Optimization-Based Control

Richard M. Murray
Control and Dynamical Systems
California Institute of Technology

Version v2.3e (28 Jan 2023)

Chapter 4

Receding Horizon Control

This chapter builds on the previous two chapters and explores the use of online optimization as a tool for control of nonlinear systems. We begin with a discussion of the technique of receding horizon control (RHC), which builds on the ideas of trajectory generation and optimization. We focus on a particular form of receding horizon control that makes use of a control Lyapunov function as a terminal cost, for which there are good stability and performance properties, and include a (optional) proof of stability. Methods for implementing receding horizon control, making use of numerical optimization possibly combined with differential flatness, are also provided. We conclude the chapter with a detailed design example, in which we explore some of the computational tradeoffs in optimization-based control as applied to a flight control experiment.

Prerequisites. Readers should be familiar with the concepts of trajectory generation and optimal control as described in Chapters 2 and 3 in this supplement. For the proof of stability for the receding horizon controller that we present, familiarity with Lyapunov stability analysis at the level given in FBS2e, Chapter 5 is assumed (but this material can be skipped if the reader is not familiar with Lyapunov stability analysis).

The material in this chapter is based in part on joint work with John Hauser, Ali Jadbabaie, Mark Milam, Nicolas Petit, William Dunbar, and Ryan Franz [MHJ+03].

4.1 Overview

The use of real-time trajectory generation techniques enables a sophisticated approach to the design of control systems, especially those in which constraints must be taken into account. The ability to compute feasible trajectories quickly enables us to make use of online computation of trajectories as an "outer feedback" loop that can be used to take into account nonlinear dynamics, input constraints, and more complex descriptions of performance goals.

Figure 4.1, a version of which was shown already in Chapter 2, provides a high level view of how real-time trajectory generation can be utilized. The dashed line from the output of the process to the trajectory generation block represents the use

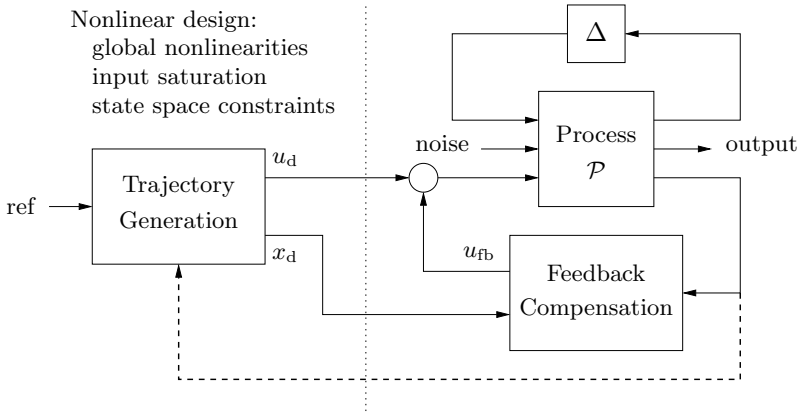


Figure 4.1: Two degree-of-freedom controller design for a process P with uncertainty Δ. The controller consists of a trajectory generator and feedback controller. The trajectory generation subsystem computes a feedforward command u_d along with the desired state x_d. The state feedback controller uses the measured (or estimated) state and desired state to compute a corrective input u_{fb}. Uncertainty is represented by the block Δ, representing unmodeled dynamics, as well as disturbances and noise.

of "on-the-fly" computation of the trajectory based on the current outputs of the process. This dynamically generated trajectory is then fed to the more traditional feedback controller. This same type of structure can also be seen in Figure 1.6, where the trajectory generation "layer" can make use of current measurements of the environment, as well as an online model of the process and upper level (supervisory controller) commands for the task to be accomplished.

The approach that we explore in this chapter is to make use of *receding horizon control*: a (optimal) feasible trajectory is computed from the current state to the desired state over a finite time horizon T, used for a short period of time $\Delta T < T$, and then recomputed based on the new system state starting at time $t + \Delta T$ until time $t+T+\Delta T$, as shown in Figure 4.2. As in the case of trajectory generation, we will normally compute the optimal trajectory assuming no process disturbances d, sensor noise n, or uncertainty Δ, relying on the feedback controller to compensate for those effects.

For the techniques that we will consider here, the problem that we solve at each time step t_i is a constrained, optimal trajectory generation problem of the form

$$
\begin{aligned}
u_{[t_i,t_i+\Delta T]} = \arg\min_{(x,u)} & \int_{t_i}^{t_i+T} L(x,u)\,d\tau + V(x(t_i+T)) \\
\text{subject to} & \\
& x(t_i) = \text{current state} \\
& \dot{x} = f(x,u) \\
& g_j(x,u) \le 0, \qquad j = 1,\ldots,r, \\
& \psi_k(x(t_i+T)) = 0, \quad k = 1,\ldots,q.
\end{aligned}
\tag{4.1}
$$

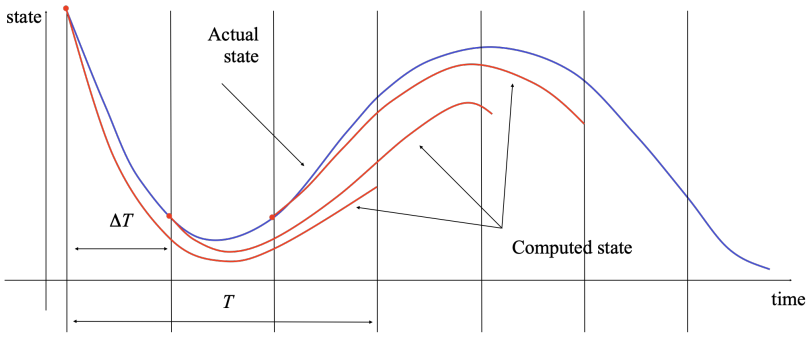


Figure 4.2: Receding horizon control. Every ΔT seconds, a trajectory generation problem is solved over a T second horizon, starting from the current state. In reality, the system will not follow the predicted path exactly, so that the red (computed) and blue (actual) trajectories will diverge. We then recompute the optimal path from the new state at time $t + \Delta T$, extending our horizon by an additional ΔT units of time.

We note that the cost consists of a trajectory cost $L(x, u)$ as well as an end-of-horizon (terminal) cost $V(x(t + T))$. In addition, we allow for the possibility of trajectory constraints on the states and inputs given by a set of functions $g_j(x, u)$ and a set of terminal constraints given by $\psi_k(x(t + T))$.

One of the challenges of properly implementing receding horizon control is that instabilities can result if the problem is not specified correctly. In particular, because we optimize the system dynamics over a finite horizon T, it can happen that choosing the optimal short term behavior can lead us *away* from the long term solution (see Exercise 4.4 for an example). To address this problem, the terminal cost $V(x(t + T))$ and/or the terminal constraints $\psi_k(x(t + T))$ must have certain properties to ensure stability (see [MRRS00] for details). In this chapter we focus on the use of terminal costs since these have certain advantages in terms of the underlying optimization problem to be solved.

Development and application of receding horizon control (also called model predictive control, or MPC) originated in process control industries where the processes being controlled are often sufficiently slow to permit its implementation with only modest computational resources. The rapid advances in computation over the last several decades have enabled receding horizon control to be used in many new applications, and they are especially prevalent in autonomous vehicles, where the trajectory generation layer is particularly important for achieving safe operation in complex environments. Proper formulation of the problem to enable rapid computation is still often required, and the use of differential flatness and other techniques (such as motion primitives) can be very important.

Finally, we note that it is often the case that the trajectory generation problems to be solved may have non-unique or non-smooth solutions, and that these can often be very sensitive to small changes in the inputs to the algorithm. Full implementation of receding horizon control techniques thus often requires careful attention to details in how the problems are solved and the use of additional methods to ensure that good solutions are obtained in specific application scenarios. Despite all of

these warnings, receding horizon control is one of the dominant methods used for control of nonlinear systems and one of the few methods that works in the presence of input (and state) constraints, leading to its wide popularity.

4.2 Receding Horizon Control with Terminal Cost

One of the earliest techniques for ensuring stability of the closed loop system under receding horizon control was to impose a terminal constraint on the optimization through use of the function $\psi(x(t+T))$ in equation (4.1). While this technique can be shown to be sound from a theoretical point of view, it can be very difficult to satisfy from a computational point of view, since imposing equality constraints on the terminal state of a trajectory generation problem can be computationally expensive. In this section we explore an alternative formulation, making use of an appropriate terminal cost function.

Stability of Receding Horizon Control using Terminal Costs

We consider the following special case of receding horizon control problem in equation (4.1):

$$\begin{aligned} &\dot{x} = f(x,u), \qquad x(0) \text{ given}, u \in \mathcal{U} \subset \mathbb{R}^m \\ &u_{[t,t+\Delta T]} = \arg\min_{(x,u)} \int_t^{t+T} L(x,u)\,d\tau + V(x(t+T)). \end{aligned} \tag{4.2}$$

The main differences between equation (4.2) and equation (4.1) are that we only consider constraints on the input ($u \in \mathcal{U}$) and we do not impose any terminal constraints.

Stability of this system is not guaranteed in general, but if we choose the trajectory and terminal cost functions carefully, it is possible to provide stability guarantees.

To illustrate how the choice of the terminal condition can provide stability, consider the case where we have an infinite horizon optimal control problem. If we start at state x at time t, we can seek to minimize the "cost to go" function

$$J(t,x) = \int_t^{\infty} L(x,u)\,dt.$$

If we let $J^*(x,t)$ represent the optimal cost to go function, then a natural choice for the terminal cost is $V(x(t+T)) = J^*(x(t+T),T)$, since then the optimal finite and infinite horizon costs are the same:

$$\begin{aligned} \min \int_t^{\infty} L(x,u)\,d\tau &= \int_t^{t+T} L(x^*,u^*)\,dt + \int_{t+T}^{\infty} L(x^*,u^*)\,dt \\ &= \int_t^{t+T} L(x^*,u^*)\,dt + \underbrace{J^*(t+T,x^*)}_{V(x^*(t+T))}. \end{aligned}$$

Intuitively, if we solve the infinite horizon problem at each update ΔT and ΔT is sufficiently small, then we anticipate that the system trajectory should converge

to the optimal trajectory, and hence to the origin (otherwise the cost would not converge to a finite value, assuming $Q_x > 0$).

