Chapter 7
Stability and Feasibility of MPC

Stability is not merely a distant abstract theoretical feature of feedback controller systems. An unstable controller may supply inputs to the closed-loop plant to a point where the actuators fail or even functional damage occurs to the plant. In the application field of active vibration control, functional damage is easily interpreted in practical terms: it is structural failure due to excessive stress and strain. In case instability results in oscillations, the cause of failure may be material fatigue as well. These dramatic effects imply not only the loss of material goods and damage of equipment but also potential injury or loss of life.

One of the classic educational toys for the control engineering community is the so-called Furuta pendulum [1, 19, 24, 63], or inverted rotational pendulum which is illustrated in Fig. 7.1. This pendulum consists of an electric motor, which drives an arm rotating in the horizontal plane. The end of this arm is equipped with a pendulum freely rotating in the vertical plane and is connected to the arm with a simple joint. The aim is to swing the pendulum to the upright position and keep it there afterward by the rotational movement of the arm. This system is underactuated, nonlinear and it is easy to see that the upper equilibrium position of the pendulum is open-loop unstable. The stability of the closed-loop control scheme is an essential feature in this demonstration application, while the loss of stable control presents itself with striking clarity: the pendulum falls down to its stable equilibrium located at the bottom of its path. Although practical issues with control stability are not so tangible and evident in all control systems, it is nevertheless important to make sure that a structure manipulated by model predictive vibration control is guaranteed to remain stable at all times.

Probably the simplest definition of stability is bounded input–bounded output (BIBO) stability. As the name implies, BIBO stability means that if a finite (bounded) input is supplied to the system a finite output response must result. On the contrary, if a finite response excites the system in a way that an infinite response would theoretically result, the system is said to be unstable. The stability of traditional control systems can be guaranteed very easily. In a case of a discrete controller in a closed form, all

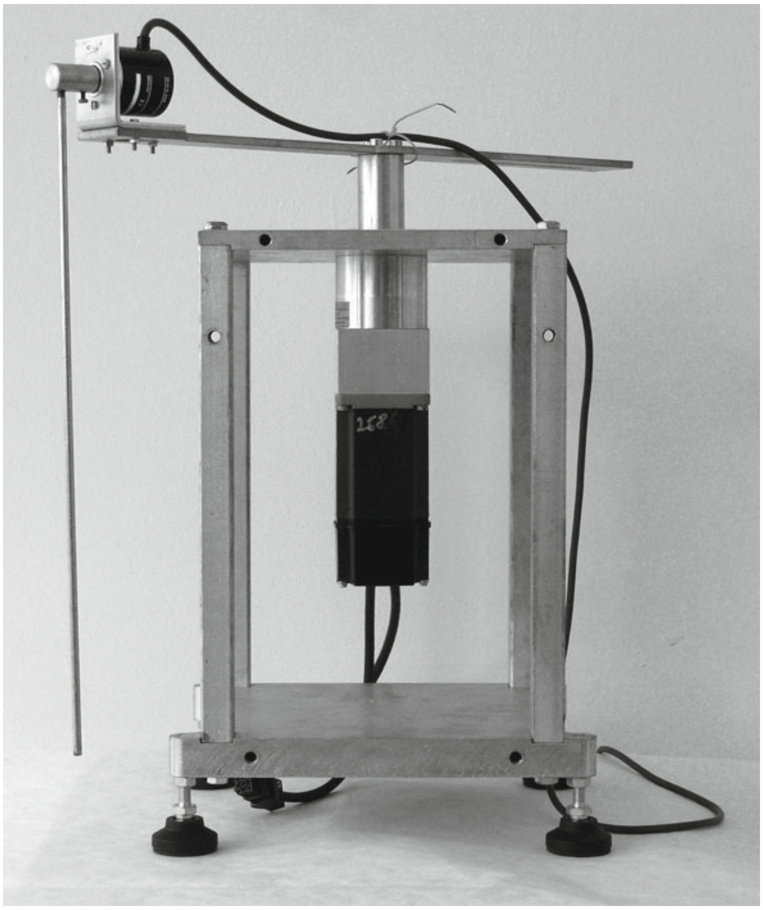


Fig. 7.1 The rotational inverted pendulum, or the Furuta pendulum, is a classic example of an open-loop unstable system. Stability of the MPC (or any other) controller is an essential feature here as well, since an insufficiently designed control strategy could easily drive the system away from its unstable equilibrium

poles of the transfer function must lie within the unit circle. However, in the case of a constrained MPC law this is not so simple anymore.

Given that one assumes no constraints for the model predictive control (MPC) problem with an infinite horizon cost, the predicted and actual input and state trajectories are identical, [1] and the stability of the process is guaranteed. This is a tractable problem, as we can readily calculate a fixed feedback matrix based on an infinite horizon cost. The predicted and actual responses would be theoretically identical in a constrained MPC problem as well, if one would use an infinitely long prediction

[1] Of course assuming no model errors or external disturbances are present.

horizon and the corresponding infinite cost. Although this would also imply a process with guaranteed stability, an infinitely long horizon would require an infinite number of optimization variables. This is clearly not possible. Instead of using an infinitely long horizon, one may utilize a finite horizon with a finite number of optimization variables in combination with terminal cost replacing the cost contribution up to infinite time—in other words the dual mode control paradigm introduced in the previous chapter. Unfortunately, this complicates the question of stability quite a bit as the constraints are only considered on a finite horizon, while beyond the horizon the feasibility of the constraints—the match of the predicted and actual trajectories and thus ultimately stability—is not guaranteed. For certain disturbances, the controller may produce an input sequence which results in an unstable response, leading to potentially catastrophic results.

The issue of MPC controller stability is reviewed in this chapter. The ultimate aim here is to provide the reader with the fundamentals to formulate a modern MPC control law, which is able to guarantee both system stability and constraint feasibility while providing maximum performance. Alternative formulations allowing for stability guarantees are also discussed, while the details of how those may help to increase computation speeds are left to Chap. 8. To the reader seeking more information on stability and alternative formulations; we may recommend the well-known books of Maciejowski and Rossiter [36, 51] and particularly the seminal work [38].

A brief review of the development of stabilized model predictive control starts our discussion on the stability of the MPC strategy. After this Sect. 7.2 inspects what the conditions of stability are for an MPC control law. Calling the well-known Lyapunov stability analysis to assistance, a very powerful conclusion can be deducted. Considering the cost as Lyapunov function will imply that if the succession of costs J_k is monotonically decreasing for the time $k \rightarrow \infty$, the controller will remain stable. The cost will be nonincreasing as long as the elements of the predicted optimal sequence $\mathbf{u}_k$ ranging from the discrete time iteration $(k+1)$ up to infinity are feasible. Next the concept of terminal constraints, and a special region within the state-space called the target set is introduced. The target set is actually bounded by the process constraints which are enforced for states ranging from the control horizon n_c up to infinity. If the states are forced to remain in this set, the system will remain feasible and thus stable as well. Fortunately, one does not have to inspect and force the inputs to conform to the constraints up to infinite time, instead it can be proved that it is enough to enforce feasibility of the constraints for an additional constraint checking horizon. The finite number of additional constraints creates a target set, which is actually identical to the maximal possible target set thus ensuring an MPC control with optimal performance while still guaranteeing stability. Based on this discussion, Sect. 7.4 will introduce the modified dual-mode quadratic programming-based MPC algorithm, which by utilizing the extra process constraints exceeding the control horizon by the new constraint checking horizon guarantees stability at all times. The following two sections of this chapter discuss alternative formulations, which can ensure the stability of MPC as well. The maximal invariant target set created by the additional constraints is a polyhedron of high complexity that further increases the necessary computational power of the MPC law. To relieve this situation partly, simplifications may be introduced. One such formulation replaces the high complexity polyhedron with a low complexity one, which is actually just a multidimensional cube in hyperspace. The other possibility is to replace the maximal invariant target set with a hyper-ellipsoid, as introduced in Sect. 7.6. A similar ellipsoidal constraint formulation is used in the upcoming chapter to create a computationally efficient MPC controller. The chapter is finished by Sect. 7.7, briefly reviewing the issues caused by mutually incompatible constraints and some strategies to avoid the infeasibility of the MPC optimization problem.

7.1 Development of MPC with Stability Guarantees

Early MPC formulations could not guarantee the stability of the closed-loop system. This however did not prevent industrial practitioners from using MPC in practical applications. The use of open-loop stable plants and long horizons in the absence of constraints prevented most stability issues; nevertheless, stability guarantees in the strict theoretical sense could not have been given.

In the beginning, stability has been investigated for finite horizon predictive controllers with a quadratic cost and in the absence of system constraints [20]. Essentially the effects of the horizon length and parameter choices were evaluated for a given controller, determining whether it is stabilizing or not. Later this rather basic approach has been deemed inappropriate [6] as several examples have shown the need for an a priori method of guaranteeing stability [58].

7.1.1 Equality Terminal Constraints

Stability guarantees for linear plants with constraints have been given later using the so-called *terminal constraints*. These terminal constraints posed a requirement on the controller, namely that the system state must equal to zero in a given number of steps beyond the horizon. By the end of the horizon n_c and beyond, the states are assumed to be zero ($x_{n_c} = 0$) while inputs assume a zero level as well [33]. Mathematically this can be translated as:

$$x_i = 0 \quad \text{for} \quad i \geq n_c \quad \text{and} \quad u_i = 0 \quad \text{for} \quad i \geq n_c \tag{7.1}$$

We may consider using an alternative interpretation to define terminal constraints. Let us require the terminal state x_{n_c} and all following states to be a part of a terminal set $x_{n_c} \in \Omega$. This terminal set is actually just a zero set $\Omega = \{0\}$.

It is possible to guarantee stability analytically for linear, unconstrained systems with an equality terminal constraint and a quadratic cost by proving that the cost function is nonincreasing. The work of Keerthi and Gilbert [28] has become a de facto basis for further stable MPC approaches. Their work proposed a constrained MPC controller based on this idea for nonlinear time varying systems, while pointing out that the cost function of the finite horizon MPC controller approaches the infinite horizon cost if the horizon is increased. Several works have proposed variations and novel algorithms which essentially make use of the fundamental idea of equality terminal constraints, such as the works of Clarke and Scattolini [17] and others [30, 40, 41]. The cost function in these early methods was finite and up to the end of the horizon J_{n_c}. A terminal cost has not been considered here.

7.1.2 *Penalty on the Terminal State*

One of the first attempts to ensure the stability of a controller is the use of a terminal penalty $\mathbf{P}_f$. The penalty of the terminal state which is added to the cost function is $J_T = x_{n_c}^T \mathbf{P}_f x_{n_c}$. As it has been previously introduced in Sect. 6.5, the most logical choice for this matrix $\mathbf{P}_f$ is the solution of the Lyapunov equation [48], which will then ensure stability through a nonincreasing cost J. This choice for $\mathbf{P}_f$ actually ensures that the addition of $J_T = x_{n_c}^T \mathbf{P}_f x_{n_c}$ to the finite horizon cost has the value of the cost from $n_c \to \infty$, thus overall giving an infinite horizon cost $J_\infty = J_{n_c} + J_T$. Mathematically express this requirement as [27]:

$$x_i \to 0 \quad \text{for} \quad i \to \infty \quad \text{and} \quad u_i \to 0 \quad \text{for} \quad i \geq n_c \tag{7.2}$$

meaning that the state and the inputs must approach zero as the time progresses toward infinity. The terminal penalty by itself would require the state to remain in a terminal set Ω which is infinitely large, that is, equal to the whole state-space $\Omega = \mathbb{R}^{n_x \times n_x}$. If the controlled system is unstable, the unstable poles must be equal to zero by the end of the horizon, while the method is utilized for the rest of the stable poles.

7.1.3 *Target Sets*

A terminal constraint in the form of an equality is too strict for most applications, as it severely limits the pool of possible initial conditions from which it is possible to steer the system into equilibrium within a finite number of steps. An equality constraint also causes a controller course with overly aggressive inputs. This strict equality requirement has been later replaced by the use of so-called terminal sets, which formulate an additional constraint in the form of an inequality. In this case, the terminal set Ω is chosen as a finite subset of the state-space $\Omega = \mathbb{R}^{n_x \times n_x}$. The aim of the controller was to steer the state into this set in a finite number of steps. The value of the cost function after the finite part J_{n_c} has been however considered to be equal to zero for this approach $J_T = 0$.

