Fast Model Predictive Control Using Online Optimization

Yang Wang and Stephen Boyd, *Fellow, IEEE*

***Abstract*—A widely recognized shortcoming of model predictive control (MPC) is that it can usually only be used in applications with slow dynamics, where the sample time is measured in seconds or minutes. A well-known technique for implementing fast MPC is to compute the entire control law offline, in which case the online controller can be implemented as a lookup table. This method works well for systems with small state and input dimensions (say, no more than five), few constraints, and short time horizons. In this paper, we describe a collection of methods for improving the speed of MPC, using online optimization. These custom methods, which exploit the particular structure of the MPC problem, can compute the control action on the order of 100 times faster than a method that uses a generic optimizer. As an example, our method computes the control actions for a problem with 12 states, 3 controls, and horizon of 30 time steps (which entails solving a quadratic program with 450 variables and 1284 constraints) in around 5 ms, allowing MPC to be carried out at 200 Hz.**

***Index Terms*—Model predictive control (MPC), real-time convex optimization.**

I. Introduction

In CLASSICAL model predictive control (MPC), the control action at each time step is obtained by solving an online optimization problem. With a linear model, polyhedral constraints, and a quadratic cost, the resulting optimization problem is a quadratic program (QP). Solving the QP using general purpose methods can be slow, and this has traditionally limited MPC to applications with slow dynamics, with sample times measured in seconds or minutes. One method for implementing fast MPC is to compute the solution of the QP explicitly as a function of the initial state [1], [2]; the control action is then implemented online in the form of a lookup table. The major drawback here is that the number of entries in the table can grow exponentially with the horizon, state, and input dimensions, so that "explicit MPC" can only be applied reliably to small problems (where the state dimension is no more than around five).

In this paper, we describe a collection of methods that can be used to greatly speed up the computation of the control action in MPC, using online optimization. Some of the ideas have already been noted in literature, and here we will demonstrate that when used in combination, they allow MPC to be implemented orders of magnitude faster than with generic optimizers.

Our main strategy is to exploit the structure of the QPs that arise in MPC [3], [4]. It has already been noted that with an appropriate variable reordering, the interior-point search direction at each step can be found by solving a block tridiagonal system of linear equations. Exploiting this special structure, a problem with state dimension n, input dimension m, and horizon T takes $O(T(n+m)^3)$ operations per step in an interior-point method, as opposed to $O(T^3(n+m)^3)$ if the special structure were not exploited. Since interior-point methods require only a constant (and modest) number of steps, it follows that the complexity of MPC is therefore *linear* rather than cubic in the problem horizon.

We should mention here, that a popular method for reducing the complexity of the MPC QP is by reformulating the QP entirely in terms of the control inputs. In this case, the QP becomes dense (the structure of the problem is lost), and requires $O(T^3m^3)$ operations per step in an interior point method. This technique is often combined with a strategy known as *move blocking*, where the input is assumed to be constant for fixed portions of the horizon, and so the number of optimization variables is further reduced (see, e.g., [5] and [6]). This can work well when the horizon is small, but we will see that even for modest values of T, this method will be slower compared with a method that fully exploits the problem structure.

Another important technique that can be used in online MPC is *warm-starting* [7], [8], in which the calculations for each step are initialized using the predictions made in the previous step. Warm-start techniques are usually not used in general interior-point methods (in part because these methods are already so efficient) but they can work very well with an appropriate choice of interior-point method, cutting the number of steps required by a factor of five or more.

The final method we introduce is (very) early termination of an appropriate interior-point method. It is not surprising that high quality control is obtained even when the associated QPs are not solved to full accuracy; after all, the optimization problem solved at each MPC step is really a planning exercise, meant to ensure that the current action does not neglect the future. We have found, however, that after only surprisingly few iterations (typically between 3 and 5), the quality of control obtained is very high, even when the QP solution obtained is poor.

We will illustrate our methods on several examples: a mechanical control system, a supply chain problem, and a randomly generated example. The mechanical control system example has $n = 12$ states, $m = 3$ controls, and a horizon $T = 30$;

Manuscript received September 13, 2008; revised November 11, 2008. Manuscript received in final form March 10, 2009. First published June 30, 2009; current version published February 24, 2010. Recommended by Associate Editor J. Lee. The work of Y. Wang was supported by a Rambus Corporation Stanford Graduate Fellowship. This work was supported in part by the Precourt Institute on Energy Efficiency, by NSF Award 0529426, by NASA Award NNX07AEIIA, and by AFOSR Award FA9550-06-1-0514.

Y. Wang and S. Boyd are with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: yw224@stanford.edu; boyd@stanford.edu).

Digital Object Identifier 10.1109/TCST.2009.2017934

each MPC step requires the solution of a QP with 450 variables, 924 inequality constraints, and 360 equality constraints. A simple, non-optimized C implementation of our method allows each MPC step to be carried out in around 5 ms, on a 3 GHz PC, which would allow MPC to be carried out at around 200 Hz. For a larger example, with $n = 30$ states, $m = 8$ controls, and horizon $T = 30$, each step requires solution of a QP with over a thousand variables, and several thousand constraints; our method can be compute each MPC step in around 25 ms, allowing an update rate of 40 Hz. (Details of the performance obtained will be given later.)

MPC can be used for several types of control (and also, estimation) problems, including tracking problems, regulator problems, and stochastic control problems; moreover, the time-varying and finite-horizon versions of each of these can be handled. In this paper we will focus on the infinite-horizon time-invariant stochastic control problem, mentioning at the end of the paper how the methods extend or apply to other types of problems.

A. Related Work

Model predictive control goes by several other names, such as rolling-horizon planning, receding-horizon control, dynamic matrix control, and dynamic linear programming. It has been applied in a wide range of applications, including chemical process and industrial control [5], [9]–[12], control of queuing systems [13], supply chain management [14], [15], stochastic control problems in economics and finance [16], [17], dynamic hedging [18], and revenue management [19], [20].

For discussion of MPC from the point of view of optimal control, see [21]–[23]; for a survey of stability and optimality results, see [24]. For closed-form (offline) methods, see [1], [2], and [25]. For dynamic trajectory optimization using optimization methods, see [26].

Previous work that addresses efficient solution of the QPs that arise in MPC includes [27]–[29], and [63]; for efficient methods for large-scale MPC problems arising in chemical process control, see [30]–[32]. Efficient methods for robust MPC are addressed by [33]–[35]. The idea of using Newton's method, applied to a smooth non-quadratic cost function can be found in [36], [37].

II. Time-Invariant Stochastic Control

A. System Dynamics and Control

In this section, we describe the basic time-invariant stochastic control problem with linear dynamics. The state dynamics are given by

$$x(t+1) = Ax(t) + Bu(t) + w(t), \quad t = 0, 1, \dots \tag{1}$$

where t denotes time, $x(t) \in \mathbf{R}^n$ is the state, $u(t) \in \mathbf{R}^m$ is the input or control, and $w(t) \in \mathbf{R}^n$ is the disturbance. The matrices $A \in \mathbf{R}^{n\times n}$ and $B \in \mathbf{R}^{n\times m}$ are (known) data. We assume that $w(t)$, for different values of t, are independent identically distributed (IID) with known distribution. We let $\bar{w} = \mathbf{E}w(t)$ denote the mean of $w(t)$ (which is independent of t).

