OPTIMIZATION TECHNICAL REPORT 02-06, AUGUST 2002, COMPUTER SCIENCES DEPT, UNIV. OF WISCONSIN
TEXAS-WISCONSIN MODELING AND CONTROL CONSORTIUM REPORT TWMCC-2002-02

NONLINEAR MODEL PREDICTIVE CONTROL VIA FEASIBILITY-PERTURBED SEQUENTIAL QUADRATIC PROGRAMMING

MATTHEW J. TENNY*, STEPHEN J. WRIGHT†, AND JAMES B. RAWLINGS‡

Abstract. Model predictive control requires the solution of a sequence of continuous optimization problems that are nonlinear if a nonlinear model is used for the plant. We describe briefly a trust-region feasibility-perturbed sequential quadratic programming algorithm (developed in a companion report), then discuss its adaptation to the problems arising in nonlinear model predictive control. Computational experience with several representative sample problems is described, demonstrating the effectiveness of the proposed approach.

1. Introduction. Model predictive control (MPC), also referred to as receding-horizon control, is a method that applies on-line optimization to a model of a system, with the aim of steering the system to a desired target state. In recent years, MPC has become a prominent advanced control technique, especially in the chemical process industry. However, for computational reasons, MPC applications largely have been limited to linear models; that is, those in which the dynamics of the system model are linear. Such models often do not capture the dynamics of the system adequately, especially in regions that are not close to the target state. In these cases, nonlinear models are necessary to describe accurately the behavior of physical systems.

From an algorithmic point of view, nonlinear model predictive control requires the repeated solution of nonlinear optimal control problems. At certain times during the control period, the state of the system is estimated, and an optimal control problem is solved over a finite time horizon (commencing at the present time), using this state estimate as the initial state. The control component at the current time is used as the input to the system. Algorithms for nonlinear optimal control, which are often specialized nonlinear programming algorithms, can therefore be used in the context of nonlinear model predictive control, with the additional imperatives that the problem must be solved in "real time," and that good estimates of the solution may be available from the state and control profiles obtained at the previous timepoint.

The linear MPC problem is well studied from an optimization standpoint. It gives rise to a sequence of optimal control problems with quadratic objectives and linear dynamics, which can be viewed as structured convex quadratic programming problems. These problems can be solved efficiently by algorithms that exploit the structure. For example, an interior-point method that uses a recursive relation to solve the linear systems at each iteration has been described by Rao, Wright, and Rawlings [21]. The nonlinear MPC problem has been less widely studied, and is a topic of recent interest.

Albuquerque et al. [1] have used a sequential quadratic programming (SQP) approach, using a primal-dual interior-point method to solve the quadratic programming subproblems. They used a general-purpose solved for sparse symmetric indefinite systems to solve the linear systems arising at each interior-point iteration, together with finite-difference methods to estimate the Hessian terms. They assumed that a good solution estimate was available for each nonlinear problem, so the algorithm contains

*Chemical Engineering Department, University of Wisconsin, Madison, WI 53706, U.S.A.
†Computer Sciences Department, 1210 W. Dayton Street, University of Wisconsin, Madison, WI 53706, U.S.A.
‡Chemical Engineering Department, University of Wisconsin, Madison, WI 53706, U.S.A.

no techniques to ensure global convergence. An example shows that a large application can be controlled successfully with their approach, in that the optimal control problem can be solved in the interval between changes to the system inputs. Later work of Bartlett, Wächter, and Biegler [2] describes the use of active-set quadratic programming solvers as possible alternatives to the interior-point solvers of [1], and concludes that for problems with few degrees of freedom, the active-set approach may be more efficient. Cervantes et al. [8] described application of an interior-point method directly to the nonlinear optimal control formulation, of the type that has recently been proposed for general nonlinear programming problems. This method generates steps by applying a modified SQP method to a barrier-function reformulation of the nonlinear program, and also makes use of a line search, a merit function, and reduced-space quasi-Newton Hessian approximations. Biegler, Cervantes, and Wächter [4] modify this approach by using a preconditioned conjugate gradient approach to solve the linear equations, allowing them to use finite-difference approximations to the exact Hessian-vector products in place of the explicit quasi-Newton reduced Hessian approximations. Rather than a single merit function, they also use a "filter" criterion to select the line search parameter, an approach that has been the subject of much recent (practical and theoretical) investigation in the context of general nonlinear programming.

Diehl et al. [10] consider the solution of the nonlinear optimal control problem in the specific context of nonlinear MPC. An SQP framework is used, with the Hessians in the quadratic program being either the Hessian of the objective function in the nonlinear MPC problem, or a partitioned quasi-Newton approximation introduced by Bock and Plitt [6]. A line search approach based on a nonsmooth penalty function is used to ensure global convergence; additional details are given in Leineweber et al. [17]. Their method also include an "initial value embedding" strategy, in which approximate derivative information (based on the previous iteration, or a reference trajectory) is used to generate the first SQP step cheaply at each new timepoint.

The works described above have a number of common features. First, they use multiple-shooting or collocation techniques to formulate the underlying continuous-time problem as a problem with finitely many variables that is suitable for solution by structured nonlinear programming techniques. Second, their iterates consist of both control variables and state variables, which are not required to be consistent with respect to the state dynamics. (In the language of nonlinear programming, the iterates are infeasible.) This "simultaneous" approach is known to have advantages in terms of stability; open-loop unstable systems have been observed to give rise to instability in algorithmic approaches that use the model equation to eliminate the states to produce a formulation involving only the control variables. As a result of the infeasible iterates, these algorithms construct a merit function (typically based on the objective function value and the sum of constraint violations) to assess the worth of different points, or else use a filter approach based on the objective function and constraint violations separately. Third, the methods typically use a line search approach to curtail steps that appear to be unacceptable.

In this paper, we describe an approach that differs from those above in two fundamental ways. First, it computes iterates containing both state and control components, but perturbs these to retain feasibility with respect to the constraints at every interation. Second, it replaces the line-search globalization approach with a scaled trust-region approach. The algorithm is essentially the feasibility-perturbed SQP method described by Wright and Tenny [23], adapted to the nonlinear MPC

context. The quadratic programming subproblems are solved by using the specialized interior-point approach described in [21].

By retaining feasibility of all iterates, the algorithm gains several significant advantages. First, the objective function can be used as a merit function, greatly simplifying the description of the algorithm. Algorithms that allow infeasible iterates must construct a merit function from some combination of objective function, constraint infeasibilities, Lagrange multiplier estimates, and various parameters. Since we insist on consistency of the model equation at every iteration, it may be objected that we run the risk of encountering the stability problems that attend the inputs-only formulation. However, by using a change of variables within the feasibility perturbation strategy, we avoid such problems. A second advantage of the feasible approach is that the latest iterate can be used as a (suboptimal) feasible solution, if it is necessary to terminate the solution process early. Third, feasibility may allow a more natural formulation of the problem than may be needed when infeasible iterates are allowed. For example, when one of the states is a concentration of a chemical, a naturally nonnegative variable, algorithms that allow infeasible points may allow (nonphysical) negative values of this variable to be considered at some iterations, leading to unpredictable behavior of the algorithm. This may be remedied by introducing an additional nonnegativity constraint and insisting on feasibility with respect to this constraint, or by a change of variables. A feasible-point algorithm, on the other hand, will not produce nonphysical values of this variable at any iteration, hence allowing a more natural formulation and obviating any additional nonnegativity constraints.

Our computational experience also shows that our algorithm requires fewer iterates than other SQP-type approaches on a wide range of problems. We believe this performance is due in large part to the retention of feasibility, and the avoidance of unpredictable algorithmic behavior that comes with allowing infeasible points.

We note that other features of the approach of Diehl et al. [10] that are specific to the nonlinear MPC context, including the initial-value embedding, can also be incorporated into our approach. We omit further discussion of these issues, however, and focus on the solution of the nonlinear optimal control problem that arises at each timepoint.

An earlier version of the algorithm of this paper was applied to a copolymerization reactor by Tenny, Rawlings, and Bindlish [22]. Because of the analysis in companion work [23], the algorithm is now on a more solid footing. We also add several enhancements, including the use of a stabilizing change of variables during the feasibility perturbation stage and the choice of a trust region scaling matrix.

The remainder of the paper is structured as follows. In Section 2, we describe briefly the feasibility-perturbed SQP algorithm, Algorithm FP-SQP, of Wright and Tenny [23]. Section 3 introduces our formulation of the nonlinear MPC problem we solve, while the SQP subproblem arising from this formulation is presented in Section 4. The specific elements of Algorithm FP-SQP are presented in the subsequent sections. Section 5 describes the perturbation technique used to maintain feasibility of the iterates. Section 6 presents the various Hessian approximations that can be used in the SQP subproblem. Our trust-region scaling matrix for the NMPC problem is derived in Section 7. Finally, Section 8 contains computational results on a variety of nonlinear control models.

2. The Trust-Region Projected Sequential Quadratic Programming Algorithm. We describe the trust-region feasibility-perturbed sequential quadratic programming algorithm, which we refer to as Algorithm FP-SQP, with respect to the

following general formulation of a constrained nonlinear optimization problem:

$$\min f(z) \;\text{ subject to }\; c(z) = 0, \;\; d(z) \leq 0, \tag{2.1}$$

where $z \in \mathsf{R}^n$ is the vector of variables, $f : \mathsf{R}^n \to \mathsf{R}$, $c : \mathsf{R}^n \to \mathsf{R}^m$, and $d : \mathsf{R}^n \to \mathsf{R}^r$ are smooth (twice continuously differentiable) functions. We denote by $\mathcal{F}$ the set of feasible points for (2.1). A full description of this algorithm and derivation of its convergence properties appears in [23].

Algorithm FP-SQP generates a sequence of feasible iterates $\{z^j\}_{j=0,1,2,\ldots}$, where a step from the current iterate z to the next iterate is obtained by first solving the following quadratic programming subproblem for Δz:

$$\min_{\Delta z} m(\Delta z) \stackrel{\text{def}}{=} \nabla f(z)^T \Delta z + \frac{1}{2} \Delta z^T H \Delta z \;\text{ subject to} \tag{2.2a}$$

$$c(z) + \nabla c(z)^T \Delta z = 0, \;\; d(z) + \nabla d(z)^T \Delta z \leq 0, \tag{2.2b}$$

$$\|D \Delta z\|_p \leq \Delta, \tag{2.2c}$$

where (2.2b) represents linearization of the constraints $c(\cdot)$ and $d(\cdot)$ around the current iterate z, (2.2c) is a trust-region constraint (where $p \in [1, \infty]$ denotes the choice of norm), and D is a scaling matrix for the trust region. We make the assumption (Assumption 1 below) that (2.2b) and (2.2c) together bound the size of the full vector Δz. This assumption holds trivially if D in (2.2c) is uniformly nonsingular, but we are interested also in cases in which D has zero eigenvalues. The matrix H in (2.2a) is assumed to be symmetric but not necessarily positive semidefinite. For best local convergence behavior, H should be a good approximation to the Hessian of the Lagrangian for the problem (2.1) on the nullspace of the gradients of the constraints active at the solution z^*.

Having solved (2.2) for Δz, we obtain a candidate step $\widetilde{\Delta z}$ by perturbing Δz in such a way as to satisfy two conditions. First, *feasibility*:

$$z + \widetilde{\Delta z} \in \mathcal{F}. \tag{2.3}$$

Second, *asymptotic exactness*: There is a continuous monotonically increasing function $\phi : \mathsf{R}^+ \to \mathsf{R}^+$ with $\phi(0) = 0$ such that

$$\|\Delta z - \widetilde{\Delta z}\|_2 \leq \phi(\|\Delta z\|_2) \, \|\Delta z\|_2. \tag{2.4}$$

The acceptability of a candidate step $\widetilde{\Delta z}$ depends on a "sufficient decrease" test, which makes use of the ratio ρ_j defined at iteration j as follows:

$$\rho_j = \frac{f(z^j) - f(z^j + \widetilde{\Delta z}^j)}{-m_j(\Delta z^j)}, \tag{2.5}$$

where z^j is the current iterate, Δz^j solves (2.2), and $\widetilde{\Delta z}^j$ satisfies (2.3) and (2.4). If ρ_j exceeds a small positive quantity η, we accept the step and set $z^{j+1} = z^j + \widetilde{\Delta z}^j$, and possibly adjust the trust-region radius Δ and scaling matrix D in preparation for the next iteration. Otherwise, we set $z^{j+1} = z^j$, decrease Δ, and calculate a new candidate step.

