
Optimization-Based Control

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

DRAFT v1.6a, 11 February 2008
© California Institute of Technology
All rights reserved.

This manuscript is for review purposes only and may not be reproduced, in whole or in part, without written consent from the author.

Chapter 3

Receding Horizon Control

This set of notes builds on the previous two chapters and explores the use of online optimization as a tool for control of nonlinear control. We begin with an high-level discussion of optimization-based control, refining some of the concepts initially introduced in Chapter 1. We then describe the technique of receding horizon control (RHC), including a proof of stability for a particular form of receding horizon control that makes use of a control Lyapunov function as a terminal cost. We conclude the chapter with a detailed design example, in which we can explore some of the computational tradeoffs in optimization-based control.

Prerequisites. Readers should be familiar with the concepts of trajectory generation and optimal control as described in Chapters 1 and 2. For the proof of stability for the receding horizon controller that we use, familiarity with Lyapunov stability analysis at the level given in ÅM08, Chapter 4 (Dynamic Behavior) is required.

3.1 Optimization-Based Control

Optimization-based control refers to the use of online, optimal trajectory generation as a part of the feedback stabilization of a (typically nonlinear) system. The basic idea is to use a *receding horizon* control technique: a (optimal) feasible trajectory is computed from the current position to the desired position over a finite time T horizon, used for a short period of time $\delta < T$, and then recomputed based on the new position. Development and application of receding horizon control (also called model predictive control, or MPC) originated in process control industries where plants being controlled are sufficiently slow to permit its implementation. An overview of the evolution of commercially available MPC technology is given in [QB97] and a survey of the current state of stability theory of MPC is given in [MRRS00].

Design approach

The basic philosophy that we propose is illustrated in Figure 3.1. We begin with a nonlinear system, including a description of the constraint set. We linearize this system about a representative equilibrium point and perform

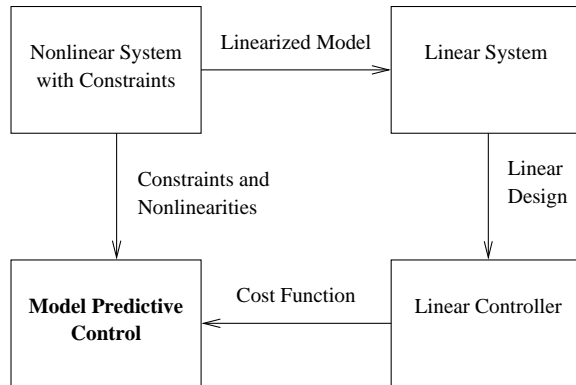


Figure 3.1: Optimization-based control approach.

a linear control design using standard (modern) tools. Such a design gives 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.

This 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 on-line trajectory generation serve as an enabler for this approach and their demonstration on representative flight control experiments shows their viability. This approach can be extended to existing nonlinear paradigms as well, as we describe in more detail below.

A key advantage of optimization-based approaches is that they allow the potential for customization of the controller based on changes in *mission*, *condition*, and *environment*. Because the controller is solving the optimization problem online, updates can be made to the cost function, to change the desired operation of the system; to the model, to reflect changes in parameter values or damage to sensors and actuators; and to the constraints,

to reflect new regions of the state space that must be avoided due to external influences. Thus, many of the challenges of designing controllers that are robust to a large set of possible uncertainties become embedded in the online optimization.

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 were used, via state and control constraints, to develop a stabilizing receding horizon scheme. Many of the nice characteristics of the CLF 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, we 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 plant, 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 seminal paper by R. E. 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) plants 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 and R given only the dynamical description of the plant 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].

There are two natural extensions of these results: extension to more general dynamical systems and extension to more general optimal control formulations. The contribution of this paper is the simultaneous extension of this approach to systems with constraints along with the extension to the more general receding horizon control framework. A first step in this approach is extension of inverse optimal results to the finite horizon case.

It is important to note that Kalman's results are restricted to the infinite horizon case ($T \rightarrow \infty$) in addition to the assumptions of linearity, time-invariance and quadratic costs. This additional assumption is necessary to derive the results associated with the algebraic Riccati equation ($\dot{P} = 0$). However, we will show that through proper application of terminal costs, the same inverse optimality problem can be soundly addressed in the case of finite horizon length. This problem is addressed by the authors in this paper in the context of Kalman's work; the review of these results will be made mathematically explicit in the next section.