The control policy must determine the current input $u(t)$ from the current and previous states $x(0), \dots, x(t)$, i.e., we will have a causal state feedback policy. We let $\phi_t : \mathbf{R}^{(t+1)n} \to \mathbf{R}^m$ denote the control policy, so that

$$u(t) = \phi_t(x(0), \dots, x(t)).$$

This is equivalent to the statement that the random variable $u(t)$ is measurable on the random variable $(w(0), \dots, w(t-1))$. An important special case is a time-invariant static state feedback control, for which $u(t) = \psi(x(t))$, where $\psi : \mathbf{R}^n \to \mathbf{R}^m$ is called the control function.

B. Objective and Constraints

We define the following objective:

$$J = \lim_{T\to\infty} \frac{1}{T} \mathbf{E} \sum_{t=0}^{T-1} \ell(x(t), u(t)) \tag{2}$$

where $\ell : \mathbf{R}^{n+m} \to \mathbf{R}$ is a convex quadratic stage cost function, with the form

$$\ell(x(t), u(t)) = \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix} + q^T x(t) + r^T u(t).$$

Here, $Q = Q^T \in \mathbf{R}^{n\times n}, S \in \mathbf{R}^{n\times m}, R = R^T \in \mathbf{R}^{m\times m}, q \in \mathbf{R}^n$, and $r \in \mathbf{R}^m$ are parameters, and we will assume

$$\begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \geq 0$$

where $\geq$ denotes matrix inequality.

We also have state and control constraints, defined as a set of l linear inequalities

$$F_x x(t) + F_u u(t) \leq f, \quad t = 0, 1, \dots \tag{3}$$

where $F_x \in \mathbf{R}^{l\times n}, F_u \in \mathbf{R}^{l\times m}$, and $f \in \mathbf{R}^l$ are problem data, and $\leq$ denotes vector (componentwise) inequality.

In many problems the objective and constraints are separable in the state and controls. This means that $S = 0$, and that the state and control constraints can be written separately, as

$$F_x x(t) \leq f_x, \quad F_u u(t) \leq f_u, \quad t = 0, 1, \dots$$

where, here, $F_x \in \mathbf{R}^{l_1\times n}, f_x \in \mathbf{R}^{l_1}, F_u \in \mathbf{R}^{l_2\times m}, f_u \in \mathbf{R}^{l_2}$. A further specialization is where, in addition, Q and R are diagonal, and the state and control constraints consist of lower and upper bounds

$$x_{\min} \leq x(t) \leq x_{\max}, \quad u_{\min} \leq u(t) \leq u_{\max}, \qquad t = 0, 1, \dots$$

where $x_{\min}, x_{\max} \in \mathbf{R}^n$ and $u_{\min}, u_{\max} \in \mathbf{R}^m$. We refer to these as box constraints.

C. Stochastic Control Problem

The stochastic control problem is to find a control policy that satisfies the state and control constraints (3), and minimizes the objective J (2). Several pathologies can occur. The stochastic control problem can be infeasible: there exists no causal state feedback policy for which the contraints hold for all t

(with probability one), and J is finite. Our stage cost can be unbounded below (since it has linear terms and the quadratic part need not be positive definite), so the stochastic control problem can also be unbounded below, i.e., there exists a policy for which $J = -\infty$. Our interest in this paper is to describe a method for efficiently computing an MPC control policy (which is itself only a heuristic suboptimal policy) and not the technical details of stochastic control, so we will not consider these issues further, and simply assume that the problem is feasible, with finite optimal cost. See, e.g., [43] for more on the technical aspects of linear stochastic control.

It can be shown that there is an optimal policy which has the form of a time-invariant static state feedback control, i.e., $u(t) = \psi_{\rm opt}(x(t))$. Unfortunately, the state feedack function $\psi_{\rm opt}$ can be effectively computed in only a few special cases, such as when there are no constraints and $w(t)$ is Gaussian.

D. Steady-State Certainty Equivalent Problem

For future reference we describe the steady-state certainty-equivalent problem, in which we assume that x and u have the constant values $\bar{x}$ and $\bar{u}$, and the process noise w has constant value equal to its mean $\bar{w}$. The dynamics equation becomes

$$\bar{x} = A\bar{x} + B\bar{u} + \bar{w}$$

the constraint becomes $F_x\bar{x} + F_u\bar{u} \leq f$, and the objective becomes $\ell(\bar{x}, \bar{u})$.

The steady-state certainty-equivalent problem is then

$$\begin{array}{ll} \text{minimize} & \ell(\bar{x}, \bar{u}) \\ \text{subject to} & \bar{x} = A\bar{x} + B\bar{u} + \bar{w} \quad F_x\bar{x} + F_u\bar{u} \leq f \end{array}$$

with variables $\bar{x} \in \mathbf{R}^n$ and $\bar{u} \in \mathbf{R}^m$. This problem is a convex QP and is readily solved. We will let $\bar{x}^\star$ denote an optimal value of $\bar{x}$ in the certainty equivalent problem.

In many cases the steady-state certainty-equivalent problem has a simple analytic solution. For example, when $\bar{w} = 0$, the objective is purely quadratic (i.e., q and r are both zero), and $f \geq 0$ (which means $\bar{x} = 0, \bar{u} = 0$ satisfy the constraints), we have $\bar{x}^\star = 0$.

E. Model Predictive Control

MPC is a heuristic for finding a good, if not optimal, control policy for the stochastic control problem. In MPC the control $u(t)$ is found at each step by first solving the optimization problem

$$\begin{array}{ll} \text{minimize} & \ell_f(x(t+T)) + \displaystyle\sum_{\tau=t}^{t+T-1} \ell(x(\tau), u(\tau)) \\ \text{subject to} & F_f x(t+T) \leq f_f \\ & F_x x(\tau) + F_u(\tau) \leq f, \qquad \tau = t, \ldots, t+T-1 \\ & x(\tau+1) = Ax(\tau) + Bu(\tau) + \bar{w}, \\ & \tau = t, \ldots, t+T-1 \end{array} \tag{4}$$

with variables $x(t+1), \ldots, x(t+T)$ and $u(t), \ldots, u(t+T-1)$. Here, T is the (planning) horizon, the function $\ell_f : \mathbf{R}^n \to \mathbf{R}$ is the terminal cost function, which we assume is quadratic

$$\ell_f(x(t+T)) = x(t+T)^T Q_f x(t+T) + q_f^T x(t+T)$$

with $Q_f \geq 0$, and $F_f x(t+T) \leq f_f$ is the terminal state constraint. There are many methods for choosing the MPC parameters T, Q_f, q_f, F_f, and f_f; see, e.g., [44]–[48].

The problem (4) is a convex QP with problem data

$$x(t),\ A,\ B,\ Q,\ Q_f,\ S,\ R,\ q,\ q_f,\ r, F_x,\ F_u,\ F_f,\ f,\ f_f,\ \bar{w}.$$

Let $u^\star(t), \ldots, u^\star(t+T-1), x^\star(t+1), \ldots, x^\star(t+T)$ be optimal for the QP (4). The MPC policy takes $u(t) = u^\star(t)$. The MPC input $u(t)$ is evidently a (complicated) function of the current state $x(t)$, so the MPC policy has a static state-feedback form $u(t) = \psi_{\rm mpc}(x(t))$. It can be shown that $\psi_{\rm mpc}$ is a piecewise-affine function, when the quadratic cost term is positive definite (see, e.g., [1]).