We specify the algorithm formally as follows.

Algorithm 2.1 (FP-SQP).

Given starting point z_0, trust-region upper bound $\bar{\Delta} > 0$, initial radius $\Delta_0 \in (0, \bar{\Delta})$, $\eta \in [0, 1/4)$, and $p \in [1, \infty]$;
for $j = 0, 1, 2, \cdots$
  Obtain Δz^j by solving (2.2);
  Seek $\widetilde{\Delta z}^j$ with the properties (2.3) and (2.4);
  if *no such $\widetilde{\Delta z}^j$ can be found;*
    $\Delta_{j+1} \leftarrow (1/2)\|D_j \Delta z^j\|_p$;
    $z^{j+1} \leftarrow z^j$; $D_{j+1} \leftarrow D_j$;
  else
    Calculate ρ_j using (2.5);
    if $\rho_j < 1/4$
      $\Delta_{j+1} \leftarrow (1/2)\|D_j \Delta z^j\|_p$;
    else if $\rho_j > 3/4$ *and* $\|D_j \Delta z^j\|_p = \Delta_j$
      $\Delta_{j+1} \leftarrow \min(2\Delta_j, \bar{\Delta})$;
    else
      $\Delta_{j+1} \leftarrow \Delta_j$;
    if $\rho_j > \eta$
      $z^{j+1} \leftarrow z^j + \widetilde{\Delta z}^j$;
      choose new scaling matrix D_{j+1};
    else
      $z^{j+1} \leftarrow z^j$; $D_{j+1} \leftarrow D_j$;
end (for).

Before stating the main convergence results for this algorithm, which are proved in the companion paper [23], we introduce some notation and assumptions.

The Lagrangian function for (2.1) is

$$\mathcal{L}(z, \mu, \lambda) \stackrel{\text{def}}{=} f(z) + \mu^T c(z) + \lambda^T d(z), \tag{2.6}$$

where $\mu \in \mathsf{R}^m$ and $\lambda \in \mathsf{R}^r$ are Lagrange multipliers. The Karush-Kuhn-Tucker conditions for (2.1) are as follows:

$$\nabla_z \mathcal{L}(z, \mu, \lambda) = \nabla f(z) + \nabla c(z)\mu + \nabla d(z)\lambda = 0, \tag{2.7a}$$
$$c(z) = 0, \tag{2.7b}$$
$$d(z) \leq 0 \perp \lambda \geq 0, \tag{2.7c}$$

where $\perp$ indicates that $\lambda^T d(z) = 0$. The Mangasarian-Fromovitz constraint qualification (MFCQ) at a feasible point z, which ensures that the linearization of the constraints (2.2b) adequately captures the local geometry of $\mathcal{F}$ near z, requires that

$$\nabla c(z) \text{ has full column rank; and} \tag{2.8a}$$
$$\text{there exists a vector } v \in \mathsf{R}^n \text{ such that}$$
$$\nabla c(z)^T v = 0 \text{ and } v^T \nabla d_i(z) < 0 \text{ for all } i \in \mathcal{A}(z). \tag{2.8b}$$

If z is a stationary point for (2.1) at which (2.8) is satisfied, then there exist vectors μ and λ such that (2.7) is satisfied by the triplet (z, μ, λ).

The level set L_0 for the starting point z^0 of Algorithm FP-SQP is defined as follows:

$$L_0 \stackrel{\text{def}}{=} \{z \,|\, c(z) = 0,\ d(z) \leq 0,\ f(z) \leq f(z^0)\} \subset \mathcal{F}.$$

Our assumption on the trust-region bound (2.2c) is as follows:

ASSUMPTION 1. *There is a constant δ such that for all points $z \in L_0$ and all positive definite scaling matrices D used by the algorithm, we have for any Δz satisfying the constraints*

$$c(z) + \nabla c(z)^T \Delta z = 0, \;\; d(z) + \nabla d(z)^T \Delta z \leq 0$$

that

$$\delta^{-1} \|\Delta z\|_2 \leq \|D \Delta z\|_p \leq \delta \|\Delta z\|_2. \tag{2.9}$$

In this assumption, the constant that relates $\|\cdot\|_2$ with the equivalent norms $\|\cdot\|_p$ (for $p \in [1, \infty]$) is absorbed into the constant δ.

Our assumption on the boundedness of the set containing the iterates and on smoothness of the functions f, c, and d can be stated as follows.

ASSUMPTION 2. *The level set L_0 is bounded, and the functions f, c, and d in (2.1) are twice continuously differentiable in an open neighborhood $\mathcal{N}(L_0)$ of this set.*

Note that L_0 is certainly closed, so that if Assumption 2 holds, it is also compact.

For the third assumption, which bounds the distance from an infeasible point z to the feasible set $\mathcal{F}$ in terms of values of the constraint functions c and d, we define $\mathcal{B}(z,t)$ to be the open Euclidean ball about z:

$$\mathcal{B}(z,t) \stackrel{\text{def}}{=} \{y \,|\, \|y - z\| < t\}.$$

Here and in all subsequent formulae, an omitted subscript on $\|\cdot\|$ denotes the Euclidean norm.

ASSUMPTION 3. *For every point $\hat{z} \in L_0$, there are positive quantities ζ and $\hat{\Delta}_3$ such that for all $z \in \mathrm{cl}(\mathcal{B}(\hat{z}, \hat{\Delta}_3))$ we have*

$$\min_{v \in \mathcal{F}} \; \|v - z\| \leq \zeta \left(\|c(z)\| + \|[d(z)]_+\| \right), \tag{2.10}$$

where $[d(z)]_+ = [\max(d_i(z), 0)]_{i=1}^r$.

In the companion report [23], we proved two global convergence results for Algorithm FP-SQP. These results differ in their assumptions on the series of Hessian matrices H_j that appear in the quadratic programming subproblem (2.2). The first result makes an assumption on these matrices that is typically satisfied by quasi-Newton updating schemes.

THEOREM 2.1. *Suppose that Assumptions 1, 2, and 3 are satisfied, and assume that all limit points of the algorithm satisfy MFCQ. Suppose in addition that the approximate Hessians H_j satisfy the bound $\|H_j\|_2 \leq \sigma_0 + j\sigma_1$, for some nonnegative constants σ_0 and σ_1. Then the algorithm has a stationary limit point.*

The second convergence results assumes uniform boundedness of the Hessians H_j. Such a bound would hold if for instance each H_j were taken to be the Hessian of the Lagrangian (2.6), for suitable definitions of the Lagrange multiplier estimates μ and λ, or some positive definite modification of this matrix.

THEOREM 2.2. *Suppose that Assumptions 1, 2, and 3 are satisfied, and that the Hessian approximations H_j satisfy $\|H_j\| \leq \sigma$ for all j and some constant σ. Then the algorithm cannot have a limit point $\bar{z}$ at which the MFCQ condition (2.8) holds but the KKT conditions (2.7) are not satisfied.*

Under additional assumptions on the algorithm and the solution of the problem, rapid local convergence can also be proved. We refer the interested reader to [23, Section 4] for details.

3. Model Predictive Control: Problem Definition. For the purposes of our study, we examine discrete-time nonlinear systems in which the control moves are injected and measurements are taken at every sampling time. Usually, the system variables are shifted so that the desired value of the state variable is zero, and the fundamental aim of the control strategy is to drive the state variables to zero. At each sampling time, an open-loop optimal control problem is solved over a finite horizon N, with the aim of identifying the input u_0 that should be injected into the system at the present time. This quantity can be obtained by solving a nonlinear optimal control problem whose initial state is the current state of the system. Because u_0 must be calculated in real time, efficient algorithms for solving this nonlinear optimal control problem are desirable. Algorithm FP-SQP is well suited for this type of application, in part because each iterate it generates is feasible and therefore can be used as a suboptimal solution, if it is necessary to terminate the algorithm prior to convergence. This property is especially important in systems with faster sampling rates.

We consider the following formulation of this N-step finite-horizon MPC problem:

$$\min_{x,u,\eta} \Phi(x,u,\eta) \stackrel{\text{def}}{=} \sum_{k=0}^{N-1} [\mathcal{C}(x_k,u_k) + \Xi(\eta_k)] + \Gamma(x_N) + \Xi(\eta_N) \quad \text{s.t.} \tag{3.1a}$$

$$x_0 \text{ given}, \;\; x_{k+1} = F(x_k,u_k), \;\; Du_k \le d, \;\; Gx_k - \eta_k \le g, \;\; \eta_k \ge 0, \tag{3.1b}$$

where x, u, and η denote the sequences of vectors representing states, inputs, and state constraint violations, respectively; that is,

$$\begin{aligned} x &= (x_1, x_2, \ldots, x_N), \\ u &= (u_0, u_1, \ldots, u_{N-1}), \\ \eta &= (\eta_1, \eta_2, \ldots, \eta_N). \end{aligned}$$

Changing the use of m and n from Section 2, we suppose that $x_k \in \mathsf{R}^n$, $k = 0, 1, \ldots, N$ and $u_k \in \mathsf{R}^m$, $k = 0, 1, \ldots, N-1$. The stage cost $\mathcal{C}(x_k, u_k)$ and the terminal penalty $\Gamma(x_N)$ in (3.1a) are convex. Moreover, they are typically quadratic and *strictly* convex with respect to the input variables u_k in their arguments. The model function $F(x_k, u_k)$ is usually the source of nonlinearity in the optimization problem, making (3.1) a nonlinear programming problem. The model equation is usually obtained by integrating a DAE model.

A more natural formulation of the MPC problem might impose the state constraints $Gx_k \le g$ explicitly, but the formulation (3.1b) represents a "softening" of these constraints by introducing the violation variables η and including penalty terms $\Xi(\cdot)$ on the violations in the objective function (3.1a). We assume, as is usual, that these penalties also are quadratic; that is,

$$\Xi(\eta_k) = \eta_k^T \Psi \eta_k + \psi^T \eta_k. \tag{3.2}$$

Softening of state constraints often makes sense in terms of the problem formulation. The input constraints $Du_k \le d$ represent physical restrictions on the available control action (for instance, a limit on available power, flow rate, or voltage), so it is natural to make the constraints "hard" in the formulation (3.1b). By contrast, state constraints often represent desired values for profitability, safety, or convenience, so that violation of these constraints is a condition to be discouraged rather than forbidden in the

formulation. Since the penalty terms for the constraint violations η_k is quadratic, and since these terms are defined by linear inequalities, they do not contribute to the "nonlinear" nature of the problem (3.1), and play little part in the development of this section.

If the constraints $Du_k \leq d$, $k = 0, 1, \ldots, N-1$ are feasible, then the full set of constraints (3.1b) is feasible. From any input sequence u that satisfies $Du_k \leq d$, we recover the states x_k and violationsη_k by setting $x_{k+1} = F(x_k, u_k)$ and $\eta_k = \max(Gx_k - g, 0)$ for all k.

For the purposes of this paper, we consider the problem (3.1) in isolation, as a single nonlinear program that we wish to solve. To put this problem in context, however, we give a brief description of MPC methodology. In MPC, a sequence of problems of the form (3.1) are solved, one at each sampling point. A starting point for the input vector sequence $\{u_k\}$ in (3.1) can be constructed by shifting the input vectors obtained at the solution form the previous sampling point forward by one stage, and using an educated guess of the remaining value u_{N-1}, based on the solution of the discrete-time linear quadratic regulator solution. A full feasible point for (3.1) can then be obtained in the manner described above.

At the very first sampling time, no previous input trajectory is available for defining the starting point. A poor choice of this initial point may cause the optimizer to find undesirable local minima. Our approach is to construct u_k and x_{k+1} sequentially for $k = 0, 1, \ldots, N-1$ as follows. We define a starting guess for u'_k by setting $u'_k = u_{k-1}$, and linearize the model F about (x_k, u'_k). We then calculate the locally optimal linear infinite horizon feedback law for the resulting linear model with quadratic objective. Application of this law yields an improved estimate u_k, which we accept as the initial guess. (In this development, we assume that the dynamics of the model do not change greatly between each sampling time.) Application of the model equation in (3.1b) now yields x_{k+1}. The first value u_0 can be attained by starting with an initial guess u'_0, linearizing the system about x_0 and the guess, and determining the optimal feedback gain K_0. This value of K_0 generates a new value for $u_0 = K_0 x_0$. The linearization and recalculation of K_0 is repeated until the guess for u_0 converges to within a tolerance. If the initial guess violates the hard constraints on the inputs, the result is adjusted to a feasible input; see (5.6).