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]. The past research on inverse optimality does 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.

Control Lyapunov Functions

For the optimal control problems that we introduce in the next section, we will make use of a terminal cost that is also a control Lyapunov function for the system. 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. A more complete treatment is given in [KKK95].

Consider a nonlinear control system

$$\dot{x} = f(x, u), \quad x \in \mathbb{R}^n, u \in \mathbb{R}^m. \quad (3.1)$$

Definition 3.1 (Control Lyapunov Function). A locally positive function $V : \mathbb{R}^n \rightarrow \mathbb{R}_+$ is called a *control Lyapunov function (CLF)* for a control system (3.1) if

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

In general, it is difficult to find a CLF 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 CLF by solving an LQR problem.

It is a well known result that the problem of minimizing the quadratic performance index,

$$J = \int_0^\infty (x^T(t)Qx(t) + u^T Ru(t))dt \quad \text{subject to} \quad \begin{aligned} \dot{x} &= Ax + Bu, \\ x(0) &= x_0, \end{aligned} \quad (3.2)$$

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

$$A^T P + PA - PBR^{-1}B^T P + Q = 0 \quad (3.3)$$

The optimal control action is given by

$$u = -R^{-1}B^T Px$$

and $V = x^T Px$ is a CLF for the system.

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

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

where the pairs (A, B) and $(Q^{\frac{1}{2}}, A)$ are assumed to be stabilizable and detectable respectively. Obviously the obtained CLF $V(x) = x^T Px$ will be valid only in a region around the equilibrium $(0, 0)$.

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].

Finite Horizon Optimal Control

We briefly review the problem of optimal control over a finite time horizon as presented in Chapter 2 to establish the notation for the chapter and set some more specific conditions required for receding horizon control. Given an initial state x_0 and a control trajectory $u(\cdot)$ for a nonlinear control system $\dot{x} = f(x, u)$, the state trajectory $x^u(\cdot; x_0)$ is the (absolutely continuous) curve

in \mathbb{R}^n satisfying

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

for $t \geq 0$.

The performance of the system will be measured by a given incremental cost $L : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}$ that is C^2 and fully penalizes both state and control according to

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

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 $p \in \mathbb{R}^n$ represent the co-state, we also require that the pre-Hamiltonian function $u \mapsto p^T f(x, u) + L(x, u) =: K(x, u, p)$ be strictly convex for each $(x, p) \in \mathbb{R}^n \times \mathbb{R}^n$ and that there is a C^2 function $\bar{u}^* : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^m : (x, p) \mapsto \bar{u}^*(x, p)$ providing the global minimum of $K(x, u, p)$. The Hamiltonian $H(x, p) := K(x, \bar{u}^*(x, p), p)$ 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 trivially 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 functions $u(\cdot)$ belong to some reasonable class of admissible controls (e.g., piecewise continuous or measurable). The function $x \mapsto J_\infty^*(x)$ is often called the *optimal value function* for the infinite horizon optimal control problem.

For the class of f and L considered, we know that $J_\infty^*(\cdot)$ is a positive definite C^2 function on a neighborhood of the origin. This follows from the *geometry* of the corresponding Hamiltonian system (see [HO01] and the references therein). In particular, since $(x, p) = (0, 0)$ is a hyperbolic critical point of the C^1 Hamiltonian vector field $X_H(x, p) := (D_2 H(x, p), -D_1 H(x, p))^T$, the local properties of $J_\infty^*(\cdot)$ are determined by the linear-quadratic approximation to the problem and, moreover, $D^2 J_\infty^*(0) = P > 0$ where P is the

stabilizing solution of the appropriate algebraic Riccati equation.

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

$$J_T(x, u(\cdot)) = \int_0^T L(x^u(\tau; x), u(\tau)) d\tau + V(x^u(T; x)) \quad (3.4)$$

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

$$\Gamma_r^\infty := \{x \in \Gamma^\infty : J_\infty^*(x) \leq r^2\}$$

are compact and path connected and moreover $\Gamma^\infty = \bigcup_{r \geq 0} \Gamma_r^\infty$. 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 incremental cost is quadratically bounded from below. We refer to sub-level sets of $J_T^*(\cdot)$ and $V(\cdot)$ using

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

and

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

These results provide the technical framework needed for receding horizon control.