We can give a simple interpretation of the MPC QP (4). In this QP, we truncate the true cost function to a horizon T steps in the future, and we replace the current and future disturbances (which are not yet known) with their mean values. We represent the truncated portion of the cost function with an approximate value function ℓ_f. We then compute an optimal trajectory for this simplified problem. We can think of $u^\star(t), \ldots, u^\star(t+T-1)$ as a plan for what we would do, if the disturbance over the next T steps were to take on its mean value. We use only the first control action in this plan, $u^\star(t)$, as our actual control. At time $t+1$, the actual value of $x(t+1)$ becomes available to us, so we carry out the planning process again, over the time period $t+1, \ldots, t+T+1$. This is repeated for each time step.

F. Explicit MPC

In explicit model predictive control, an explicit form of the piecewise-affine control law $\psi_{\rm mpc}$ is computed offline. We compute, offline, the polyhedral regions over which the control is affine, as well as the offset and control gain for each region [1], [2], [49]. The online control algorithm is then reduced to a lookup table: the region associated with the current state $x(t)$ is first determined, and then the control law associated with that region is applied.

This method is very attractive for problems with a small number of states and controls, simple constraints, and a modest horizon, for which the number of regions is manageable. For other problems (say, with $n \geq 5, m \geq 3$, and $l \geq 12$) the number of regions can be very large, so explicit MPC is no longer practically feasible. (There is some hope that a good control law can be obtained by simplifying the piecewise-affine function $\phi_{\rm mpc}$, e.g., using the methods described in [50]–[52], replacing it with a piecewise-affine function with a manageable number of regions.)

Even when the number of regions is manageable (say, a few thousand), it can still be faster to solve the QP (4), using the methods described in this paper, rather than implementing the lookup table required in explicit MPC. Furthermore, explicit MPC cannot handle cases where the system, cost function, or constraints are time-varying. On the other hand, the QP (4) can be easily modified for the case where the matrices $A, B, Q, R, S, F_x, F_u, q, r, f$ and $\bar{w}$ vary over time (we only need to keep track of the time index for these matrices).

III. Primal Barrier Interior-Point Method

In this section, we describe a basic primal barrier interior-point for solving the QP (4), that exploits its special structure. Much of this material has been reported elsewhere, possibly in a different form or context (but seems to not be widely enough appreciated among those who use or study MPC); we collect it here in one place, using a unified notation. In Section IV, we describe variations on the method described here, which give even faster methods.

We first rewrite the QP (4) in a more compact form. We define an overall optimization variable

$$z=(u(t),x(t+1),\ldots,u(t+T-1),x(t+T))\in\mathbf{R}^{T(m+n)}$$

and express the QP (4) as

$$\begin{array}{ll}\text{minimize} & z^THz+g^Tz\\ \text{subject to} & Pz\leq h,\quad Cz=b\end{array}\tag{5}$$

where

$$H=\begin{bmatrix}R&0&0&\cdots&0&0&0\\0&Q&S&\cdots&0&0&0\\0&S^T&R&\cdots&0&0&0\\\vdots&\vdots&\vdots&\ddots&\vdots&\vdots&\vdots\\0&0&0&\cdots&Q&S&0\\0&0&0&\cdots&S^T&R&0\\0&0&0&\cdots&0&0&Q_f\end{bmatrix}$$

$$P=\begin{bmatrix}F_u&0&0&\cdots&0&0&0\\0&F_x&F_u&\cdots&0&0&0\\\vdots&\vdots&\vdots&\ddots&\vdots&\vdots&\vdots\\0&0&0&\cdots&F_x&F_u&0\\0&0&0&\cdots&0&0&F_f\end{bmatrix}$$

$$C=\begin{bmatrix}-B&I&0&0&\cdots&0&0&0\\0&-A&-B&I&\cdots&0&0&0\\0&0&0&-A&\cdots&0&0&0\\\vdots&\vdots&\vdots&\vdots&\ddots&\vdots&\vdots&\vdots\\0&0&0&0&\cdots&I&0&0\\0&0&0&0&\cdots&-A&-B&I\end{bmatrix}$$

$$g=\begin{bmatrix}r+2S^Tx(t)\\q\\r\\\vdots\\q\\r\\q_f\end{bmatrix}$$

$$h=\begin{bmatrix}f-F_xx(t)\\f\\\vdots\\f\\f_f\end{bmatrix}$$

$$b=\begin{bmatrix}Ax(t)+\bar{w}\\\bar{w}\\\bar{w}\\\vdots\\\bar{w}\\\bar{w}\end{bmatrix}.$$

Evidently these matrices have considerable structure; for example, H is block tridiagonal. In special cases we can have even more structure. When the problem is state control separable, for example, H is block diagonal.

A. Primal Barrier Method

We will use an infeasible start primal barrier method to solve the QP [3, Ch. 11], [53]. We replace the inequality constraints in the QP (5) with a barrier term in the objective, to get the approximate problem

$$\begin{array}{ll}\text{minimize} & z^THz+g^Tz+\kappa\phi(z)\\ \text{subject to} & Cz=b\end{array}\tag{6}$$

where $\kappa>0$ is a barrier parameter, and ϕ is the log barrier associated with the inequality constraints, defined as

$$\phi(z)=\sum_{i=1}^{lT+k}-\log\left(h_i-p_i^Tz\right)$$

where $p_1^T,\ldots,p_{lT+k}^T$ are the rows of P. (We take $\phi(z)=\infty$ if $Pz\not<h$.) The problem (6) is a convex optimization problem with smooth objective and linear equality constraints, and can be solved by Newton's method, for example.

As κ approaches zero, the solution of (6) converges to a solution of the QP (5): it can be shown that the solution of (6) is no more than $\kappa(lT+k)$ suboptimal for the QP (5) (see, e.g., [3, Sect. 11.2.2]). In a basic primal barrier method, we solve a sequence of problems of the form (6), using Newton's method starting from the previously computed point, for a decreasing sequence of values of κ. A typical method is to reduce κ by a factor of 10 each time a solution of (6) is computed (within some accuracy). Such a method can obtain an accurate solution of the original QP with a total effort of around 50 or so Newton steps. (Using the theory of self-concordance [3, Sect. 11.5], [54], one can obtain a rigorous upper bound on the total number of Newton steps required to solve the QP to a given accuracy. But this bound is far larger than the number of steps always observed in practice.)

B. Infeasible Start Newton Method

We now focus on solving the problem (6) using an infeasible start Newton method [3, Sect. 10.3.2]. We associate a dual variable $\nu\in\mathbf{R}^{Tn}$ with the equality constraint $Cz=b$. The optimality conditions for (6) are then

$$\begin{aligned}r_d&=2Hz+g+\kappa P^Td+C^T\nu=0,\\ r_p&=Cz-b=0\end{aligned}\tag{7}$$

where $d_i=1/(h_i-p_i^Tz)$, and p_i^T denotes the ith row of P. The term κP^Td is the gradient of $\kappa\phi(z)$. We also have the implicit constraint here that $Pz<h$. We call r_p the primal residual, and r_d the dual residual. The stacked vector $r=(r_d,r_p)$ is called the residual; the optimality conditions for (6) can then be expressed as $r=0$.

In the infeasible start Newton method, the algorithm is initialized with a point z^0 that strictly satisfies the inequality constraints ($Pz^0 < h$), but need not satisfy the equality constraints $Cz = b$, (thus, the initial z^0 can be infeasible, which is where the method gets its name). We can start with any ν^0.