4. The SQP Subproblem. In the problem (3.1), inputs and states at the current sampling time k directly affect future states $x_{k+1}, x_{k+2}, \ldots$, but have no effect on states and inputs at previous times. Due to this causal nature, the optimization problems are highly structured. For an appropriate "interleaved" ordering of the components of u, x, and η, the Hessian of the objective and the Jacobian of the constraints in (3.1) are block-banded. Rao, Wright, and Rawlings [21] exploit this structure for the case of linear model F and convex quadratic $\mathcal{C}$ and Φ by developing a customized primal-dual interior-point method for the resulting quadratic programming problem. With some alterations, this method can be used to solve the subproblems arising in the SQP algorithm of Section 2, applied to (3.1). We devise other features of the SQP algorithm to ensure that this structure can be exploited at the level of the quadratic programming subproblem; for example, by choosing the approximate Hessians H in (2.2a) to have the same structure as the exact Hessians.

For the remainder of the MPC discussion, it is convenient to introduce the fol-

lowing definitions, based on the formulation in (3.1):

$$Q_k = \frac{\partial^2 \Phi}{\partial x_k^2} = \frac{\partial^2 \mathcal{C}(x_k, u_k)}{\partial x_k^2}, \tag{4.1a}$$

$$R_k = \frac{\partial^2 \Phi}{\partial u_k^2} = \frac{\partial^2 \mathcal{C}(x_k, u_k)}{\partial u_k^2}, \tag{4.1b}$$

$$M_k = \frac{\partial^2 \Phi}{\partial u_k \partial x_k} = \frac{\partial^2 \mathcal{C}(x_k, u_k)}{\partial u_k \partial u_k}, \tag{4.1c}$$

$$A_k = \frac{\partial F(x_k, u_k)}{\partial x_k}, \tag{4.1d}$$

$$B_k = \frac{\partial F(x_k, u_k)}{\partial u_k} \tag{4.1e}$$

In the SQP approach applied to (3.1), the subproblem has a structure similar to (3.1), except that the model equation is linearized, and the objective is replaced by a quadratic whose second-order terms are approximations to the Hessian of the Lagrangian function for (3.1). To be precise, the subproblem is as follows:

$$\min_{\Delta x, \Delta u, \Delta \eta} \frac{1}{2} \Delta u_0^T \tilde{R}_0 \Delta u_0 + r_0^T \Delta u_0 + \tag{4.2}$$
$$\sum_{k=1}^{N-1} \left\{ \frac{1}{2} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} \begin{bmatrix} \tilde{Q}_k & \tilde{M}_k \\ \tilde{M}_k^T & \tilde{R}_k \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} + \begin{bmatrix} q_k \\ r_k \end{bmatrix}^T \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} \right\}$$
$$+ \frac{1}{2} \Delta x_N^T \tilde{Q}_N \Delta x_N + q_N^T \Delta x_N + \sum_{k=1}^{N} \Xi(\eta_k + \Delta \eta_k)$$

subject to

$$\Delta x_0 = 0, \tag{4.3a}$$

$$\Delta x_{k+1} = A_k \Delta x_k + B_k \Delta u_k, \quad k = 0, 1, \ldots, N-1, \tag{4.3b}$$

$$D(u_k + \Delta u_k) \le d, \quad k = 0, 1, \ldots, N-1, \tag{4.3c}$$

$$G(x_k + \Delta x_k) - (\eta_k + \Delta \eta_k) \le g, \quad k = 1, 2, \ldots, N, \tag{4.3d}$$

$$\eta_k + \Delta \eta_k \ge 0, \quad k = 1, 2, \ldots, N, \tag{4.3e}$$

$$\|\Sigma_k \Delta u_k\|_\infty \le \Delta, \quad k = 0, 1, \ldots, N-1. \tag{4.3f}$$

Note that feasibility of the current iterate x, u, η (in particular, $x_{k+1} = F(x_k, u_k)$) is exploited in defining the linearization of the model equation (4.3b). The blocks that make up the Lagrangian Hessian approximation are denoted by $\tilde{Q}_k$, $\tilde{R}_k$, and $\tilde{M}_k$ in (4.2). We discuss various choices for these approximations in Section 6. We have assumed that the constraint violation penalties $\Xi(\cdot)$ are quadratic, as in (3.2). Note that the trust-region constraint (4.3f) is applied only to the $\{\Delta u\}$ components. We discuss the choice of scaling matrices Σ_k in Section 7.

5. Feasibility Perturbation. In this section, we describe the perturbation technique that is used to recover a feasible step $\widetilde{\Delta x}, \widetilde{\Delta u}, \widetilde{\Delta \eta}$ from the trust-region SQP step $\Delta x, \Delta u, \Delta \eta$. The perturbed step should satisfy the asymptotic exactness condition (2.4).

The easiest way to perturb the step is to set $\widetilde{\Delta u} = \Delta u$, and then recover the new states from the model equation (3.1b); that is, $\widetilde{\Delta x}$ satisfies

$$x_{k+1} + \widetilde{\Delta x}_{k+1} = F(x_k + \widetilde{\Delta x}_k, u_k + \Delta u_k), \quad k = 0, 1, \ldots, N-1, \tag{5.1}$$

(with $\widetilde{\Delta x}_0 = \Delta x_0 = 0$). The perturbed constraint violations $\{\widetilde{\Delta\eta}\}$ are recovered from (4.3d) and (4.3e) by setting

$$\eta_k + \widetilde{\Delta\eta}_k = \max\left(G(x_k + \widetilde{\Delta x}_k) - g, 0\right), \quad k = 1, 2, \ldots, N. \tag{5.2}$$

This scheme is obvious, and often works well, but it has potential problems when the system is open-loop unstable at the set-point; that is, $\partial F/\partial x(0,0)$ has spectral radius greater than 1. The system is ill-conditioned in the sense that the forward integration used to recover $\widetilde{\Delta x}$ in (5.1) contains "increasing modes", which amplify perturbations and cause large differences between Δx_k and $\widetilde{\Delta x}_k$ as k grows. In this case, it is unlikely that $x + \widetilde{\Delta x}$ will be close to a true projection of $x + \Delta x$ onto the feasible set for (3.1b).

A more stable perturbation technique is obtained by using a change of variable, motivated by linear MPC theory, to stabilize the forward integration. We define a set of stagewise variables v as follows:

$$v_k = (u_k + \Delta u_k) - K_k(x_k + \Delta x_k), k = 0, 1, \ldots, N-1, \tag{5.3}$$

where K_k is a feedback gain matrix constructed such that $|\mathrm{eig}(A_k + B_k K_k)| < 1$. Several choices of K_k are possible. One method is to use *pole placement*, choosing the poles of the closed loop system arbitrarily inside the unit circle, and then choosing K_k such that the eigenvalues of $A_k + B_k K_k$ coincide with these poles. A second method is to obtain K_k by solving the optimal infinite horizon linear quadratic regulator problem at each (A_k, B_k) pair. This K_k is then guaranteed to stabilize the linear system defined by (A_k, B_k). It is often possible to reuse the same K_k over a number of different stages k, provided the eigenvalues of $(A_k + B_k K_k)$ stay inside the unit circle, and thereby reduce the computational load associated with calculating each K_k online. In many cases, a single stabilizing "time-invariant" K_k can be used at all stages, usually by solving the optimal infinite-horizon linear-quadratic regulator problem on the linearized system at the target.

We use a third approach to select the feedback gain matrices K_k, based on solution of a finite-horizon discrete-time linear-quadratic regulator for time-varying systems. This approach is guaranteed to yield K_k for which $|\mathrm{eig}(A_k + B_k K_k)| < 1$ for many, but not all systems. The procedure for defining K_k is developed from dynamic programming arguments, starting from the end of the prediction horizon and working towards the beginning. The result is the discrete-time Riccati equation [3]. First, define $\Pi_N = Q_N$, and then apply the following recursions for $k = N-1, N-2, \ldots, 1$:

$$\begin{aligned} K_k &= -\left(R_k + B_k^T \Pi_{k+1} B_k\right)^{-1} \left(M_k^T + B_k^T \Pi_{k+1} A_k\right) \\ \Pi_k &= Q_k + K_k^T R_k K_k + M_k K_k + K_k^T M_k^T + \left(A_k + B_k K_k\right)^T \Pi_{k+1} \left(A_k + B_k K_k\right). \end{aligned}$$

If one of the K_k matrices generated by this technique does not stabilize (A_k, B_k), then K_k may be selected instead from one of the other techniques mentioned above. However, experience has shown that it is often better to use the K_k from the above approach because it most closely resembles the solution of the quadratic subproblem. The feedback gains K_k obtained from the discrete Riccati scheme are consistent with the optimal solution for the unconstrained time-varying linear system along the trajectory of the current guess. This scheme for calculating the K_k is not as computationally intensive as the first two approaches described above, and it represents the system at the current iterate well.

Given the stabilizing feedback gain matrices K_k and the SQP step, we can define v from the formulae (5.3). We may view v as the deviation from a stable closed-loop trajectory. We then determine the perturbed components $\widetilde{\Delta x}$ and $\widetilde{\Delta u}$ as follows:

$$\widetilde{\Delta u}_k = K_k(x_k + \widetilde{\Delta x}_k) - u_k + v_k, \quad k = 0, 1, \ldots, N-1, \tag{5.4a}$$

$$\widetilde{\Delta x}_{k+1} = F(x_k + \widetilde{\Delta x}_k, u_k + \widetilde{\Delta u}_k) - x_k, \quad k = 0, 1, \ldots, N-1. \tag{5.4b}$$

By combining (5.3) with (5.4a), we see immediately that

$$\widetilde{\Delta u}_k - \Delta u_k = K_k(\widetilde{\Delta x}_k - \Delta x_k). \tag{5.5}$$

Note $\widetilde{\Delta x}_0 = \Delta x_0 = 0$ in (5.4a) and (5.5), so from the latter equation we have in particular that $\widetilde{\Delta u}_0 = \Delta u_0$.

When there is a constraint $Du_k \leq d$ on the inputs, we modify the procedure above by solving the following subproblem after the calculation of each $\widetilde{\Delta u}_k$ from (5.4a):

$$\min_{\widehat{\Delta u}_k} (\widehat{\Delta u}_k - \widetilde{\Delta u}_k)^T R_k (\widehat{\Delta u}_k - \widetilde{\Delta u}_k) \text{ subject to } D(u_k + \widehat{\Delta u}_k) \leq d. \tag{5.6}$$

We then make the replacement $\widetilde{\Delta u}_k \leftarrow \widehat{\Delta u}_k$, and proceed with (5.4b).

The definition (5.2) can again be used to recover the perturbed step in the η components.

In Appendix A, we show that this feasibility projection approach satisfies the asymptotic exactness condition (2.4) under reasonable assumptions, and also suggest why the stabilization scheme improves the results.

In the case of open-loop unstable models, the SQP subproblem itself may be ill-conditioned. The change of variables in (5.3) may also be used in the QP to improve the conditioning. In fact, the re-conditioning can be performed by using the same values for K_k as we derived above for the perturbation.

6. Approximate Hessians. We now describe various ways to choose the matrices $\tilde{Q}_k$, $\tilde{R}_k$, and $\tilde{M}_k$ that appear in the objective of the SQP subproblem (4.2). Since the variables η enter the constraints of (3.1) linearly and the objective quadratically, the terms in the Hessian of the quadratic involving these variables are constant, so it is not necessary to seek approximations of these terms. Hence, for clarity, we omit these variables from the formulation considered in this section and the next, although our implementations described in Section 8 solve the full problem (3.1).