3.2 Receding Horizon Control with CLF Terminal Cost

In receding horizon control, a finite horizon optimal control problem is solved, generating an open-loop state and control trajectories. The resulting control trajectory is then applied to the system for a fraction of the horizon length. This process is then repeated, resulting in a sampled data feedback law. Although receding horizon control has been successfully used in the process control industry, its application to fast, stability critical nonlinear systems has been more difficult. This is mainly due to two issues. The first is that the finite horizon optimizations must be solved in a relatively short period of time. Second, it can be demonstrated using linear examples that a naive application of the receding horizon strategy can have disastrous effects, often rendering a system unstable. Various approaches have been proposed to tackle this second problem; see [MRRS00] for a comprehensive review of this literature. The theoretical framework presented here also addresses the stability issue directly, but is motivated by the need to relax the computational demands of existing stabilizing MPC formulations.

Receding horizon control provides a practical strategy for the use of model information through on-line optimization. Every δ seconds, an optimal control problem is solved over a T second horizon, starting from the current state. The first δ seconds of the optimal control $u_T^*(\cdot; x(t))$ is then applied to the system, driving the system from $x(t)$ at current time t to $x_T^*(\delta, x(t))$ at the next sample time $t + \delta$ (assuming no model uncertainty). We denote this receding horizon scheme as $\mathcal{RH}(T, \delta)$.

In defining (unconstrained) finite horizon approximations to the infinite horizon problem, the key design parameters are the terminal cost function $V(\cdot)$ and the horizon length T (and, perhaps also, the increment δ). What choices will result in success?

It is well known (and easily demonstrated with linear examples), that simple truncation of the integral (i.e., $V(x) \equiv 0$) may have disastrous effects if $T > 0$ is too small. Indeed, although the resulting value function may be nicely behaved, the “optimal” receding horizon closed loop system can be unstable.

A more sophisticated approach is to make good use of a suitable terminal cost $V(\cdot)$. Evidently, the best choice for the terminal cost is $V(x) = J_\infty^*(x)$ since then the optimal finite and infinite horizon costs are the same. Of course, if *the* optimal value function were available there would be no need to solve a trajectory optimization problem. What properties of the optimal value function should be retained in the terminal cost? To be effective, the terminal cost should account for the discarded tail by ensuring that the

origin can be reached from the terminal state $x^u(T; x)$ in an efficient manner (as measured by L). One way to do this is to use an appropriate control Lyapunov function which is also an upper bound on the cost-to-go.

The following theorem shows that the use of a particular type of CLF is in fact effective, providing rather strong and specific guarantees.

Theorem 3.1. [JYH01] *Suppose that the terminal cost $V(\cdot)$ is a control Lyapunov function such that*

$$\min_{u \in \mathbb{R}^m} (\dot{V} + L)(x, u) \leq 0 \quad (3.5)$$

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 an $\bar{r}(T) \geq r_v$ such that $\Gamma_{\bar{r}(T)}^T$ 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 $\Lambda \subset \Gamma_{\bar{r}(T)}^T$ for all $T \geq T^*$.

Theorem 3.1 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 $\Gamma_{r_v}^T$. 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 $\Gamma_{\bar{r}(T)}^T$, 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

$$J_T^*(x_T^*(\delta; x)) \leq J_T^*(x) - \int_0^\delta q(L(x_T^*(\tau; x), u_T^*(\tau; x))) d\tau. \quad (3.6)$$

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 (3.6) 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 CLF trajectory (x^k, u^k) .

We now proceed to show that the inequality (??) 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 now use the bound

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

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

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

Finally, using the optimality of u_T^* we have that $J_T^*(x_T^*(\delta; x)) \leq J_T(x_T^*(\delta; x), \tilde{u}(\cdot))$ we obtain equation (3.6). \square

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 CLF 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 $x^{CLF}(\cdot) \in \mathcal{X}$, 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).

3.3 Receding Horizon Control Using Differential Flatness

In this section we demonstrate how to use 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

$$\dot{x} = f(x) + g(x)u, \quad (3.7)$$

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 (3.7) that minimizes the performance index (3.4), subject to a vector of initial, final, and trajectory constraints

$$\begin{aligned} lb_0 &\leq \psi_0(x(t_0), u(t_0)) \leq ub_0, \\ lb_f &\leq \psi_f(x(t_f), u(t_f)) \leq ub_f, \\ lb_t &\leq S(x, u) \leq ub_t, \end{aligned} \quad (3.8)$$