We maintain an approximate z (with $Pz < h$) and ν, at each step. If the residuals are small enough, we quit; otherwise we refine our estimate by linearizing the optimality conditions (7) and computing primal and dual steps $\Delta z, \Delta\nu$ for which $z + \Delta z, \nu + \Delta\nu$ give zero residuals in the linearized approximation.

The primal and dual search steps Δz and $\Delta\nu$ are found by solving the linear equations

$$\begin{bmatrix} 2H + \kappa P^T \mathbf{diag}(d)^2 P & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta\nu \end{bmatrix} = - \begin{bmatrix} r_d \\ r_p \end{bmatrix}. \quad (8)$$

(The term $\kappa P^T \mathbf{diag}(d)^2 P$ is the Hessian of $\kappa\phi(z)$.) Once Δz and $\Delta\nu$ are computed, we find a step size $s \in (0, 1]$ using a backtracking line search on the norm of the residual r, making sure that $Pz < h$ holds for the updated point (see, e.g., [3, Sect. 9.2]). We then update our primal and dual variables as $z := z + s\Delta z$ and $\nu := \nu + s\Delta\nu$. This procedure is repeated until the norm of the residual is below an acceptable threshold.

It can be shown that primal feasibility (i.e., $Cz = b$) will be achieved in a finite number of steps, assuming the problem (6) is strictly feasible. Once we have $r_p = 0$, it will remain zero for all further iterations. Furthermore, z and ν will converge to an optimal point. The total number of Newton steps required to compute an accurate solution depends on the initial point z^0 (and ν^0). This number of steps can be formally bounded using self-concordance theory, but the bounds are much larger than the number of steps typically required.

C. Fast Computation of the Newton Step

If we do not exploit the structure of the (8), and solve it using a dense $\mathrm{LDL^T}$ factorization, for example, the cost is $(1/3)T^3(2n + m)^3$ flops. But we can do much better by exploiting the structure in our problem.

We will use block elimination [3, App. C]. Before we proceed, let us define $\Phi = 2H + \kappa P^T \mathbf{diag}(d)^2 P$, which is block diagonal, with the first block $m \times m$, the last block $n \times n$ and the remaining $T - 1$ blocks $(n + m) \times (n + m)$. Its inverse is also block diagonal; we write it as

$$\Phi^{-1} = \begin{bmatrix} \tilde{R}_0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \tilde{Q}_1 & \tilde{S}_1 & \cdots & 0 & 0 & 0 \\ 0 & \tilde{S}_1^T & \tilde{R}_1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \tilde{Q}_{T-1} & \tilde{S}_{T-1} & 0 \\ 0 & 0 & 0 & \cdots & \tilde{S}_{T-1} & \tilde{R}_{T-1} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \tilde{Q}_T \end{bmatrix}.$$

Solving (8) by block elimination involves the following sequence of steps.

1) Form the Schur complement $Y = C\Phi^{-1}C^T$ and $\beta = -r_p + C\Phi^{-1}r_d$.
2) Determine $\Delta\nu$ by solving $Y\Delta\nu = -\beta$.
3) Determine Δz by solving $\Phi\Delta z = -r_d - C^T\nu$.

The Schur complement Y has the block tridiagonal form

$$Y = \begin{bmatrix} Y_{11} & Y_{12} & 0 & \cdots & 0 & 0 \\ Y_{21} & Y_{22} & Y_{23} & \cdots & 0 & 0 \\ 0 & Y_{32} & Y_{33} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & Y_{T-1,T-1} & Y_{T-1,T} \\ 0 & 0 & 0 & \cdots & Y_{T,T-1} & Y_{TT} \end{bmatrix}$$

where

$$\begin{aligned} Y_{11} &= B\tilde{R}_0 B^T + \tilde{Q}_1, \\ Y_{ii} &= A\tilde{Q}_{i-1}A^T + A\tilde{S}_{i-1}B^T \\ &\quad + B\tilde{S}_{i-1}^T A^T + B\tilde{R}_{i-1}B^T + \tilde{Q}_i, \qquad i = 2, \ldots, T \\ Y_{i,i+1} &= Y_{i+1,i}^T = -\tilde{Q}_i A^T - \tilde{S}_i B^T, \qquad i = 1, \ldots, T-1. \end{aligned}$$

We can form Y as follows. First we compute the Cholesky factorization of each of the blocks in Φ, which requires $(1/3)m^3 + (T - 1)(1/3)(m + n)^3$ flops, which is order $T(m + n)^3$. Forming Y is then done by backward and forward-substitution with columns taken from A and B, and then multiplying by the associated blocks in C; this requires order $T(n^3 + n^2 m)$ flops. Thus, step 1 requires order $T(n + m)^3$ flops.

Step 2 is carried out by Cholesky factorization of Y, followed by backward and forward-substitution. The matrix Y is block tridiagonal, with T (block) rows, with $n \times n$ blocks. It can be factored efficiently using a specialized method described below for block tridiagonal matrices (which is related to the Riccati recursion in control theory) [27]–[29], [55], [56], or by treating it as a banded matrix, with bandwidth $3n$. Both of these methods require order Tn^3 flops ([3], [4], [29]–[32], [55], [56]). Step 2 therefore requires order Tn^3 flops.

The Cholesky factorization of $Y = LL^T$, where L is lower triangular, is found as follows. The Cholesky factor L has the lower bidiagonal block structure

$$L = \begin{bmatrix} L_{11} & 0 & 0 & \cdots & 0 & 0 \\ L_{21} & L_{22} & 0 & \cdots & 0 & 0 \\ 0 & L_{32} & L_{33} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & L_{T-1,T-1} & 0 \\ 0 & 0 & 0 & \cdots & L_{T,T-1} & L_{TT} \end{bmatrix}$$

where L_{ii} are $n \times n$ lower triangular with positive diagonal entries, and $L_{i+1,i}$ are general $n \times n$ matrices. Directly from $LL^T = Y$ we find that

$$\begin{aligned} L_{11}L_{11}^T &= Y_{11} \\ L_{ii}L_{i+1,i}^T &= Y_{i,i+1}, \quad i = 1, \ldots, T-1, \\ L_{ii}L_{ii}^T &= Y_{ii} - L_{i,i-1}L_{i,i-1}^T, \quad i = 2, \ldots, T. \end{aligned}$$

Thus, we can find L_{11} by Cholesky factorization of Y_{11}, then we find L_{21} by solving $L_{11}L_{21}^T = Y_{12}$ by forward substitution; then we can find L_{22} by Cholesky factorization of $Y_{22} - L_{21}L_{21}^T$, and so on. Each of these steps requires order n^3 flops.

The cost of step 3 is dominated by the other steps, since the Cholesky factorization of Φ was already computed in step 1. The

overall effort required to compute the search directions Δz and $\Delta\nu$, using block elimination, is order $T(m+n)^3$. Note that this order grows linearly with the horizon T, as opposed to cubicly, if the (8) are solved using a generic method.

For special cases, further savings in complexity are possible. For example, when the problem is state control separable, with diagonal Q and R, and box constraints, the matrix Φ is diagonal. In this case, the over all complexity is order $T(n^3+n^2m)$ flops, which grows linearly in both T and m.