Inside the set Ω instead of requiring the states and inputs to be zero as in the case of equality constraints, a local stabilizing control law took over. The course of inputs

u_i with $i \geq n_c$ was no more equal to zero instead the fixed feedback control law $\mathbf{K}$ then steers the state to the origin as time progresses toward infinity. Mathematically this can be given as:

$$x_i \to 0 \quad \text{for} \quad i \to \infty \quad \text{and} \quad u_i = \mathbf{K}x_i \quad \text{for} \quad i \geq n_c \tag{7.3}$$

This concept has been introduced previously in Sect. 6.5 as dual-mode predictive control, where mode 1 assumes free variables and mode 2 the fixed feedback law. This approach has been introduced in [60] and later in [39] for continuous nonlinear systems with constraints. The local stabilizing feedback law $\mathbf{K}$ is most frequently chosen as the linear quadratic gain [55, 56].

7.1.4 Combination of Target Sets and Terminal Penalties

The stable MPC formulation used nowadays is the combination of the concepts presented in the previous two subsections: that is, terminal penalty $\mathbf{P}_f$ and target sets. Here the terminal cost is nonzero, rendering the total predicted cost is equal to the infinite horizon cost $J_\infty = J_{n_c} + J_T$. Moreover, the state is forced to a target set, where a stabilized fixed feedback law is taking over according to (7.3). This approach will be expanded in more detail in the rest of the chapter, while the dual-mode approach has been already considered in Chap. 6. To recapitulate the basics of modern, constrained MPC approaches with stability guarantees, we may state that they utilize the concepts of:

- Nominal stabilizing control law $u_i = \mathbf{K}x_i$ for $i \geq n_c$ also referred to as dual-mode control; which assumes the use of nonzero inputs beyond the control horizon n_c
- Target set Ω defining a state constraint at the end of the control horizon n_c, with the property that this set is contracting, e.g. once the state enters it cannot leave anymore
- Terminal penalty, $J_T = x_{n_c}^T \mathbf{P}_f x_{n_c}$ which allows to take into account a total cost $J_\infty = J_{n_c} + J_T$ equivalent to an infinitely long horizon.

In the absence of constraints, choosing a terminal penalty creates an infinite horizon cost which results in an ideal situation, where stability is ensured and online optimization is not needed. This in fact is the basis of the unconstrained controller presented previously in Sect. 6.6. However, by introducing input, state, and output constraints to the system, the predicted cost J_∞ diverges from the real cost. To relieve this situation the idea was to use a finite set Ω around the state-space equilibrium, in which a local stabilizing law took over, thus the sum of the finite horizon and terminal cost was in fact equal to the real cost.

The use of a Lyapunov function as a cost to ensure stability for systems without system constraints has been considered relatively early in [16] by Chen and Shaw. This approach turned out to be valid and current stable MPC formulations are based on this idea. Subsequently a Lyapunov function-based stability guarantee has been worked out for continuous systems [37], while the first application of this concept for a discrete constrained system has been presented in [28] and later in [4]. These works present a general treatment of the stability issue in MPC which is based on the monotonicity of the decreasing cost function.

7.1.5 State Contractility and Others

Instead of using a Lyapunov cost function, an alternative way to ensure stability is to require that the two-norm of the state is contracting. Mathematically this is given as:

$$\|x_{k+1}\| \leq \alpha \|x_k\| \quad \text{where} \quad \alpha < 1 \tag{7.4}$$

Stability is achieved independently of the parameters of the cost function. This method has been characterized in [44, 45, 64] while it is further expanded for nonlinear systems by de Oliveira [18]. Later Bemporad has proposed an MPC method with stability guarantees utilizing a quadratic Lyapunov function similarly to the methods employing target sets [3]. As it turns out, contractility-based methods have been useful to earn guaranteed stability, albeit with considerable performance loss. Because the norm of the system state is required to be contracting at all times, its course is constantly forced to be outside the ideal trajectory. In essence, the contractility condition introduced above is a Lyapunov function, thus suitable for stability guarantees.

Yet other approaches are based on confining the final state to a terminal set while requiring this set to be stabilizing [38, 43, 47].

7.2 Closed-Loop Stability of the Infinite Horizon MPC Law

The possible issues with stability of a constrained MPC system are demonstrated if one plots the evolution of the cost J_k overtime for a system without disturbance and a perfect model match. The value of the optimal cost function should be decreasing steadily; however due to nonlinear nature of the closed-loop MPC law the cost can be increasing, even if the overall response remains stable. This variation indicates that the closed-loop input trajectory does not follow an optimal predicted trajectory since the predicted cost does not steadily decrease overtime. Given an otherwise stable linear time-invariant system, it is possible to choose an initial condition, which will render the MPC controlled constrained system unstable.

Formally, the stability of the constrained MPC law can be evaluated by considering the cost function J_k as a Lyapunov function. The aim of the Lyapunov stability analysis is not to assess stability for each individual controller; on the contrary, the aim is to establish the conditions under which the MPC control law will be stable in general. Given the knowledge of these conditions, one may create a modified MPC controller that does ensure guaranteed stability. As it will be demonstrated, the stability of the closed-loop system is closely related to the feasibility of the process. The generic stability analysis for a constrained MPC strategy has been first introduced by Mayne et al. in [38].

According to the Lyapunov stability theorem [35] which has been applied to discrete systems by Bertram and Kalman [5], we may define x_0 as an *equilibrium* point of a system $x_{k+1} = f(x)$ if and only if $f(x_0) = x_0$. The function $f(x)$ is actually the state equation as defined by (6.2). The natural equilibrium point of our interest is then $x = 0$ which is located at $f(0) = 0$. It is possible to define a *stable equilibrium* similarly. The state $x = 0$ is a stable equilibrium of a system if for all $k > 0$ the state remains within an arbitrarily small region of the state-space containing $x = 0$ whenever the initial condition $x(0)$ lies sufficiently close to $x = 0$. Mathematically this can be given as for all $R > 0$ exists $r > 0$ such that for all $k > 0$ [2, 14, 35]:

$$\|x(0)\| < r \Rightarrow \|x(k)\| < R \tag{7.5}$$

According to Lyapunov's second method for stability whenever $\|x\|$ is sufficiently small, $x = 0$ will be a stable equilibrium point if there exists a continuously differentiable scalar function $V(x)$ which [14, 35]:

1. positive definite
2. $V(f(x)) - V(x) \leq 0$ holds

and where $x_{k+1} = f(x)$.

It is possible to define asymptotic convergence in a similar fashion. According to this $l(x)$ will converge to zero, meaning that as $k \to \infty$ the series $l(x) \to 0$, if $V(x)$ is a continuously differentiable scalar function and is

1. positive definite
2. $V(f(x)) - V(x) \leq -l(x) \leq 0$ holds

After rearranging the second condition we get the following statement

$$l(x) \leq V(x_k) - V(x_{k+1}) \tag{7.6}$$

which after summing both sides $k = 0, 1, 2, \ldots, \infty$ will render to

$$\sum_{k=0}^{\infty} l(x) \leq V(x_0) - \lim_{k \to \infty} V(x_k) \tag{7.7}$$

The right hand side of this equation is finite, which has an important consequence on the convergence of $l(x)$. The finite nature of the right hand side of the above equation comes from the fact that the $V(x_k) \geq 0$ or in other words the function is positive definite. The second Lyapunov stability gives $V(x_k) \geq V(x_{k+1})$, meaning that the function value at the next step is smaller than the previous. This implies that $V(x_k)$

approaches a finite limit as $k \to \infty$. This is because if the first term on the right side is finite, the second term must be smaller than the first one, and their difference is also finite. *If the right hand side of the equation is a finite number the value of $l(x)$ must approach zero with $k \to \infty$*. This comes from the fact that if one sums an infinite series and the sum is a finite value, then the series must converge to zero.

In this light, it is possible to perform a stability analysis for the MPC control law. Assuming that the MPC problem is feasible, the optimal predicted cost (6.30) is simply a function of the current state x_k, considering the cost function J_k as a Lyapunov function:

$$J_k(x_k) = V(x_k) \tag{7.8}$$

To fulfill the asymptotic convergence property defined previously, the cost function has to be positive definite. The optimal predicted cost J_k is a positive definite function of x_k if either of the following two conditions hold [14, 36]:

1. $\mathbf{Q}$ is positive definite
2. The pair $(\mathbf{A}, \mathbf{Q}^{1/2})$ is observable

where $\mathbf{Q} = \mathbf{Q}^{1/2T}\mathbf{Q}^{1/2}$. The first condition ensures that the first term and hence the entire sum in (6.30) is positive definite.

If the terminal weight $\mathbf{P}_f$ is chosen in a way that J_k is an infinite horizon cost and the optimal predicted input sequence $\mathbf{u}_k$ computed at time (k) is feasible for the optimization problem at $(k+1)$ then the optimal predicted cost is nonincreasing and satisfies [38]:

$$J_{k+1} - J_k \leq -\left(x_k^T \mathbf{Q} x_k + u_k^T \mathbf{R} u_k\right) \tag{7.9}$$

Given that the optimal predicted input sequence $\mathbf{u}_k$ is feasible for the problem at $(k+1)$, (7.9) holds because the optimal cost at current time (k) must be at least as small (or smaller) as the cost for the *tail* of the optimal sequence predicted for the previous sample. The condition of the nonincreasing cost in (7.9) is also referred to as the *direct stability* method and it is utilized to prove the stability of the constrained MPC law in several works [15, 28, 37, 39, 56].

To demonstrate the previous statement, let us take the optimal predicted sequence $\mathbf{u}_k$ at (k) and take it as a basis for the prediction at the next time step, that is, at $(k+1)$:

$$\tilde{\mathbf{u}}_{k+1} = [u_{k+1}\ u_{k+2}\ \ldots\ u_{k+n_c-1}\ \mathbf{K}x_{k+n_c}] \tag{7.10}$$

where $\tilde{\mathbf{u}}_{k+1}$ is referred to as the *tail* of $\mathbf{u}_k$ and it is illustrated in Fig. 7.2. The last element of the tail is given by the fixed feedback law in mode 2, therefore it is according to $u_{k+n_c} = \mathbf{K}x_{k+n_c}$.

The cost function J_k in (6.30) expresses an infinite horizon cost. The cost $\tilde{J}_k$ associated with the tail $\tilde{\mathbf{u}}_{k+1}$ is the cost J_k at (k) minus the term which remains to that time:

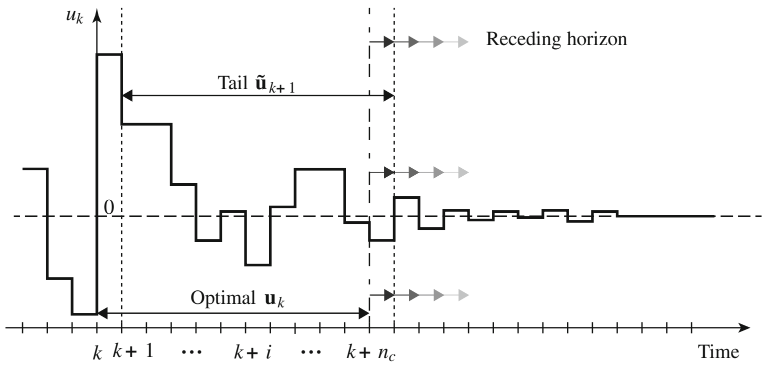


Fig. 7.2 The optimal prediction $\mathbf{u}_k$ at time (k) and its tail $\tilde{\mathbf{u}}_{k+1}$

$$\begin{aligned}
\tilde{J}_{k+1} &= \sum_{i=1}^{n_c}\left(x_{k+i}^T\mathbf{Q}x_{k+i} + u_{k+i}^T\mathbf{R}u_{k+i}\right) + x_{k+n_c+1}^T\mathbf{P}_f x_{k+n_c+1} \\
&= \sum_{i=1}^{\infty}\left(x_{k+i}^T\mathbf{Q}x_{k+i} + u_{k+i}^T\mathbf{R}u_{k+i}\right) \\
&= \sum_{i=0}^{\infty}\left(x_{k+i}^T\mathbf{Q}x_{k+i} + u_{k+i}^T\mathbf{R}u_{k+i}\right) - \left(x_k^T\mathbf{Q}x_k + u_k^T\mathbf{R}u_k\right) \\
&= J_k - \left(x_k^T\mathbf{Q}x_k + u_k^T\mathbf{R}u_k\right)
\end{aligned}$$

In reality, the tail $\tilde{\mathbf{u}}_{k+1}$ will be suboptimal at that time because it is based on the optimal predictions at the previous step (k). The optimal value at $(k+1)$ will satisfy [14]

$$J_{k+1} \le \tilde{J}_{k+1} = J_k - \left(x_k^T\mathbf{Q}x_k + u_k^T\mathbf{R}u_k\right) \tag{7.12}$$

implying the condition given by (7.9).