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

$$\min_{(x,u)} J(x, u) \quad \text{subject to} \quad \begin{cases} \dot{x} = f(x) + g(x)u, \\ lb \leq c(x, u) \leq ub. \end{cases} \quad (3.9)$$

Numerical Solution Using Collocation

A numerical approach to solving this optimal control problem is to use the direct collocation method outlined in Hargraves and Paris [HP87]. The idea behind this approach is to transform the optimal control problem into a nonlinear programming problem. This is accomplished by discretizing time into a grid of $N - 1$ intervals

$$t_0 = t_1 < t_2 < \dots < t_N = t_f \quad (3.10)$$

and approximating the state x and the control input u as piecewise polynomials \hat{x} and \hat{u} , respectively. Typically a cubic polynomial is chosen for the states and a linear polynomial for the control on each interval. Collocation is then used at the midpoint of each interval to satisfy equation (3.7). Let $\hat{x}(x(t_1), \dots, x(t_N))$ and $\hat{u}(u(t_1), \dots, u(t_N))$ denote the approximations to x and u , respectively, depending on $(x(t_1), \dots, x(t_N)) \in \mathbb{R}^{nN}$ and $(u(t_1), \dots, u(t_N)) \in \mathbb{R}^N$ corresponding to the value of x and u at the grid

points. Then one solves the following finite dimension approximation of the original control problem (3.9):

$$\min_{y \in \mathbb{R}^M} F(y) = J(\hat{x}(y), \hat{u}(y)) \quad \text{subject to} \quad \begin{cases} \dot{\hat{x}} - f(\hat{x}(y)) + g(\hat{x}(y))\hat{u}(y) = 0, \\ lb \leq c(\hat{x}(y), \hat{u}(y)) \leq ub, \\ \forall t = \frac{t_j + t_{j+1}}{2} \quad j = 1, \dots, N-1 \end{cases} \quad (3.11)$$

where $y = (x(t_1), u(t_1), \dots, x(t_N), u(t_N))$, and $M = \dim y = (n+1)N$.

Seywald [Sey94] suggested an improvement to the previous method (see also [Bry99, p. 362]). Following this work, one first solves a subset of system dynamics in equation (3.9) for the the control in terms of combinations of the state and its time derivative. Then one substitutes for the control in the remaining system dynamics and constraints. Next all the time derivatives \dot{x}_i are approximated by the finite difference approximations

$$\dot{\hat{x}}(t_i) = \frac{x(t_{i+1}) - x(t_i)}{t_{i+1} - t_i}$$

to get

$$\left. \begin{array}{l} p(\dot{\hat{x}}(t_i), x(t_i)) = 0 \\ q(\dot{\hat{x}}(t_i), x(t_i)) \leq 0 \end{array} \right\} \quad i = 0, \dots, N-1.$$

The optimal control problem is turned into

$$\min_{y \in \mathbb{R}^M} F(y) \quad \text{subject to} \quad \begin{cases} p(\dot{\hat{x}}(t_i), x(t_i)) = 0 \\ q(\dot{\hat{x}}(t_i), x(t_i)) \leq 0 \end{cases} \quad (3.12)$$

where $y = (x(t_1), \dots, x(t_N))$, and $M = \dim y = nN$. As with the Hargraves and Paris method, this parameterization of the optimal control problem (3.9) can be solved using nonlinear programming.

The dimensionality of this discretized problem is lower than the dimensionality of the Hargraves and Paris method, where both the states and the input are the unknowns. This induces substantial improvement in numerical implementation.

Differential Flatness Based Approach

The results of Seywald give a constrained optimization problem in which we wish to minimize a cost functional subject to $n-1$ equality constraints, corresponding to the system dynamics, at each time instant. In fact, it is usually possible to reduce the dimension of the problem further. Given an output, it is generally possible to parameterize the control and a part of the state in terms of this output and its time derivatives. In contrast to the previous approach, one must use more than one derivative of this output for this purpose.