D. Warm Start

In a typical primal barrier method, we solve (6) for a decreasing sequence of values of κ. When we decrease κ and solve (6) again, for the new value of κ, we can simply use the previously computed z as the initial z for the first Newton step. This warm start method is extremely effective in reducing the number of Newton steps required for each value of κ. (Indeed, it is the key to the primal barrier method.) It can be proved, using the theory of self-concordance, that when κ is decreased by any constant factor, the total number of Newton steps required to solve (6) to a given accuracy, starting from the previously computed solution, is bounded by a polynomial of the problem dimensions. While the existence of such a bound is reassuring, its value is much larger than the number of Newton steps actually required, which is typically small (say, between 5 and 10) [3, Sect. 10.3.3].

We can also use this warm start idea to initialize z the *first* time we solve (6), at time step t, using the previously computed trajectory for time step $t-1$. Roughly speaking, in MPC we compute the current action by working out an entire plan for the next T time steps. We can use the previously computed plan, suitably shifted in time, as a good starting point for the current plan.

We initialize the primal barrier method for solving (6) for time step t with the trajectories for x and u computed during the previous time step, with $\hat{u}$ and $\hat{x}$ appended at the end. One simple choice for $\hat{u}$ and $\hat{x}$ is $\bar{u}^\star$ and $\bar{x}^\star$. Suppose at time $t-1$ the computed trajectory is

$$\tilde{z} = (\tilde{u}(t-1), \tilde{x}(t-1), \ldots, \tilde{u}(t+T-2), \tilde{x}(t+T-1)).$$

We can initialize the primal barrier method, for time step t, with

$$z^{\text{init}} = (\tilde{u}(t), \tilde{x}(t+1), \ldots, \tilde{x}(t+T-1), \bar{u}^\star, \bar{x}^\star).$$

Assuming $\tilde{z}$ satisfies the equality constraints, so will z^{init}, except at the first and last time steps. If $\tilde{z}$ satisfies the inequality constraints strictly, then so will z^{init}, except possibly at the first and last time steps. In this case, we can modify $u(t)$ and $\hat{u}$ so that the inequality constraints are strictly satisfied.

Several other warm start methods can be used in special cases. For example, let us consider the case with box constraints. We can work out the linear control obtained using the associated LQR problem, ignoring the constraints, or using the techniques described in [44], and then project this trajectory a small distance into the feasible set (which is a box). This warm start initialization has the (possible) advantage of depending only on $x(t)$, and not on the previous states or any controller state.

IV. Approximate Primal Barrier Method

In this section, we describe some simple variations on the basic infeasible start primal barrier method described above. These variations produce only a good approximate solution of the basic MPC QP (5), but with no significant decrease in the quality of the MPC control law (as measured by the objective J). These variations, however, can be computed much faster than the primal barrier method described previously.

A. Fixed κ

Our first variation is really a simplification of the barrier method. Instead of solving (6) for a decreasing sequence of values of κ, we propose to use one fixed value, which is never changed. Moreover, we propose that this fixed value is not chosen to be too small; this means that the suboptimality bound $\kappa(lT+k)$ will not be small.

For a general QP solver, using a single fixed value of κ would lead to a very poor algorithm, that could well take a very large number of steps, depending on the problem data, and in addition only computes an approximate solution. We propose the use of a fixed value of κ here for several reasons. First, we must remember that the goal is to compute a control that gives a good objective value, as measured by J, and not to solve the QP (5) accurately. (Indeed, the QP is nothing but a heuristic for computing a good control.) In extensive numerical experiments, we have found that the quality of closed-loop control obtained by solving the approximate QP (6) instead of the exact QP (5) is extremely good, even when the bound on suboptimality in solving the QP is not small. This was also observed by Wills and Heath [57] who explain this phenomenon as follows. When we substitute the problem (6) for (5), we can interpret this as solving an MPC problem exactly, where we interpret the barrier as an additional nonquadratic state and control cost function terms.

The particular value of κ to use turns out to not matter much (in terms of Newton iterations required, or final value of J achieved); any value over a very wide range (such as a factor of 10^3) seems to give good results. A good value of κ can be found for any particular application by increasing it, perhaps by a factor of 10 each time, until simulation shows a drop in the quality of control achieved. The previous value of κ can then be used.

The idea of fixing a barrier parameter to speed up the approximate solution of a convex problem was described in [58], where the authors used a fixed barrier parameter to compute a nearly optimal set of grasping forces extremely rapidly.

A second advantage we get by fixing κ is in warm starting from the previously computed trajectory. By fixing κ, each MPC iteration is nothing more than a Newton process. In this case, warm starting from the previously computed trajectory reliably gives a very good advantage in terms of the number of Newton steps required. In contrast, warm start for the full primal barrier method offers limited, and erratic, advantage.

By fixing κ, we can reduce the number of Newton steps required per MPC iteration from a typical value of 50 or so for a primal barrier method, to a value on the order of 5. (The exact number depends on the application; in some cases it can be even smaller.)

B. Fixed Iteration Limit

Our second variation on the primal barrier method is also a simplification. By fixing κ we have reduced each MPC calculation to carrying out a Newton method for solving (6). In a standard Newton method, the iteration is stopped only when the norm of the residual becomes small, or some iteration limit $K^{\max}$ is reached. It is considered an algorithm failure when the iteration limit is reached before the residual norm is small enough. (This happens, for example, when the problem does not have a strictly feasible point.) In MPC, the computation of the control runs periodically with a fixed period, and has a hard run-time constraint, so we have to work with the worst case time required to compute the control, which is $K^{\max}$ times the time per Newton step.

Now we come to our second simplification. We simply choose a very small value of $K^{\max}$, typically between 3 and 10. (When the MPC control is started up, however, we have no previous trajectory, so we might use a larger value of $K^{\max}$.) Indeed, there is no harm in simply running a fixed number $K^{\max}$ of Newton steps per MPC iteration, independent of how big or small the residual is.

When $K^{\max}$ has some small value such as 5, the Newton process can terminate with a point z that is not even primal feasible. Thus, the computed plan does not even satisfy the dynamics equations. It does, however, respect the constraints; in particular, $u(t)$ satisfies the current time constraint $F_x(t) + F_u(t) \leq f$ (assuming these constraints are strictly feasible). In addition, of course, the dual residual need not be small.

One would think that such a control, obtained by such a crude solution to the QP, could be quite poor. But extensive numerical experimentation shows that the resulting control is of very high quality, with only little (or no) increase in J when compared to exact MPC. Indeed, we have observed that with $K^{\max}$ as low as 1, the control obtained is, in some cases, not too bad. We do not recommend $K^{\max} = 1$; we only mention it as an interesting variation on MPC that requires dramatically less computation.

We do not fully understand why this control works so well, even in cases where, at many time instances, the optimization is terminated before achieving primal feasibility. One plausible explanation goes back to the basic interpretation of MPC: At each step, we work out an entire plan for the next T steps, but only use the current control. The rest of the plan is not really used; the planning is done to make sure that the current control does not have a bad effect on the future behavior of the system. Thus it seems reasonable that the current control will be good, even if the plan is not carried out very carefully.

It is interesting to note that when the algorithm does not fully converge in $K^{\max}$ iterations, the resulting control law is not a static state feedback, as (exact) MPC is. Indeed, the controller state is the plan z, and the control $u(t)$ does indeed depend (to some extent) on $x(t-1)$.

C. Summary and Implementation Results

We have developed a simple implementation of our approximate primal barrier method, written in C, using the LAPACK library [59], [60] to carry out the numerical linear algebra computations. Our current C implementation, available at http://www.stanford.edu/~boyd/fast_mpc.html, handles the case of separable purely quadratic objective (i.e., $q = 0, r = 0$) and box constraints. We report here the timing results for 12 problems with different dimensions, constructed using the method described in Section V-C, on a 3 GHz AMD Athlon running Linux.