The important aspect of this stability analysis helps to formulate algorithms that do ensure the stability of the closed-loop system. To summarize, the previously introduced conditions [14]:

If J_k is a positive definite infinite horizon cost, then $x_k = 0$ is a stable equilibrium for the closed-loop system and x_k converges asymptotically to zero, if the tail $\tilde{\mathbf{u}}_{k+1}$ is feasible for all $k > 0$.

The second method to prove the stability of a constrained MPC law also referred to as the *indirect stability* method originates from [16]:

$$J_k - J_{k+1} > 0 \quad \text{for} \quad x \neq 0 \tag{7.13}$$

If it is possible to prove that the left side of Eq. (7.13) is positive, the stability of the control course is proven as well. This approach is discussed in more detail in the works of Chen and Shaw [16] and others [6, 46].

7.3 Stability Through Terminal Constraints

It has been demonstrated in the Sect. 7.2 that a closed-loop MPC control law will remain asymptotically stable, given that the *tail* of the input predictions generated at time (k) will satisfy constraints at times $k = 1, 2, 3, \dots$ and onwards. Fortunately, it is not necessary to force the system to comply with the constraints from the present time all to infinity, since this problem would not be possible to formulate with a finite number of constraints.

This requirement on the feasibility of the tail $\tilde{\mathbf{u}}_{k+1}$ is at least partly satisfied by the problem formulation itself. The constraints will be satisfied for $\tilde{\mathbf{u}}_{k+1}$ in the first $n_c - 1$ sampling intervals of the n_c steps long prediction horizon. This is true because the optimal predictions at time (k) must satisfy the constraints. However, $\tilde{\mathbf{u}}_{k+1}$ has one more element, that is, the n_c-th element $u_{k+n_c} = \mathbf{K}x_{k+n_c}$ as defined by the mode 2 fixed feedback control law.

To guarantee the feasibility of the last element and therefore the whole tail at $(k + 1)$, we must define an additional constraint at time (k). Constraints additional to the constraints arising from the problem definition are referred to as *terminal constraints*. Terminal constraints are defined in mode 2, where the fixed feedback law is active and therefore they are defined in terms of the terminal state prediction that is, x_{k+n_c}.

We can define a region of the state-space Ω in which the terminal state prediction x_{k+n_c} must lie in order to satisfy the terminal constraints. If we have a system with input constraints $\underline{u} \leq u_k \leq \overline{u}$ and state constraints $\underline{x} \leq x_k \leq \overline{x}$, to ensure that the tail will satisfy constraints over the whole prediction horizon n_c we must include a terminal constraint in the following form

$$x_{k+n_c} \in \Omega \quad \Rightarrow \quad \begin{array}{l} \underline{u} \leq \mathbf{K}x_{k+n_c} \leq \overline{u} \\ \underline{x} \leq x_{k+n_c} \leq \overline{x} \end{array} \tag{7.14}$$

defining a region in the state-space Ω. The set Ω is a region where we want to steer the state by the end of the horizon, and it is referred to as the *target set*. The terminal constraints must be computed in a way that they ensure the feasibility of the MPC optimization recursively, that is the tail predictions ensure constraints including the terminal constraints themselves.

This is however not enough, as the predictions generated by the tail $\tilde{\mathbf{u}}_{k+1}$ at $(k+1)$ must also satisfy the next terminal constraint, or in other words the terminal state x_{k+n_c+1} at $(k + 1)$ must also be a part of the region Ω:

$$x_{k+n_c+1} \in \Omega \tag{7.15}$$

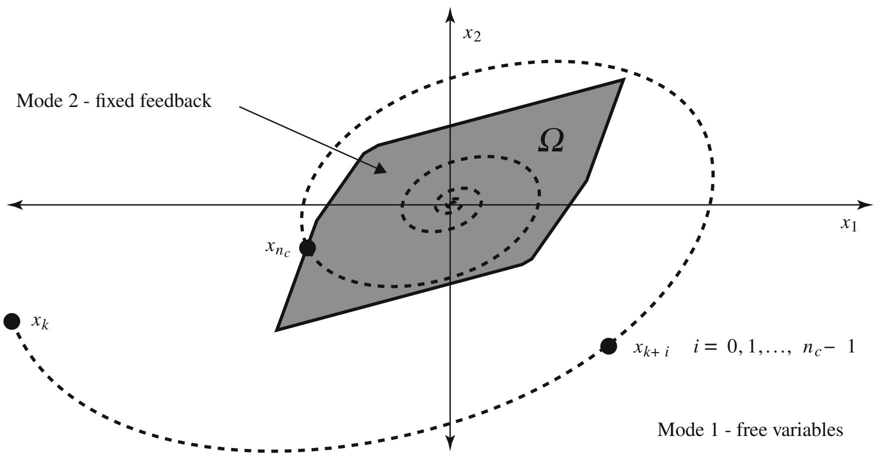


Fig. 7.3 Illustration of the invariant target set Ω with the state trajectory. The area outside the target set represents mode 1 predictions, where inputs are free optimization variables. The target set is mode 2, where the inputs are assumed to be calculated by a fixed feedback law

A target set is illustrated for a two-dimensional state-space in Fig. 7.3. The area outside the target set is where the inputs are assumed to be free optimization variables, while inside the target set the inputs are calculated by the fixed feedback law. Once the state trajectory enters target set Ω it cannot exit it. This is in fact the invariance property of the set Ω, which can be interpreted in a way that the set is contracting. If the state is the part of the set at (k) so must be at the next and subsequent times as well.

To ensure the stability of the closed-loop MPC system, we must ensure the feasibility of the tail $\tilde{\mathbf{u}}_{k+1}$ at time $(k+1)$ whenever the MPC optimization is feasible at time (k). To achieve this we must ensure recursive feasibility. The conditions to ensure this are

1. system constraints are in Ω
2. Ω is invariant in mode 2

The first condition is simply a restatement of (7.14). Invariance of the region Ω then means that if the terminal state is part of the region, then so must be its closed-loop iteration according to the fixed feedback law [8, 9]:

$$x_{k+n_c} \in \Omega \Longrightarrow (\mathbf{A} + \mathbf{BK})x_{k+n_c} \in \Omega \tag{7.16}$$

The constrained MPC optimization has been defined by Algorithm 6.1 without stability guarantees. Now we will add the terminal constraints, so that the stability of the constrained MPC optimization will be now guaranteed. If we would like to steer system (6.1) into the origin, we may define the following algorithm:

Algorithm 7.1 To find the solution of the constrained model predictive control problem with guaranteed stability, perform the following set of operations at each sampling instant:

- Observe or measure actual system state at sample x_k.
- Minimize the following cost function with respect to constraints:

$$\min_{\mathbf{u}_k} J(\mathbf{u}_k, x_k) = \sum_{i=0}^{n_c-1} \left(x_{k+i}^T \mathbf{Q} x_{k+i} + \mathbf{u}_{k+i}^T \mathbf{R} \mathbf{u}_{k+i} \right) + x_{k+n_c}^T \mathbf{P}_f x_{k+n_c}$$

  where $\mathbf{u}_k = \left[u_k, u_{k+1}, u_{k+2}, \ldots, u_{k+n_c-1} \right]$, $\mathbf{Q} = \mathbf{Q}^T \geq 0$ is a state penalization matrix, $\mathbf{R} = \mathbf{R}^T \geq 0$ is an input penalization matrix, n_c is the prediction horizon and $\mathbf{P}_f$ is the solution of the unconstrained, infinite horizon quadratic regulation problem. The typical MPC cost function must be subject to the following system and terminal constraints:

$$\underline{u} \leq u_{k+i} \leq \overline{u}, \qquad i = 0, 1, 2, \ldots, n_c - 1 \tag{7.17}$$

$$\underline{x} \leq x_{k+i} \leq \overline{x}, \qquad i = 1, 2, \ldots, n_c \tag{7.18}$$

$$x_{k+n_c} \in \Omega \tag{7.19}$$

$$x_{k+0} = x_k \tag{7.20}$$

$$x_{k+i+1} = \mathbf{A} x_{k+i} + \mathbf{B} u_{k+i}, \qquad i \geq 0 \tag{7.21}$$

$$y_{k+i} = \mathbf{C} x_{k+i}, \qquad i \geq 0 \tag{7.22}$$

$$u_{k+i} = \mathbf{K} x_{k+i}, \qquad i \geq n_c \tag{7.23}$$

  where $\mathbf{K}$ is a stabilizing feedback gain.
- Apply the first element of the vector of optimal control moves $\mathbf{u}_k$ to the controlled system, and restart the procedure.

If Ω satisfies system constraints and is invariant in mode 2, then a system which is feasible at k will also remain feasible for $k = 1, 2, 3, \ldots, \infty$.

We may define the *region of attraction*, which is a subset of the state-space defining the set of all initial conditions from which is possible to drive the state predictions inside Ω over the n_c steps long mode 1 horizon [29]. Let us denote this set with S_Ω, then formally the above statement means that S_Ω is a collection of all initial conditions x_0 for which exists a vector of inputs $\mathbf{u}$ such that by the end of the horizon x_{n_c} will be a part of the set Ω:

$$S_\Omega = \left\{ x_0 : \text{exists } \ \mathbf{u}_k \text{ such that } \begin{array}{l} x_{n_c} \in \Omega \\ \underline{u} \leq u_{k+i} \leq \overline{u} \\ \underline{x} \leq x_{k+i+1} \leq \overline{x} \end{array} \right\} \tag{7.24}$$

where $i = 0, 1, \ldots, n_c - 1$. The region of attraction is the operating region of the MPC law. No state outside the region of attraction is feasible; therefore, it is in our

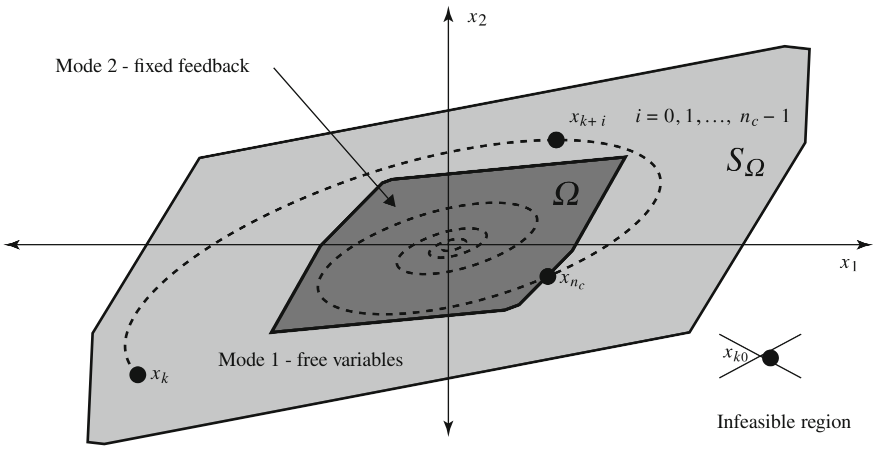


Fig. 7.4 Illustration of the region of attraction S_Ω, which is a set of all feasible initial states x_k. The target set Ω is illustrated as the smaller *dark* area and it is a subset of the region of attraction

interest to make it as large as possible. If one chooses to increase the horizon n_c the size of the region of attraction S_Ω will also increase as well, since the number of steps in which is possible to reach the target set is larger. Another approach is to increase the size of the target set Ω.