When the whole state and the input can be parameterized with one

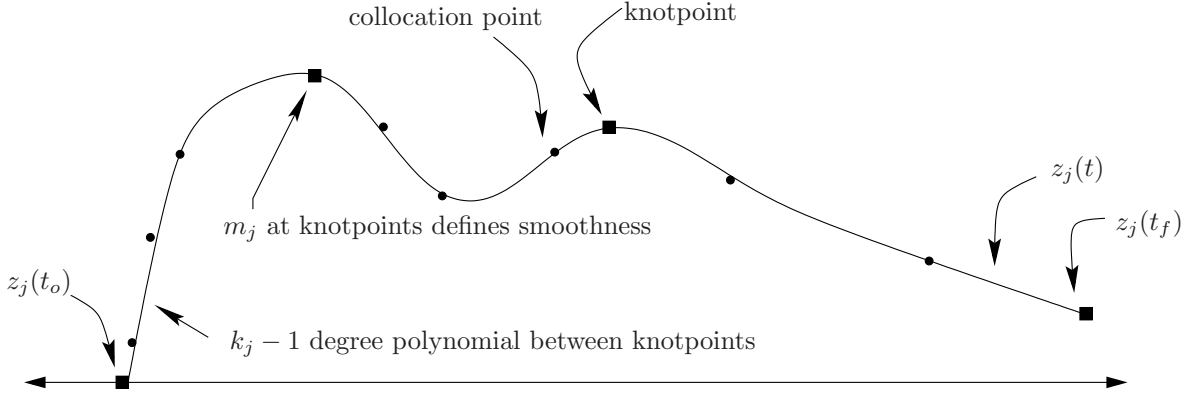


Figure 3.2: Spline representation of a variable.

output, the system is differentially flat, as described in Section 1.3. 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

$$\begin{aligned} x &= \alpha(z, \dot{z}, \dots, z^{(q)}) \\ u &= \beta(z, \dot{z}, \dots, z^{(q)}) \\ \Phi(z, \dot{z}, \dots, z^{n-r}) &= 0 \end{aligned} \quad (3.13)$$

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.

Unlike the approach of Seywald, it is not realistic to use finite difference approximations as soon as $r > 2$. In this context, it is convenient to represent z using B-splines. 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 3.2. Doing so we get

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

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 $(z_1, z_2, \dots, z_{n-r})$ is thus represented by $M = \sum_{j \in \{1, r+1, \dots, n\}} p_j$ coefficients.

In general, w collocation points are chosen uniformly over the time interval $[t_0, t_f]$ (though optimal knots placements or Gaussian points 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:

$$\min_{y \in \mathbb{R}^M} F(y) \quad \text{subject to} \quad \begin{cases} \Phi(z(y), \dot{z}(y), \dots, z^{(n-r)}(y)) = 0 \\ lb \leq c(y) \leq ub \end{cases} \quad (3.14)$$

where

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

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

A software package called Nonlinear Trajectory Generation (NTG) has been written to solve optimal control problems in the manner described above (see [MMM00] for details). 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.

3.4 Implementation on the Caltech Ducted Fan

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 shown in Figure 3.3.

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 Combat Aerial Vehicles (UCAVs). 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. Also, the wing/fan assembly at the end of the boom is allowed to rotate about its center of mass. Optical encoders mounted on the ducted fan, gearing wheel, 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.