TABLE I
TIME TAKEN TO SOLVE EACH QP FOR RANDOMLY GENERATED EXAMPLES

n	m	T	QP size	$K^{\max}$	Time (ms)	SDPT3 (ms)
4	2	10	60/168	3	0.94	150
4	2	20	120/328	3	1.87	250
4	2	30	180/488	3	2.79	400
10	3	10	130/380	3	1.84	450
10	3	20	260/740	3	3.68	800
10	3	30	390/1100	3	5.52	1400
16	4	10	200/592	3	2.50	500
16	4	20	400/1152	3	5.04	1300
16	4	30	600/1712	3	7.58	2600
30	8	10	380/1120	5	7.63	1400
30	8	20	760/2180	5	16.42	4800
30	8	30	1140/3240	5	25.95	3400

To be sure that the inputs computed by our method delivered control performance essentially equivalent to exact MPC, we simulated each example with exact MPC, solving the QP exactly, using the generic optimization solver SDPT3 [61], called by CVX [62]. (The reported times, however, include only the SDPT3 CPU time.) SDPT3 is a state-of-the-art primal-dual interior-point solver, that exploits sparsity. The parameters $K^{\max}$ and κ in our approximate primal barrier method were chosen to give control performance, as judged by Monte Carlo simulation to estimate average stage cost, essentially the same as the control performance obtained by solving the QP exactly; in any case, never more than a few percent worse (and in some cases, better).

Table I lists results for 12 problems. The column listing QP size gives the total number of variables (the first number) and the total number of constraints (the second number). We can see that the small problems (which, however, would be considered large problems for an explicit MPC method) are solved in under a millisecond, making possible kiloHertz control rates. The largest problem, which involves a QP that is not small, with more than a thousand variables and several thousand constraints, is solved in around 26 ms, allowing a control rate of several tens of hertz. We have solved far larger problems as well; the time required by our approximate primal barrier method grows as predicted, or even more slowly.

We can also see that the approximate barrier method far outperforms the generic (and very efficient) solver SDPT3, which only exploits sparsity. Of course the comparison is not entirely fair; in each call to SDPT3, the sparsity pattern must be detected, and a good elimination ordering determined; in contrast, in our code, the sparsity exploitation has already been done. (Indeed we can see this effect: for the problem with $n = 30, m = 8$, and $T = 30$, the time taken by SDPT3 is actually lower than for $T = 20$, since SDPT3 chooses to treat the former problem as sparse, but the latter one as dense.) The numbers make our

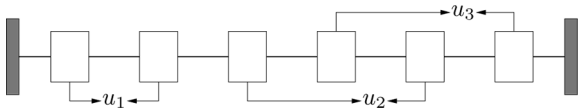

Fig. 1. Oscillating masses model. Bold lines represent springs, and the dark regions on each side represent walls.

point: Generic convex optimization solvers, even very efficient ones, are not optimized for repeatedly solving small and modest sized problems very fast.

V. Examples

In the next section, we present three specific examples: a mechanical control system, a supply chain, and a randomly generated system. We compare the performance of our fast approximate MPC method with exact MPC, and give timing specifications for the two cases which our current C implementation can handle.

A. Oscillating Masses

The first example consists of a sequence of six masses connected by springs to each other, and to walls on either side, as shown in Fig. 1. There are three actuators, which exert tensions between different masses. The masses have value 1, the springs all have spring constant 1, and there is no damping. The controls can exert a maximum force of ± 0.5, and the displacements of the masses cannot exceed ± 4.

We sample this continuous time system, using a first order hold model, with a period of 0.5 (which is around 3 times faster than the period of the fastest oscillatory mode of the open-loop system). The state vector $x(t) \in \mathbf{R}^{12}$ is the displacement and velocity of the masses. The disturbance $w(t) \in \mathbf{R}^6$ is a random force acting on each mass, with a uniform distribution on $[-0.5, 0.5]$. (Thus, the disturbance is as large as the maximum allowed value of the actuator forces, but acts on all six masses.) For MPC we choose a horizon $T = 30$, and separable quadratic objective with $Q = I, R = I, S = 0, q = 0, q_f = 0, r = 0$. We choose Q_f using a heuristic method described in [44]. The problem dimensions are $n = 12, m = 3, l = 18$ and $k = 24$. The steady-state certainty equivalent optimal state and control are, of course, zero.

All simulations were carried out for 1100 time steps, discarding the first 100 time steps, using the same realization of the random force disturbance. The initial state $x(0)$ is set to the steady-state certainty equivalent value, which is zero in this case. We first compare exact MPC (computed using CVX [62], which relies on the solver SDPT3 [61]) with approximate MPC, computed using a fixed positive value of κ, but with no iteration limit. Exact MPC achieves an objective value $J = 5.93$ (computed as the average stage cost over the 1000 time steps). Fig. 2 shows the distributions of stage cost for exact MPC and for approximate MPC with $\kappa = 1, 10^{-1}, 10^{-2}$, and 10^{-3}. For $\kappa \leq 10^{-2}$, the control performance is essentially the same as for exact MPC; for $\kappa = 10^{-1}$, the objective is less than 3% larger than that obtained by exact MPC, and for $\kappa = 1$, the objective is about 18% larger than exact MPC.

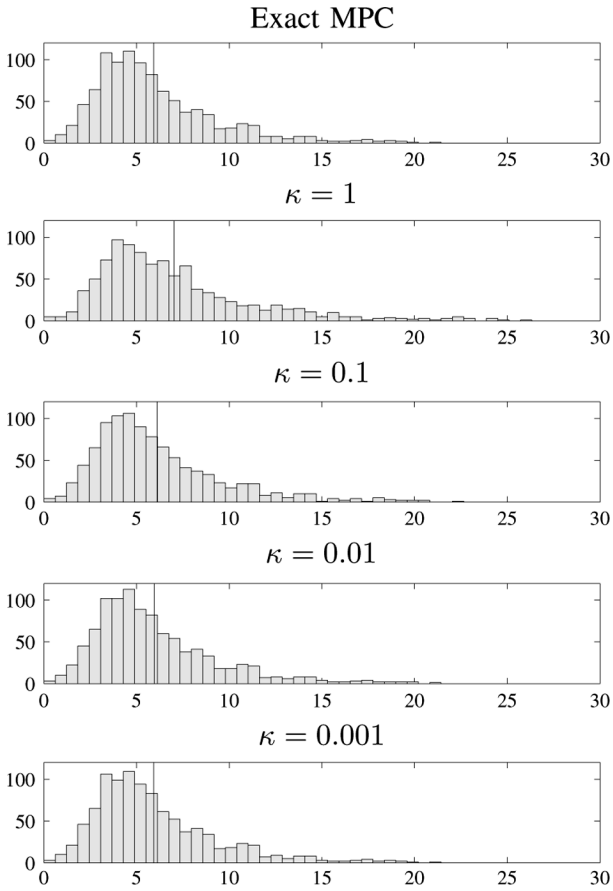

Fig. 2. Histograms of stage costs, for different values of κ, for oscillating masses example. Solid vertical line shows the mean of each distribution.