By omitting the soft state constraints from (3.1), we obtain

$$\min_{x,u} \Phi(x,u) \stackrel{\text{def}}{=} \sum_{k=0}^{N-1} \mathcal{C}(x_k, u_k) + \Gamma(x_N) \quad \text{s.t.} \tag{6.1a}$$

$$x_0 \text{ given}, \quad x_{k+1} = F(x_k, u_k), \quad Du_k \leq d, \quad k = 0, 1, \ldots, N-1, \tag{6.1b}$$

while the SQP subproblem has the following form:

$$\min_{\Delta x, \Delta u} \frac{1}{2}\Delta u_0^T \tilde{R}_0 \Delta u_0 + r_0^T \Delta u_0 + \frac{1}{2}\Delta x_N^T \tilde{Q}_N \Delta x_N + q_N^T \Delta x_N + \tag{6.2}$$

$$\sum_{k=1}^{N-1} \left\{ \frac{1}{2} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} \begin{bmatrix} \tilde{Q}_k & \tilde{M}_k \\ \tilde{M}_k^T & \tilde{R}_k \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} + \begin{bmatrix} q_k \\ r_k \end{bmatrix}^T \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} \right\}$$

subject to

$$\Delta x_0 = 0, \tag{6.3a}$$
$$\Delta x_{k+1} = A_k \Delta x_k + B_k \Delta u_k, \quad k = 0, 1, \ldots, N-1, \tag{6.3b}$$
$$D(u_k + \Delta u_k) \le d, \quad k = 0, 1, \ldots, N-1, \tag{6.3c}$$
$$\|\Sigma_k \Delta u_k\|_\infty \le \Delta, \quad k = 0, 1, \ldots, N-1. \tag{6.3d}$$

The Lagragian for the problem (6.1) is as follows:

$$\begin{aligned}\mathcal{L}(x, u, \lambda, \mu) &= \Phi(x, u) + \sum_{k=0}^{N-1} \lambda_k^T (F(x_k, u_k) - x_{k+1}) + \mu_k^T (Du_k - d) \\ &= \sum_{k=0}^{N-1} \left[\mathcal{C}(x_k, u_k) + \lambda_k^T (F(x_k, u_k) - x_{k+1}) + \mu_k^T (Du_k - d)\right] + \Gamma(x_N).\end{aligned} \tag{6.4}$$

We can decompose the Lagrangian in a stagewise fashion, as follows:

$$\mathcal{L}(x, u, \lambda, \mu) = \mathcal{L}_0(u_0, \lambda_0, \mu_0) + \sum_{k=1}^{N-1} \mathcal{L}_k(x_k, u_k, \lambda_{k-1}, \lambda_k, \mu_k) + \mathcal{L}_N(x_N, \lambda_{N-1}), \tag{6.5}$$

where

$$\begin{aligned}\mathcal{L}_0(u_0, \lambda_0, \mu_0) &= \mathcal{C}(x_0, u_0) + \lambda_0^T F(x_0, u_0) + \mu_0^T (Du_0 - d), \\ \mathcal{L}_k(x_k, u_k, \lambda_{k-1}, \lambda_k, \mu_k) &= \mathcal{C}(x_k, u_k) + \lambda_k^T F(x_k, u_k) - \lambda_{k-1}^T x_k + \mu_k^T (Du_k - d), \\ \mathcal{L}_N(x_N, \lambda_{N-1}) &= \Gamma(x_N) - \lambda_{N-1}^T x_N.\end{aligned}$$

Note that each x_k and u_k appear only in $\mathcal{L}_k$, and that each $\mathcal{L}_k$ depend on just a few of the Lagrange multiplier components.

Using the definitions above, we now discuss various options for choosing the Hessian terms in the SQP subproblem (6.2), (6.3).

6.1. Exact Hessian. The first option is to use the exact Hessians of the Lagrangian; that is,

$$\hat{Q}_k = \frac{\partial^2 \mathcal{L}_k}{\partial x_k^2}, \quad \hat{R}_k = \frac{\partial^2 \mathcal{L}_k}{\partial u_k^2}, \quad \hat{M}_k = \frac{\partial^2 \mathcal{L}_k}{\partial u_k \partial x_k}. \tag{6.6}$$

The full Hessian of $\mathcal{L}$ with respect to the variables x and u, using a stagewise interleaving of these variables, has the following block-banded structure:

$$\nabla^2_{x,u}\mathcal{L} = \begin{bmatrix} \hat{R}_0 & & & & & & & & \\ & \hat{Q}_1 & \hat{M}_1 & & & & & & \\ & \hat{M}_1^T & \hat{R}_1 & & & & & & \\ & & & \hat{Q}_2 & \hat{M}_2 & & & & \\ & & & \hat{M}_2^T & \hat{R}_2 & & & & \\ & & & & & \ddots & & & \\ & & & & & & \hat{Q}_{N-1} & \hat{M}_{N-1} & \\ & & & & & & \hat{M}_{N-1}^T & \hat{R}_{N-1} & \\ & & & & & & & & \hat{Q}_N \end{bmatrix} \tag{6.7}$$

Note in particular that the Hessian of the Lagrangian is structured identically to the Hessian of the objective in (6.1), and to the Hessian of the linear MPC problem, because there are no coupled interactions between future and past states and inputs other than through the model equality constraint.

We can also use a finite-difference approximation to the terms in (6.6). Note that each $\hat{Q}_k$ consists of Q_k from (4.1a) added to a contribution involving λ_k and the second partial derivatives of the model function F. If the model is linear, we have $\hat{Q}_k = Q_k$. Similar comments apply for $\hat{R}_k$ and $\hat{M}_k$.

These choices for the Hessians of the SQP subproblem have the advantage of rapid local convergence, under appropriate assumptions. The disadvantage is that second derivatives of the model F may be difficult to compute by hand, and time-consuming to approximate with a difference approximation. Moreover, the block-diagonal matrices

$$\mathcal{H}_k \stackrel{\text{def}}{=} \begin{bmatrix} \tilde{Q}_k & \tilde{M}_k \\ \tilde{M}_k^T & \tilde{R}_k \end{bmatrix} = \begin{bmatrix} \hat{Q}_k & \hat{M}_k \\ \hat{M}_k^T & \hat{R}_k \end{bmatrix} \tag{6.8}$$

may not be positive definite, in which case the SQP subproblem will not be a convex quadratic program. Since the solver described in [21] requires a convex quadratic objective, these matrices may need to be modified in order for the solver to work.

6.2. Hessians of the Objective. An approximation that is often effective is to simply ignore the contributions from F and use

$$\tilde{Q}_k = Q_k, \quad \tilde{R}_k = R_k, \quad \tilde{M}_k = M_k. \tag{6.9}$$

This is known as the "Gauss-Newton" choice in some papers. Although this choice is not asymptotically equivalent to the ideal choice of Section 6.1 unless the model function F is linear, these matrices often have the right scale. Moreover, the matrix

$$\mathcal{H}_k \stackrel{\text{def}}{=} \begin{bmatrix} \tilde{Q}_k & \tilde{M}_k \\ \tilde{M}_k^T & \tilde{R}_k \end{bmatrix} = \begin{bmatrix} Q_k & M_k \\ M_k^T & R_k \end{bmatrix} \tag{6.10}$$

is constant and positive semidefinite when (as is usually the case) the cost function $\mathcal{C}$ in (6.1) is quadratic and convex.

6.3. Full Quasi-Newton Approximation. Quasi-Newton variants of SQP for nonlinear programming have been the subject of extensive research; see [19, Chapter 18]. Indeed, most existing implementations of SQP use quasi-Newton Hessian approximations, either to the full Langrangian Hessian, or to the projection of this Hessian onto the nullspace of the active constraints. Here, we consider only methods of the former kind. The Hessian approximations are updated after each iteration, using information about the step just taken and the difference in the Lagrangian first derivatives between the current iterate and the previous one. Specifically, we have the step vector s defined by

$$s = \begin{bmatrix} x^+ - x \\ u^+ - u \end{bmatrix}, \tag{6.11}$$

where x^+ and u^+ denote the new iterates and x and u the current iterates; and

$$\begin{aligned} y &= \begin{bmatrix} \nabla_x \mathcal{L}(x^+, u^+, \lambda, \mu) - \nabla_x \mathcal{L}(x, u, \lambda, \mu) \\ \nabla_u \mathcal{L}(x^+, u^+, \lambda, \mu) - \nabla_u \mathcal{L}(x, u, \lambda, \mu) \end{bmatrix} \\ &= \begin{bmatrix} \nabla_x \Phi(x^+, u^+) - \nabla_x \Phi(x, u) + \sum_{k=0}^{N-1} (A_k(x_k^+, u_k^+) - A_k(x_k, u_k))^T \lambda_k \\ \nabla_u \Phi(x^+, u^+) - \nabla_u \Phi(x, u) + \sum_{k=0}^{N-1} (B_k(x_k^+, u_k^+) - B_k(x_k, u_k))^T \lambda_k \end{bmatrix}, \end{aligned} \tag{6.12}$$

where we have used the notation (4.1) to define A_k and B_k.

The damped BFGS approach (see Powell [20] and Nocedal and Wright [19, p. 540]) maintains a positive definite approximation H to the full Lagrangian Hessian (6.7), and updates it after each step according to the following rule:

ALGORITHM 6.1 (damped BFGS).
if $s^T y < 0.2 s^T H s$
define $\theta = 0.8 s^T H s / (s^T H s - s^T y)$
set $y \leftarrow \theta y + (1-\theta) H s$
end if
update H *as follows:*

$$H \leftarrow H - \frac{H s s^T H}{s^T H s} + \frac{y y^T}{y^T s}. \tag{6.13}$$

A sensible starting initial guess for H is the matrix (6.7) with the approximations (6.9), possibly with the addition of a multiple of the identity matrix to ensure strict positive definiteness.

A naive application of this procedure to our problem is not practical, since after just one update H becomes in general a fully dense matrix. Since it does not preserve the block structure of (6.7), we cannot use the efficient quadratic programming technique to solve the SQP subproblem.

6.4. Sparsified Quasi-Newton Approximation. An ad-hoc alternative to the approach just discussed is to impose the desired sparsity pattern (that is, the pattern in (6.7)) on the Hessian approximation H after each step. In other words, we carry out the procedure above to update H, and then zero out all parts of H that are outside the block-diagonal band in (6.7). It can be shown that this approach maintains positive definiteness of the approximations (given a positive definite initial approximation to H). However, approaches that enforce specific sparsity patterns in this fashion do not enjoy a good reputation, even for unconstrained problems.

6.5. Partitioned Quasi-Newton Approaches. We now consider approaches that maintain separate approximations to the Hessians of each of the component Lagrangians $\mathcal{L}_k$ in (6.5). Griewank and Toint [13] proposed methods of this type for partially separable nonlinear systems, and these methods were first applied to nonlinear control problems by Bock and Plitt [6]. In our problem, the Lagrangian in (6.5) is in fact *completely* separable in the state and input variables. Each pair (x_k, u_k) appears only in the term $\mathcal{L}_k$, and the coupling between stages comes only in the model equation.

We apply this approach by defining "stagewise" versions of the s and y vectors in (6.11) and (6.12). For $k = 1, 2, \ldots, N-1$, we have

$$s_k = \begin{bmatrix} x_k^+ - x_k \\ u_k^+ - u_k \end{bmatrix}, \tag{6.14}$$

and

$$\begin{aligned} y_k &= \begin{bmatrix} \frac{\partial}{\partial x_k}\mathcal{L}_k(x_k^+, u_k^+, \lambda_{k-1}, \lambda_k, \mu_k) - \frac{\partial}{\partial x_k}\mathcal{L}_k(x_k, u_k, \lambda_{k-1}, \lambda_k, \mu_k) \\ \frac{\partial}{\partial u_k}\mathcal{L}_k(x_k^+, u_k^+, \lambda_{k-1}, \lambda_k, \mu_k) - \frac{\partial}{\partial u_k}\mathcal{L}_k(x_k, u_k, \lambda_{k-1}, \lambda_k, \mu_k) \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial}{\partial x_k}\mathcal{C}(x_k^+, u_k^+) - \frac{\partial}{\partial x_k}\mathcal{C}(x_k, u_k) + (A_k(x_k^+, u_k^+) - A_k(x_k, u_k))^T \lambda_k \\ \frac{\partial}{\partial u_k}\mathcal{C}(x_k^+, u_k^+) - \frac{\partial}{\partial u_k}\mathcal{C}(x_k, u_k) + (B_k(x_k^+, u_k^+) - B_k(x_k, u_k))^T \lambda_k \end{bmatrix}, \end{aligned} \tag{6.15}$$

where we again used the notation (4.1) for A_k and B_k. For the initial and final stages, we have

$$\begin{array}{rcl} s_0 & = & u_0^+ - u_0, \\ y_0 & = & \frac{\partial}{\partial u_0}\mathcal{C}(x_0^+, u_0^+) - \frac{\partial}{\partial u_0}\mathcal{C}(x_0, u_0) + (A_0(x_0^+, u_0^+) - A_0(x_0, u_0))^T\lambda_0 \\ s_N & = & x_N^+ - x_N \\ y_N & = & \frac{\partial}{\partial x_N}\Gamma(x_N^+) - \frac{\partial}{\partial x_N}\Gamma(x_N). \end{array} \tag{6.16}$$

We now use s_k and y_k to maintain an approximation H_k to the kth diagonal block in (6.7), which for all but the initial and final stages has the form

$$H_k = \begin{bmatrix} \tilde{Q}_k & \tilde{M}_k \\ \tilde{M}_k^T & \tilde{R}_k \end{bmatrix} \tag{6.17}$$

A damped-BFGS variant of the partitioned approach uses the update strategy of Algorithm 6.1, applied to each stage k separately. To be precise, we apply Algorithm 6.1 to each stage $k = 0, 1, \ldots, N$, with s_k, y_k, and H_k replacing s, y, and H, respectively. We then obtain decompose the approximate Hessians H_k according to (6.17), to obtain the matrices $\tilde{Q}_k$, $\tilde{R}_k$, and $\tilde{M}_k$ to be used in (6.2).