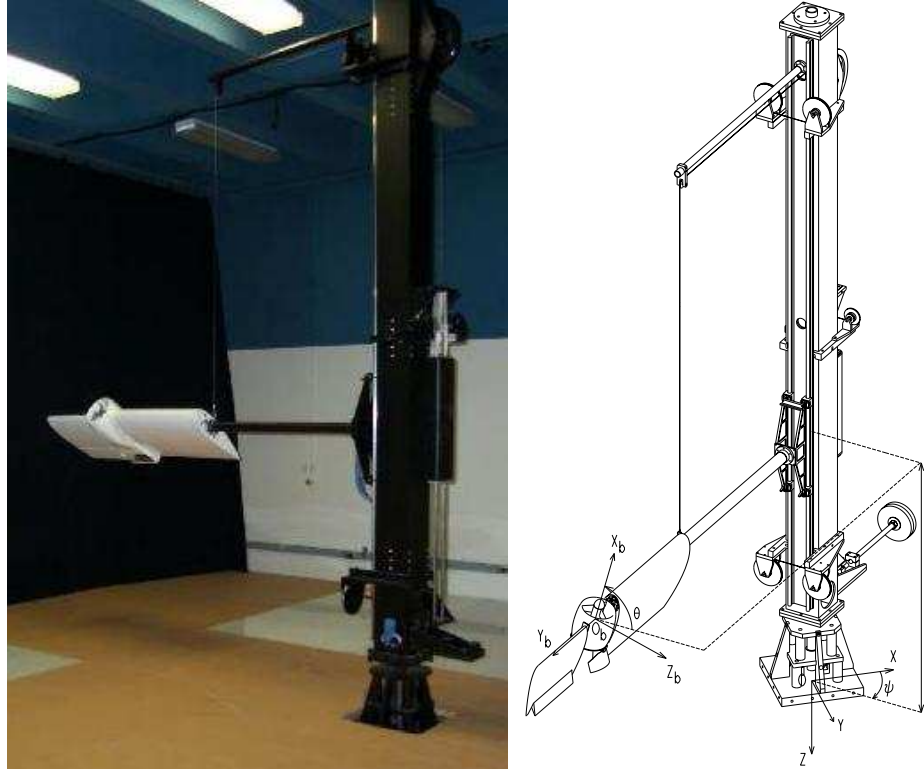


Figure 3.3: Caltech ducted fan.

The sensors are read and the commands sent by a dSPACE 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 ISA bus to interface with a PC. The dSPACE system provides a real-time interface to the 4 processors and I/O card to the 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

$$\begin{aligned}
 m\ddot{x} + F_{X_a} - F_{X_b} \cos \theta - F_{Z_b} \sin \theta &= 0 \\
 m\ddot{z} + F_{Z_a} + F_{X_b} \sin \theta - F_{Z_b} \cos \theta &= mg_{\text{eff}} \\
 J\ddot{\theta} - M_a + \frac{1}{r_s} I_p \Omega \dot{x} \cos \theta - F_{Z_b} r_f &= 0,
 \end{aligned} \tag{3.15}$$

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

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

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 demonstrate the trajectory generation results by using NTG to generate minimum time trajectories in real time. An LQR-based regulator is used to stabilize the system, allow us to focus on the trajectory generation properties. 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 Reference Trajectory

The results in this section rely on the traditional two degree of freedom design paradigm described in the Introduction. 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 thrust not modeled as a function of velocity and angle of attack contribute to the aerodynamic uncertainty. The friction in the vertical direction is also not considered in the model. The prismatic joint has an unbalanced load creating an effective moment on the bearings. The vertical frictional force of the ducted fan stand varies with the vertical acceleration of the ducted fan as well as the forward velocity. 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 with the optimal gain matrix is gain scheduled on the (estimated) forward velocity. The Kalman filter outperformed other methods that computed the derivative using only the position data and a filter.

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 (intervals), sixth order, C^4 (multiplicity), 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. By choosing the outputs to be parameterized in this way, we are in effect controlling the frequency content of inputs. Since we are not including the actuators in the model, it would be undesirable to have inputs with a bandwidth higher than the actuators. 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.

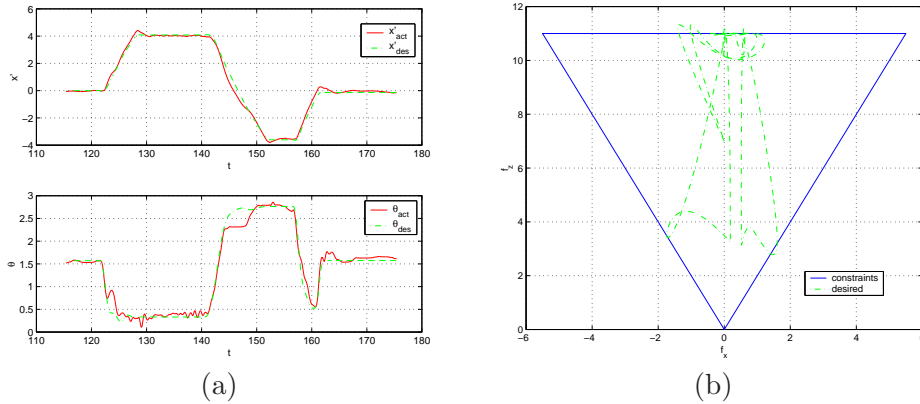


Figure 3.4: Forward flight test case: (a) θ and \dot{x} desired and actual, (b) desired F_{X_b} and F_{Z_b} with bounds.