Of course, if the optimal value function were available there would be no need to solve a trajectory optimization problem, since we could just use the gradient of the cost function to choose the input u. Only in special cases (such as the linear quadratic regulator problem) can the infinite horizon optimal control problem be solved in closed form, and we thus seek to find a simpler set of conditions under which we can guarantee stability of closed loop, receding horizon controller. The following theorem summarizes on such set of conditions:

Theorem 4.1 (based on [JYH01])**.** *Consider the receding horizon control problem in equation* (4.2) *and suppose that the trajectory cost* $L(x, u)$ *and terminal cost* $V(x(t+T))$ *satisfy*

$$\min_{u \in \mathcal{U}} \left(\frac{\partial V}{\partial x} f(x, u) + L(x, u) \right) \leq 0 \tag{4.3}$$

for all x *in a neighborhood of the origin. Then, for* every $T > 0$ *and* $\Delta T \in (0, T]$*, there exist constants* $M > 0$ *and* $c > 0$ *such that the resulting receding horizon trajectories converge to 0 exponentially fast:*

$$\|x(t)\| \leq M e^{-ct} \|x(0)\|.$$

Before providing more insights into when these conditions can be satisfied, it is useful to take think about the implications of Theorem 4.1. In particular, it provides us a method for defining a stabilizing feedback controller for a fully nonlinear system in the presence of input constraints. This latter feature is particularly important, since it turns out that input constraints are ubiquitous in control systems and must other methods, including LQR and gain scheduling, are not able to take them into account in a systematic and rigorous way. It is because of this ability to handle constraints that receding horizon control is so widely used.

Control Lyapunov Functions

To provide insight into the conditions in Theorem 4.1, we need to define the concept of a *control Lyapunov function*. The material in this subsection is rather advanced in nature and can be skipped on first reading. Readers should be familiar with (regular) Lyapunov stability analysis at the level given in FBS2e, Chapter 5 prior to tackling the concepts in this section.

Control Lyapunov functions are an extension of standard Lyapunov functions and were originally introduced by Sontag [Son83]. They allow constructive design of nonlinear controllers and the Lyapunov function that proves their stability. We give a brief description of the basic idea of control Lyapunov functions here; a more complete treatment is given in [KKK95].

Consider a nonlinear control system

$$\dot{x} = f(x, u), \qquad x \in \mathbb{R}^n,\ u \in \mathbb{R}^m, \tag{4.4}$$

and recall that a function $V(x)$ is a *positive definite function* if $V(x) \geq 0$ for all $x \in \mathbb{R}^n$ and $V(x) = 0$ if and only if $x = 0$. A function $V(x)$ is *locally positive definite* if it is positive definite on a ball of radius ϵ around the origin, $B_\epsilon(0) = \{x \colon \|x\| < \epsilon\}$.

Definition 4.1 (Control Lyapunov Function)**.** A locally positive function $V : \mathbb{R}^n \to \mathbb{R}_+$ is called a *control Lyapunov function (CLF)* for a control system (4.4) if

$$\inf_{u \in \mathbb{R}^m} \left(\frac{\partial V}{\partial x} f(x,u) \right) < 0 \qquad \text{for all } x \neq 0.$$

Intuitively, a control Lyapunov function is a positive definite function for which it is always possible to choose an input u that makes the function decrease if we apply that input to the control system (4.4). Since the function V is positive definite, if we always choose a u that makes it decrease then eventually the value of the Lyapunov function must converge to 0 and hence the state $x(t)$ must also converge to zero. It turns out that this property is enough to show that the system is *stabilizable* using continuous (though not necessarily linear) feedback laws of the form $u = -k(x)$.

In general, it is difficult to find a control Lyapunov function for a given system. However, for many classes of systems, there are specialized methods that can be used. One of the simplest is to use the Jacobian linearization of the system around the desired equilibrium point and generate a control Lyapunov function by solving an LQR problem. To see how this works, we consider first the case of a linear system with quadratic cost function.

As described in Chapter 3, the problem of minimizing the quadratic performance index

$$J = \int_0^\infty (x^\mathsf{T}(t) Q_x x(t) + u^\mathsf{T}(t) Q_u u(t))\, dt \qquad \text{subject to} \qquad \begin{aligned} \dot{x} &= Ax + Bu, \\ x(0) &= x_0, \end{aligned} \tag{4.5}$$

results in finding the positive definite solution of the following Riccati equation:

$$A^\mathsf{T} P + PA - PBR^{-1}B^\mathsf{T} P + Q = 0. \tag{4.6}$$

The optimal control action is given by

$$u^* = -R^{-1}B^\mathsf{T} P x$$

and $V = x^\mathsf{T} P x$ is a control Lyapunov function for the system since it can be shown (with a bit of algebra) that

$$\min_u \frac{\partial V}{\partial x} f(x,u) \leq \frac{\partial V}{\partial x} f(x,u^*) = -x^\mathsf{T}(Q_x + PBQ_u^{-1}B^\mathsf{T} P)x \leq 0.$$

In the case of the nonlinear system $\dot{x} = f(x,u)$, A and B are taken as

$$A = \left. \frac{\partial f(x,u)}{\partial x} \right|_{(0,0)} \qquad B = \left. \frac{\partial f(x,u)}{\partial u} \right|_{(0,0)}$$

where the pairs (A,B) and $(Q_x^{\frac{1}{2}}, A)$ are assumed to be stabilizable and detectable respectively. The control Lyapunov function $V(x) = x^\mathsf{T} P x$ is valid in a region around the equilibrium $(0,0)$, as shown in Exercise 4.1.

More complicated methods for finding control Lyapunov functions are often required and many techniques have been developed. An overview of some of these methods can be found in [Jad01].

Solving Receding Horizon Optimal Control Problems

We now return to the problem of implementing the receding horizon controller in equation (4.3). As illustrated in Figure 4.2, at every time instant t_i we compute the optimal trajectory that minimizes the cost function

$$J(x,t) = \int_t^{t+T} L(x,u)\,d\tau + V(x(t+T))$$

subject to the satisfying the equations of motion with constrained inputs:

$$\dot{x} = f(x,u), \qquad u \in \mathcal{U} \subset \mathbb{R}^m.$$

This is precisely the optimal control problem that we considered in Chapter 3 and so the numerical methods used in that chapter can be utilized.

One conceptually simple way to implement the optimization required to solve this optimal control problem is to parameterize the inputs of the system u, either by setting the values of u at discrete time points or by choosing a set of basis functions for u and searching over linear combinations of the basis functions. These methods are often referred to as "shooting" methods, since they integrate ("shoot") the equations forward in time and then attempt to compute the changes in parameter values to allow the system to minimize the cost and satisfy any constraints. While crude, this approach does work for simple systems and can be used to gain insights into the properties of a receding horizon controller based on simulations (where real-time computation is not needed).

Example 4.1 Double integrator with bounded input
To illustrate the implementation of a receding horizon controller, we consider a linear system corresponding to a double integrator with bounded input:

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{clip}(u) \qquad \text{where} \qquad \text{clip}(u) = \begin{cases} -1 & u < -1, \\ u & -1 \le u \le 1, \\ 1 & u > 1. \end{cases}$$

We implement a model predictive controller by choosing

$$Q_x = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \qquad Q_u = \begin{bmatrix} 1 \end{bmatrix}, \qquad P_1 = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}.$$

Figure 4.3 shows the results of this computation, with the inputs plotted for the planning horizon, showing that the final computed input differs from the planned inputs over the horizon. (The code for computing these solutions is given in Section 4.3.) ∇

Implementing receding horizon control for realistic problems requires care in formulating the optimization optimization so that it can be done in real-time. For example, for a typical autonomous vehicle (land, air, or sea), a reasonable optimization horizon might be the next 10-60 s and a typical update period might be as short as 10-100 ms. There are a variety of methods for speeding up computations, as well as taking into account finite computation times. Some of these are

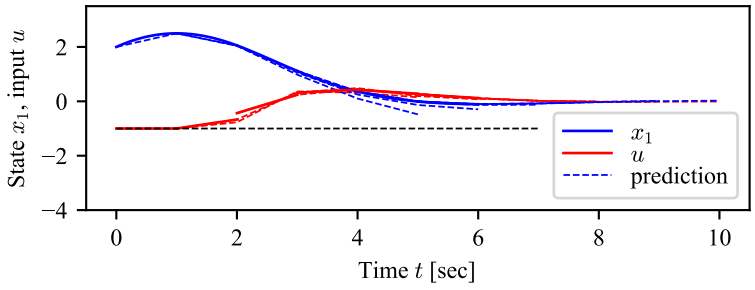


Figure 4.3: Receding horizon controller for a double integrator. Dashed lines show the planned trajectory over each horizon; solid lines show the closed loop trajectory. The horizontal dashed line on each plot shows the lower limit of the input.

described in more detail in Section 4.6, where we describe a specific implementation of receding horizon control on a flight control testbed.

A relatively efficient numerical approach to solving the optimal control problem is the direct collocation method (see [Kel17] for a good overview). The idea behind this approach is to transform the optimal control problem into a constrained nonlinear programming problem. This is accomplished by discretizing time into a set of $N-1$ intervals defined by grid points

$$t_0 = t_1 < t_2 < \ldots < t_N = t_\mathrm{f}$$

and approximating the state x and the control input u as piecewise polynomials $\tilde{x}$ and $\tilde{u}$. For example, on each interval the states can be approximated by a cubic polynomial and the control can be approximated by a linear polynomial. The value of the polynomial at the midpoint of each interval is then used to satisfy the dynamics

$$\dot{x} = f(x, u).$$

To solve the problem, let $\tilde{x}(x(t_1), \ldots, x(t_N))$ and $\tilde{u}(u(t_1), \ldots, u(t_N))$ denote the approximations to x and u, which depend on $(x(t_1), \ldots, x(t_N)) \in \mathbb{R}^{nN}$ and $(u(t_1), \ldots, u(t_N)) \in \mathbb{R}^N$, representing the value of x and u at the grid points. We then solve the following finite dimension approximation of the original control problem (4.1):

$$\min_{\xi \in \mathbb{R}^M} F(\xi) = J(\tilde{x}(\xi), \tilde{u}(\xi))$$

$$\text{subject to} \quad \begin{cases} \dot{\tilde{x}} - f(\tilde{x}(\xi, \tilde{u}(\xi) = 0, \\ lb \le c(\tilde{x}(\xi), \tilde{u}(\xi)) \le ub, \\ \qquad \forall t = \dfrac{t_j + t_{j+1}}{2} \quad j = 1, \ldots, N-1 \end{cases} \tag{4.7}$$

where $\xi = (x(t_1), u(t_1), \ldots, x(t_N), u(t_N))$, and $M = \dim \xi = (n+1)N$. Note here that we are optimizing over both the x and u variables at the grid points and then seeking to (approximately) satisfying the differential equation, compared to the

shooting method, which optimizes just over the inputs at the grid points and then integrates the differential equation.