The region of attraction S_Ω is illustrated in Fig. 7.4 as the larger *shaded* area. All states within the region of attraction are feasible, which means that it is possible to steer them into the target set within n_c steps. The target set Ω is a subset of the region of attraction. States outside the region of attraction are infeasible and would violate constraints sometime in the future. A state trajectory is denoted with the *dotted* spiral shaped line. Mode 1 control is effective within the region of attraction, however the fixed feedback law in mode 2 is assumed active within the target set.

7.4 Maximal Invariant Terminal Set

For a given horizon length it is necessary to ensure the largest possible region of attraction, thus enlarging the operating region of the MPC controller. To do this, one must formulate the terminal constraints in a way that the largest possible target set Ω is created. The question of maximal admissible sets has been discussed by Gilbert and Tan in [23], while its ellipsoidal approximation has been given by Mayne et al. [38].

The largest possible target set is ensured if one creates a set of terminal constraints, which enforce system constraints over a horizon n_a called the *constraint checking horizon*. For an LTI system, the length of this constraint checking horizon is constant and determined offline.

Recursive feasibility requires that the system constraints are satisfied over the entire mode 2 horizon. This means that state and input constraints must be enforced for:

$$\underline{u} \le u_{k+i} \le \overline{u}, \quad i = n_c, n_c+1, n_c+2, \ldots, \infty \tag{7.25}$$

$$\underline{x} \le x_{k+i} \le \overline{x}, \quad i = n_c, n_c+1, n_c+2, \ldots, \infty \tag{7.26}$$

from which it is logical to assume that the largest possible target set is defined, if we enforce system constraints over the mode 2 horizon by assuming that Ω is formulated as follows:

$$\Omega = \left\{ x : \underline{u} \le \mathbf{K}(\mathbf{A}+\mathbf{B}\mathbf{K})^i x \le \overline{u}, \underline{x} \le (\mathbf{A}+\mathbf{B}\mathbf{K})^i x \le \overline{x} \right\} \tag{7.27}$$

for $i = n_c, n_c+1, n_c+2, \ldots, \infty$. Fortunately, instead of having an infinite set of constraints one may satisfy the constraints over the whole infinitely long mode 2 horizon by an n_a steps long finite horizon. Therefore, (7.27) will be defined instead of an infinitely long constraint checking horizon over $i = n_c, n_c+1, \ldots, n_a-1, n_a$.

To prove that a finite steps long constraint checking horizon can ensure system constraints over the infinite mode 2 horizon, let us assume a case where only input constraints $\overline{u}$ and $\underline{u}$ are considered. Moreover, let us denote Π_i as the set of initial conditions for which the input constraints are satisfied over a certain horizon length of n steps under a mode 2 fixed feedback law given by $u = \mathbf{K}x$ [14, 36]:

$$\Pi_i = \left\{ x : \underline{u} \le \mathbf{K}(\mathbf{A}+\mathbf{B}\mathbf{K})^i x \le \overline{u} \right\} \tag{7.28}$$

For this case, we may also assume that the set of initial conditions satisfying system constraints for the infinitely long mode 2 horizon can be replaced by a set of initial conditions determined by a finite steps long constraint checking horizon:

$$\Pi_\infty = \Pi_{n_a} \tag{7.29}$$

The closed-loop system $(\mathbf{A}+\mathbf{B}\mathbf{K})$ defines a stable matrix with eigenvalues smaller than the one according to $|\kappa\{\mathbf{A}+\mathbf{B}\mathbf{K}\}| < 1$. As we increase i into infinity, this matrix term will approach zero:

$$(\mathbf{A}+\mathbf{B}\mathbf{K})^i \to 0 \quad \text{as} \quad i \to \infty \tag{7.30}$$

Let us denote the perpendicular distance of the hyperplane defined by $(\mathbf{A}+\mathbf{B}\mathbf{K})^i x = \overline{u}$ from $x = 0$ by o_i. For any given x this distance o_i will tend to infinity as $i \to \infty$ [14]:

$$o_i = \frac{\overline{u}}{\left\| \mathbf{K}(\mathbf{A}+\mathbf{B}\mathbf{K})^i \right\|_2} \to \infty \quad \text{as} \quad i \to \infty \tag{7.31}$$

since the term $(\mathbf{A}+\mathbf{B}\mathbf{K})^i$ will tend to zero with increasing i because the closed-loop matrix is stable. This geometrically means that the upper constraints $\overline{u}$ and lower constraints $\underline{u}$ define hyperplanes associated with the increasing horizon. The strip between these hyperplanes will increase in size, and will be wider and wider as i approaches infinity.

For a constrained system, it is always possible to conceive an initial state that will go over system constraints sooner or later in the future, indicating that mode 2 constraints create a finite set in the overall state-space. This means that constraints must be violated sometimes in the future if the initial state x_0 is large enough, therefore Π_∞ must be actually finite assuming an observable pair $(\mathbf{A}+\mathbf{BK})$, $\mathbf{K}$. The strip of state-space defined by the upper and lower input constraint

$$\left\{x : \underline{u} \leq \mathbf{K}(\mathbf{A}+\mathbf{BK})^i x \leq \overline{u}\right\} \tag{7.32}$$

contains Π_∞ for all $i > n_a$ for a finite n_a. This implies that a finite horizon n_a must exist which ensures system constraint feasibility for the whole infinitely long mode two horizon. Then according to this, all points located in the finite Π_{n_a} are also located in Π_∞ .

Let us search for $\Pi_\infty = \Pi_{n_a}$ by assuming that increasing the horizon n_a will eventually cause that the next set will be equivalent to the previous and no improvements can be made in the size or $\Pi_{n_a+1} = \Pi_{n_a}$. Formally we can state that [14]:

Theorem 7.1 $\Pi_\infty = \Pi_{n_a}$ *if and only if* $\Pi_{n_a+1} = \Pi_{n_a}$ To see that Theorem 7.1 is true, let us follow the next line of thought:

Proof Assume that if $x \in \Pi_{n_a}$ then x is also in the next set $x \in \Pi_{n_a+1}$ for some n_a:

$$x \in \Pi_{n_a} \Longrightarrow x \in \Pi_{n_a+1} \Longrightarrow \underline{u} \leq \mathbf{K}(\mathbf{A}+\mathbf{BK})^{n_a+1} x \leq \overline{u} \tag{7.33}$$

Let us replace x with $x = (\mathbf{A}+\mathbf{BK})\check{x}$ and substitute it to the previous equation to get

$$\underline{u} \leq \mathbf{K}(\mathbf{A}+\mathbf{BK})^{n_a+1} \check{x} \leq \overline{u} \Longrightarrow \underline{u} \leq \mathbf{K}(\mathbf{A}+\mathbf{BK})^{n_a+2} \check{x} \leq \overline{u} \tag{7.34}$$

This means that whenever the state $x \in \Pi_{n_a+1}$ it is also $x \in \Pi_{n_a+2}$. If we apply this repeatedly, we can get to the conclusion that Eq. (7.33) also implies that for some n_a the state will be in the infinite mode 2 horizon $x \in \Pi_{n_a+1}$.

The practical meaning of Theorem 7.1 and Eq. (7.33) is that, in the case of input constraints it is sufficient to check whether the mode 2 constraints enforced over n_a also satisfy constraints over n_a+1 steps. If this is true, then n_a is the constraint checking horizon. At each iteration, one must check whether the following statement is true:

$$\underline{u} \leq \mathbf{K}(\mathbf{A}+\mathbf{BK})^i x \leq \overline{u}, i = 0, 1, 2, \ldots, n_a \Longrightarrow \underline{u} \leq \mathbf{K}(\mathbf{A}+\mathbf{BK})^{n_a+1} x \leq \overline{u} \tag{7.35}$$

For a case with input constraints, the constraint checking horizon algorithm is [13]:

Algorithm 7.2 To compute the constraint checking horizon, initialize with an $n_a = 0$ long horizon in mode 2 and

1. Evaluate two linear programs,[2] one for the lower constraint $\underline{u}$ and one for the upper constraint $\overline{u}$ subject to constraints:

$$u_{\max,j} = \max_{x} \mathbf{K}_j(\mathbf{A}+\mathbf{BK})^{n_a+1}x \quad s.t. \quad \underline{u} \leq \mathbf{K}(\mathbf{A}+\mathbf{BK})^{i}x \leq \overline{u} \tag{7.36}$$

$$u_{\min,j} = \min_{x} \mathbf{K}_j(\mathbf{A}+\mathbf{BK})^{n_a+1}x \quad s.t. \quad \underline{u} \leq \mathbf{K}(\mathbf{A}+\mathbf{BK})^{i}x \leq \overline{u} \tag{7.37}$$

   where the constraints are formulated for $i = 0, 1, 2, \ldots, n_a$ and for input constraints[3] $j = 1, 2, \ldots, n_u$.
2. Check whether $u_{\max,j} \leq \overline{u}_j$ and $u_{\min,j} \geq \overline{u}_j$ for each $j = 1, 2, \ldots, n_u$. If this is satisfied, terminate the program and note n_a as the constraint checking horizon, otherwise continue.
3. Increase $n_a = n_a + 1$ horizon and resume at 2…

In other words, at each iteration (for each constrained input) we check whether the constraints over the first n_a steps will also satisfy constraints for the $n_a + 1$-th step. To do this we simply compute the maximal (minimal) input value which would be possible to compute with the fixed feedback law $\mathbf{K}(\mathbf{A}+\mathbf{BK})^{n_a+1}x$ at the next step at $n_a + 1$, given that the computation is constrained from steps $i = 0, \ldots, n_a$ with $\underline{u} \leq \mathbf{K}(\mathbf{A}+\mathbf{BK})^{i}x \leq \overline{u}$. If the computed maximal (minimal) value is larger (smaller) than the constraints, the process is repeated and the horizon is increased. On the other hand, if the constraints are satisfied, the program is terminated and the current n_a is the constraint checking horizon. To satisfy constraints over the infinite mode 2 long horizon, it is enough to solve a finite set of linear programs at each iteration to get to a finite constraint checking horizon n_a.

A maximal invariant target set is illustrated for the two-dimensional state-space ($n_x = 2$) for a system with symmetric input constraints in Fig. 7.5. The terminal constraints iterated through the constraint checking horizon enclose strips in the hyperspace. The width of these hyperplanes grows and they will be rotated around the origin as the constraint checking horizon grows. Finally there will be a set of strips, which encloses the maximal possible invariant target set $\Pi_{n_a} = \Pi_\infty$.

7.4.1 Implementing the Terminal Constraints

With the explicit knowledge of the constraint checking horizon n_a after performing Algorithm 7.2 we can change Algorithm 7.1 defining the dual-mode infinite horizon MPC problem with guaranteed stability to:

Algorithm 7.3 To find the solution of the constrained infinite horizon dual-mode model predictive control problem with guaranteed stability, perform the following set of operations at each sampling instant:

[2] Each with n_u complexity, if the input has n_u dimensions.

[3] Note that although x is the optimization variable, we are not searching which x maximizes (minimizes) this function but on the contrary, the value of the function which is not a state but an input value.