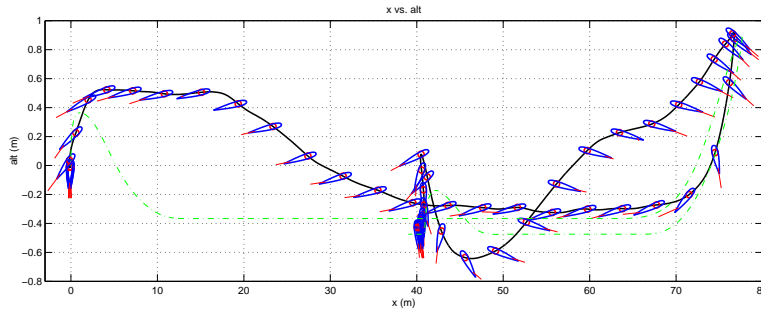


Figure 3.5: Forward flight test case: altitude and x position (actual (solid) and desired (dashed)). Airfoil represents actual pitch angle (θ) of the ducted fan.

Forward Flight

To obtain the forward flight test data, the operator commanded a desired forward velocity and vertical position with the joysticks. We set the trajectory update time, δ to 2 seconds. By rapidly changing the joysticks, NTG produces high angle of attack maneuvers. Figure 3.4(a) depicts the reference trajectories and the actual θ and \dot{x} over 60 sec. Figure 3.4(b) 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 3.5 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.

Receding Horizon Control

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 CLF terminal cost to implement the receding horizon controller described in Section 3.2. We also limit the operation of the system to near hover, so that we can use the local linearization to find the terminal CLF.

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

$$\begin{aligned} L(x, u) &= \frac{1}{2} \hat{x}^T Q \hat{x} + \frac{1}{2} \hat{u}^T R \hat{u} \\ V(x) &= \gamma \hat{x}^T P \hat{x} \end{aligned} \tag{3.16}$$

where

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

$\gamma = 0.075$ and P is the unique stable solution to the algebraic Riccati equation corresponding to the linearized dynamics of equation (1.5) at hover and the weights Q and R . Note that if $\gamma = 1/2$, then $V(\cdot)$ is the CLF for the system corresponding to the LQR problem. Instead V is a relaxed (in magnitude) CLF, which achieved better performance in the experiment. In either case, V is valid as a CLF 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 CLF. 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

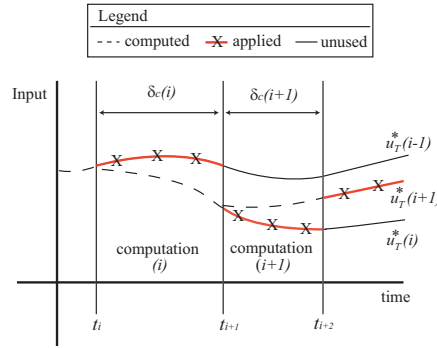


Figure 3.6: Receding horizon input trajectories

states θ_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 MPC computations
t_i	Value of current time when MPC 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 3.6 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 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 3.6 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 3.7(a) shows a comparison of the average computation time as time proceeds. For each second after the offset was initiated, the data corresponds 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 CLF terminal cost versus the terminal constraints in the minimum-time, real-time trajectory generation 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 CLF is large, likely far from its region of attraction. Figure 3.7(b)

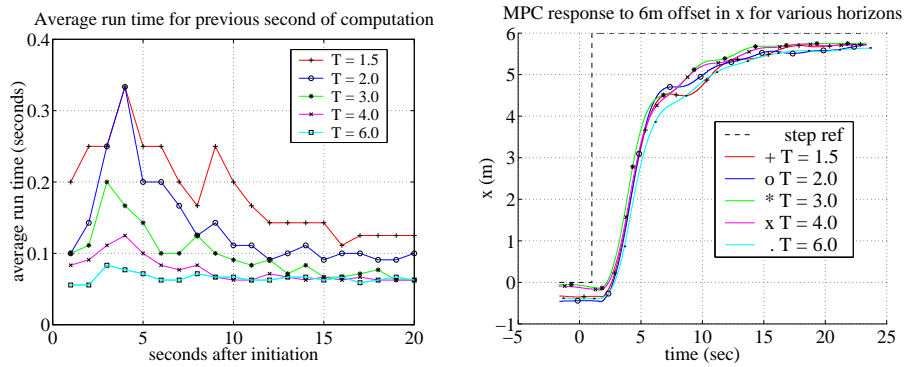


Figure 3.7: Receding horizon control: (a) moving one second average of computation time for MPC implementation with varying horizon time, (b) response of MPC controllers to 6 meter offset in x for different horizon lengths.

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.

3.5 Further Reading

Exercises

3.1 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.)