Collocation techniques turn out to be much more numerically well-conditioned than shooting methods, and there are excellent software packages available for solving collocation-based optimization problems.

Proof of Stability (with J. E. Hauser and A. Jadbabaie)

In this final (optional) subsection, we return to Theorem 4.1 and provide a mathematically rigorous version of the theorem and a sketch of its proof. In order to show the stability of the proposed approach, and give full conditions on the terminal cost $V(x(T))$, we briefly review the problem of optimal control over a finite time horizon as presented in Chapter 3 to establish some notation and set some more specific conditions required for receding horizon control. This material is based on [MHJ+03].

Given an initial state x_0 and a control trajectory $u(\cdot)$ for a nonlinear control system $\dot{x} = f(x,u)$, let $x^u(\cdot;x_0)$ represent the state trajectory. We can write this solution as a continuous curve

$$x^u(t;x_0) = x_0 + \int_0^t f(x^u(\tau;x_0),u(\tau))\,d\tau$$

for $t \geq 0$. We require that the trajectories of the system satisfy an *a priori* bound

$$\|x(t)\| \leq \beta(x,T,\|u(\cdot)\|_1) < \infty, \qquad t \in [0,T],$$

where β is continuous in all variables and monotone increasing in T and $\|u(\cdot)\|_1 = \|u(\cdot)\|_{L_1(0,T)}$. Most models of physical systems will satisfy a bound of this type.

The performance of the system will be measured by an integral cost $L\colon \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$. We require that L be twice differentiable (C^2) and fully penalize both state and control according to

$$L(x,u) \geq c_q(\|x\|^2 + \|u\|^2), \qquad x \in \mathbb{R}^n, u \in \mathbb{R}^m$$

for some $c_q > 0$ and $L(0,0) = 0$. It follows that the quadratic approximation of L at the origin is positive definite,

$$\left.\frac{\partial L}{\partial x}\right|_{(0,0)} \geq c_q I > 0.$$

To ensure that the solutions of the optimization problems of interest are well behaved, we impose some convexity conditions. We require the set $f(x,\mathbb{R}^m) \subset \mathbb{R}^n$ to be convex for each $x \in \mathbb{R}^n$. Letting $\lambda \in \mathbb{R}^n$ represent the co-state, we also require that the pre-Hamiltonian function $\lambda^{\mathsf{T}} f(x,u) + L(x,u) =: K(x,u,\lambda)$ be strictly convex for each $(x,\lambda) \in \mathbb{R}^n \times \mathbb{R}^n$ and that there is a C^2 function $\bar{u}^*\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^m$ providing the global minimum of $K(x,u,\lambda)$. The Hamiltonian $H(x,\lambda) := K(x,\bar{u}^*(x,\lambda),\lambda)$ is then C^2, ensuring that extremal state, co-state, *and* control trajectories will all be sufficiently smooth (C^1 or better). Note that these conditions are automatically satisfied for control affine f and quadratic L.

The cost of applying a control $u(\cdot)$ from an initial state x over the infinite time interval $[0, \infty)$ is given by

$$J_\infty(x, u(\cdot)) = \int_0^\infty L(x^u(\tau; x), u(\tau))\, d\tau.$$

The optimal cost (from x) is given by

$$J^*_\infty(x) = \inf_{u(\cdot)} J_\infty(x, u(\cdot)),$$

where the control function $u(\cdot)$ belongs to some reasonable class of admissible controls (e.g., piecewise continuous). The function $J^*_\infty(x)$ is often called the *optimal value function* for the infinite horizon optimal control problem. For the class of f and L considered, it can be verified that $J^*_\infty(\cdot)$ is a positive definite C^2 function in a neighborhood of the origin [HO01].

For practical purposes, we are interested in finite horizon approximations of the infinite horizon optimization problem. In particular, let $V(\cdot)$ be a non-negative C^2 function with $V(0) = 0$ and define the finite horizon cost (from x using $u(\cdot)$) to be

$$J_T(x, u(\cdot)) = \int_0^T L(x^u(\tau; x), u(\tau))\, d\tau + V(x^u(T; x)), \tag{4.8}$$

and denote the optimal cost (from x) as

$$J^*_T(x) = \inf_{u(\cdot)} J_T(x, u(\cdot))\ .$$

As in the infinite horizon case, one can show, by geometric means, that $J^*_T(\cdot)$ is locally smooth (C^2). Other properties will depend on the choice of V and T.

Let Γ^∞ denote the domain of $J^*_\infty(\cdot)$ (the subset of $\mathbb{R}^n$ on which J^*_∞ is finite). It is not too difficult to show that the cost functions $J^*_\infty(\cdot)$ and $J^*_T(\cdot)$, $T \geq 0$, are continuous functions on Γ_∞ [Jad01]. For simplicity, we will allow $J^*_\infty(\cdot)$ to take values in the extended real line so that, for instance, $J^*_\infty(x) = +\infty$ means that there is no control taking x to the origin.

We will assume that f and L are such that the minimum value of the cost functions $J^*_\infty(x)$, $J^*_T(x)$, $T \geq 0$, is attained for each (suitable) x. That is, given x and $T > 0$ (including $T = \infty$ when $x \in \Gamma^\infty$), there is a (C^1 in t) optimal trajectory $(x^*_T(t; x), u^*_T(t; x))$, $t \in [0, T]$, such that $J_T(x, u^*_T(\cdot; x)) = J^*_T(x)$. For instance, if f is such that its trajectories can be bounded on finite intervals as a function of its input size, e.g., there is a continuous function β such that $\|x^u(t; x_0)\| \leq \beta(\|x_0\|, \|u(\cdot)\|_{L_1[0,t]})$, then (together with the conditions above) there will be a minimizing control (cf. [LM67]). Many such conditions may be used to good effect; see [Jad01] for a more complete discussion.

It is easy to see that $J^*_\infty(\cdot)$ is proper on its domain so that the sub-level sets

$$\Gamma^\infty_r := \{x \in \Gamma^\infty \colon J^*_\infty(x) \leq r^2\}$$

are compact and path connected and moreover $\Gamma^\infty = \bigcup_{r \geq 0} \Gamma^\infty_r$. Note also that Γ^∞ may be a proper subset of $\mathbb{R}^n$ since there may be states that cannot be driven to the origin. We use r^2 (rather than r) here to reflect the fact that our integral cost

is quadratically bounded from below. We refer to sub-level sets of $J_T^*(\cdot)$ and $V(\cdot)$ using

$$\Gamma_r^{\mathsf{T}} := \text{path connected component of } \{x \in \Gamma^\infty \colon J_T^*(x) \leq r^2\} \text{ containing } 0,$$

and

$$\Omega_r := \text{path connected component of } \{x \in \mathbb{R}^n \colon V(x) \leq r^2\} \text{ containing } 0.$$

These results provide the technical framework needed for receding horizon control. The following restated version of Theorem 4.1 provides a more rigorous description of the results of this section.

Theorem 1'. [JYH01] *Consider the receding horizon control problem in equation* (4.2) *and suppose that the terminal cost* $V(\cdot)$ *is a control Lyapunov function such that*

$$\min_{u \in \mathbb{R}^m} (\dot{V} + L)(x, u) \leq 0 \tag{4.9}$$

for each $x \in \Omega_{r_v}$ *for some* $r_v > 0$. *Then, for* every $T > 0$ *and* $\delta \in (0, T]$, *the resulting receding horizon trajectories go to zero exponentially fast. For each* $T > 0$, *there is a constant* $\bar{r}(T) \geq r_v$ *such that* $\Gamma_{\bar{r}(T)}^{\mathsf{T}}$ *is contained in the region of attraction of* $\mathcal{RH}(T, \delta)$. *Moreover, given any compact subset* Λ *of* Γ^∞, *there is a* T^* *such that* $\Lambda \subset \Gamma_{\bar{r}(T)}^{\mathsf{T}}$ *for all* $T \geq T^*$.

Theorem 1' shows that for *any* horizon length $T > 0$ and *any* sampling time $\delta \in (0, T]$, the receding horizon scheme is exponentially stabilizing over the set $\Gamma_{r_v}^{\mathsf{T}}$. For a given T, the region of attraction estimate is enlarged by increasing r beyond r_v to $\bar{r}(T)$ according to the requirement that $V(x_T^*(T; x)) \leq r_v^2$ on that set. An important feature of the above result is that, for operations with the set $\Gamma_{\bar{r}(T)}^{\mathsf{T}}$, there is no need to impose stability ensuring constraints which would likely make the online optimizations more difficult and time consuming to solve.

Sketch of proof. Let $x^u(\tau; x)$ represent the state trajectory at time τ starting from initial state x and applying a control trajectory $u(\cdot)$, and let $(x_T^*, u_T^*)(\cdot, x)$ represent the optimal trajectory of the finite horizon, optimal control problem with horizon T. Assume that $x_T^*(T; x) \in \Omega_r$ for some $r > 0$. Then for any $\delta \in [0, T]$ we want to show that the optimal cost $x_T^*(\delta; x)$ satisfies

$$J_T^*\big(x_T^*(\delta; x)\big) \leq J_T^*(x) - \int_0^\delta q\big(L(x_T^*(\tau; x), u_T^*(\tau; x))\, d\tau. \tag{4.10}$$

This expression says that solution to the finite-horizon, optimal control problem starting at time $t = \delta$ has cost that is less than the cost of the solution from time $t = 0$, with the initial portion of the cost subtracted off.. In other words, we are closer to our solution by a finite amount at each iteration of the algorithm. It follows using Lyapunov analysis that we must converge to the zero cost solution and hence our trajectory converges to the desired terminal state (given by the minimum of the cost function).

To show equation (4.10) holds, consider a trajectory in which we apply the optimal control for the first T seconds and then apply a closed loop controller using a

stabilizing feedback $u = -k(x)$ for another T seconds. (The stabilizing compensator is guaranteed to exist since V is a control Lyapunov function.) Let $(x_T^*, u_T^*)(t; x)$, $t \in [0, T]$ represent the optimal control and $(x^k, u^k)(t-T; x_T^*(T; x))$, $t \in [T, 2T]$ represent the control with $u = -k(x)$ applied where k satisfies $(\dot{V} + L)(x, -k(x)) \leq 0$. Finally, let $(\tilde{x}(t), \tilde{u}(t))$, $t \in [0, 2T]$ represent the trajectory obtained by concatenating the optimal trajectory (x_T^*, u_T^*) with the control Lyapunov function trajectory (x^k, u^k).