As well as being efficient, this approach maintains positive semidefiniteness of the diagonal blocks, so the SQP subproblem can be passed to the convex quadratic programming solver without complications.

An alternative approach is to use the symmetric rank-1 (SR1) update (see [19, Section 8.2]) in place of the damped BFGS approach. The update formula for the stage-k block is as follows:

$$H_k \leftarrow H_k + \frac{(y_k - H_k s_k)(y_k - H_k s_k)^T}{(y_k - H_k s_k)^T s_k}, \tag{6.18}$$

where the update is skipped if the denominator in (6.18) is too small; that is, if the following criterion is satisfied:

$$\left|(y_k - H_k s_k)^T s_k\right| < 10^{-6}\|y_k - H_k s_k\|\,\|s_k\|. \tag{6.19}$$

The Hessian approximations $\tilde{Q}_k$, $\tilde{R}_k$, and $\tilde{M}_k$ are then recovered from the decomposition (6.17).

The symmetric rank-one (SR1) update is in a sense more natural than BFGS, as it does not maintain positive definiteness (and there is no reason to expect the diagonal blocks of (6.7) to be positive definite). However, the indefinite approximations may cause problems for the quadratic programming solver. We handle this by passing to the quadratic programming solver a version of H_k in which the negative eigenvalues have been replaced by zero. That is, we form the eigenvalue decomposition

$$H_k = V_k \Lambda_k V_k^T,$$

where Λ_k is a diagonal matrix containing the eigenvalues, and V_k^T is orthogonal. We then redefine H_k to be

$$H_k \leftarrow V_k \Lambda_k^+ V_k^T,$$

where Λ_k^+ is obtained from Λ_k by replacing the negative diagonals by zero. Since the matrices H_k are small in our applications, the cost of performing these eigenvalue decompositions is relatively trivial.

7. Trust Region Scaling. In our problem (3.1), the states x and the constraint violations η are fully determined by the inputs u and the constraints (3.1b). Therefore, rather than apply a trust region constraint to all the variables in the SQP method, we define the trust region only in terms of the inputs. In this section, we discuss the issue of *scaling* the trust region constraint; that is, choosing the matrices Σ_k in (4.3f). It is well known that scaling can have a significant impact on the practical performance of trust-region algorithms.

In trust-region algorithms for unconstrained problems, the subproblems usually have the following form:

$$\min_{\Delta z} \nabla f(z)^T \Delta z + \frac{1}{2}\Delta z^T H \Delta z, \quad \text{subject to } \|D\Delta z\| \le \Delta. \tag{7.1}$$

Often, D is chosen to be a diagonal matrix whose diagonal elements are related to the diagonals of H. When the trust-region constraint is a Euclidean norm, the optimality conditions for (7.1) yield

$$(H + \xi D^T D)\Delta z = -\nabla f(z), \tag{7.2}$$

for some Lagrange multiplier $\xi \ge 0$. Hence, a common choice is to make each diagonal of D the square root of the corresponding diagonal of H. Similar choices of D are appropriate for other norms, as we discuss at the end of this section.

Motivated by this choice, we use the constraints in the our SQP subproblem to eliminate all but the input variables, and obtain a subproblem of the form (7.1) in which Δz is made up of $\Delta u_0, \Delta u_1, \ldots, \Delta u_{N-1}$. We then base the scaling matrix on the diagonal blocks of the Hessian in this subproblem.

For simplicity, we work with a special case of (3.1) in which neither the input constraints $Du_k \le d$ nor the soft state constraints are present. (Our derivation is exactly the same as in the general case, but less cluttered.) To be specific, our MPC problem is as follows:

$$\min_{x,u,\eta} \Phi(x,u,\eta) \stackrel{\text{def}}{=} \sum_{k=0}^{N-1} [\mathcal{C}(x_k,u_k) + \Xi(\eta_k)] + \Gamma(x_N) + \Xi(\eta_N) \quad \text{s.t.} \tag{7.3a}$$

$$x_0 \text{ given}, \quad x_{k+1} = F(x_k,u_k), \quad k = 0,1,\ldots,N-1. \tag{7.3b}$$

With a Euclidean-norm trust region, the corresponding SQP subproblem is

$$\min_{\Delta x,\Delta u} \frac{1}{2}\Delta u_0^T \tilde{R}_0 \Delta u_0 + r_0^T \Delta u_0 + \frac{1}{2}\Delta x_N^T \tilde{Q}_N \Delta x_N + q_N^T \Delta x_N + \tag{7.4}$$
$$\sum_{k=1}^{N-1} \left\{ \frac{1}{2} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} \begin{bmatrix} \tilde{Q}_k & \tilde{M}_k \\ \tilde{M}_k^T & \tilde{R}_k \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} + \begin{bmatrix} q_k \\ r_k \end{bmatrix}^T \begin{bmatrix} \Delta x_k \\ \Delta u_k \end{bmatrix} \right\}$$

subject to

$$\Delta x_0 = 0, \tag{7.5a}$$

$$\Delta x_{k+1} = A_k \Delta x_k + B_k \Delta u_k, \quad k = 0,1,\ldots,N-1, \tag{7.5b}$$

$$\|\Sigma_k \Delta u_k\|_2 \le \Delta, \quad k = 0,1,\ldots,N-1. \tag{7.5c}$$

We aggregate the variables in this subproblem as follows:

$$\Delta x = (\Delta x_1, \Delta x_2, \ldots, \Delta x_N), \quad \Delta u = (\Delta u_0, \Delta u_1, \ldots, \Delta u_{N-1}),$$

and also aggregate the data matrices as follows:

$$A = \begin{bmatrix} -I & & & & \\ A_1 & -I & & & \\ & A_2 & -I & & \\ & & \ddots & \ddots & \\ & & & A_{N-1} & -I \end{bmatrix}, \quad B = \begin{bmatrix} B_0 & & & \\ & B_1 & & \\ & & \ddots & \\ & & & B_{N-1} \end{bmatrix}, \tag{7.6a}$$

$$M = \begin{bmatrix} 0 & & & & \\ \tilde{M}_1 & 0 & & & \\ 0 & \tilde{M}_2 & 0 & & \\ \vdots & & & \ddots & \\ 0 & 0 & \dots & \tilde{M}_{N-1} & 0 \end{bmatrix}, \quad \tilde{Q} = \begin{bmatrix} \tilde{Q}_1 & & & \\ & \tilde{Q}_2 & & \\ & & \ddots & \\ & & & \tilde{Q}_N \end{bmatrix}, \tag{7.6b}$$

$$\Sigma = \begin{bmatrix} \Sigma_0 & & & \\ & \Sigma_1 & & \\ & & \ddots & \\ & & & \Sigma_{N-1} \end{bmatrix}, \quad \tilde{R} = \begin{bmatrix} \tilde{R}_0 & & & \\ & \tilde{R}_1 & & \\ & & \ddots & \\ & & & \tilde{R}_{N-1} \end{bmatrix}, \tag{7.6c}$$

$$q = (q_1, q_2, \dots, q_N), \quad r = (r_0, r_1, \dots, r_{N-1}). \tag{7.6d}$$

Using this notation, the objective (7.4) can be written in a more compact form:

$$\frac{1}{2} \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix}^T \begin{bmatrix} \tilde{Q} & \tilde{M}^T \\ \tilde{M} & \tilde{R} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta u \end{bmatrix}^T \begin{bmatrix} q \\ r \end{bmatrix}, \tag{7.7}$$

as can the constraints (7.5b):

$$A\Delta x + B\Delta u = 0. \tag{7.8}$$

Since A is square and nonsingular, we can use (7.8) to eliminate Δx, and write the objective (7.7) in terms of Δu alone, as follows:

$$\frac{1}{2}\Delta u^T \hat{Q} \Delta u + \hat{r}^T \Delta u, \tag{7.9}$$

where

$$\begin{aligned} \hat{Q} &= \tilde{R} + B^T A^{-T} \tilde{Q} A^{-1} B - (\tilde{M} A^{-1} B + B^T A^{-T} \tilde{M}^T) \\ \hat{r} &= r - B^T A^{-T} q. \end{aligned}$$

Following the motivation above, we choose Σ to be a block-diagonal matrix such that the diagonal blocks of $\Sigma^T \Sigma = \Sigma^2$ are identical to those of $\hat{Q}$. Since from (7.6), A is block lower triangular and B is block diagonal, A^{-1} and $A^{-1}B$ are both block lower triangular. From the structure of $\tilde{M}$ in (7.6b), we have that the diagonal blocks of $(\tilde{M} A^{-1} B + B^T A^{-T} \tilde{M}^T)$ are zero. Hence, the diagonal blocks of $\hat{Q}$ are simply those of $\tilde{R} + B^T A^{-T} \tilde{Q} A^{-1} B$, which are as follows:

$$\begin{aligned} \hat{Q}_{11} &= \tilde{R}_0 + B_0^T \tilde{Q}_1 B_0 + B_0^T A_1^T \tilde{Q}_2 A_1 B_0 + \cdots + B_0^T A_1^T \dots A_{N-1}^T \tilde{Q}_N A_{N-1} \dots A_1 B_0, \\ \hat{Q}_{22} &= \tilde{R}_1 + B_1^T \tilde{Q}_2 B_1 + B_1^T A_2^T \tilde{Q}_3 A_2 B_1 + \cdots + B_1^T A_2^T \dots A_{N-1}^T \tilde{Q}_N A_{N-1} \dots A_2 B_1, \\ &\vdots \\ \hat{Q}_{NN} &= \tilde{R}_{N-1} + B_{N-1}^T \tilde{Q}_N B_{N-1}. \end{aligned}$$

Therefore, we can construct the scaling matrices for the Euclidean-norm trust region by the following recursive relationship: Define $\mathcal{G}_N = \tilde{Q}_N$, and then apply the following formula for $k = N, N-1, \ldots, 2$:

$$\mathcal{G}_{k-1} = \tilde{Q}_{k-1} + A_{k-1}^T \mathcal{G}_k A_{k-1}.$$

Then we have

$$\hat{Q}_{kk} = \tilde{R}_{k-1} + B_{k-1}^T \mathcal{G}_k B_{k-1}, \quad k = 1, 2, \ldots, N,$$

so our choice for the scaling matrices is

$$\Sigma_{k-1} = \hat{Q}_{kk}^{1/2}, \quad k = 1, 2, \ldots, N. \tag{7.10}$$

In practice, the use of a 2-norm trust region introduces a nonlinear constraint into the subproblem, making it no longer a quadratic program. Rather than use the same scaling matrices Σ_k for the ∞-norm as for the 2-norm, we construct tangent planes to the ellipsoidal 2-norm constraint using an eigenvalue decomposition. For each k, we calculate the orthogonal V_k and positive diagonal Λ_k such that

$$\Sigma_k^T \Sigma_k = \Sigma_k^2 = V_k \Lambda_k^2 V_k^T.$$

We then define the ∞-norm constraint as follows:

$$\|\Lambda_k V_k^T \Delta u\|_\infty \leq \Delta, \quad k = 0, 1, \ldots, N-1.$$

This constraint defines the smallest multi-dimensional box that circumscribes the Euclidean-norm trust region.

8. Computational Results. In this section, we describe computational experience with Algorithm FP-SQP applied to nonlinear MPC. We describe in detail two examples that are typical of problems that arise in industrial practice, showing how the various mechanisms described in Sections 5, 6, and 7 contribute to the effectiveness of the approach. We then summarize computational experience with these two examples and three others.

Our first example involves a continuously stirred tank reactor (CSTR).