3.2 Consider the optimal control problem given in Example 2.2:

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

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 2.2. Now consider the infinite horizon cost

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

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

$$\begin{aligned} a &= 2 & b &= 0.5 & x(t_0) &= 4 \\ c &= 0.1, 10 & t_f &= 0.5, 1, 10 \end{aligned}$$

(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?

Using the solution given in equation (2.7), implement the finite-time optimal controller in a receding horizon fashion with an update time of $\delta = 0.5$. Using the parameter values in problem 1(b), Compare the responses of the receding horizon controllers to the LQR controller you designed for problem 1, from the same initial condition. What do you observe as c and t_f increase?

(Hint: you can write a MATLAB 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.)

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

$$\begin{aligned} \dot{x}_1 &= 0.8x_1 - 0.5x_2 + 0.5u \\ \dot{x}_2 &= x_1 + 0.5u \end{aligned}$$

subject to the constraint that $|u| \leq a$ where a is a positive 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) Use SIMULINK or `ode45` to simulate the the 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.)

Bibliography

- [Bry99] A. E. Bryson. *Dynamic optimization*. Addison Wesley, 1999.
- [dB78] C. de Boor. *A Practical Guide to Splines*. Springer-Verlag, 1978.
- [GMSW] P. E. Gill, W. Murray, M. A. Saunders, and M. Wright. *User's Guide for NPSOL 5.0: A Fortran Package for Nonlinear Programming*. Systems Optimization Laboratory, Stanford University, Stanford, CA 94305.
- [HO01] J. Hauser and H. Osiuga. On the geometry of optimal control: The inverted pendulum example. In *American Control Conference*, 2001.
- [HP87] C. Hargraves and S. Paris. Direct trajectory optimization using nonlinear programming and collocation. *AIAA J. Guidance and Control*, 10:338–342, 1987.
- [Isi89] A. Isidori. *Nonlinear Control Systems*. Springer-Verlag, 2nd edition, 1989.
- [Jad01] A. Jadbabaie. *Nonlinear Receding Horizon Control: A Control Lyapunov Function Approach*. PhD thesis, California Institute of Technology, Control and Dynamical Systems, 2001.
- [JSK99] M. Jankovic, R. Sepulchre, and P. V. Kokotović. CLF based designs with robustness to dynamic input uncertainties. *Systems Control Letters*, 37:45–54, 1999.
- [JYH01] A. Jadbabaie, J. Yu, and J. Hauser. Unconstrained receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 46, 2001.
- [Kal64] R. E. Kalman. When is a linear control system optimal? *J. Basic Engrg. Trans. ASME Ser. D*, 86:51–60, 1964.
- [KKK95] M. Krstić, I. Kanellakopoulos, and P. Kokotović. *Nonlinear and Adaptive Control Design*. Wiley, 1995.
- [LM67] E. B. Lee and L. Markus. *Foundations of Optimal Control Theory*. Robert E. Krieger Publishing Company, 1967.
- [MA73] P. J. Moylan and B. D. O. Anderson. Nonlinear regulator theory and an inverse optimal control problem. *IEEE Trans. on Automatic Control*, 18(5):460–454, 1973.
- [MM99] M. B. Milam and R. M. Murray. A testbed for nonlinear flight control techniques: The Caltech ducted fan. In *Proc. IEEE International Conference on Control and Applications*, 1999.
- [MMM00] M. B. Milam, K. Mushambi, and R. M. Murray. A computational approach to real-time trajectory generation for constrained mechanical systems. In *Proc. IEEE Control and Decision Conference*, 2000.
- [MRRS00] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

- [PND99] J. A. Primbs, V. Nevistić, and J. C. Doyle. Nonlinear optimal control: A control Lyapunov function and receding horizon perspective. *Asian Journal of Control*, 1(1):1–11, 1999.
- [QB97] S. J. Qin and T. A. Badgwell. An overview of industrial model predictive control technology. In J.C. Kantor, C.E. Garcia, and B. Carnahan, editors, *Fifth International Conference on Chemical Process Control*, pages 232–256, 1997.
- [Sey94] H. Seywald. Trajectory optimization based on differential inclusion. *J. Guidance, Control and Dynamics*, 17(3):480–487, 1994.
- [SJK97] R. Sepulchre, M. Jankovic, and P. V. Kokotović. *Constructive Nonlinear Control*. Springer, London, 1997.
- [Son83] E. D. Sontag. A Lyapunov-like characterization of asymptotic controllability. *SIAM Journal of Control and Optimization*, 21:462–471, 1983.