputs, in particular some of its constraints, from the top layer.

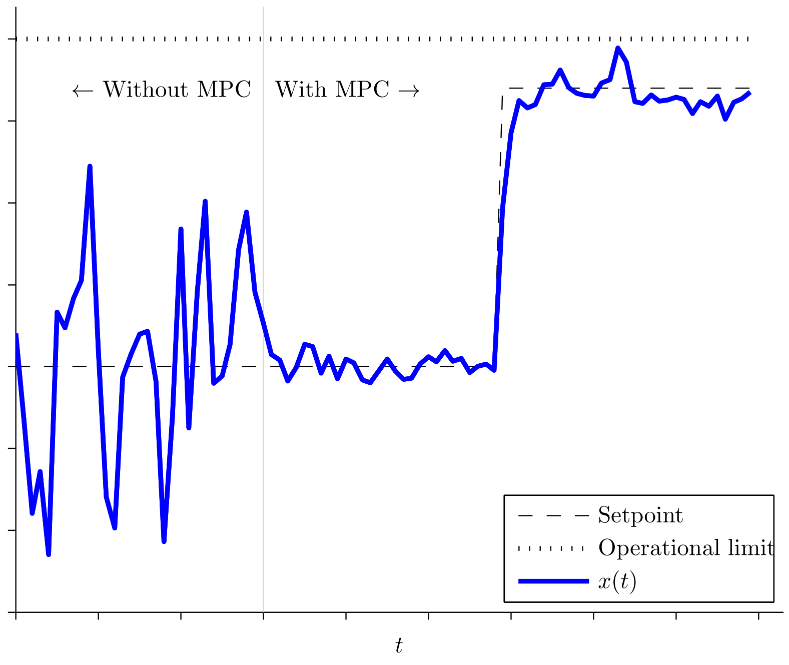


Figure 14: An MPC controller improves performance and may therefore operate closer to an operational limit. This is often called “squeeze and shift”.

Most processes are limited by constraints. A distillation column, for instance, may be constrained by boiler capacity, limits on the pressure differential across the column, or product purity specifications. It is difficult, if not impossible, to dynamically track the constraints that limit production without resorting to mathematical optimization. In this sense MPC provides a tool that automatically adjusts the control strategy to push against, and thereby exploit, the constraints that limit production according to the chosen objective function.

MPC performance also depends on the performance of the regulatory controllers. It is always important to have well tuned controllers in the regulatory control layer, e.g., well tuned PID controllers. The effect of this is similar to the “squeeze and shift” effect discussed above.

4.6.3 Feedforward control

Feedforward control is an efficient way to improve performance when disturbance measurements are available. This is commonly used in regulatory control where for instance a level controller for a tank may include a feedforward term from an upstream measurement. Thereby the level control can react before a disturbance affects the tank level. Feedforward action can easily be embedded into MPC by extending the dynamic model with a disturbance model. Focussing on linear MPC, cf. (47), feedforward is included by extending the linear model (47b) as shown below

$$x_{t+1} = A_t x_t + B_t u_t + C v_t \tag{88}$$

where v_t are measured disturbances. The downside of this is a more complex model since a disturbance model Cv_t is needed. However, this extension hardly affects computation time.

One example of feedforward MPC is its use in distillation column control where the inflow rate, and possibly composition and temperature, is used as feedforward measurements.

4.6.4 MPC models

An MPC controller needs a dynamic model. In this note we have focussed on state space models. In principle there are two approaches to modeling; *experimental models*[41] and *first principles models*[42]. Experimental models are developed from data and are popular in linear MPC applications, in particular the *step response model* since this is a model that is easy to identify from experimental data. These models are usually formulated in an *input output form* rather than a state space model.

First principles models usually require more development cost than experimental models (Foss et al., 1998). They do, however, have certain distinct advantages. First, since these models are based on physics, e.g., mass and energy conservation laws, they tend to provide better prediction capabilities than experimental models, in particular beyond the operating range covered by experimental data. Second, physics based models need less data for model identification and validation.