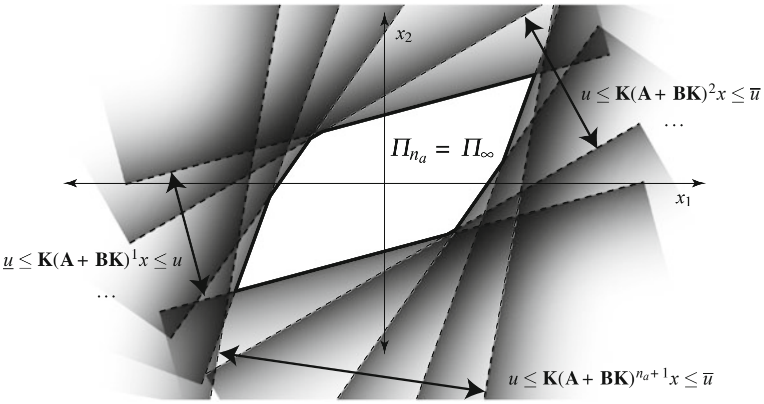

x_2
x_1
$\Pi_{n_a} = \Pi_\infty$
$u \leq \mathbf{K}(\mathbf{A}+\mathbf{BK})^2 x \leq \overline{u}$
...
$\underline{u} \leq \mathbf{K}(\mathbf{A}+\mathbf{BK})^1 x \leq u$
...
$u \leq \mathbf{K}(\mathbf{A}+\mathbf{BK})^{n_a+1} x \leq \overline{u}$

Fig. 7.5 Illustration of a maximal invariant target set in two-dimensional state-space. The strips defined by the terminal constraints create a maximal invariant target set $\Pi_{n_a} = \Pi_\infty$

- Observe or measure actual system state at sample x_k.
- Minimize the following cost function with respect to constraints:

$$\min_{\mathbf{u}_k} J(\mathbf{u}_k, x_k) = \sum_{i=0}^{n_c-1} \left(x_{k+i}^T \mathbf{Q} x_{k+i} + \mathbf{u}_{k+i}^T \mathbf{R} \mathbf{u}_{k+i} \right) + x_{k+n_c}^T \mathbf{P}_f x_{k+n_c}$$

where $\mathbf{u}_k = \left[u_k, u_{k+1}, u_{k+2}, \ldots, u_{k+n_c-1}\right]$, $\mathbf{Q} = \mathbf{Q}^T \geq 0$ is a state penalization matrix, $\mathbf{R} = \mathbf{R}^T \geq 0$ is an input penalization matrix, n_c is the prediction horizon and $\mathbf{P}_f$ is the solution of the unconstrained, infinite horizon quadratic regulation problem, The MPC cost function must be subject to the following system and terminal constraints:

$$\underline{u} \leq u_{k+i} \leq \overline{u}, \qquad i = 0, 1, 2, \ldots, n_c - 1 \tag{7.38}$$

$$\underline{x} \leq x_{k+i} \leq \overline{x}, \qquad i = 1, 2, \ldots, n_c \tag{7.39}$$

$$\underline{u} \leq \mathbf{K}(\mathbf{A}+\mathbf{BK})^i x_{k+n_c} \leq \overline{u}, \qquad i = 0, 1, 2, \ldots, n_c, \ldots, n_c + n_a \tag{7.40}$$

$$\underline{x} \leq (\mathbf{A}+\mathbf{BK})^i x_{k+n_c} \leq \overline{x}, \qquad i = 0, 1, 2, \ldots, n_c, \ldots, n_c + n_a \tag{7.41}$$

$$x_{k+0} = x_k \tag{7.42}$$

$$x_{k+i+1} = \mathbf{A} x_{k+i} + \mathbf{B} u_{k+i}, \qquad i \geq 0 \tag{7.43}$$

$$y_{k+i} = \mathbf{C} x_{k+i}, \qquad i \geq 0 \tag{7.44}$$

$$u_{k+i} = \mathbf{K} x_{k+i}, \qquad i \geq n_c \tag{7.45}$$

where $\mathbf{K}$ is a stabilizing feedback gain, n_c is the control and prediction horizon and n_a is the control checking horizon.
- Apply the first element of the vector of optimal control moves $\mathbf{u}_k$ to the controlled system, and restart the procedure.

7.4.2 Horizon Length

The inclusion of the terminal constraints in this quadratic programming problem does not increase the computational cost significantly, since most of the online computation effort is spent on the QP itself. The additional terminal constraints over the n_a steps long constraint checking horizon are linear. As it will be demonstrated later, computational time may be saved by using other types of terminal constraints. This is not resulting because of the direct online computational savings on the additional constraints, but rather because alternative constraint formulations may also allow for alternative formulations of the minimization of the cost itself.

The size of the feasible initial conditions S_Ω will increase with an increased horizon n_c. This proves to be essential with lightly damped vibrating systems, as the large discrepancy between actuator capabilities and expected deformations calls for a large region of attraction. It is likely that in a vibration attenuation application the horizon n_c will be kept at high values anyways.

In addition to enlarging the region of attraction S_Ω, increasing the horizon has other effects as well. That is the increase of the performance of the closed-loop system, or the decrease of the closed-loop cost function

$$J = \sum_{k=0}^{\infty} \left(x_k^T \mathbf{Q} x_k + u_k^T \mathbf{R} u_k \right) \tag{7.46}$$

The performance of the MPC law will increase because of the reduction of the predicted cost.

However, this performance increase has its limits, and there is a certain horizon n_c over which the optimality of the closed-loop system will not improve. This limit is known as the *constrained LQ-optimal performance* and it ensures the performance equivalent to an infinite number of degrees of freedom MPC problem. The reason why the performance cannot be increased beyond a further limit is given by the fact that terminal constraints must be inactive for a sufficiently large n_c, so there cannot be any further reduction to the cost. In the problem of active vibration damping the requirement for a large region of attraction will dominate when the horizon n_c is designed; therefore it is likely that a constrained LQ optimal performance will be reached anyway. It is possible to perform a simulation analysis, where one calculates closed-loop performance J for different horizons in order to assess whether an increased horizon brings an optimality improvement. Alternatively, one may compare the closed-loop cost J to the cost predicted at the initial time J_0 and see if they are identical. If the closed-loop cost is smaller, improvement can be made, but if the two are identical, the constrained LQ-optimal performance has been already reached.

7.5 Simplified Polyhedral Target Sets

As demonstrated before, the inclusion of a constraint checking horizon n_a in dual-mode MPC is the most straightforward way to ensure stability a priori while also reaching maximum performance. However, polyhedral target sets created by process constraints may be very complex in certain cases, especially with higher order prediction models. This complexity directly translates to the computational effort necessary to acquire the evolution of future process inputs at each sampling instant. It is therefore sometimes desirable to approximate the complex polytope $\Pi_{n_a} = \Pi_\infty$ created by constraints with a simplified shape. These polyhedral target sets are in fact created by an assembly of simpler elements: hyperspaces[4] and the half spaces bounded by them [25]. According to this, it is also possible to create an invariant target set that is bounded by a smaller number of hyperspaces but still ensures the stability of the MPC law. Several alternative stabilized MPC approaches rely on such simplifications. The upside is the reduction of the computational effort; however, as the ideal target set is only approximated with a simplified equivalent, the performance of the control law will suffer as well.

Let us consider a simple regulation problem where we would like to steer our system state into zero. For this regulation problem, we are aiming to minimize the cost function such as in (6.30) with respect to constraints. But, instead of using the high complexity target set Π_{n_a} to ensure stability as discussed before, let us imagine a simplified polyhedral target set Π_s instead. The state x shall remain within this set which shall be defined by:

$$\{x : \mathbf{V}_s x \leq 1\} \quad \text{where} \quad \mathbf{V}_s \in \mathbb{R}^{n \times n} \tag{7.47}$$

where $\mathbf{V}_s$ is a matrix defining a simplified polyhedral target set. Essentially, this defines a hypercube and our aim is to determine what matrix $\mathbf{V}_s$ will be. The conditions for invariance are defined by the properties of the set, see for example the paper by Bitsoris [7].

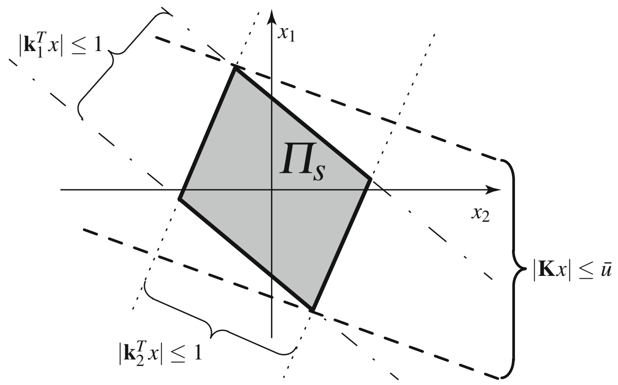
A low complexity polyhedral invariant set is illustrated for a second order system in Fig. 7.6. If at time (k) the set is defined by $|\mathbf{V}_s x_k| \leq 1$, then at the next time step $(k+1)$ should be smaller and described by the following relation:

$$|\mathbf{V}_s x_k| \leq 1 \longrightarrow |\mathbf{V}_s \boldsymbol{\Phi} x_k| \leq 1 \quad \text{where} \quad \boldsymbol{\Phi} = (\mathbf{A} + \mathbf{BK}) \tag{7.48}$$

It is possible to rewrite this by inserting $\mathbf{V}_s^{-1}\mathbf{V}_s = 1$ and obtain $\left|\mathbf{V}_s \boldsymbol{\Phi} \mathbf{V}_s^{-1} \mathbf{V}_s x_k\right| \leq 1$. Let us denote elements of the product $\mathbf{V}_s \boldsymbol{\Phi} \mathbf{V}_s^{-1}$ with k_{ij} and elements of the product $\mathbf{V}_s x_k$ with l_j. The definition of the invariant set (7.47) actually states that in the worst-case scenario the elements of $\mathbf{V}_s x_k$ will equal to 1: $|l_1| = \cdots = |l_n| = 1$. We can utilize a second order system to illustrate the situation:

$$\max_{x \in \Pi_s} |k_{11} l_1 + k_{12} l_2| = |k_{11}| + |k_{12}| \tag{7.49}$$
$$\max_{x \in \Pi_s} |k_{21} l_1 + k_{22} l_2| = |k_{21}| + |k_{22}| \tag{7.50}$$

[4] It is also possible to represent polyhedra in a vertex-based representation instead of hyperspaces, see the book by Ziegler [66].



$|\mathbf{k}_1^T x| \leq 1$
x_1
Π_s
x_2
$|\mathbf{K}x| \leq \bar{u}$
$|\mathbf{k}_2^T x| \leq 1$

Fig. 7.6 Low complexity polyhedral invariant set, where $\mathbf{k}_1 = [k_{11} \quad k_{12}]^T$ and $\mathbf{k}_1 = [k_{21} \quad k_{22}]^T$

then

$$\left|\begin{matrix} |k_{11}| & |k_{12}| \\ |k_{21}| & |k_{22}| \end{matrix}\right| \left|\begin{matrix} 1 \\ 1 \end{matrix}\right| = \left|\begin{matrix} 1 \\ 1 \end{matrix}\right| \tag{7.51}$$

From this, it is clear that the condition of invariance may be transformed to a very convenient form:

$$\left|\mathbf{V}_s \boldsymbol{\Phi} \mathbf{V}_s^{-1}\right| 1 = 1 \tag{7.52}$$

In addition to (7.52) which sufficiently defines invariance, feasibility conditions need to be defined as well. The simplest case is to have symmetric constraints only on the control input. In this case we have to ensure:

$$|\mathbf{K}x_k| \leq \bar{u} \longrightarrow |\mathbf{K}\mathbf{V}_s^{-1}\mathbf{V}_s x_k| \leq \bar{u} \tag{7.53}$$

where $\mathbf{V}_s^{-1}\mathbf{V}_s$ was inserted to the second equation. Similarly to the invariance condition, the definition of the invariant set ensures that $|\mathbf{V}_s x_k| \leq 1$, therefore in the worst case $|l_1| = \cdots = |l_n| = 1$. This simplifies the problem of feasibility to:

$$|\mathbf{K}\mathbf{V}_s^{-1}| 1 \leq \bar{u} \tag{7.54}$$

To find our invariant set defined by $\mathbf{V}_s$ let us state the eigenvalue problem for the matrix $\boldsymbol{\Phi}$:

$$\begin{aligned} \boldsymbol{\Phi}\Delta &= \Delta\Lambda \\ \Delta^{-1}\boldsymbol{\Phi}\Delta &= \Lambda \\ \Delta &= [\delta_1 \ \delta_2 \ \ldots \delta_{n_c}] \\ \Lambda &= \begin{bmatrix} \kappa_1 & \cdots 0 \\ \kappa_2 & \vdots \\ \vdots & \ddots \\ 0 \ \cdots & \kappa_n \end{bmatrix} \end{aligned} \tag{7.55}$$

where $\varDelta$ is a matrix of eigenvectors δ_i, $\varLambda$ is a diagonal matrix containing eigenvalues κ_i. We can utilize the inverse of the eigenvector matrix scaled with α_s to choose a suitable $\mathbf{V}_s$:

$$\mathbf{V}_s = \alpha_s \varDelta^{-1} \tag{7.56}$$

the conditions for invariance will transform to

$$|\alpha_s \varDelta^{-1} \boldsymbol{\Phi} \alpha_s^{-1} \varDelta|1 \leq 1 \longrightarrow |\varLambda|1 \leq 1 \tag{7.57}$$