We now proceed to show that the inequality (4.10) holds. The cost of using $\tilde{u}(\cdot)$ for the first T seconds starting from the initial state $x_T^*(\delta; x))$, $\delta \in [0, , T]$ is given by

$$\begin{aligned} J_T(x_T^*(\delta; x), \tilde{u}(\cdot)) &= \int_\delta^{T+\delta} L(\tilde{x}(\tau), \tilde{u}(\tau))\, d\tau + V(\tilde{x}(T+\delta)) \\ &= J_T^*(x) - \int_0^\delta L(x_T^*(\tau; x), u_T^*(\tau; x))\, d\tau - V(x_T^*(T; x)) \\ &\quad + \int_T^{T+\delta} L(\tilde{x}(\tau), \tilde{u}(\tau))\, d\tau + V(\tilde{x}(T+\delta)). \end{aligned}$$

Note that the second line is simply a rewriting of the integral in terms of the optimal cost J_T^* with the necessary additions and subtractions of the additional portions of the cost for the interval $[\delta, T+\delta]$. We can how use the bound

$$L(\tilde{x}(\tau), \tilde{u}(\tau)) \leq \dot{V}(\tilde{x}(\tau), \tilde{u}(\tau), \qquad \tau \in [T, 2T],$$

which follows from the definition of the control Lyapunov function V and stabilizing controller $k(x)$. This allows us to write

$$\begin{aligned} J_T(x_T^*(\delta; x), \tilde{u}(\cdot)) &\leq J_T^*(x) - \int_0^\delta L(x_T^*(\tau; x), u_T^*(\tau; x))\, d\tau - V(x_T^*(T; x)) \\ &\quad - \int_T^{T+\delta} \dot{V}(\tilde{x}(\tau), \tilde{u}(\tau))\, d\tau + V(\tilde{x}(T+\delta)) \\ &= J_T^*(x) - \int_0^\delta L(x_T^*(\tau; x), u_T^*(\tau; x))\, d\tau - V(x_T^*(T; x)) \\ &\quad - V(\tilde{x}(\tau))\Big|_T^{T+\delta} + V(\tilde{x}(T+\delta)) \\ &= J_T^*(x) - \int_0^\delta L(x_T^*(\tau; x), u_T^*(\tau; x)). \end{aligned}$$

Finally, using the optimality of u_T^* we have that $J_T^*(x_T^*(\delta; x)) \leq J_T(x_T^*(\delta; x), \tilde{u}(\cdot))$ and we obtain equation (4.10). □

An important benefit of receding horizon control is its ability to handle state and control constraints. While the above theorem provides stability guarantees when there are no constraints present, it can be modified to include constraints on states and controls as well. In order to ensure stability when state and control constraints are present, the terminal cost $V(\cdot)$ should be a local control Lyapunov function satisfying $\min_{u \in \mathcal{U}} \dot{V} + L(x, u) \leq 0$ where $\mathcal{U}$ is the set of controls where the control

constraints are satisfied. Moreover, one should also require that the resulting state trajectory $x^{\mathrm{CLF}}(\cdot) \in \mathcal{X}$, where $\mathcal{X}$ is the set of states where the constraints are satisfied. (Both $\mathcal{X}$ and $\mathcal{U}$ are assumed to be compact with origin in their interior). Of course, the set Ω_{r_v} will end up being smaller than before, resulting in a decrease in the size of the guaranteed region of operation (see [MRRS00] for more details).

4.3 Implementation in Python

The optimal control module of the python-control package allows implementation of receding horizon control by solving an optimization problem based on the current state of a nonlinear system. We begin by defining an optimal control problem using the `OptimalControlProblem` class:

```
ocp = obc.OptimalControlProblem(
    sys, timepts, cost, constraints, terminal_cost)
```

To describe an optimal control problem we need an input/output system, a list of time points, a cost function, and (optionally) a set of constraints on the state and/or input, either along the trajectory (via the `trajectory_constraint` keyword) or at the terminal time (via the `terminal_constraint` keyword). The `OptimalControlProblem` class sets up an optimization over the inputs at each point in time, using the integral and terminal costs as well as the trajectory and terminal constraints.

Once an optimal control problem has been defined, the `compute_trajectory` method can be used to solve for an optimal trajectory from a given state x:

```
res = ocp.compute_trajectory(x)
t, u = res.time, res.inputs
```

This is the method that the `opt.solve_ocp` function uses to compute an optimal trajectory. In the context of model predictive control, we would repeatedly call `compute_trajectory` from the state at each update time t_i and then apply the input u for the next ΔT seconds.

For discrete time systems, the `create_mpc_iosystem` method can be used to create an input/output system that implements the control law:

```
ctrl = ocp.create_mpc_iosystem()
```

The resulting object takes as input the current state of the system as returns as output the commanded input from the MPC controller.

For continuous time system, receding horizon control must be implemented "manually", by computing the optimal input at time instant t_i and then simulating the system over the interval $[t_i, t_i + \Delta T]$. This is illustrated in the following example.

Example 4.2 Double integrator with bounded input
We the consider linear system corresponding to a double integrator with bounded input described in Example 4.1. The equations of motion are given by

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathrm{clip}(u) \qquad \text{where} \qquad \mathrm{clip}(u) = \begin{cases} -1 & u < -1, \\ u & -1 \le u \le 1, \\ 1 & u > 1. \end{cases}$$

This function can be created with the Python code

```
def doubleint_update(t, x, u, params):
    return np.array([x[1], np.clip(u, -1, 1)])

proc = ct.NonlinearIOSystem(
    doubleint_update, None, name="double integrator",
    inputs = ['u'], outputs=['x[0]', 'x[1]'], states=2, dt=True)
```

We now define an optimal control problem with quadratic costs and input constraints:

```
# Cost function
Qx = np.diag([1, 0])         # state cost
Qu = np.diag([1])            # input cost
P1 = np.diag([0.1, 0.1])     # terminal cost

# Constraints (using input_poly_constraint to illustrate its use)
traj_constraints = opt.input_poly_constraint(
    proc, np.array([[1, 0], [-1, 0]]), np.array([1, 1]))

# Horizon
T = 5
timepts = np.linspace(0, T, 5, endpoint=True)

# Set up the optimal control problem
ocp = opt.OptimalControlProblem(
    proc, timepts,
    opt.quadratic_cost(proc, Qx, Qu),
    trajectory_constraints=traj_constraints,
    terminal_cost=opt.quadratic_cost(proc, P1, None)
)
```

This optimal control problem contains all of the information required to compute the optimal input from a state x over the specified time horizon.

To use this optimal control problem in a receding horizon fashion, we manually compute the trajectory at each time point:

```
x = X0                       # initial condition (updated as we go)
Tf = 10                      # total simulation time
for t in np.linspace(0, Tf-T, 6, endpoint=True):
    # Compute the optimal trajectory over the horizon
    res = ocp.compute_trajectory(x, return_states=True)

    # Simulate the system for the update period
    time = np.linspace(0, res.time[1], 20)
    soln = ct.input_output_response(proc, time, inputs, x)

    # Update the state for the next iteration
    x = soln.states[:, -1]
```

Figure 4.3 in the previous section shows the results of this computation, with the inputs plotted for the planning horizon, showing that the final computed input differs from the planned inputs over the horizon. ∇

4.4 Receding Horizon Control Using Differential Flatness

For systems that are differentially flat, it is possible to use the flatness-based structure of the system to implement receding horizon control. In this section we summarize how to exploit differential flatness to find fast numerical algorithms for solving the optimal control problems required for the receding horizon control results of the previous section.

We consider the affine nonlinear control system

$$\dot{x} = f(x) + g(x)u, \tag{4.11}$$

where all vector fields and functions are smooth. For simplicity, we focus on the single input case, $u \in \mathbb{R}$. We wish to find a trajectory of equation (4.11) that minimizes the performance index (4.8), subject to a vector of initial, final, and trajectory constraints

$$\begin{aligned} lb_0 &\leq \psi_0(x(t_0), u(t_0)) \leq ub_0, \\ lb_{\mathrm{f}} &\leq \psi_{\mathrm{f}}(x(t_{\mathrm{f}}), u(t_{\mathrm{f}})) \leq ub_{\mathrm{f}}, \\ lb_t &\leq S(x, u) \leq ub_t, \end{aligned} \tag{4.12}$$

respectively. For conciseness, we will refer to this optimal control problem as

$$\min_{(x,u)} J(x,u) \qquad \text{subject to} \qquad \begin{cases} \dot{x} = f(x) + g(x)u, \\ lb \leq c(x,u) \leq ub. \end{cases} \tag{4.13}$$

For a system that is differentially flat, we can compute feasible trajectories in terms of the flat output trajectory $z(\cdot)$, as described in Section 2.3. When the parameterization is only partial, the dimension of the subspace spanned by the output and its derivatives is given by r the *relative degree* of this output [Isi89]. In this case, it is possible to write the system dynamics as

$$\begin{aligned} x &= \alpha(z, \dot{z}, \ldots, z^{(q)}), \\ u &= \beta(z, \dot{z}, \ldots, z^{(q)}), \\ \Phi(z, \dot{z}, \ldots, z^{n-r}) &= 0, \end{aligned} \tag{4.14}$$

where $z \in \mathbb{R}^p$, $p > m$ represents a set of outputs that parameterize the trajectory and $\Phi : \mathbb{R}^n \times \mathbb{R}^m$ represents $n - r$ remaining differential constraints on the output. (In the case that the system is flat, $r = n$ and we eliminate these differential constraints.)