To study the effect of the iteration limit $K^{\max}$, we fix $\kappa = 10^{-2}$, and carry out simulations for $K^{\max} = 1, K^{\max} = 3$, and $K^{\max} = 5$. The distribution of stage costs is shown in Fig. 3. For $K^{\max} = 3$ and $K^{\max} = 5$, the quality of control obtained (as measured by average stage cost) is essentially the same as for exact MPC. For $K^{\max} = 1$, in which only one Newton step is taken at each iteration, the quality of control is measurably worse than for exact MPC, but it seems surprising to us that this control is even this good. For $K^{\max} = 1$, primal feasibility is attained in 5% of the steps, while for $K^{\max} = 3$, primal feasibility is attained in only 68% of the steps, and for $K^{\max} = 5$, primal feasibility is attained in 95% of the steps. It is apparent (and a bit shocking) that primal feasibility is not essential to obtaining satisfactory control performance.

For this example, a reasonable choice of parameters would be $\kappa = 10^{-2}$ and $K^{\max} = 5$, which yields essentially the same average stage cost as exact MPC, with a factor of 10 or more speedup (based on 50 iterations for exact solution of the QP). The control produced by $\kappa = 10^{-2}$ and $K^{\max} = 5$ is very similar to, but not the same as, exact MPC. Fig. 4 shows $x_1(t)$, the displacement of the first mass, and $u_1(t)$, the first control, for both exact MPC and fast MPC. The state trajectories are almost indistinguishable, while the control trajectories show a few small deviations.

Our simple C implementation can carry out one Newton step for the oscillating masses problem in around 1 ms. With

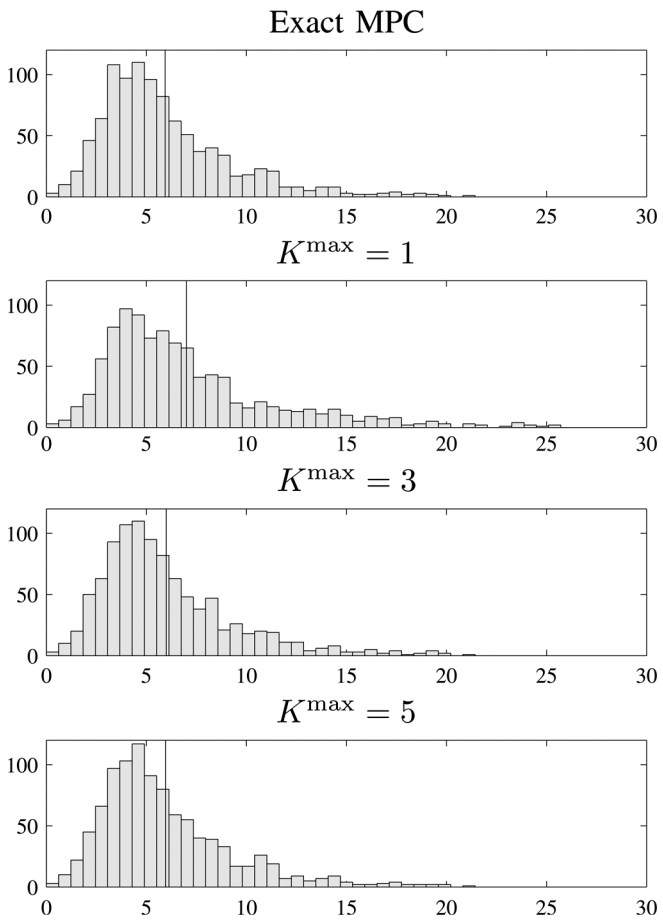

Fig. 3. Histograms of stage costs, for different values of $K^{\max}$, with $\kappa = 10^{-2}$, for oscillating masses example. The solid vertical line shows the mean of each distribution.

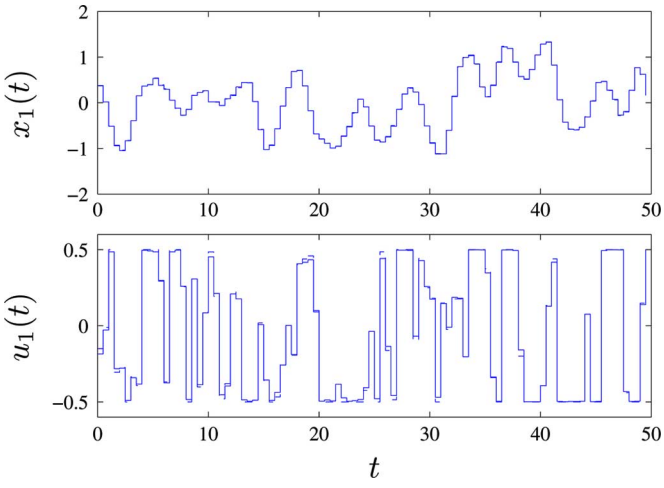

Fig. 4. Simulation of oscillating masses example with $\kappa = 10^{-2}, K^{\max} = 5$. The solid curves show fast MPC; the dashed curves show exact MPC.

$K^{\max} = 5$, our approximate MPC control can be implemented with sample time of 5 ms. It follows that the MPC control can be carried out at a rate up to 200 Hz.

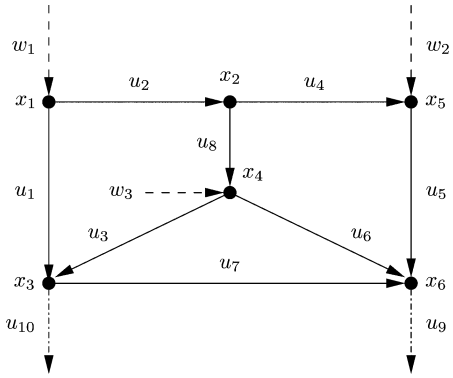

Fig. 5. Supply chain model. Dots represent nodes or warehouses. Arrows represent links or commodity flow. Dashed arrows are inflows and dash-dot arrows are outflows.

B. Supply Chain

This example is a single commodity supply chain consisting of 6 nodes, representing warehouses or buffers, interconnected with 13 uni-directional links, representing some transport mechanism or flow of the commodity, as shown in Fig. 5. Three of the flows, represented by dashed arrows, are inflows, and beyond our control; the remaining 10 flows, however, are the controls. Two of these controls are outflows, represented in dashed-dotted line type.

At each time step, an uncertain amount of commodity enters the network along the three inflow links; the control law chooses the flow along the other 10 edges, including the two outflow links. The system state is the quantity of the commodity present at each node, so $x(t) \in \mathbf{R}^6$. The state is constrained to satisfy $x(t) \geq 0$. The control is $u(t) \in \mathbf{R}^{10}$, each component of which is constrained to lie in the interval $[0, 2.5]$. (Thus, each link has a capacity of 2.5.) The disturbance $w(t)$ is the inflow, so $w(t) \in \mathbf{R}^3$. The components of w are IID with exponential distribution, with mean one. There is one more constraint that links the controls and the states: The total flow out of any node, at any time, cannot exceed the amount of commodity available at the node. Unlike the constraints in the oscillating masses example, this problem is not separable in the state and control. The problem dimensions are $n = 6, m = 10$, and $l = 32$.

The objective parameters are $Q = I, R = 0, S = 0, q = \mathbf{1}, r = \mathbf{1}, Q_f = I, q_f = \mathbf{1}$. (Here $\mathbf{1}$ denotes the vector with all entries one.) This means that there is a storage cost at each node, with value $x_i(t)+x_i(t)^2$, and a charge equal to the flow on each edge. The storage cost gives incentive for the commodity to be routed out of the network through the outflow links u_9 and u_{10}. For this problem, the steady-state certainty equivalent problem is not trivial; it must be computed by solving a QP.