4.6.5 Practical MPC formulations

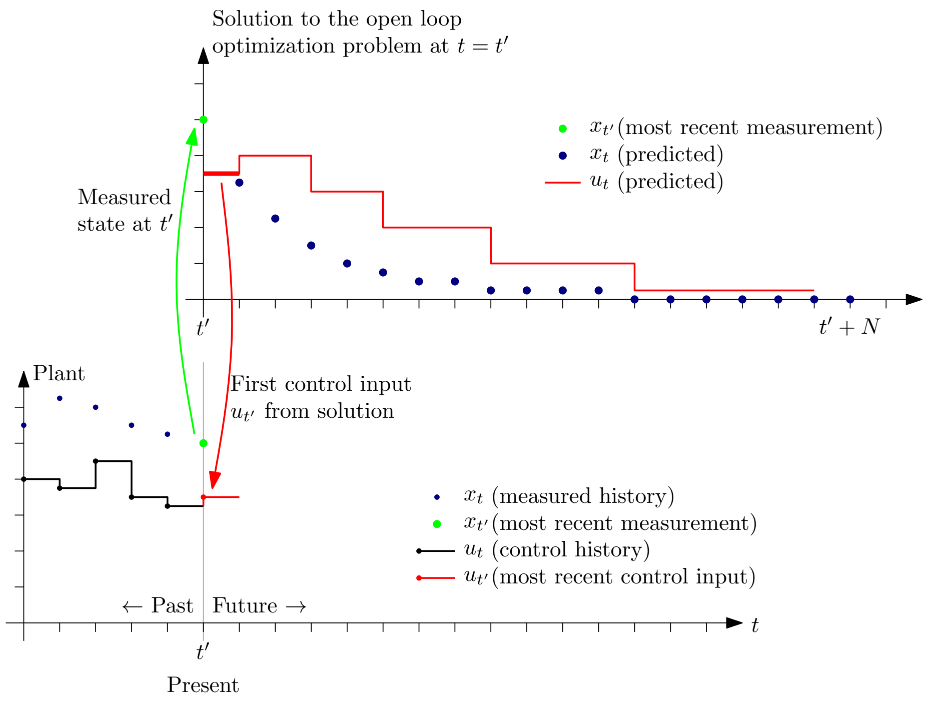
The basic linear MPC problem (47) is not used directly in industrial applications. We have already discussed several extensions like ensuring feasibility at all times and feedforward control. There are many other extension, some of which are listed below.

Control input blocking is commonly used and shown in Figure 15. It is useful to compare this figure with the upper part of Figure 6 where the number of control input decision variables is $N \cdot n_u$. Control input blocking reduces this number considerably. N may typically be 20 while the number of control moves with control input blocking may be 4 or 5. Control input blocking reduces runtime. This reduction may however marginal for linear MPC since the number of states is much large than the control input dimension as discussed in Section 3.3. Practice has shown, however, that control input blocking works well. In NMPC applications control input blocking may reduce runtime significantly

[41]Experimental models are also called black box models and data driven models.

[42]First principles models are also called physics based models, mechanistic models and white box models.

and may therefore be an important contribution to secure a robust NMPC application.



Solution to the open loop optimization problem at $t = t'$
$x_{t'}$ (most recent measurement)
x_t (predicted)
u_t (predicted)
Measured state at t'
t'
$t' + N$
Plant
First control input $u_{t'}$ from solution
x_t (measured history)
$x_{t'}$ (most recent measurement)
u_t (control history)
$u_{t'}$ (most recent control input)
← Past
Future →
t
t'
Present

Figure 15: Illustration of the MPC with control input blocking.

Some terms, which are extensively used in practice, are explained below.