A computationally attractive representation for the flat outputs is to use B-splines, which are a set of piecewise polynomials defined across a set of knot points with given degree and smoothness. On each interval between two knot points, we have a polynomial of a given degree and the spline is continuous up to a given smoothness at interior breakpoints. B-splines are chosen as basis functions because of their ease of enforcing continuity across knot points and ease of computing their derivatives. A pictorial representation of such an approximation is given in Figure 4.4. In this formulation, each flat output z_j is represented as

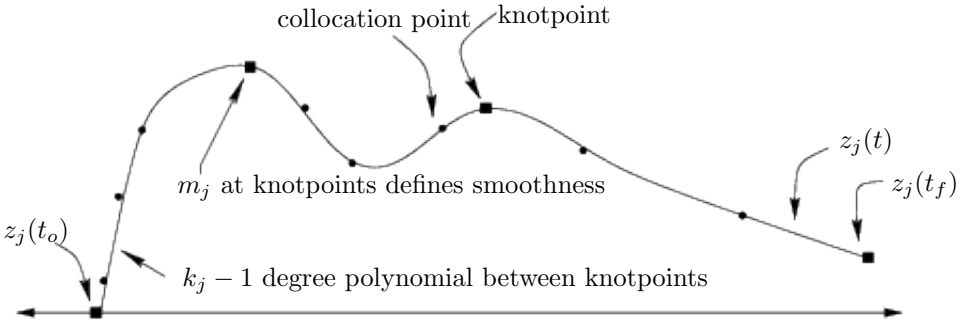


Figure 4.4: Spline representation of a variable.

$$z_j = \sum_{i=1}^{p_j} B_{i,k_j}(t) C_i^j, \qquad p_j = l_j(k_j - m_j) + m_j$$

where $B_{i,k_j}(t)$ is the B-spline basis function defined in [dB78] for the output z_j with order k_j, C_i^j are the coefficients of the B-spline, l_j is the number of knot intervals, and m_j is number of smoothness conditions at the knots. The set $(z_1, z_2, \ldots, z_{n-r})$ is thus represented by $M = \sum_{j\in\{1,r+1,\ldots,n\}} p_j$ coefficients.

In general, w collocation points are chosen uniformly over the time interval $[t_o, t_\mathrm{f}]$ (though non-uniform knots placements may also be considered). Both dynamics and constraints will be enforced at the collocation points. The problem can be stated as the following nonlinear programming form:

$$\min_{y\in\mathbb{R}^M} F(y) \qquad \text{subject to} \qquad \begin{cases} \Phi(z(y), \dot{z}(y), \ldots, z^{(n-r)}(y)) = 0, \\ \quad lb \le c(y) \le ub, \end{cases} \tag{4.15}$$

where

$$y = (C_1^1, \ldots, C_{p_1}^1, C_1^{r+1}, \ldots, C_{p_{r+1}}^{r+1}, \ldots, C_1^n, \ldots, C_{p_n}^n).$$

The coefficients of the B-spline basis functions can be found using nonlinear programming.

4.5 Choosing Cost Functions

The receding horizon control methodology is a very powerful tool for design of feedback controllers for constrained, nonlinear control systems. While the controllers that it produces are guaranteed to be stable under appropriate conditions, the choice of cost functions is left to the designer and can often require substantial trial and error. In this section we describe some tools for helping obtain cost functions based on insights from linear systems theory.

Design approach

The basic philosophy that we propose is illustrated in Figure 4.5. We begin with

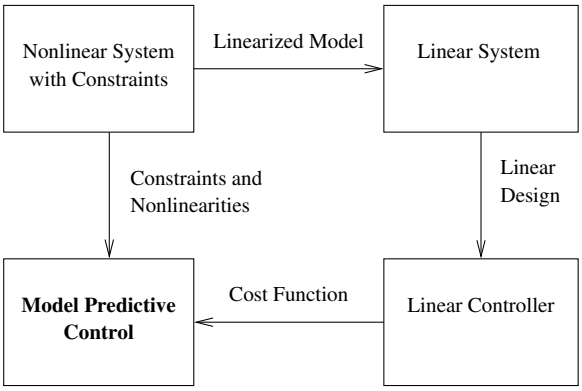


Figure 4.5: Optimization-based control approach.

a nonlinear system, including a description of the constraint set. We linearize this system about a representative equilibrium point and perform a linear control design using standard control design tools. Such a design can provide provably robust performance around the equilibrium point and, more importantly, allows the designer to meet a wide variety of formal and informal performance specifications through experience and the use of sophisticated linear design tools.

The resulting linear control law then serves as a *specification* of the desired control performance for the entire nonlinear system. We convert the control law specification into a receding horizon control formulation, chosen such that for the linearized system, the receding horizon controller gives comparable performance. However, because of its use of optimization tools that can handle nonlinearities and constraints, the receding horizon controller is able to provide the desired performance over a much larger operating envelope than the controller design based just on the linearization. Furthermore, by choosing cost formulations that have certain properties, we can provide proofs of stability for the full nonlinear system and, in some cases, the constrained system.

The advantage of the proposed approach is that it exploits the power of humans in designing sophisticated control laws in the absence of constraints with the power of computers to rapidly compute trajectories that optimize a given cost function in the presence of constraints. New advances in online trajectory generation serve as an enabler for this approach and their demonstration on representative flight control experiments shows their viability [MFHM05]. This approach can be extended to existing nonlinear paradigms as well, as we describe in more detail below.

An advantage of optimization-based approaches is that they allow the potential for online customization of the controller. By updating the model that the optimization uses to reflect the current knowledge of the system characteristics, the controller can take into account changes in parameters values or damage to sensors or actuators. In addition, environmental models that include dynamic constraints can be included, allowing the controller to generate trajectories that satisfy complex operating conditions. These modifications allow for many state- and environment-dependent uncertainties to including the receding horizon feedback loop, providing potential robustness with respect to those uncertainties.

A number of approaches in receding horizon control employ the use of terminal state equality or inequality constraints, often together with a terminal cost, to ensure closed loop stability. In Primbs et al. [PND99], aspects of a stability-guaranteeing, global control Lyapunov function (CLF) were used, via state and control constraints, to develop a stabilizing receding horizon scheme. Many of the nice characteristics of the control Lyapunov function controller together with better cost performance were realized. Unfortunately, a global control Lyapunov function is rarely available and often not possible.

Motivated by the difficulties in solving constrained optimal control problems, researchers have developed an alternative receding horizon control strategy for the stabilization of nonlinear systems [JYH01]. In this approach, closed loop stability is ensured through the use of a terminal cost consisting of a control Lyapunov function that is an incremental upper bound on the optimal cost to go. This terminal cost eliminates the need for terminal constraints in the optimization and gives a dramatic speed-up in computation. Also, questions of existence and regularity of optimal solutions (very important for online optimization) can be dealt with in a rather straightforward manner.

Inverse Optimality

The philosophy presented here relies on the synthesis of an optimal control problem from specifications that are embedded in an externally generated controller design. This controller is typically designed by standard classical control techniques for a nominal process, absent constraints. In this framework, the controller's performance, stability and robustness specifications are translated into an equivalent optimal control problem and implemented in a receding horizon fashion.

One central question that must be addressed when considering the usefulness of this philosophy is: *Given a control law, how does one find an equivalent optimal control formulation?* The paper by Kalman [Kal64] lays a solid foundation for this class of problems, known as *inverse optimality*. In this paper, Kalman considers the class of linear time-invariant (LTI) processes with full-state feedback and a single input variable, with an associated cost function that is quadratic in the input and state variables. These assumptions set up the well-known linear quadratic regulator (LQR) problem, by now a staple of optimal control theory.

In Kalman's paper, the mathematical framework behind the LQR problem is laid out, and necessary and sufficient algebraic criteria for optimality are presented in terms of the algebraic Riccati equation, as well as in terms of a condition on the return difference of the feedback loop. In terms of the LQR problem, the task of synthesizing the optimal control problem comes down to finding the integrated cost weights Q_x and Q_u given only the dynamical description of the process represented by matrices A and B and of the feedback controller represented by K. Kalman delivers a particularly elegant frequency characterization of this map [Kal64], which we briefly summarize here.

We consider a linear system

$$\dot{x} = Ax + Bu \qquad x \in \mathbb{R}^n, u \in \mathbb{R}^m \tag{4.16}$$

with state x and input u. We consider only the single input, single output case for

now ($m = 1$). Given a control law

$$u = -Kx$$

we wish to find a cost functional of the form

$$J = \int_0^\mathsf{T} x^\mathsf{T} Q_x x + u^\mathsf{T} Q_u u \, dt + x^\mathsf{T}(T) P_T x(T) \tag{4.17}$$

where $Q_x \in \mathbb{R}^{n\times n}$ and $Q_u \in \mathbb{R}^{m\times m}$ define the integrated cost, $P_T \in \mathbb{R}^{n\times n}$ is the terminal cost, and T is the time horizon. Our goal is to find $P_T > 0$, $Q_x > 0$, $Q_u > 0$, and $T > 0$ such that the resulting optimal control law is equivalent to $u = Kx$.

The optimal control law for the quadratic cost function (4.17) is given by

$$u = -R^{-1} B^\mathsf{T} P(t),$$

where $P(t)$ is the solution to the Riccati ordinary differential equation

$$-\dot{P} = A^\mathsf{T} P + PA - PBR^{-1}B^\mathsf{T} P + Q \tag{4.18}$$

with terminal condition $P(T) = P_T$. In order for this to give a control law of the form $u = -Kx$ for a constant matrix K, we must find P_T, Q_x, and Q_u that give a constant solution to the Riccati equation (4.18) and satisfy $R^{-1}B^\mathsf{T} P = K$. It follows that P_T, Q_x and Q_u should satisfy

$$\begin{aligned} A^\mathsf{T} P_T + P_T A - P_T B Q_u^{-1} B^\mathsf{T} P_T + Q = 0 \\ -Q_u^{-1} B^\mathsf{T} P_T = K. \end{aligned} \tag{4.19}$$

We note that the first equation is simply the normal algebraic Riccati equation of optimal control, but with P_T, Q, and R yet to be chosen. The second equation places additional constraints on R and P_T.

Equation (4.19) is exactly the same equation that one would obtain if we had considered an infinite time horizon problem, since the given control was constant and hence $P(t)$ was forced to be constant. This infinite horizon problem is precisely the one that Kalman considered in 1964, and hence his results apply directly. Namely, in the single-input single-output case, we can always find a solution to the coupled equations (4.19) under standard conditions on reachability and observability [Kal64]. The equations can be simplified by substituting the second relation into the first to obtain

$$A^\mathsf{T} P_T + P_T A - K^\mathsf{T} R K + Q = 0.$$

This equation is linear in the unknowns and can be solved directly (remembering that P_T, Q_x and Q_u are required to be positive definite).

The implication of these results is that any state feedback control law satisfying these assumptions can be realized as the solution to an appropriately defined receding horizon control law. Thus, we can implement the design framework summarized in Figure 4.5 for the case where our (linear) control design results in a state feedback controller.