Equation (7.57) tells us that the eigenvalues of the closed-loop system need to be real and from within the unit disk, more formally: $|\kappa_i| \leq 1$. It is possible to manipulate the eigenvalues of the closed-loop system by pole placement, although this fundamental problem may have a surprisingly large computational complexity [10, 11]. By conforming to the former requirement, it is possible now to rewrite the conditions for feasibility:

$$|\mathbf{K}\alpha_s^{-1}\varDelta| \leq \bar{u} \longrightarrow \alpha_s^{-1}|\mathbf{K}\varDelta|1 \leq \bar{u} \tag{7.58}$$

Finally, the conditions for feasibility will transform into a convenient form:

$$\alpha_s \geq |\mathbf{K}\varDelta|1/\bar{u} \tag{7.59}$$

We can summarize the algorithm for model predictive control with guaranteed stability, utilizing simplified polyhedral invariant target sets as follows:

Algorithm 7.4

- Find the multiplier α_s using relation (7.59)
- Find $\mathbf{V}_s$ defining the simplified polyhedral target set using (7.56)
- Perform the minimization of J_k subject to constraints $|\mathbf{u}_k| \leq \bar{u}$ and $|\mathbf{V}_s x_{n_c}| \leq 1$

The constraints need to be fed to the quadratic programming solver, mostly in the form $\mathbf{A}_c\mathbf{u}_k \leq \mathbf{b}_0 + \mathbf{B}_c x_k$. We have an additional constraint, defining the terminal state $-1 \leq \mathbf{V}_s x_{n_c} \leq 1$, where $x_{k+n_c} = \mathbf{M}_{n_c} x_k + \mathbf{N}_{n_c}\mathbf{u}_k$. Generally, we can define these constraints according to the following relation:

$$\begin{bmatrix} 1 & \cdots & 0 \\ \vdots & 1 & \vdots \\ 0 & \cdots & 1 \\ & \mathbf{V}_s\mathbf{N}_{n_c} & \end{bmatrix} \mathbf{u}_k \leq \begin{bmatrix} \bar{u} \\ \vdots \\ \bar{u} \\ \mathbf{I} \end{bmatrix} + \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \\ & \mathbf{V}_s\mathbf{M}_{n_c} & \end{bmatrix} x_k \tag{7.60}$$

State and output constraints may be defined similarly.

A practical issue with the construction of low complexity invariant target sets is the occurrence of complex eigenvalues κ_i of the closed-loop matrix $\boldsymbol{\Phi}$. In this case, the definition matrix of the polyhedral set $\varGamma$ will contain a pair of complex conjugate eigenvectors and the set will remain open [27]. To solve this situation and close the target set elementary rotation matrices may be used [50], which essentially break down eigenvectors to their real and imaginary components. Such and similar operations may cause that the condition of invariance is not met for certain complex conjugate eigenvalues. Kouvaritakis et al. in their conference article in [31] review for which eigenvalues is the invariance condition still valid. A formulation for continuous systems is also possible [49], while its application to the pole-placement of gain matrices for fixed feedback systems is described by Rusko in [52].

The introduction of non-symmetric constraints $\overline{u} \neq \underline{u}$ or rate of change constraints $\triangle\overline{u}$, $\triangle\underline{u}$, $\triangle\overline{x}$, $\triangle\underline{x}$, and $\triangle\overline{y}$, $\triangle\underline{y}$, may require more complex formulations [26, 27, 31].

7.6 Elliptic Invariant Target Sets

The previous section introduced a formulation where the complex polyhedral set $\Pi_{n_a} = \Pi_\infty$ created by the process constraints in the constraint checking horizon of optimal dual-mode QPMPC were replaced by a simplified invariant target set Π_s. An additional possibility to replace the maximal target set Π_∞ with a simpler approximation is the use of elliptic invariant target sets [38]. Geometric stability guarantees based on the elliptic invariant set formulation are the cornerstone of the efficient algorithm considered in Sect. 8.1 of the upcoming chapter. In general, the construction of ellipsoidal target sets is based on the Lyapunov or Ricatti equation [65] and linear matrix inequalities (LMI).

The shape of the invariant ellipsoidal target set will be an ellipse in the case of a second, an ellipsoid for a third order system. For a second order system, an ellipsoidal set is illustrated in Fig. 7.7. In case the system order is larger than three, we talk about a hyperellipsoid. Unfortunately, it is difficult to illustrate hyperellipsoids graphically without creating confusion; therefore, the illustrations will assume a second order system. Generally, we may describe the target set by the following expression:

$$E_x = \left\{ x | x^T \Gamma x \leq 1 \right\} \tag{7.61}$$

The aim is to find the matrix Γ such that the ellipsoid will enclose the largest invariant target set within the bounds and conforming the constraints. Naturally, it is desirable to make the target set—this case an ellipsoid—as large as possible. The conditions for invariance relate to the basic idea of invariant sets. If a system state in a certain point of time is contained within the ellipsoid, so must it be at the next time step:

$$x_k \in E_x \rightarrow x_{k+1} \in E_x \rightarrow x_{k+2} \in E_x \rightarrow \ldots \tag{7.62}$$

This condition may be expressed by stating that the ellipsoid in the next time step must be smaller or at least equal to the one in the current time steps:

$$x_{k+1}^T \Gamma x_{k+1} \leq x_k^T \Gamma x_k \tag{7.63}$$

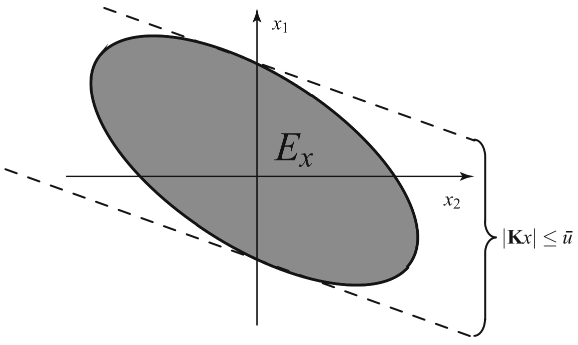


Fig. 7.7 Illustration of an elliptic invariant target set

We can take advantage of the fact that $x_{k+1} = (\mathbf{A}+\mathbf{BK})x_k$, where $\mathbf{K}$ is actually $\mathbf{K}_{LQ}$ in dual-mode. Therefore, if $(\mathbf{A} + \mathbf{BK})$ is substituted by the closed-loop matrix Φ, we obtain

$$x_k^T \boldsymbol{\Phi}^T \Gamma \boldsymbol{\Phi} x_k \leq x_k^T \Gamma x_k \tag{7.64}$$

From this, by rearranging we get

$$-x_k^T \Gamma x_k + x_k^T \boldsymbol{\Phi}^T \Gamma \boldsymbol{\Phi} x_k \leq 0 \tag{7.65}$$

The final condition for invariance of the hyperellipsoid is:

$$\Gamma - \boldsymbol{\Phi}^T \Gamma \boldsymbol{\Phi} \geq 0 \tag{7.66}$$

In addition to the invariance condition, there are input and possibly state constraints present. Let us consider the case of the simple symmetric input constraints, defined by:

$$|\mathbf{K}x| \leq \overline{u} \tag{7.67}$$

Utilizing the identity $\Gamma^{-\frac{1}{2}} \Gamma^{\frac{1}{2}} = \mathbf{I}$, we may transform (7.67) to:

$$|\mathbf{K}\Gamma^{-\frac{1}{2}} \Gamma^{\frac{1}{2}} x|^2 \leq \overline{u}^2 \tag{7.68}$$

which may also be equivalently denoted as:

$$||\mathbf{K}\Gamma^{-\frac{1}{2}}||^2 ||\Gamma^{\frac{1}{2}} x||^2 \leq \overline{u}^2 \tag{7.69}$$

The second term on the left side of (7.69) is simply $x^T \Gamma^{\frac{1}{2}^T} \Gamma^{\frac{1}{2}} x = x^T \Gamma x$. According to Eq. (7.61) $x^T \Gamma x \leq 1$ which implies that the second term can have a value of one in the worst case. Therefore, we may rewrite the conditions of feasibility:

$$||\mathbf{K}\Gamma^{-1}\mathbf{K}^T|| \leq \overline{u}^2 \tag{7.70}$$

To calculate Γ from the conditions of invariance, one needs to employ semidefinite programming or SDP as it is often referred to. First, it is necessary to transform the invariance conditions to a more convenient form using Schur complements. According to Schur complements, the following is valid [13]:

$$\begin{vmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{C} \end{vmatrix} \geq 0 \Longleftrightarrow \begin{array}{l} \mathbf{A} - \mathbf{B}\mathbf{C}^{-1}\mathbf{B}^T \geq 0,\ \mathbf{C} > 0 \\ \mathbf{C} - \mathbf{B}^T\mathbf{A}\mathbf{B} \geq 0,\ \mathbf{A} > 0 \end{array} \tag{7.71}$$

A function in the form $\mathbf{F}(p) > 0$ where $\mathbf{F}(p) = \mathbf{p}_{11}\mathbf{I}_{11} + \mathbf{p}_{12}\mathbf{I}_{12} + \cdots$ is called a linear matrix inequality[5] [12]. The invariance condition is (7.66) and in addition to that it is necessary for the eigenvalues of Γ to be positive, that is, $\Gamma > 0$. If we multiply both sides of the invariance condition by Γ^{-1} we obtain:

$$\begin{array}{r} \Gamma^{-1}(\Gamma - \boldsymbol{\Phi}^T\Gamma\boldsymbol{\Phi})\Gamma^{-1} \geq 0 \\ \Gamma > 0 \end{array} \Longleftrightarrow \begin{array}{r} \Gamma^{-1} - \Gamma^{-1}\boldsymbol{\Phi}^T\Gamma\boldsymbol{\Phi}\Gamma^{-1} \geq 0 \\ \Gamma > 0 \end{array} \tag{7.72}$$

According to Schur complements, it is possible to rewrite this relation to

$$\begin{vmatrix} \Gamma^{-1} & \Gamma^{-1}\boldsymbol{\Phi} \\ \boldsymbol{\Phi}\Gamma^{-1} & \Gamma^{-1} \end{vmatrix} \geq 0 \tag{7.73}$$

Although we have Γ^{-1} instead of Γ, it is possible to calculate Γ^{-1} and invert it afterward. For feasibility, we have

$$\mathbf{K}_i\Gamma^{-1}\mathbf{K}_i^T < \overline{u}_i^2 \tag{7.74}$$

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_1 \\ \mathbf{K}_2 \\ \vdots \\ \mathbf{K}_{n_u} \end{bmatrix}, \quad \bar{\mathbf{u}} = \begin{bmatrix} \overline{u}_1 \\ \overline{u}_2 \\ \vdots \\ \overline{u}_{n_u} \end{bmatrix}$$

where the index i represents the i-th row of the LQ optimal gain matrix $\mathbf{K}$ and the i-th element of the vector $\bar{\mathbf{u}}$ for a general multiple-input system. As previously mentioned, it is necessary to maximize the volume of the hyperellipsoid, or in the case of a second order system, the area of an ellipse-subject to feasibility and invariance constraints. If an ellipse is described by (7.61), then its major and minor axes are defined by the reciprocals of the eigenvalues of the matrix Γ (Fig. 7.8). The area of an ellipse (valid for a second order system) may be calculated by:

$$\mathrm{Vol} E_x = \frac{\pi}{\kappa_1\kappa_2} = \pi \det \Gamma^{-1} \tag{7.75}$$

[5] Most mathematics and optimization-related publications use the notation $\mathbf{F}(p)\succ 0$ to denote the concept of positive definiteness. This book will denote this concept with simple relation signs.