EXAMPLE 8.1. *Consider a CSTR in which the exothermic reaction $A \longrightarrow B$ is taking place. The temperature of the reactor is reduced by adjusting the temperature of the coolant fluid in a heat exchange coil inside the vessel. The goal is to determine the optimal trajectory of future coolant temperatures such that the system reaches a desired steady state.*

The equations governing this system are as follows:

$$\dot{C_A} = \frac{q}{V}(C_{Af} - C_A) - k_0 \exp\left(-\frac{E}{RT}\right) C_A, \tag{8.1a}$$

$$\dot{T} = \frac{q}{V}(T_f - T) + \frac{(-\Delta H)}{\rho C_p} k_0 \exp\left(-\frac{E}{RT}\right) C_A + \frac{UA}{V \rho C_p}(T_c - T), \tag{8.1b}$$

where C_A and T describe the state of the system, T_c is the input, and the remaining quantities are parameters whose values, as presented in [15], are given in Table 8.1. The variable C_A is the concentration of species A in mol/L, T is the temperature of the reactor in K, and T_c is the temperature of the coolant.

q	100 L/min	E/R	8750 K
C_{Af}	1 mol/L	k_0	7.2×10^{10} min^{-1}
UA	5×10^4 J/min $\cdot$ K	V	100 L
ρ	1000 g/L	T_f	350 K
C_p	0.239 J/g $\cdot$ K	ΔH	-5×10^4 J/mol

TABLE 8.1
Parameters for the CSTR Model of Example 8.1.

Given target values $C_{A,\text{target}}$ *and* T_{target} *for the states and* $T_{c,\text{target}}$ *for the input, we define the states and input for this system as follows:*

$$x = \begin{bmatrix} C_A - C_{A,\text{target}} \\ T - T_{\text{target}} \end{bmatrix}, \quad u = T_c - T_{c,\text{target}}.$$

We use the target values $C_{A,\text{target}} = 0.5$ *M,* $T_{\text{target}} = 350$ *K, and* $T_{c,\text{target}} = 300$ *K. The initial state of the system is* $x_0 = (1.0, 350)^T$.

We obtain a discrete-time problem from this model by defining a sampling interval of $\Delta t = 0.05$ minutes, and performing numerical integration of the equations (8.1) between sampling times using the LSODE code [16]. We define the components of the cost function as follows:

$$\mathcal{C}(x_k, u_k) = x_k^T Q x_k + u_k^T R u_k, \quad \Gamma(x_N) = x_N^T P x_N \tag{8.2}$$

where $Q = \begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix}$ and $R = 2$. The matrix P is the Lyapunov penalty associated with the discrete linear quadratic regulator problem with penalties Q and R on the linearized system at the set point. It has the following value:

$$P = \begin{bmatrix} 99164.7 & 2104.17 \\ 2104.17 & 73.2818 \end{bmatrix}.$$

The prediction horizon N is 60 time steps, or 3 minutes. The input u is constrained so that the coolant temperature may not be colder than 230 K; that is, $T_c \geq 230$.

For this example, the choice of target operating point is open-loop unstable. For temperatures just above the set point, the plant ignites, converting most of the reactant to product, and releasing more heat than the coolant can remove. Although we do not explicitly impose a "soft" state constraint to discourage this situation, it is certainly undesirable to steer the system through an ignition.

We find from solving the formulation (3.1) that the problem has several local solutions, which can be identified by starting the algorithm from different starting points. Figure 8.1 shows the profiles for the input variable u at three selected local solutions, along with the final objective value corresponding to each profile. The trajectory with the lowest objective function is believed to be the global optimum, arrived at by using the initial guess procedure described in Section 3. The other two minima plotted here are much less desirable, as they result from ignition. The controller in both these cases saturates at the minimum coolant temperature (that is, T_c remains at its lower bound of 230 for several successive sampling times) but is incapable of preventing ignition. These solutions were obtained by adding random noise at the level of 10% to the initial guess used to obtain the global solution.

An interesting property about the input profiles in the figure is that they all asymptotically converge to the steady-state input coolant temperature of 300 K. Each

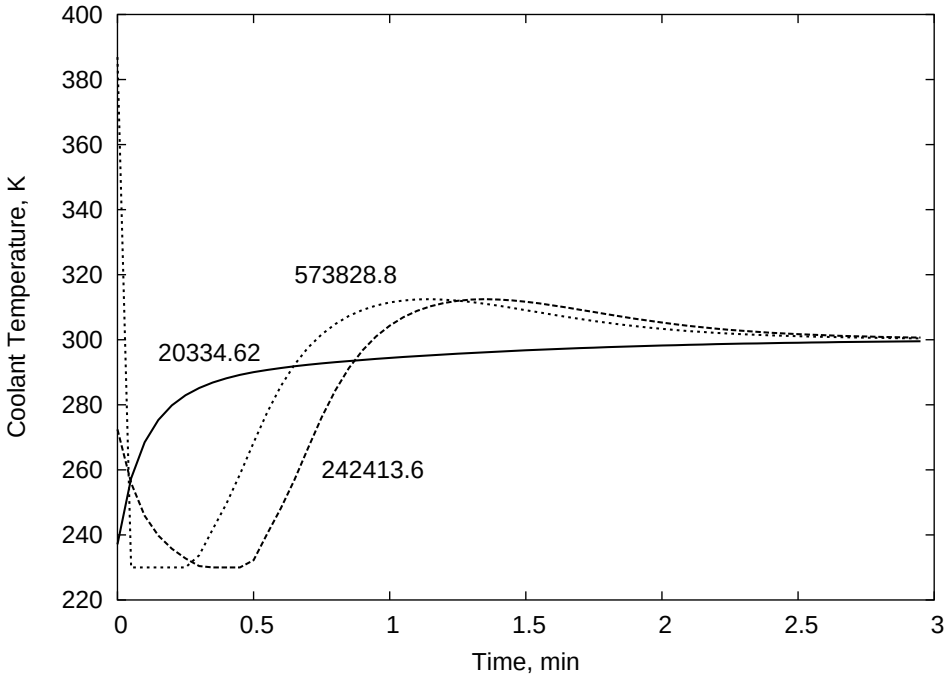

FIG. 8.1. *Input profiles and objective function values of local minima of CSTR example*

local solution still stabilizes the system, even though ignition occurs. Though ignition makes these solutions unacceptable in this case, the stabilization indicates that the controller is well designed. Local solutions may give acceptable control performance in other cases.

The existence of local solutions leads us to conclude that, as in other nonconvex optimization problems, the choice of starting point may be crucial to the quality of the computed solution. This observation has implications for the way we generate starting points for solving the MPC problem (3.1) at subsequent sampling times. The technique proposed in Section 3 of shifting the solution of (3.1) at the previous timepoint to obtain a starting point for the current timepoint may not be adequate. For instance, if $F(x, u)$ is not an adequate model of the true system, the state x_1 obtained by solving (3.1) at the previous timepoint may be distant from the estimate of x_0 obtained by applying a state estimation procedure at the current timepoint. Likewise, if the state and input targets change between sampling times, either due to a change in setpoint or as a result of rejecting non-zero mean disturbances, the shifting procedure may produce a poor initial guess. This could lead the algorithm to produce a potentially undesirable local optimum for (3.1). We can avoid these difficulties by repeating the initialization procedure in Section 3.

A key component in the formulation of Algorithm FP-SQP is the use of stabilization in the feasibility perturbation procedure, described in Section 5. Omission of stabilization can result in state trajectories $\widetilde{\Delta x}_k$ that diverge wildly from the SQP steps Δx_k. This phenomenon was noted for Example 8.1, as illustrated in Figure 8.2. This figure is a phase portrait, with the first component C_A and second component T of the system state plotted on the horizontal and vertical axes, respectively. From the

initial guess indicated on the plot, the first iteration of Algorithm FP-SQP yielded a step Δx for the subproblem, where $x + \Delta x$ is labelled as "QP Solution" in the figure. When we recovered a feasible step $\widetilde{\Delta x}$ by means of the stabilized feasibility perturbation procedure described in Section 5, we obtained a perturbed step $\widetilde{\Delta x}$ that is quite close in this phase portrait to the original SQP step; see the curve labelled "Stabilized Projection." If, however, we omit the stabilization (setting $K_k = 0$ for all k), we obtain a perturbed step $\widetilde{\Delta x}$ that is distant from the original SQP step; see the curve labelled "Naive Projection," for which some of the points are off the scale. In fact, the unstabilized perturbation predicts an ignition of the system. The unstabilized perturbation leads to poor algorithmic performance in this example—the Algorithm FP-SQP takes hundreds of iterations to converge. The trust region radius Δ must be shrunk to a very small value before an acceptable perturbed step is generated by the naive perturbation approach.

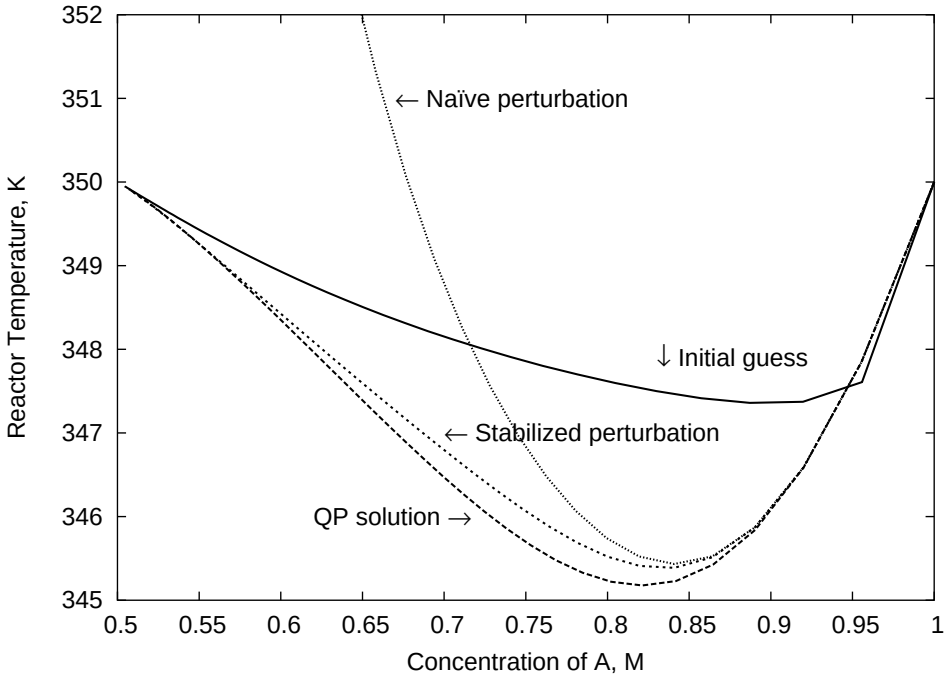

FIG. 8.2. *Phase portrait of states during the first iteration of the CSTR example*

We now turn to our second example.

EXAMPLE 8.2. *An electromagnetically actuated mass spring damper system is governed by the equations*

$$\dot{p} = v \tag{8.3}$$

$$\dot{v} = -\frac{k}{m}p - \frac{c}{m}v + \frac{\alpha}{m}\frac{C}{(d_0 - p)^\gamma} \tag{8.4}$$

where the states p and v represent position and velocity, respectively, and the input C is a function of the current applied to the coil (see [18]). The parameters in this model take on the following values: $\alpha = 4.5 \times 10^{-5}$, $\gamma = 1.99$, $c = 0.6590$, $k = 38.94$, $d_0 = 0.0102$, and $m = 1.54$. We apply the constraint $0 \le C \le 3$ to the input.

Given target values p_{target} *and* v_{target} *for the states and* C_{target} *for the inputs, we define the state variable* x *and the input* u *as follows:*

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} p - p_{\text{target}} \\ v - v_{\text{target}} \end{bmatrix}, \quad u = C - C_{\text{target}}.$$

We used the target values $p_{\text{target}} = .0074$ *and* $v_{\text{target}} = 0$ *(that is, we try to steer the mass to a specified position at rest); and the target input is* $C_{\text{target}} = .0532$. *The initial state of the system is* $p_0 = 0$, $v_0 = .012$.

The sampling time is .01 and the prediction horizon is 100 time steps. We choose the penalty matrices Q, R, and P as in (8.2), to be

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad R = 1, \quad P = \begin{bmatrix} 18776.1 & 1746.93 \\ 1746.93 & 67.751 \end{bmatrix}.$$

To demonstrate the evolution from initial guess to final solution, we present in Figure 8.3 the input profile at each iteration of Algorithm FP-SQP, where the Hessian of the objective was used to approximate the Lagrangian Hessian, as in (6.9). Because all iterates of FP-SQP are feasible, we could use any of these iterates as a feasible suboptimal solution. In this case, Algorithm FP-SQP converges in 10 iterations, but the input profiles become almost indistinguishable after iteration 6.