The above results can be generalized to nonlinear systems, in which one takes a nonlinear control system and attempts to find a cost function such that the given controller is the optimal control with respect to that cost.

The history of inverse optimal control for nonlinear systems goes back to the early work of Moylan and Anderson [MA73]. More recently, Sepulchre et al. [SJK97] showed that a nonlinear state feedback obtained by Sontag's formula from a control Lyapunov function (CLF) is inverse optimal. The connections of this inverse optimality result to passivity and robustness properties of the optimal state feedback are discussed in Jankovic *et al.* [JSK99]. Most results on inverse optimality do not consider the constraints on control or state. However, the results on the unconstrained inverse optimality justify the use of a more general nonlinear loss function in the integrated cost of a finite horizon performance index combined with a real-time optimization-based control approach that takes the constraints into account.

4.6 Implementation on the Caltech Ducted Fan (with M. Milam and N. Petit)

To demonstrate the use of the techniques described in the previous section, we present an implementation of optimization-based control on the Caltech Ducted Fan, a real-time, flight control experiment that mimics the longitudinal dynamics of an aircraft. The experiment is show in Figure 4.6. The work in this section is based on the work of Mark Milam and Ryan Franz [Mil03, MFHM05] and is drawn from [MHJ+03]. It illustrates some of the practical considerations in implementing receding horizon control and describes techniques for addressing computational limitations.

The experimental results in this section were implemented using the Nonlinear Trajectory Generation (NTG) [MM02] software package. The sequential quadratic programming package NPSOL by [GMSW] is used as the nonlinear programming solver in NTG. When specifying a problem to NTG, the user is required to state the problem in terms of some choice of outputs and its derivatives. The user is also required to specify the regularity of the variables, the placement of the knot points, the order and regularity of the B-splines, and the collocation points for each output. The python-control package also includes an implementation of B-splines and these can be used for both flat and non-flat optimal trajectory generation problems.

Description of the Caltech Ducted Fan Experiment

The Caltech ducted fan is an experimental testbed designed for research and development of nonlinear flight guidance and control techniques for Uninhabited Aerial Vehicles (UAVs). The fan is a scaled model of the longitudinal axis of a flight vehicle and flight test results validate that the dynamics replicate qualities of actual flight vehicles [MM99].

The ducted fan has three degrees of freedom: the boom holding the ducted fan is allowed to operate on a cylinder, 2 m high and 4.7 m in diameter, permitting horizontal and vertical displacements. A counterweight is connected to the vertical axis of the stand, allowing the effective mass of the fan to be adjusted. Also, the wing/fan assembly at the end of the boom is allowed to rotate about its center of

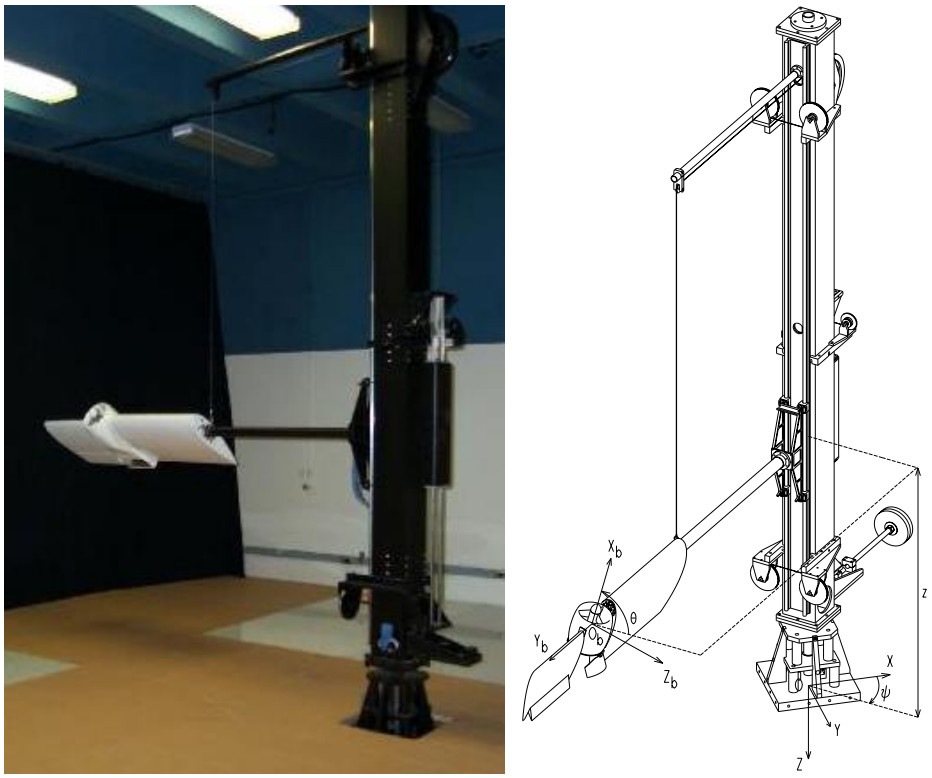


Figure 4.6: Caltech ducted fan.

mass. Optical encoders mounted on the ducted fan, counterweight pulley, and the base of the stand measure the three degrees of freedom. The fan is controlled by commanding a current to the electric motor for fan thrust and by commanding RC servos to control the thrust vectoring mechanism.

The sensors are read and the commands sent by a DSP-based multi-processor system, comprised of a D/A card, a digital I/O card, two Texas Instruments C40 signal processors, two Compaq Alpha processors, and a high-speed host PC interface. A real-time interface provides access to the processors and I/O hardware. The NTG software resides on both of the Alpha processors, each capable of running real-time optimization.

The ducted fan is modeled in terms of the position and orientation of the fan, and their velocities. Letting x represent the horizontal translation, z the vertical translation and θ the rotation about the boom axis, the equations of motion are given by

$$\begin{aligned} m\ddot{x} + F_{X_a} - F_{X_b}\cos\theta - F_{Z_b}\sin\theta &= 0, \\ m\ddot{z} + F_{Z_a} + F_{X_b}\sin\theta - F_{Z_b}\cos\theta &= mg_{\text{eff}}, \\ J\ddot{\theta} - M_a + \frac{1}{r_s}I_p\Omega\dot{x}\cos\theta - F_{Z_b}r_f &= 0, \end{aligned} \tag{4.20}$$

where $F_{X_a} = D\cos\gamma + L\sin\gamma$ and $F_{Z_a} = -D\sin\gamma + L\cos\gamma$ are the aerodynamic forces and F_{X_b} and F_{Z_b} are thrust vectoring body forces in terms of the lift (L),

drag (D), and flight path angle (γ). I_p and Ω are the moment of inertia and angular velocity of the ducted fan propeller, respectively. J is the moment of ducted fan and r_f is the distance from center of mass along the X_b axis to the effective application point of the thrust vectoring force. The angle of attack α can be derived from the pitch angle θ and the flight path angle γ by

$$\alpha = \theta - \gamma.$$

The flight path angle can be derived from the spatial velocities by

$$\gamma = \arctan \frac{-\dot{z}}{\dot{x}}.$$

The lift (L) ,drag (D), and moment (M) are given by

$$L = qSC_L(\alpha) \qquad D = qSC_D(\alpha) \qquad M = \bar{c}SC_M(\alpha),$$

respectively. The dynamic pressure is given by $q = \frac{1}{2}\rho V^2$. The norm of the velocity is denoted by V, S the surface area of the wings, and ρ is the atmospheric density. The coefficients of lift ($C_L(\alpha)$), drag ($C_D(\alpha)$) and the moment coefficient ($C_M(\alpha)$) are determined from a combination of wind tunnel and flight testing and are described in more detail in [MM99], along with the values of the other parameters.

Real-Time Trajectory Generation

In this section we describe the implementation of the trajectory generation algorithms by using NTG to generate minimum time trajectories in real time. An LQR-based regulator is used to stabilize the system. We focus in this section on aggressive, forward flight trajectories. The next section extends the controller to use a receding horizon controller, but on a simpler class of trajectories.

Stabilization around a reference trajectory

The results in this section rely on the traditional two degree of freedom design paradigm described in Chapter 2. In this approach, a local control law (inner loop) is used to stabilize the system around the trajectory computed based on a nominal model. This compensates for uncertainties in the model, which are predominantly due to aerodynamics and friction. Elements such as the ducted fan flying through its own wake, ground effects and velocity- and angle-of-attack dependent thrust contribute to the aerodynamic uncertainty. Actuation models are not used when generating the reference trajectory, resulting in another source of uncertainty.

Since only the position of the fan is measured, we must estimate the velocities. We use an extended Kalman filter (described in later chapters) with the optimal gain matrix is gain scheduled on the (estimated) forward velocity.

The stabilizing LQR controllers were gain scheduled on pitch angle, θ, and the forward velocity, $\dot{x}$. The pitch angle was allowed to vary from $-\pi/2$ to $\pi/2$ and the velocity ranged from 0 to 6 m/s. The weights were chosen differently for the hover-to-hover and forward flight modes. For the forward flight mode, a smaller

weight was placed on the horizontal (x) position of the fan compared to the hover-to-hover mode. Furthermore, the z weight was scheduled as a function of forward velocity in the forward flight mode. There was no scheduling on the weights for hover-to-hover. The elements of the gain matrices for each of the controller and observer are linearly interpolated over 51 operating points.

Nonlinear trajectory generation parameters

We solve a minimum time optimal control problem to generate a feasible trajectory for the system. The system is modeled using the nonlinear equations described above and computed the open loop forces and state trajectories for the nominal system. This system is not known to be differentially flat (due to the aerodynamic forces) and hence we cannot completely eliminate the differential constraints.

We choose three outputs, $z_1 = x$, $z_2 = z$, and $z_3 = \theta$, which results in a system with one remaining differential constraint. Each output is parameterized with four, sixth order C^4 piecewise polynomials over the time interval scaled by the minimum time. A fourth output, $z_4 = T$, is used to represent the time horizon to be minimized and is parameterized by a scalar. There are a total of 37 variables in this optimization problem. The trajectory constraints are enforced at 21 equidistant breakpoints over the scaled time interval.