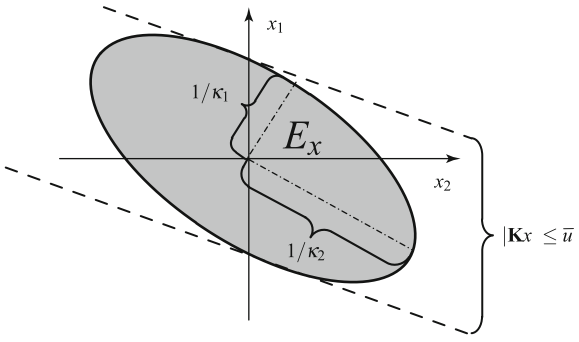


Fig. 7.8 Major and minor axes of an ellipsoid E_x expressed by the reciprocals of the eigenvalues of the definition matrix $\mathit{\Gamma}$

and the volume of a generic hyperellipsoid is calculated by evaluating:

$$\mathrm{Vol} E_x = h \det \mathit{\Gamma}^{-1} \tag{7.76}$$

where h is an unknown number. Finding $\mathit{\Gamma}$ is an offline optimization problem:

Algorithm 7.5 Perform the maximization of the determinant according to [62] in offline mode:

$$\max \mathrm{Vol} E_x = h \det \mathit{\Gamma}^{-1} \tag{7.77}$$

Subject to the invariance condition (7.73), feasibility condition (7.74) and possibly a shape conditioning constraint, for example:

$$h\mathbf{I} < \mathit{\Gamma}^{-1} \text{ or trace } (\mathit{\Gamma}) < 1 \text{ etc} \ldots \tag{7.78}$$

The so-called shape conditioning ensures that the hyperellipsoid axes will not be distorted in a particular direction. That said, it avoids an infinitely thin and long ellipsoid-which otherwise would have the maximal volume and conform to all the conditions. To realize this in practice, one needs to deploy a solver for semidefinite programming (SDP) [42]. For this purpose, a rational choice is the freely available *SeDuMi* solver [46, 59]; with an LMI parser called *YALMIP* [34]. The next problem to solve is the actual online computation. The online algorithm may be described by:

Algorithm 7.6 Solve the following optimization problem at each time instant:

$$\min_u J_k = \mathbf{u}_k^T \mathbf{H} \mathbf{u}_k + 2x_k^T \mathbf{G} \mathbf{u}_k + x_k^T \mathbf{F} x_k \tag{7.79}$$

$$\textit{Feasibility}\text{: } \mathbf{A}_c \mathbf{u}_k \leq \mathbf{b}_0 + \mathbf{B}_c x_k$$

$$\textit{Invariance}\text{: } x_{n_c}^T \mathit{\Gamma} x_{n_c} \leq 1$$

This algorithm presents a quadratic optimization problem with quadratic constraints. The evaluation of this task in the original form may be formidable. Fortunately, this formulation may be changed to a second order cone programming (SOCP) problem. Constraints for a second order cone programming problem are given in the following form:

$$||\mathbf{A}x + \mathbf{b}||_2 \le \mathbf{C}x + \mathbf{d} \tag{7.80}$$

where x is the variable to be optimized, and **A**, **b**, **C** and **d** are given optimization parameters. The expression on the left side of the equation is in the two norm, also referred to as Euclidean norm, where

$$||\mathbf{A}x + \mathbf{b}||_2 = \left[(\mathbf{A}x + \mathbf{b})^T (\mathbf{A}x + \mathbf{b})\right]^{\frac{1}{2}} \tag{7.81}$$

In the light of this information, the original definition of ellipse $x_{n_c}^T \Gamma x_{n_c} \le 1$ can be described equivalently as

$$||\Gamma^{\frac{1}{2}} x_{n_c}||_2 \le 1 \tag{7.82}$$

where x_{n_c} denotes the state at the end of the prediction horizon: $x_{n_c} = \mathbf{M}_{n_c} x_k + \mathbf{N}_{n_c} \mathbf{u}_k$. The new invariance condition in (7.79) will be

$$||\Gamma^{\frac{1}{2}} \mathbf{M}_{n_c} x_k + \Gamma^{\frac{1}{2}} \mathbf{N}_{n_c} \mathbf{u}_k||_2 \le 1 \tag{7.83}$$

The quadratic optimization problem can be transformed into a second order cone programming form as well:

$$\begin{aligned} J_k = {} & (\mathbf{H}^{\frac{1}{2}} \mathbf{u}_k + \mathbf{H}^{-\frac{1}{2}} \mathbf{G}^T x_k)^T (\mathbf{H}^{\frac{1}{2}} \mathbf{u}_k + \mathbf{H}^{-\frac{1}{2}} \mathbf{G}^T x_k) \\ & + x_k^T \mathbf{F} x_k - x_k^T \mathbf{G} \mathbf{H}^{-1} \mathbf{G}^T x_k \end{aligned} \tag{7.84}$$

The last two terms of the equation are negligible, since we have a minimization problem to solve. Therefore, we have a new expression in the following form:

$$||\mathbf{H}^{\frac{1}{2}} \mathbf{u}_k + \mathbf{H}^{-\frac{1}{2}} \mathbf{G}^T x_k||_2 \le h \tag{7.85}$$

where h is a new scalar optimization variable. The optimization algorithm is transformed to:

Algorithm 7.7 Evaluate the following second order cone programming problem at each sampling instant:

$$\min_{\mathbf{u}_k, h} (h) \tag{7.86}$$

Subject to the transformed invariance condition (7.83) and the feasibility condition. The new transformed optimization problem is now expressed as an additional constraint:

.

$$||\mathbf{H}^{\frac{1}{2}} \mathbf{u}_k + \mathbf{H}^{-\frac{1}{2}} \mathbf{G}^T x_k||_2 \le h \tag{7.87}$$

7.7 Infeasibility Handling

There are many kinds of constraints in the MPC formulation that must be taken into account in practical control applications: safety limitations, physical restrictions, technological requirements, product quality specifications, etc. The importance of constraints is also reinforced by the fact that, in practice, the optimal operating point very often lies on one or more of the boundaries defined by the applied constraints.

The relatively simple implementation of equality or inequality constraints into the task of minimizing the criterion function in predictive control [see e.g. the generic QP problem in Eq. (6.61)] may introduce very significant problems related to their compatibility. Incompatible constraints can cause that the optimization problem may be insolvable, respectively the given optimization problem is *infeasible*. The term infeasibility can be defined as an inability to satisfy all the constraints simultaneously. These problems arise when the restrictions on the relevant variables define an empty area and the optimization problem of minimizing the criterion function does not provide an adequate solution. An infeasible set of constraints may occur as a result of disturbances, operator failure, actuator or control system failure, bad design; ultimately causing the optimization problem to become incompatible in certain steps. It may also happen that the numerical algorithm minimizing the criterion leads the system outside the feasible region. The problem of feasibility is often referred to as a compatibility problem or the realization problem of the constraints. A general solution to the feasibility problem does not exist and therefore the issue needs attention in a real control application. In general, the constraints in MPC algorithms can be interpreted as [51]:

- *Hard constraints* are constraints that must always be satisfied. For example, hard constraints may be physical limits on actuators or safety limits. A control scheme ideally shall not use tactics to violate hard constraints, as this is either physically impossible or would lead to catastrophic results. If hard constraints would be violated, a mismatch between the predicted and actual closed-loop plant would occur leading to serious consequences and even loss of stability.
- *Soft constraints* are those, which should be satisfied only if possible. It is assumed that if necessary, soft constraints can be violated (ignored). Soft constraints are usually enforced on output or state variables, although they could also be applied to inputs. In a practical sense, the constraints are nonessential, only preferred.
- *Terminal constraints* are somewhat artificial in a sense that they arise from the stability guarantee conditions of the control algorithm, which have been discussed in detail previously. They can be defined in the form of equality or inequality conditions given on terminal state and terminal region. In fact, they represent a mixture of hard and soft constraints.

We may distinguish two types of infeasibility problems for various incompatible constraint configurations [53]:

- *Type I infeasibility* is caused by incompatibility between equality and inequality constraints, e.g. the inequality constraints define a nonempty region $\Phi_{\text{in}} \neq 0$ and $\Phi_{\text{in}} \cap \Phi_{\text{eq}} \equiv 0$, where Φ_{eq} is the region created by the equality constraints
- *Type II infeasibilities* are caused by incompatibility between the inequality constraints because they define an empty region $\Phi_{\text{in}} \equiv 0$, e.g. $u_k \leq 1$ and $y_k \geq 2$ when the process has unity DC gain

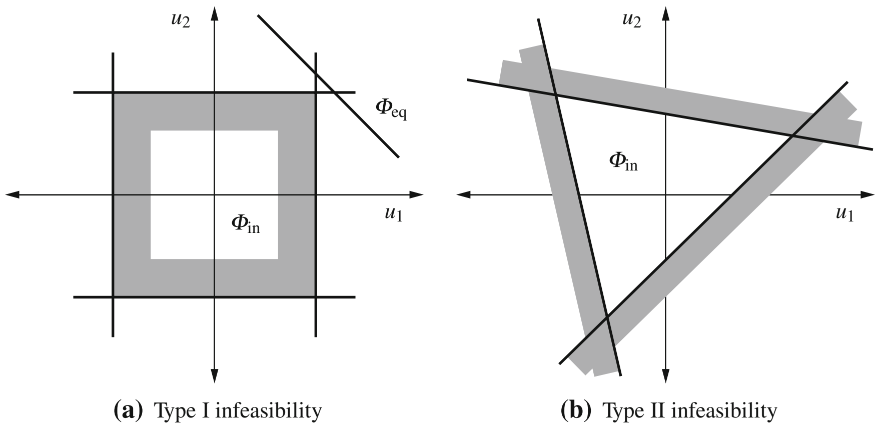


Fig. 7.9 Illustration of the two main types of infeasible constraint configurations for an input constrained problem with two elements. **a** Type I infeasibility. **b** Type II infeasibility

The issue of Type I infeasibilities is very important since some earlier MPC stabilizing strategies (such as CRHPC, SGPC or SIORHC) rely on the end point equality constraints in Eq. (7.1) to ensure stability. It can be shown that, if at least one inequality constraint is imposed, it is always possible to find a set-point sequence causing this type of incompatibility. Careful design of constraints cannot guarantee feasibility; hence, there are situations when such a stabilizing control is infeasible. Type II infeasibilities usually arise because of either poor design, or the nature of the plant and the presence of disturbances. The resulting mismatch between the predicted and actual plant behavior can than lead to serious consequences in online control. A simple example of the both types of infeasibilities is illustrated in Fig. 7.9, where incompatible constraint configurations of Type I and II are given for a two-dimensional input vector $u = [u_1 \quad u_2]^T$.

One may see that there is a possibility that the minimization problem of the MPC cost function subject to design constraints may not have a solution at all. This is why it is necessary to devise procedures for the effective handling of infeasibility. All practical MPC implementations should have means to recover from infeasibility, shall that occur during the real-time control process.

Let us briefly discuss some typical techniques for avoiding infeasibility. One way of handling (Type I) infeasibilities is the *set point management* technique [21, 22]. An obvious case of infeasibility is due to rapid set point changes. This implies a large change in the terminal constraint set (due to a shift in steady-state values) and hence these may become inconsistent. The key philosophy of set point management algorithms is to establish a set point different from the true one, when changes in true set point would otherwise cause infeasibility. The controller set point therefore implements slower changes in the value than the true set point would necessitate. Simple algorithms implementing this set point change strategy can be found in [51].

The next two techniques referred to as *constraint removal* and *constraint softening* can be applied for handling infeasibilities of both types. The idea is, when an infeasibility arises, control is continued as unconstrained. Feasibility is checked at every sample and a full set of constraints are reintroduced back as soon as they become feasible. If the set of constraints is found to be inconsistent, then some constraint must be either relaxed or removed. A simple process-dependent removal strategy could be developed using the following logic [51]:

Algorithm 7.8 At each sampling instant, perform the following algorithm:

- Test for feasibility, if found infeasible then

1. relax (or remove) the least important soft constraint, test for feasibility
2. if the remaining set of constraints is feasible, pass on to the MPC algorithm and start optimization
3. else repeat the cycle again from step 1

- else pass on full set of constraints to the MPC algorithm and start the optimization.