- *CV* refers to a controlled variable. This is equal to γ_t, which has been discussed several times in this note. We emphasize again that CV, or γ_t, differs from the measured data y_t as discussed in conjunction with (59).
- *MV* refers to a manipulated variable and is the equivalent of the control input u_t.
- *DV* refers to a disturbance variable and equals the measured disturbance v_t in (88).

References

Anderson, B. D. O. and Moore, J. B. (1990). *Optimal Control: Linear Quadratic Methods.* Prentice Hall.

Audet, C. and Dennis JR., J. E. (2006). Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17(1):188–217.

Bellantoni, J. F. and Dodge, K. W. (1967). A square root formulation of the Kalman-Schmidt filter. *AIAA Journal*, 5(7):1309–1314.

Brown, R. G. and Hwang, P. Y. C. (2012). *Introduction to Random Signals and Applied Kalman Filtering.* Wiley, fourth edition.

Chen, C.-T. (1999). *Linear System Theory and Design.* Oxford Universtiy Press, third edition.

Chmielewski, D. and Manousiouthakis, V. (1996). On constrained infinite-time linear quadratic optimal control. *Systems & Control Letters*, 29(3):121–129.

Conn, A. R., Scheinberg, K., and Vicente, L. N. (2009). *Introduction to Derivative-Free Optimization.* SIAM.

Cutler, C. R. and Ramaker, B. L. (1980). Dynamic matrix control — A computer control algorithm. In *Proceedings of the Joint Automatic Control Conference*, San Francisco, CA.

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

Foss, B. A., Lohmann, B., and Marquardt, W. (1998). A field study of the industrial modeling process. *Journal of Process Control*, 8(5-6):325–338.

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

Goodwin, G. C., Seron, M. M., and Doná, J. A. (2004). *Constrained Control and Estimation: An Optimisation Approach.* Springer.

Hooke, R. and Jeeves, T. A. (1961). "Direct Search" solution of numerical and statistical problems. *Journal of the ACM*, 8(2):212–229.

Imsland, L., Findeisen, R., Bullinger, E., Allgöwer, F., and Foss, B. A. (2003). A note on stability, robustness and performance of output feedback nonlinear model predictive control. *Journal of Process Control*, 13(7):633–644.

Julier, S. J. and Uhlmann, J. K. (2004). Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45.

Khalil, H. K. (2002). *Nonlinear Systems.* Prentice-Hall, third edition.

Lewis, F. L. (1986). *Optimal Control.* Wiley.

Maciejowski, J. (2002). *Predictive Control with Constraints.* Pearson Prentice Hall.

Mayne, D. Q., Rawlings, J. B., Rao, C. V., and Scokaert, P. O. M. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814.

Muske, K. R. and Rawlings, J. B. (1993). Model predictive control with linear models. *AIChE Journal*, 39(2):262–287.

Nemhauser, G. L. and Wolsey, L. A. (1999). *Integer and Combinatorial Optimization.* Wiley.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization.* Springer, second edition.

Qin, S. J. and Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764.

Rao, C. V., Rawlings, J. B., and Mayne, D. Q. (2003). Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations. *IEEE Transactions on Automatic Control*, 48(2):246–258.

Rawlings, J. B. and Mayne, D. Q. (2009). *Model Predictive Control: Theory and Design.* Nob Hill Publishing.

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

Richalet, J., Rault, A., Testud, J. L., and Papon, J. (1978). Model predictive heuristic control: applications to industrial processes. *Automatica*, 14(5):413–428.

Rossiter, J. A. (2003). *Model-Based Predictive Control: A Practical Approach.* CRC Press.

Safonov, M. G. and Athans, M. (1977). Gain and phase margin for multiloop LQG regulators. *IEEE Transactions on Automatic Control*, 22(2):173–179.

Scokaert, P. O. M. and Rawlings, J. B. (1998). Constrained linear quadratic regulation. *IEEE Transactions on Automatic Control*, 43(8):1163–1169.

Torzcon, V. (1997). On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25.