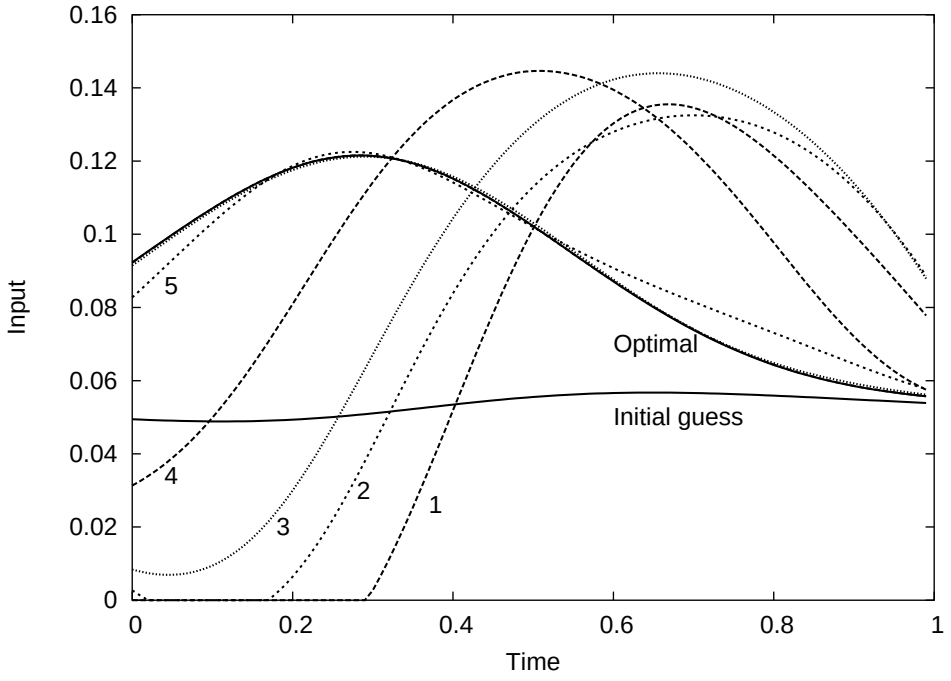

FIG. 8.3. *Input profiles at each iteration of Example 8.2*

Returning to the feasibility perturbation scheme of Section 5, we show that the property (2.4) holds for the perturbation scheme presented here, when applied to Example 8.2. In Table 8.2, we tabulate the ratio

$$\|(\widetilde{\Delta x}, \widetilde{\Delta u}) - (\Delta x, \Delta u)\|_2 / \|(\Delta x, \Delta u)\|_2$$

Iteration	Objective value	$\frac{\|\Delta z - \widetilde{\Delta z}\|_2}{\|\Delta z\|_2}$
Initial	1.48171132254992	
1	1.03826539433288	7.980E-2
2	0.986956571853988	1.293E-2
3	0.896987789509686	4.476E-2
4	0.356871962432300	6.654E-1
5	0.217035426442347	2.300E-2
6	0.214667335503278	2.703E-3
7	0.214630294390683	2.399E-5
8	0.214629782597039	7.499E-7
9	0.214629774955025	8.200E-9
10	0.214629774925470	1.280E-9

TABLE 8.2
Asymptotic Exactness of Feasibility perturbation in Example 8.2

at each iteration, where $(\Delta x, \Delta u)$ is the SQP step and $(\widetilde{\Delta x}, \widetilde{\Delta u})$ is the feasibility-perturbed SQP step. We also show the objective value at each iteration. From the table, it is clear that as the algorithm iterates, the discrepancy between the SQP step and its feasibility-perturbed variant vanishes as the iterates near the solution, so that (2.4) is satisfied for the perturbation scheme used here. Similar results were obtained on other examples.

The Hessian of the Lagrangian in this example is indefinite, which indicates that the Hessian update schemes must take action to approximate this Hessian with a positive definite matrix. In the case of the BFGS approaches, the modified update mechanism is invoked more frequently. For the finite-difference-Hessian and the SR1 schemes, the resulting negative eigenvalues are set to zero, as described in Section 6, so that the subproblem may be solved by the convex QP solver of [21].

In Table 8.3, we compare variants of the Algorithm FP-SQP that use the various Hessian approximation strategies from Section 6. We also show results obtained with the commercial nonlinear optimization package NPSOL [12]. NPSOL is an SQP code that uses a BFGS approximation to the Hessian of the Lagrangian, as in Section 6.3. It uses dense linear algebra, and therefore is unable to take advantage of the structure of MPC problems. However, since the total number of variables in our examples is not particularly large (at most a few hundred), our problems can still be solved by this code in reasonable time. NPSOL does not restrict itself to feasible iterates and uses an augmented Lagrangian merit function to determine whether to accept steps. We applied NPSOL to the MPC problem in two ways. The first approach, denoted by NPSOLz in Table 8.3, refers to the application of NPSOL directly to the formulation (3.1). The second approach, denoted by NPSOLu, is the equivalent optimization problem that results from substituting the model equality constraints directly into the objective function, thereby eliminating the states x_k from the problem and yielding an optimization over only the input variables u_k and the constraint violations η_k.

Algorithm FP-SQP was implemented in Octave [11]. As mentioned earlier, the code LSODE was called from Octave to perform the numerical integrations between sampling times, necessary to obtain a discrete model. DDASAC [7] was used to determine the parametric sensitivities of the model equation with respect to their states and inputs. In the finite-difference-Hessian variant of FP-SQP (Section 6.1) finite differencing over the gradients calculated by DDASAC at perturbed points was used to

obtain an approximate Lagrangian Hessian. Our platform was a 1.2 GHz AMD Athlon running Debian Linux. Initial points were calculated by the procedure described in Section 3. For Algorithm FP-SQP, these initial guesses were good enough that the trust region did not become active on any of the problems for which convergence was reported.

The parameter values used to initialize Algorithm 2.1 are $\eta = 0$ and $\Delta_0 = 1000$. For optimality, one of two tolerances must be satisfied. The absolute tolerance condtion requires $\|u^j - u^{j-1}\|_\infty \leq 1 \times 10^{-8}$. The relative tolerance is satisfied when $\|u^j - u^{j-1}\|_\infty / (\|u^{j-1}\|_\infty + \epsilon) \leq 1 \times 10^{-6}$, in which ϵ is defined as the smallest real number representable in floating point arithmentic.

We apply these codes to five test problems. The first two are Examples 8.1 and 8.2 described above. The third example, CSTR4, is a benchmark tank reactor model described in [9]. This system has four states, two inputs, and a prediction horizon of thirty samples. The fourth example, PEND, is the inverted pendulum system studied by Hauser and Osinga [14]. In this nonlinear model, a moving cart with an inverted pendulum must be steadied by appropriate changes to the velocity of the cart. The system has two states and one input, and the horizon length is thirty samples. The fifth example, COPOLY, is a copolymerization reaction and separation developed by Bindlish [5]. This model has fifteen states and three inputs, and the prediction horizon is twenty sample times.

Table 8.3 shows the number of iterations and the run times for NPSOL and Algorithm FP-SQP applied to each of the five examples. The run times contain limited information. Because Octave is a interpreted language, there is significant overhead reflected in the CPU times for the implementation of FP-SQP. Because the quasi-Newton approximation used in the NPSOL variants does not exploit the structure of the problem, its runtimes serve in part to show the importance of exploting this structure. Finally, because the sensitivity calculations are so expensive in our implementation, the variant of FP-SQP that used a finite-difference Hessian required an inordinate amount of time. The use of a more efficient sensitivity code, or evaluation of the gradients in parallel on a multiprocessor system, could be used to improve the runtime for this variant.

In normal operation, the initial guess to the optimizer would incorporate the solution to the NMPC problem at the previous time step, rather than the procedure described at the end of Section 3, which was used in Table 8.3. Barring large disturbances, the initial guess from the previous timepoint would usually be solved in fewer iterations than the iteration counts shown in Table 8.3. Since robust behavior in the presence of disturbance is a necessary property of a practical control system, however, the performance of the algorithm without prior information on the starting point is germane to our evaluation.

The results for NPSOL show that the original formulation (3.1) (the NPSOLz variant) is more robust than the one in which the states are eliminated for Example 8.1, because of the open-loop instability of this problem. We note that the variants of FP-SQP had little difficulty with this problem, indicating that our stabilized perturbation procedure described in Section 5 was effective. The NPSOLz variant generally has longer runtimes that NPSOLu, because the number of variables (and therefore the sizes of the dense matrices to be factored) is greater. More noteworthy are the observations that NPSOL fails altogether on Examples 8.1 and COPOLY, and takes appreciably more iterations than the FP-SQP variants on Examples 8.2 and PEND. We believe that the maintenance of feasibility in FP-SQP gives it an advantage in

robustness over NPSOL.

Method	Ex. 8.1	Ex. 8.2	CSTR4	PEND	COPOLY
Finite-Difference	4	13	4	5	3
Hessian	18.34	246.93	39.24	78.56	196.60
Objective Hessian	9	10	7	8	5
	11.67	50.75	7.04	26.48	20.27
Partitioned BFGS	6	FAIL	8	8	4
	8.37	N/A	9.92	26.73	16.54
Sparsified BFGS	7	10	7	8	4
	10.06	52.64	7.45	27.15	17.19
Paritioned SR1	6	12	7	11	4
	8.07	60.78	8.86	36.62	16.13
NPSOLu	FAIL	50	3	12	FAIL
	N/A	2279.73	16.30	127.53	N/A
NPSOLz	23	>100	4	16	FAIL
	6783.61	162989.12	4870.23	7834.80	N/A

TABLE 8.3
Comparison of iterations and CPU time to optimality tolerance

Table 8.3 also reveals that different variants of FP-SQP often take a similar number of iterations, with one notable exception. The partitioned BFGS procedure fails to converge on Example 8.2. This failure is a result of poor scaling of the Hessian approximation; the trust region remains inactive, but the steps become smaller and smaller, to the point at which the Hessian is so badly scaled that no further progress can be made. As noted earlier, the finite-difference Hessian variant is not competitive, because of the expense of the repeated sensitivity calculations with DDASAC needed to assemble the approximate Hessian.

In the application to MPC, both efficiency and robustness are necessary. The nonlinear programming algorithm should be efficient enough to find a reasonable approximate solution in the limited time available (the time interval between sampling points), while being robust enough to deal with possibly large disturbances in the system. Given the results of this section, we can draw the following conclusions.

- It is essential to exploit the stagewise structure of the problem.
- For models of the type used in our examples, which are typical of many industrial models, methods that require finite-difference calculation of Hessian approximations are generally too slow.
- Structured quasi-Newton schemes are frequently efficient, but sometimes fail in unpredictable ways.
- The simple scheme of using the objective Hessian as an approximation to the Lagrangian Hessian often works well, even when the objective Hessian is singular.
- An effective practical methodology may be to use the objective Hessian scheme in concert with the structured quasi-Newton schemes, switching from one scheme to another when a failure occurs.

Although the algorithms seemed not to have much difficulty identifying the global solutions on these five examples, there may be situations in which we are led to a local solution, particularly when there is a model-plant mismatch, or after a disturbance. If additional time is available between sampling points, it may be well spent running

these FP-SQP variants from a number of different initial guesses, and checking that the original computed minimizer is at least as good as the minimizers obtained from these speculative starting points.

To gain more insight into the behavior of the update schemes, Figure 8.4 shows the local convergence behavior of each algorithm on Example 8.1. In this plot, the relative error, defined as the ratio of the difference of the current and final objective to the final objective value, at iteration j (horizontal) is plotted against the relative error at iteration $j+1$. Convergence rates can be inferred by determining the slope of the line near the solution (the lower left corner) in this log-log plot. A slope of 2 indicates quadratic convergence, while a slope of 1 indicates linear convergence, It is clear from the figure that the finite-difference Hessian variant yields a quadratic convergence rate. The variants based on the Hessian of the objective or the sparsified BFGS update produce nearly linear convergence. The partitioned BFGS variant is definitely superlinear, while the partitioned SR1 variant is slightly better than linear. The two partitioned quasi-Newton updates have somewhat jagged convergence profiles, however, possibly foreshadowing their occasional failure on other examples.