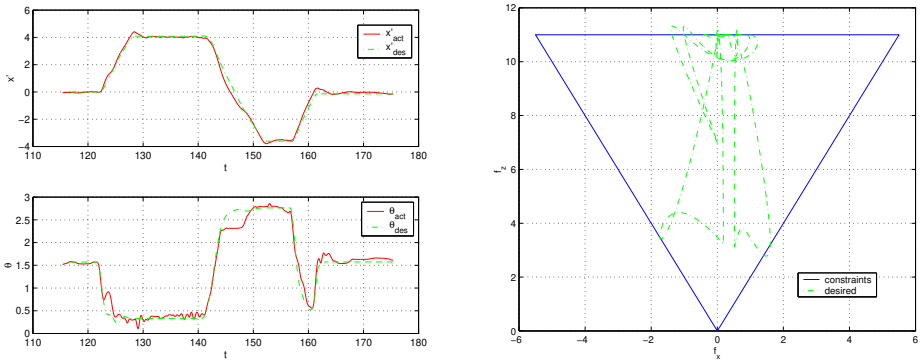
There are many considerations in the choice of the parameterization of the outputs. Clearly there is a trade between the parameters (variables, initial values of the variables, and breakpoints) and measures of performance (convergence, run-time, and conservative constraints). Extensive simulations were run to determine the right combination of parameters to meet the performance goals of our system.

Forward flight

To obtain the forward flight test data, an operator commanded a desired forward velocity and vertical position with joysticks. We set the trajectory update time δ to 2 seconds. By rapidly changing the joysticks, NTG produces high angle of attack maneuvers. Figure 4.7aa depicts the reference trajectories and the actual θ and $\dot{x}$ over 60 s. Figure 4.7b shows the commanded forces for the same time interval. The sequence of maneuvers corresponds to the ducted fan transitioning from near hover to forward flight, then following a command from a large forward velocity to a large negative velocity, and finally returning to hover.

Figure 4.8 is an illustration of the ducted fan altitude and x position for these maneuvers. The air-foil in the figure depicts the pitch angle (θ). It is apparent from this figure that the stabilizing controller is not tracking well in the z direction. This is due to the fact that unmodeled frictional effects are significant in the vertical direction. This could be corrected with an integrator in the stabilizing controller.

An analysis of the run times was performed for 30 trajectories; the average computation time was less than one second. Each of the 30 trajectories converged to an optimal solution and was approximately between 4 and 12 seconds in length. A random initial guess was used for the first NTG trajectory computation. Subsequent NTG computations used the previous solution as an initial guess. Much improvement can be made in determining a "good" initial guess. Improvement in the initial guess will improve not only convergence but also computation times.



(a) System state (b) Input forces

Figure 4.7: Forward flight test case: (a) θ and $\dot{x}$ desired and actual, (b) desired F_{X_b} and F_{Z_b} with bounds.

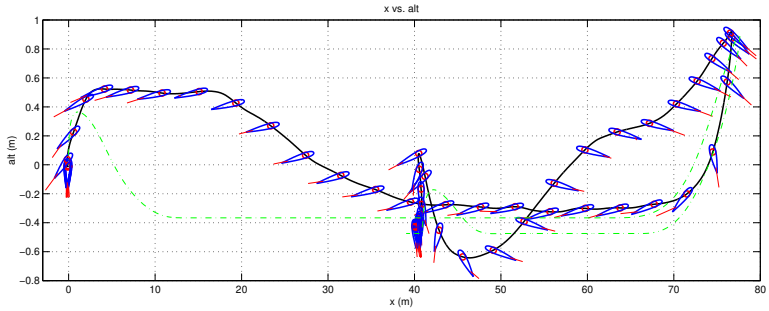


Figure 4.8: Forward flight test case: altitude and x position (actual (solid) and desired (dashed)). Airfoil represents actual pitch angle (θ) of the ducted fan.

Receding Horizon Control Implementation

The results of the previous section demonstrate the ability to compute optimal trajectories in real time, although the computation time was not sufficiently fast for closing the loop around the optimization. In this section, we make use of a shorter update time δ, a fixed horizon time T with a quadratic integral cost, and a control Lyapunov function terminal cost to implement a receding horizon controller as described in Section 4.2. We focus on the operation of the system to near hover, so that we can use the local linearization to find the terminal control Lyapunov function.

We have implemented the receding horizon controller on the ducted fan experiment where the control objective is to stabilize the hover equilibrium point. The quadratic cost is given by

$$\begin{aligned} L(x,u) &= \frac{1}{2}\hat{x}^\mathsf{T} Q \hat{x} + \frac{1}{2}\hat{u}^\mathsf{T} R \hat{u}, \\ V(x) &= \gamma \hat{x}^\mathsf{T} P \hat{x}, \end{aligned} \tag{4.21}$$

where

$$\begin{aligned}\hat{x} &= x - x_{eq} = (x, z, \theta - \pi/2, \dot{x}, \dot{z}, \dot{\theta}),\\ \hat{u} &= u - u_{eq} = (F_{X_b} - mg, F_{Z_b}),\\ Q &= \text{diag}\{4, 15, 4, 1, 3, 0.3\},\\ R &= \text{diag}\{0.5, 0.5\}.\end{aligned}$$

For the terminal cost, we choose $\gamma = 0.075$ and P is the unique stable solution to the algebraic Riccati equation corresponding to the linearized dynamics of equation (4.20) at hover and the weights Q and R. Note that if $\gamma = 1/2$, then $V(\cdot)$ is the control Lyapunov function for the system corresponding to the LQR problem. Instead V is a relaxed (in magnitude) control Lyapunov function, which achieved better performance in the experiment. In either case, V is valid as a control Lyapunov function only in a neighborhood around hover since it is based on the linearized dynamics. We do not try to compute off-line a region of attraction for this control Lyapunov function. Experimental tests omitting the terminal cost and/or the input constraints leads to instability. The results in this section show the success of this choice for V for stabilization. An inner-loop PD controller on $\theta, \dot{\theta}$ is implemented to stabilize to the receding horizon states $\theta_T^*, \dot{\theta}_T^*$. The θ dynamics are the fastest for this system and although most receding horizon controllers were found to be nominally stable without this inner-loop controller, small disturbances could lead to instability.

The optimal control problem is set-up in NTG code by parameterizing the three position states (x, z, θ), each with 8 B-spline coefficients. Over the receding horizon time intervals, 11 and 16 breakpoints were used with horizon lengths of 1, 1.5, 2, 3, 4 and 6 seconds. Breakpoints specify the locations in time where the differential equations and any constraints must be satisfied, up to some tolerance. The value of $F_{X_b}^{\max}$ for the input constraints is made conservative to avoid prolonged input saturation on the real hardware. The logic for this is that if the inputs are saturated on the real hardware, no actuation is left for the inner-loop θ controller and the system can go unstable. The value used in the optimization is $F_{X_b}^{\max} = 9$ N.

Computation time is non-negligible and must be considered when implementing the optimal trajectories. The computation time varies with each optimization as the current state of the ducted fan changes. The following notational definitions will facilitate the description of how the timing is set-up:

i	Integer counter of RHC computations
t_i	Value of current time when RHC computation i started
$\delta_c(i)$	Computation time for computation i
$u_T^*(i)(t)$	Optimal output trajectory corresponding to computation i, with time interval $t \in [t_i, t_i + T]$

A natural choice for updating the optimal trajectories for stabilization is to do so as fast as possible. This is achieved here by constantly resolving the optimization. When computation i is done, computation $i + 1$ is immediately started, so $t_{i+1} = t_i + \delta_c(i)$. Figure 4.9 gives a graphical picture of the timing set-up as the optimal input trajectories $u_T^*(\cdot)$ are updated. As shown in the figure, any computation i for $u_T^*(i)(\cdot)$ occurs for $t \in [t_i, t_{i+1}]$ and the resulting trajectory is applied for $t \in [t_{i+1}, t_{i+2}]$. At $t = t_{i+1}$ computation $i + 1$ is started for trajectory $u_T^*(i + 1)(\cdot)$, which is applied as soon as it is available ($t = t_{i+2}$). For the experimental runs

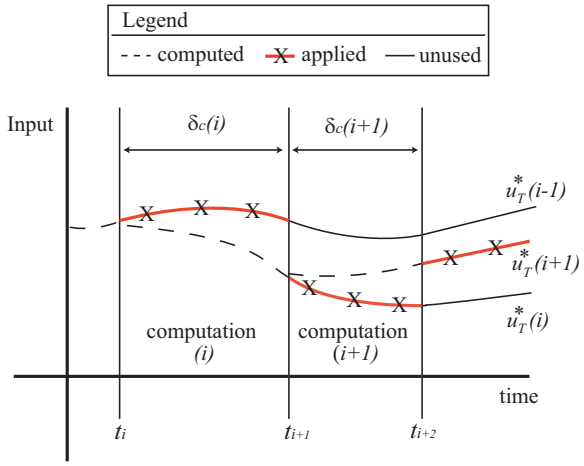
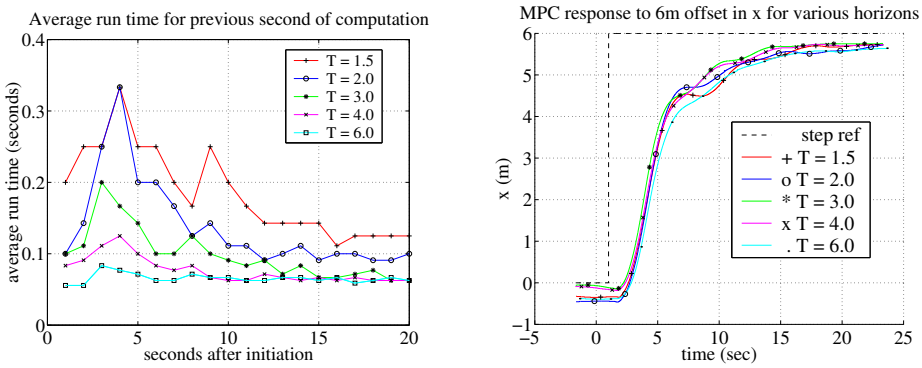


Figure 4.9: Receding horizon input trajectories.

detailed in the results, $\delta_c(i)$ is typically in the range of $[0.05, 0.25]$ seconds, meaning 4 to 20 optimal control computations per second. Each optimization i requires the current measured state of the ducted fan and the value of the previous optimal input trajectories $u_T^*(i-1)$ at time $t = t_i$. This corresponds to, respectively, 6 initial conditions for state vector x and 2 initial constraints on the input vector u. Figure 4.9 shows that the optimal trajectories are advanced by their computation time prior to application to the system. A dashed line corresponds to the initial portion of an optimal trajectory and is not applied since it is not available until that computation is complete. The figure also reveals the possible discontinuity between successive applied optimal input trajectories, with a larger discontinuity more likely for longer computation times. The initial input constraint is an effort to reduce such discontinuities, although some discontinuity is unavoidable by this method. Also note that the same discontinuity is present for the 6 open-loop optimal state trajectories generated, again with a likelihood for greater discontinuity for longer computation times. In this description, initialization is not an issue because we assume the receding horizon computations are already running prior to any test runs. This is true of the experimental runs detailed in the results.