Naturally, the algorithm relates to those constraints that are predicted to be violated, relaxing nonactive constraints will change nothing. The hope is that once enough soft constraints have been relaxed, the whole constraint set will become feasible and one can continue. The decision-making process is taken by a supervisory controller level, before the constraints are downloaded to the MPC algorithm. A more subtle variant is a *hierarchical strategy* [32], where the user is asked to assign a priority index to each group of constraints at the design stage. This index expresses the relative importance of a particular group. At every sample, the full set of constraints is checked for compatibility. At the time of infeasibilities, the supervisory controller level uses the priority indices to determine a set of low priority constraints, which must be removed to reestablish feasibility. These constraints are then reintroduced into the control law as soon as possible. An important modification is manipulation with the *lower constraint horizon*, where the removal of constraints is performed by increases made in the value of the horizon. At the time of infeasibilities, the conflict between constraints is resolved by defining a new value of the lower constraint horizon such that the set of constraints is feasible.

The technique of constraint softening involves removing certain constraints at times of infeasibilities and adding a term to the cost function that penalizes violations of temporarily discarded constraints. Similar to the removal strategy, a hierarchical constraint softening can be introduced.

The topic of feasibility and handling infeasible constraint configurations is an essential one, since the MPC algorithm is defined well only when constraints are feasible. More techniques and procedures concerning feasibility issues and the maintenance of feasibility can be found in works by Scokaert et al. and others [51, 53, 54, 57, 61].

References

1. Åstrom K, Furuta K (2000) Swinging up a pendulum by energy control. Automatica 36(2):287–295. doi:10.1016/S0005-1098(99)00140-5, http://www.sciencedirect.com/science/article/pii/S0005109899001405
2. Belavý C (2009) Teória Automatického Riadenia II: Návody na cvičenia, Slovenská vysoká škola technická v Bratislave: Strojnícka Fakulta, 1st edn. Bratislava, (Theory of automatic control II: seminar guide) in Slovak language
3. Bemporad A (1998) A predictive controller with artificial Lyapunov function for linear systems with input/state constraints. Automatica 34(10):1255–1260
4. Bemporad A, Chisci L, Mosca E (1994) On the stabilizing property of the zero terminal state receding horizon regulation. Automatica 30(12):2013–2015
5. Bertram JE, Kalman RE (1960) Control systems analysis and design via second method of Ljapunov. Trans ASME, J Basic Eng 82:371–400
6. Bitmead RR, Gevers M, Wertz V (1990) Adaptive optimal control: the thinking man's GPC. Prentice Hall, Englewood Cliffs
7. Bitsoris G (1988) On the positive invariance of polyhedral sets for discrete-time systems. Syst Control Lett 11(3):243–248
8. Blanchini F (1994) Ultimate boundedness control for uncertain discrete-time systems via set-induced Lyapunov functions. IEEE Trans Autom Control 39(2):428–433
9. Blanchini F (1999) Set invariance in control. Automatica 35(11):1747–1767
10. Blondel V, Tsitsiklis JN (1996) NP-hardness of some linear control design problems. SIAM J Control Optim 35:2118–2127
11. Blondel VD, Tsitsiklis JN (2000) A survey of computational complexity results in systems and control. Automatica 36(9):1249–1274. doi:10.1016/S0005-1098(00)00050-9, http://www.sciencedirect.com/science/article/pii/S0005109800000509
12. Boyd S, Ghaoui L, Feron E, Balakrishnan V (1994) Linear matrix inequalities in systems and control theory, 1st edn. Society for Industrial and Applied Mathematics, Philadelphia
13. Boyd S, Ghaoui LE, Feron E, Balakrishnan V (1994) Linear matrix inequalities in system and control theory. Studies in Applied Mathematics, SIAM, Philadelphia
14. Cannon M (2005) Model predictive control, lecture notes. Michaelmas Term 2005 (4 Lectures), Course code 4ME44. University of Oxford, Oxford
15. Chen H, Allgöver F (1998) A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. Automatica 34(10):1205–1217
16. Chen CC, Shaw L (1982) On receding horizon feedback control. Automatica 18:349–352
17. Clarke DW, Scattolini R (1991) Constrained receding-horizon predictive control. IEE Proc Part D 138(4):347–354
18. de Oliveira SL (1996) Model predictive control for constrained nonlinear systems. PhD thesis, California Institute of Technology, Pasadena
19. Furuta K (1992) Swing-up control of inverted pendulum using pseudo-state feedback. J Syst Control Eng 206(14):263–269. doi:10.1243/PIME_PROC_1992_206_341_02
20. Garcia CE, Prett DM, Morari M (1989) Model predictive control: theory and practice—a survey. Automatica 25(3):335–348
21. Gilbert EG, Kolmanovsky I (1995) Discrete-time reference governors and the non-linear control of systems with state and control constraints. Int J Robust Nonlinear Control 5:487–504
22. Gilbert EG, Kolmanovsky I (1999) Fast reference governors for systems with state and control constraints and disturbance inputs. Int J Robust Nonlinear Control 9:1117–1141
23. Gilbert EG, Tan KT (1991) Linear systems with state and control constraints: the theory and application of maximal output admissible sets. IEEE Trans Autom Control 36(9):1008–1020
24. Iwase M, Astom KJ, Furuta K, Akesson J (2006) Analysis of safe manual control by using Furuta pendulum. In: Computer aided control system design, 2006 IEEE international conference on control applications, 2006 IEEE international symposium on intelligent control, 2006 IEEE, pp 568–572. doi:10.1109/CACSD-CCA-ISIC.2006.4776708

25. Jirstrand M (1998) Constructive methods for inequality constraints in control. PhD thesis, Department of Electrical Engineering, Linköping University, Linköping
26. Karas A (2002) Stabilizujúce prediktívne riadenie systémov s obmedzeniami. PhD thesis, Slovak University of Technology in Bratislava, Bratislava (Stabilizing predictive control of systems with constraints.) in Slovak language
27. Karas A, Rohal'-Ilkiv B, Belavý C (2007) Praktické aspekty prediktívneho riadenia, 1st edn. Slovak University of Technology in Bratislava / Slovenská E-Akadémia, n.o., Bratislava (Practical aspects of predictive control) in Slovak language
28. Keerthi SS, Gilbert EG (1988) Optimal, infinite horizon feedback law for a general class of constrained discrete time systems: stability and moving-horizon approximations. J Optim Theory Appl 57:265–293
29. Kerrigan EC (2000) Robust constraint satisfaction: invariant sets and predictive control. PhD thesis, Control Group, Department of Engineering, University of Cambridge, Cambridge
30. Kouvaritakis B, Rossiter JA, Chang AOT (1992) Stable generalised predictive control: an algorithm with guaranteed stability. IEE Proc Part D 139(4):349–362
31. Kouvaritakis B, Cannon M, Karas A, Rohal'-Ilkiv B, Belavý C (2002) Asymmetric constraints with polyhedral sets in MPC with application to coupled tanks system. In: IEEE 2002 conference on decision and control, Las Vegas, pp 4107–4112. doi:10.1109/CDC.2002.1185011
32. Kuznetsov AG (1996) Constrained predictive control: a brief survey. Journal A (Benelux publication of the Belgian Federation of Automatic Control) 37(2):3–8
33. Kwon WH, Pearson AE (1978) On feedback stabilization of time-varying discrete linear systems. IEEE Trans Autom Control 23:479–481
34. Lofberg J (2004) YALMIP: a toolbox for modeling and optimization in MATLAB. In: Proceedings of the CACSD conference, Taipei
35. Lyapunov AM (1893) Problème général da la stabilité du mouvement. Academic Press, New York (reprinted in 1966)
36. Maciejowski JM (2000) Predictive control with constraints, 1st edn. Prentice Hall, Upper Saddle River
37. Mayne DQ, Michalska H (1990) Receding horizon control of non-linear systems. IEEE Trans Autom Control 35(5):814–824
38. Mayne DQ, Rawlings JB, Rao CV, Scokaert POM (2000) Constrained model predictive control: stability and optimality. Automatica 36(6):789–814
39. Michalska H, Mayne DQ (1993) Robust receding horizon control of constrained nonlinear systems. IEEE Trans Autom Control 38(11):1623–1633
40. Mosca E, Zhang J (1992) Stable redesign of predictive control. Automatica 28(6):1229–1233
41. Mosca E, Lemos JM, Zhang J (1990) Stabilising I/O receding-horizon control. In: Proceedings of 29th IEEE conference on decision and control, Honolulu
42. Nesterov Y, Nemirovskii A (1994) Interior-point polynomial methods in convex programming, Studies in applied mathematics. vol 13, SIAM, Philadelphia
43. Nevistić V, Primbs JA (1997) Finite receding horizon linear quadratic control: a unifying theory for stability and performance analysis. Technical report CIT-CDS 97-001, California Institute of Technology, Pasadena
44. Polak E, Yang TH (1993) Moving horizon control of linear systems with input saturation and plant uncertainty: part 1: robustness. Int J Control 58(3):613–638
45. Polak E, Yang TH (1993) Moving horizon control of linear systems with input saturation and plant uncertainty—part 2: disturbance rejection and tracking. Int J Control 58(3):639–663
46. Pólik I (2005) Addendum to the SeDuMi user guide version 1.1. Technical report, McMaster University, Advanced Optimization Lab, Hamilton, Ontario. http://sedumi.ie.lehigh.edu/
47. Primbs JA, Nevistić V (1997) Constrained finite receding horizon linear quadratic control. Technical report CIT-CDS 97-002, California Institute of Technology, Pasadena
48. Rawlings JB, Muske KR (1993) The stability of constrained receding horizon control. IEEE Trans Autom Control 38(10):1512–1516

49. Rohal'-Ilkiv B (2004) A note on calculation of polytopic invariant and feasible sets for linear continuous-time systems. Annu Rev Control 28:59–64
50. Rohal'-Ilkiv B, Belavý C, Karas A (2001) Stable infinite-horizon predictive control with amplitude and rate input constraints. In: 13th international conference on process control-process control '01, Štrbské Pleso, Vysoké Tatry, Slovak Republic, CD-ROM
51. Rossiter JA (2003) Model-based predictive control: a practical approach, 1st edn. CRC Press, Boca Raton
52. Rusko M (2004) A note to LQ / $\mathscr{H}_2$ optimal sector pole placement. In: 6th international scientific-technical conference on process control 2004, Kouty nad Desnou, conference CD, paper R251
53. Scokaert POM (1994) Constrained predictive control. Technical report OUEL 2023/94, Department of Engineering Science, Oxford University, Parks Road, Oxford
54. Scokaert POM, Clarke DW (1994) Stabilising properties of constrained predictive control. IEE Proc Part D 141(5):295–304
55. Scokaert POM, Rawlings JB (1996) Infinite horizon linear quadratic control with constraints. In: Proceedings of IFAC'96 world congress, San Francisco, vol 7a-04-1, pp 109–114
56. Scokaert POM, Rawlings JB (1998) Constrained linear quadratic regulation. IEEE Trans Autom Control 43(8):1163–1169
57. Scokaert POM, Rawlings JB (1999) Feasibility issues in model predictive control. AIChE J 45(8):1649–1659
58. Soeterboek R (1992) Predictive control-a unified approach. Prentice Hall, New York
59. Sturm JF (2001) SeDuMi 1.05 R5 user's guide. Technical report, Department of Economics, Tilburg University, Tilburg. http://sedumi.ie.lehigh.edu/
60. Sznaier M, Damborg MJ (1987) Suboptimal control of linear systems with state and control inequality constraints. In: Proceedings of the 26th IEEE conference on decision and control, pp 761–762
61. Vada J, Slupphaug O, Johansen TA, Foss BA (2001) Linear mpc with optimal prioritized infeasibility handling: application, computational issues and stability. Automatica 37:1835–1843
62. Wu SP, Vandenberghe L, Boyd S (1996) MAXDET-software for determinant maximization problems-user's guide. Information Systems Laboratory, Electrical Engineering Department, Stanford University
63. Xu Y, Iwase M, Furuta K (2001) Time optimal swing-up control of single pendulum. J Dyn Syst Meas Control 123(3):518–527. doi:10.1115/1.1383027, http://link.aip.org/link/?JDS/123/518/1
64. Zheng ZQ, Morari M (1995) Stability of model predictive control with mixed constraints. IEEE Trans Autom Control 40(10):1818–1823
65. Zhou K, Doyle J, Glover K (1994) Robust optimal control. Prentice-Hall, Englewood Cliffs
66. Ziegler MG (1995) Lectures on polytopes. Springer, New York