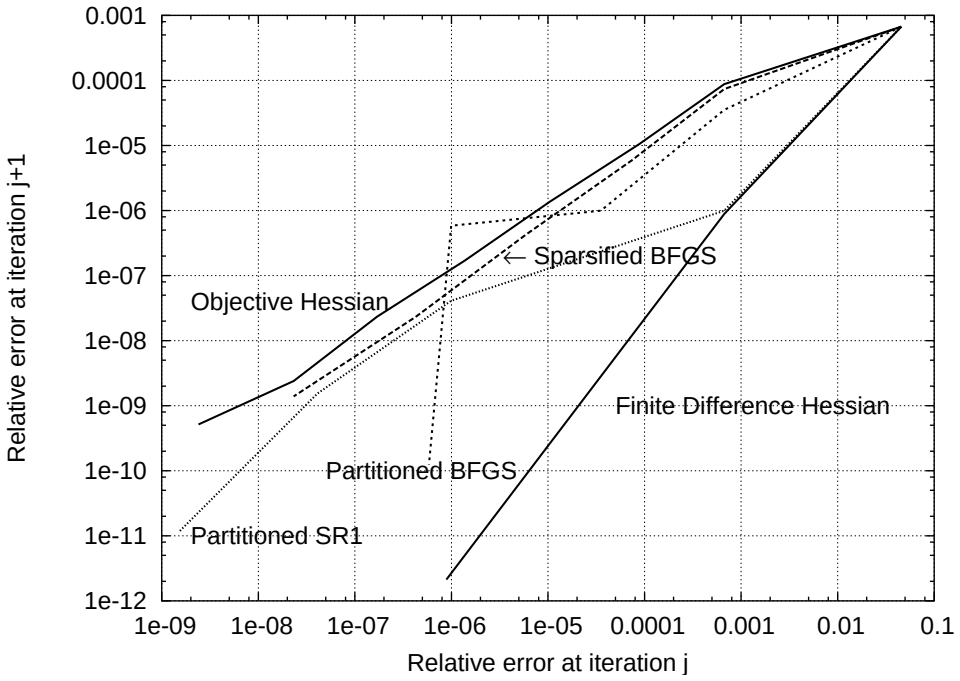

FIG. 8.4. *Comparison of convergence rates for Example 8.1*

Acknowledgments. We gratefully acknowledge the financial support of the industrial members of the Texas-Wisconsin Modeling and Control Consortium. Research of the first and third authors was supported in part by NSF grants #CTS-0105360, while research of the second author was supported in part by NSF grants #MTS-0086559, #ACI-0196485, and #EIA-0127857. All simulations were performed using Octave (`http://www.octave.org`). Octave is freely distributed under the terms of the GNU General Public License.

Appendix A. Asymptotic Exactness of Feasibility-Perturbed Step.

In this section, we show that the feasibility perturbation scheme proposed in Section 5 satisfies the asymptotic exactness condition (2.4). For this purpose, we assume that the constraints $Du_k \leq d$ are not present (although the proof can be extended to handle this case).

The key conditions relating the perturbed steps $\widetilde{\Delta u}_k$, $k = 0, 1, \ldots, N-1$ and $\widetilde{\Delta x}_k$, $k = 1, 2, \ldots, N$ to their original SQP-step counterparts Δu_k and Δx_k are (5.4b) and (5.5). By setting $k = 0$ in (5.5), we see that

$$\widetilde{\Delta u}_0 = \Delta u_0. \tag{A.1}$$

To analyze the relationship between the original and perturbed SQP steps, we simplify the notation as follows:

$$y = (x_1, x_2, \ldots, x_N), \quad w = (u_1, u_2, \ldots, u_{N-1}), \quad w_0 = u_0,$$
$$c(y, w, w_0) = [x_{k+1} - F(x_k, u_k)]_{N-1}^{k=0}, \quad K = \begin{bmatrix} K_1 & & & & 0 \\ & K_2 & & & 0 \\ & & \ddots & & \vdots \\ & & & K_{N-1} & 0 \end{bmatrix},$$

so that (5.5) becomes

$$\widetilde{\Delta w} - \Delta w = K(\widetilde{\Delta y} - \Delta y), \tag{A.2}$$

while (5.4b) becomes

$$c(y + \widetilde{\Delta y}, w + \widetilde{\Delta w}, w_0 + \widetilde{\Delta w_0}) = 0. \tag{A.3}$$

Because $(\Delta y, \Delta w, \Delta w_0)$ is an SQP step, we have

$$c_y(y, w, w_0)\Delta y + c_w(y, w, w_0)\Delta w + c_{w_0}(y, w, w_0)\Delta w_0 = 0. \tag{A.4}$$

The feasibility-perturbed step $(\widetilde{\Delta y}, \widetilde{\Delta w})$ satisfies the algebraic conditions (A.2) and (A.3), which can be viewed a parametrized system of nonlinear equations, with parameter $\widetilde{\Delta w_0} = \Delta u_0$. Note that this system is "square;" that is, the number of equations equals the number of unknowns. Hence, we can obtain the asymptotic exactness result by applying the implicit function theorem. The Jacobian of this system at $(\widetilde{\Delta y}, \widetilde{\Delta w}) = (0, 0)$ is

$$\begin{bmatrix} c_y(y, w, w_0) & c_w(y, w, w_0) \\ K & -I \end{bmatrix}, \tag{A.5}$$

while its residual at the point $(\widetilde{\Delta y}, \widetilde{\Delta w}) = (\Delta y, \Delta w)$ is

$$\begin{aligned} &\begin{bmatrix} c(y + \Delta y, w + \Delta w, w_0 + \Delta w_0) \\ K(\Delta y - \Delta y) - (\Delta w - \Delta w) \end{bmatrix} \\ &= \begin{bmatrix} c_y \Delta y + c_w \Delta w + c_{w_0} \Delta w_0 \\ 0 \end{bmatrix} + O(\|(\Delta y, \Delta w, \Delta w_0)\|^2) \\ &= O(\|(\Delta y, \Delta w, \Delta w_0)\|^2), \end{aligned} \tag{A.6}$$

where we used smoothness of c and (A.4). If we assume that K is chosen so that the matrix (A.5) is nonsingular, and if $(\Delta y, \Delta w, \Delta w_0)$ is sufficiently small, the implicit function theorem together with (A.6) implies that

$$(\Delta y - \widetilde{\Delta y}, \Delta w - \widetilde{\Delta w}) = O\left\|(\Delta y, \Delta w, \Delta w_0)\right\|^2.$$

It follows immediately from (A.1) that

$$\frac{\left\|(\Delta y, \Delta w, \Delta w_0) - (\widetilde{\Delta y}, \widetilde{\Delta w}, \widetilde{\Delta w_0})\right\|}{\|(\Delta y, \Delta w, \Delta w_0)\|} = O\left(\|(\Delta y, \Delta w, \Delta w_0)\|\right) = o(1), \tag{A.7}$$

as required.

Note that the choice $K = 0$ (no stabilization) is sufficient to satisfy the assumptions above, provided that $c_y(y, w, w_0)$ is nonsingular. However, the nonzero choice of K in Section 5 is designed to ensure that (A.5) is much better conditioned than $c_y(y, w, w_0)$, and thus that the constant in the $O(\cdot)$ term in (A.7) is smaller.

Finally, we show that asymptotic exactness also holds for the η_k components. From the SQP step condition, we have similarly to (5.2) that

$$\eta_k + \Delta\eta_k = \max(G(x_k + \Delta x_k) - g, 0).$$

By comparing this expression with (5.2), we obtain

$$\|\Delta\eta_k - \widetilde{\Delta\eta}_k\| \leq \|G(\Delta x_k - \widetilde{\Delta x}_k)\| = O(\|\Delta x_k - \widetilde{\Delta x}_k\|).$$

Asymptotic exactness in these components now follows immediately from (A.7).

REFERENCES

[1] Joao Albuquerque, Vipin Gopal, George Staus, Lorenz T. Biegler, and Erik B. Ydstie. Interior point SQP strategies for large-scale, structured process optimization problems. *Computers & Chemical Engineering*, 23(4):543–554, 1999.

[2] R. A. Bartlett, A. Wächter, and L. T. Biegler. Active set vs. interior point strategies for model predictive control. In *Proceedings of the American Control Conference*, pages 4229–4233, Chicago, IL, June 2000.

[3] Dimitri P. Bertsekas. *Dynamic Programming*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1987.

[4] L. T. Biegler, A. M. Cervantes, and A. Wächter. Advances in simultaneous strategies for dynamic process optimization. Technical report, Departmnet of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA, February 2001.

[5] Rahul Bindlish. *Modeling and Control of Polymerization Processes*. PhD thesis, University of Wisconsin–Madison, 1999.

[6] Hans Georg Bock and Karl J. Plitt. A multiple shooting algorithm for direct solution of optimal ontrol problems. In *Proceedings of the 9th IFAC World Conference*, pages 242–247, Budapest, 1984. Pergamon Press.

[7] Malamas Caracotsios. *Model Parametric Sensitivity Analysis and Nonlinear Parameter Estimation. Theory and Applications*. PhD thesis, University of Wisconsin–Madison, 1986.

[8] A. M. Cervantes, A. Wächter, R. H. Tütüncü, and L. T. Biegler. A reduced space interior-point strategy for optimization of differential algebraic systems. *Computers & Chemical Engineering*, 24:983–990, 2000.

[9] H. Chen, A. Kremling, and F. Allgöwer. Nonlinear predictive control of a benchmark CSTR. In *Proceedings of the 3rd European Control Conference ECC'95*, pages 3247–3252, Rome, Italy, 1995.

[10] Moritz Diehl, H. Georg Bock, Johannes P. Schlöder, Rolf Findeisen, Zoltan Nagy, and Frank Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.

[11] John W. Eaton. Octave: Past, present and future. In Kurt Hornik and Fritz Leisch, editors, *Proceedings of the 2nd International Workshop on Distributed Statistical Computing*, Vienna, Austria, March 2001.

[12] Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. User's guide for SOL/NPSOL (Version 4.0): A Fortran package for nonlinear programming, technical report SOL 86-2. Technical report, Systems Optimization Laboratory, Department of Operations Research, Stanford University, 1986.

[13] A. Griewank and Ph. L. Toint. Partitioned variable metric updates for large structured optimization problems. *Numerische Mathematik*, 39:119–137, 1982.

[14] John Hauser and Hinke Osinga. On the geometry of optimal control: the inverted pendulum example. In *Proceedings of the American Control Conference*, pages 1721–1726, Washington, DC, 2001.

[15] Michael A. Henson and Dale E. Seborg. *Nonlinear Process Control*. Prentice Hall PTR, Upper Saddle River, New Jersey, 1997.

[16] Alan C. Hindmarsh. ODEPACK, a systematized collection of ODE solvers. In R. S. Stepleman, editor, *Scientific Computing*, pages 55–64, Amsterdam, 1983. North-Holland.

[17] D. B. Leineweber, I. Bauer, H. G. Bock, and J. P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Preprint 2001-23, Interdisciplinary Center for Scientific Computing (IWR), University of Heidelberg, D-69120 Heidelberg, Germany, 2001.

[18] R. H. Miller, I. Kolmanovsky, E. G. Gilbert, and P. D. Washabaugh. Control of constrained nonlinear systems: a case study. *IEEE Control Systems Magazine*, 20(1):23–32, 2000.

[19] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 1999.

[20] M. J. D. Powell. A fast algorithm for nonlinearly constrained optimization calculations. In G. A. Watson, editor, *Dundee Conference on Numerical Analysis*, volume 7, Dundee, June 1977. Springer-Verlag.

[21] Christopher V. Rao, Stephen J. Wright, and James B. Rawlings. On the application of interior point methods to model predictive control. *Journal of Optimization Theory and Applications*, 99:723–757, 1998.

[22] Matthew J. Tenny, James B. Rawlings, and Rahul Bindlish. Feasible real-time nonlinear model predictive control. In James B. Rawlings, Babatunde A. Ogunnaike, and John W. Eaton, editors, *Chemical Process Control– VI: Sixth International Conference on Chemical Process Control*, volume 97, Tucson, Arizona, January 2001. AIChE Symposium Series.

[23] S. J. Wright and M. J. Tenny. A feasible trust-region sequential quadratic programming algorithm. Optimization Technical Report 02-05, University of Wisconsin-Madison, Computer Sciences Departments, August 2002. Also Texas-Wisconsin Modeling and Control Consortium Report TWMCC-2002-01.