The experimental results show the response of the fan with each controller to a 6 meter horizontal offset, which is effectively engaging a step-response to a change in the initial condition for x. The following details the effects of different receding horizon control parameterizations, namely as the horizon changes, and the responses with the different controllers to the induced offset.

The first comparison is between different receding horizon controllers, where time horizon is varied to be 1.5, 2.0, 3.0, 4.0 or 6.0 seconds. Each controller uses 16 breakpoints. Figure 4.10a shows a comparison of the average computation time as time proceeds. For each second after the offset was initiated, the data correspond to the average run time over the previous second of computation. Note that these computation times are substantially smaller than those reported for real-time trajectory generation, due to the use of the control Lyapunov function terminal cost versus the terminal constraints in the minimum-time, real-time trajectory genera-



(a) Average run time (b) Step responses

Figure 4.10: Receding horizon control: (a) moving one second average of computation time for RHC implementation with varying horizon time, (b) response of RHC controllers to 6 meter offset in x for different horizon lengths.

tion experiments.

There is a clear trend toward shorter average computation times as the time horizon is made longer. There is also an initial transient increase in average computation time that is greater for shorter horizon times. In fact, the 6 second horizon controller exhibits a relatively constant average computation time. One explanation for this trend is that, for this particular test, a 6 second horizon is closer to what the system can actually do. After 1.5 seconds, the fan is still far from the desired hover position and the terminal cost control Lyapunov function is large, likely far from its region of attraction. Figure 4.10b shows the measured x response for these different controllers, exhibiting a rise time of 8–9 seconds independent of the controller. So a horizon time closer to the rise time results in a more feasible optimization in this case.

4.7 Further Reading

Receding horizon control (more commonly referred to as model predictive control or MPC) has a long history and there are many good textbooks available. The book by Rawlings, Mayne, and Diehl [RMD17] is an excellent resource (available for free download), as well as the textbook (also freely available) by Borelli, Bemporad, and Morari [BBM17], which matches the material in the MATLAB-based MPT toolbox. An overview of the early evolution of commercially available MPC technology is given in [QB97] and a survey of the state of stability theory of MPC circa 2000 is given in [MRRS00].

Exercises

4.1. Consider a nonlinear control system

$$\dot{x} = f(x, u)$$

with linearization

$$\dot{x} = Ax + Bu.$$

Show that if the linearized system is reachable, then there exists a (local) control Lyapunov function for the nonlinear system. (Hint: start by proving the result for a stable system.)

4.2. Consider the optimal control problem given in Example 3.2:

$$\dot{x} = ax + bu, \qquad J = \tfrac{1}{2}\int_{t_0}^{t_f} u^2(t)\,dt + \tfrac{1}{2}cx^2(t_f),$$

where $x \in \mathbb{R}$ is a scalar state, $u \in \mathbb{R}$ is the input, the initial state $x(t_0)$ is given, and $a, b \in \mathbb{R}$ are positive constants. We take the terminal time t_f as given and let $c > 0$ be a constant that balances the final value of the state with the input required to get to that position. The optimal control for a finite time $T > 0$ is derived in Example 3.2. Now consider the infinite horizon cost

$$J = \tfrac{1}{2}\int_{t_0}^{\infty} u^2(t)\,dt$$

with $x(t)$ at $t = \infty$ constrained to be zero.

(a) Solve for $u^*(t) = -bPx^*(t)$ where P is the positive solution corresponding to the algebraic Riccati equation. Note that this gives an explicit feedback law ($u = -bPx$).

(b) Plot the state solution of the finite time optimal controller for the following parameter values

$$a = 2 \qquad b = 0.5 \qquad x(t_0) = 4$$
$$c = 0.1,\ 10 \qquad t_f = 0.5,\ 1,\ 10$$

(This should give you a total of 6 curves.) Compare these to the infinite time optimal control solution. Which finite time solution is closest to the infinite time solution? Why?

(c) Using the solution given in equation (3.5), implement the finite-time optimal controller in a receding horizon fashion with an update time of $\delta = 0.5$. Using the parameter values in part (b), compare the responses of the receding horizon controllers to the LQR controller you designed in part (a), from the same initial condition. What do you observe as c and t_f increase?

(Hint: you can write a Python script to do this by performing the following steps:

(i) set $t_0 = 0$

(ii) using the closed form solution for x^* from problem 1, plot $x(t)$, $t \in [t_0, t_f]$ and save $x_\delta = x(t_0 + \delta)$

(iii) set $x(t_0) = x_\delta$ and repeat step (ii) until x is small.)

4.3. In this problem we will explore the effect of constraints on control of the linear unstable system given by

$$\dot{x}_1 = 0.8x_1 - 0.5x_2 + 0.5u, \qquad \dot{x}_2 \quad = x_1 + 0.5u,$$

subject to the constraint that $|u| \leq a$ where a is a postive constant.

(a) Ignore the constraint ($a = \infty$) and design an LQR controller to stabilize the system. Plot the response of the closed system from the initial condition given by $x = (1, 0)$.

(b) Simulate the initial condition response of system for some finite value of a with an initial condition $x(0) = (1, 0)$. Numerically (trial and error) determine the smallest value of a for which the system goes unstable.

(c) Let $a_{\min}(\rho)$ be the smallest value of a for which the system is unstable from $x(0) = (\rho, 0)$. Plot $a_{\min}(\rho)$ for $\rho = 1$, 4, 16, 64, 256.

(d) *Optional*: Given $a > 0$, design and implement a receding horizon control law for this system. Show that this controller has larger region of attraction than the controller designed in part (b). (Hint: solve the finite horizon LQ problem analytically, using the bang-bang example as a guide to handle the input constraint.)

4.4. [Instability of MPC with short horizons (Mark Cannon, Oxford University, 2020)] Consider a linear, discrete time system with dynamics

$$x[k+1] = \begin{bmatrix} 1 & 0.1 \\ 0 & 2 \end{bmatrix} x[k] + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} u[k], \qquad y[k] = \begin{bmatrix} 1 & 0 \end{bmatrix} x[k]$$

with finite time horizon cost given by

$$J(x[k], u) = \sum_{i=0}^{N-1} \Big(y^2[k] + u^2[k] \Big) + y^2[N].$$

(a) Show that the predicted state of the system can be written in the form

$$\begin{bmatrix} x[k] \\ x[k+1] \\ \vdots \\ x[k+N] \end{bmatrix} = \mathcal{M} x[k] + \mathcal{L} \begin{bmatrix} u[k] \\ u[k+1] \\ \vdots \\ u[k+N-1] \end{bmatrix}$$

and give formulas for $\mathcal{M}$ and $\mathcal{L}$ in terms of A, B, and C for the case $N = 3$.

(b) Show that the cost function can be written as

$$J(x[k], u) = \bar{u}^{\mathsf{T}}[k] H \bar{u}[k] + 2x^{\mathsf{T}}[k] F \bar{u}[k] + x^{\mathsf{T}}[k] G x[k]$$

and give expressions for F, G, and H.

(c) Show that the RHC controller that minimizes the cost function for a horizon length of N can be written as $u = -Kx$ and find an expression for K in terms of F, G, and H. Show that for $N = 3$ the feedback gain is given by

$$K = \begin{bmatrix} 0.1948 & 0.1168 \end{bmatrix}.$$

(d) Compute the closed loop eigenvalues for the system with a receding horizon controller with $N = 3$ and show that the system is unstable. What is the smallest value of N such that the system is stable?

(e) Change the terminal cost use the optimal cost-to-go function returned by the `dlqr` command in MATLAB or Python. Verify that the closed loop system is stable for $N = 1, \ldots, 5$.

4.5. Consider the double integrator system from Example 4.1. A discrete time representation of this system with sampling time of 1 second is given by

$$x[k+1] = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{clip}(u), \qquad \text{where} \qquad \text{clip}(u) = \begin{cases} -1 & u < -1, \\ u & -1 \le u \le 1, \\ 1 & u > 1. \end{cases}$$

We choose the same weighting matrices as in Example 4.1:

$$Q_x = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \qquad Q_u = \begin{bmatrix} 1 \end{bmatrix}, \qquad P_1 = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}.$$

(a) Construct a discrete-time receding horizon control law for the system and recreate Figure 4.3 using $x_0 = (2, 1)$ as the initial condition. Your plot should should show the actual trajectory for x and u as solid lines and the predicted trajectories from the optimization as dashed lines.

(b) The discrete time equivalent of the conditions in Theorem 4.1 are

$$\min_u V(f(x,u)) - V(x) + L(x,u) \le 0 \quad \text{for all} \quad x,$$

where f represents the discrete time dynamics $x[k+1] = f(x[k], u[k])$. Check to see if these conditions are satisfied for this system using the weights above along the states that are visited along the trajectory of the system in (a). (For Theorem 4.1 to hold you would need to show this condition at all states x, so we are just checking a subset in this problem.)

(c) Replace the terminal cost P_1 with the solution to the discrete time algebraic Riccati equation (which can be obtained using the `dlqr` command in MATLAB or Python), recompute and plot the initial condition response of the receding horizon controller, and check that whether satisfies the stability condition along the states in the trajectory.

(d) Modify the terminal cost P_1 obtained in part (c) by 10X in each direction ($P_1' = 0.1P_1$ and $P_1' = 10P_1$), recompute and plot the initial condition response of the receding horizon controller, and check that whether satisfies the stability condition along the trajectory.

4.6. Consider the dynamics of the vectored thrust aircraft described in Examples 2.4 and 3.5. Assume that the inputs must satisfy the constraints

$$|F_1| \leq 0.1\,|F_2|, \qquad 0 \leq F_2 \leq 1.5\,mg.$$

Design a receding horizon controller for the system that stabilizes the origin using an optimization horizon of $T = 5$ s and an update period of $\Delta T = 0.1$ s. Demonstrate the performance of your controller from initial conditions starting at initial position $(x_0, y_0) = (0 \text{ m}, 5 \text{ m})$ and desired final position $(x_\text{f}, y_\text{f}) = (10 \text{ m}, 5 \text{ m})$ (all other states should be zero).