For MPC control of the supply chain, we used a horizon $T = 10$. In our simulations, we use initial state $x(0) = \bar{x}^\star$. In this example, we find that with $\kappa = 0.01, K^{\max} = 10$, our approximate MPC gives essentially the same quality of control as exact MPC. Fig. 6 shows $x_1(t)$ and $u_1(t)$ for both exact MPC (dashed line) and approximate MPC (solid line). Evidently, the controls and states for the two are almost indistinguishable.

Our current C implementation of the fast MPC method does not handle coupled state input constraints, which are present in

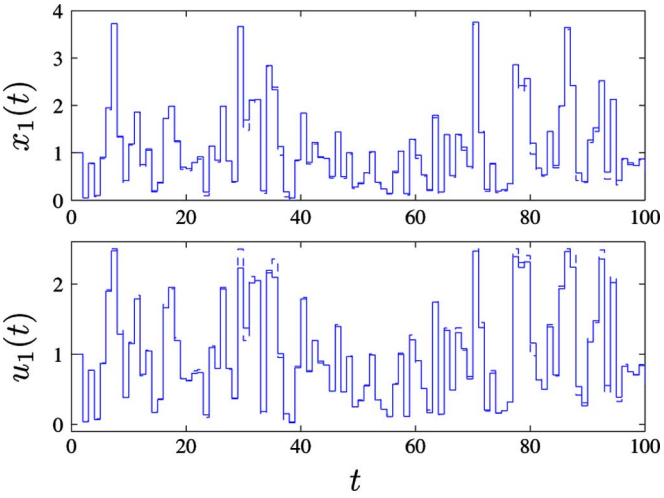

Fig. 6. Simulation of supply chain with $\kappa = 0.01, K^{\max} = 10$. Solid line: Fast MPC, Dashed line: Exact MPC.

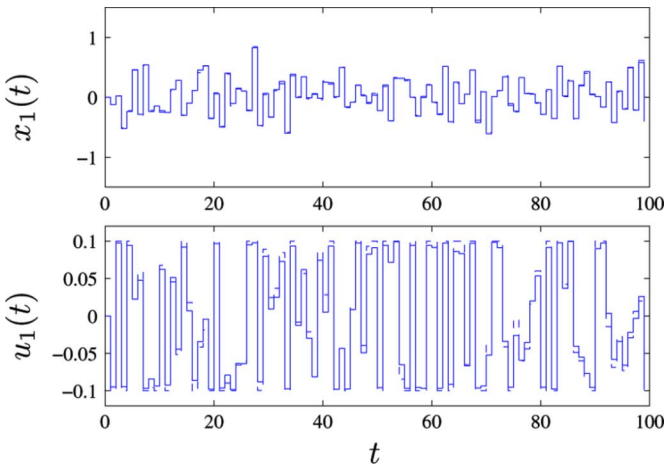

Fig. 7. Simulation of random system with $\kappa = 0.01, K^{\max} = 5$. Solid line: fast MPC; Dashed line: exact MPC.

this example, so we cannot report the timing. (All the simulations shown in this example were found using a MATLAB implementation of the fast MPC method, which is slow for many reasons.) We can, however, form a reasonable estimate of the performance that would be obtained, based on the complexity analysis given above and extrapolating from other examples. Our estimate is that the fast MPC implementation would take around 1.2 ms per Newton step, and so around 12 ms per solve.

C. Random System

Our third example is a randomly generated system, where the entries of A and B are zero mean unit variance normal random variables. We then scale A so that its spectral radius is one, so the system is neutrally stable. We have already given timing results for this family of problems in Table I; here we consider a specific problem, to check that out approximate solution of the QP has little or no effect on the control performance obtained.

The particular example we consider has $n = 30$ states and $m = 8$ controls. The constraints are box constraints: $|x_i| \leq 5, |u_i| \leq 0.1$. The disturbance $w(t) \in \mathbf{R}^{30}$ has IID entries, uniformly distributed on $[-0.3, 0.3]$. (The constraints and disturbance magnitudes were chosen so that the controls are often saturated.) The cost parameters are $Q = I, R = I, q = 0, r = 0, q_f = 0$ and we choose Q_f using the method described in [44]. For MPC we use a horizon of $T = 30$, with a randomly generated starting point $x(0)$.

As in the previous examples, we found that approximate primal barrier MPC, with parameters $\kappa = 0.01$ and $K^{\max} = 5$, yields a control essentially the same as exact MPC. A sample trace is shown in Fig. 7.

Using our C implementation, we can carry out one Newton step for this example in around 5 ms. With $K^{\max} = 5$, our method allows MPC to be implemented with a sample time of 25 ms, so control can be carried out at a rate of 40 Hz.

VI. Extensions and Variations

Our discussion so far has focussed on the time-invariant infinite-horizon stochastic control problem. Here, we describe how the same methods can be applied in many variations on this particular MPC formulation. First, it should be clear that the exact same methods can be used when the system, cost function, or constraints are time-varying; we only need to keep track of the time index in the matrices A, B, Q, R, S, F_x, and F_u, and the vectors q, r, f, and $\bar{w}$. The same infeasible-start Newton method, and the same method for rapidly computing the Newton step, can be used. All that changes is that these data matrices (possibly) change at each time step. Other extensions that are straightforward include multi-rate problems, or problems with piecewise-linear or piecewise-constant inputs, or the addition of a final state constraint (instead of our final state cost term).

The method can be applied to nonlinear systems as well, using standard techniques. At each Newton step we simply linearize the dynamics (using Jacobians or particle filter methods), at the current value of z, and compute the step using this linearized model. (We would, however, carry out the line search using the true primal residual, with nonlinear dynamics, and not the primal residual in the linearized dynamics.) In this case, the data matrices change each Newton step (as the linearization is updated). We have not experimented much with applying these methods to problems with nonlinear dynamics; we expect that a larger number of iterations will be required to give good control performance. Our ability to rapidly compute each Newton step will be useful here as well.

We can use any smooth convex stage cost functions, with little change. We can incorporate nonsmooth convex stage cost functions, by introducing local variables that yield a (larger) smooth problem (see, e.g., [3, Ch. 4]). These added variables are "local", i.e., interact only with $x(t), u(t)$, so their contribution to the Hessian will also be local, and the same methods can be applied. One example is moving horizon estimation with a Huber cost function, which gives a smoothing filter that is very robust to outliers in process disturbance or noise [3, Sect. 6.1]. We can also add any convex constraints that are "local" in time, i.e., that link state or control over a fixed number of time steps; such constraints lead to KKT matrices with the same banded form.

VII. Conclusion and Implications

By combining several ideas, some of which are already known in MPC or other contexts, we can dramatically increase the speed with which online computation of MPC control laws can be carried out. The methods we have described complement offline methods, which give a method for fast control computation when the problem dimensions are small. Combined with ever-increasing available computer power, the possibility of very fast online computation of MPC control laws suggests to us that MPC can be used now, or soon, in many applications where it has not been considered feasible before.

Much work, both practical and theoretical, remains to be done in the area of fast online MPC methods. While our extensive simulations suggest that fast online MPC works very well, a formal stability analysis (or, even better, performance guarantee) would be a welcome advance.

Acknowledgment

The authors would like to thank M. Morari, S. Meyn, A. Bemporad, and P. Parrilo for very helpful discussions. They would also like to thank K. Koh and A. Mutapcic for advice on the C implementation, and D. Gorinevski for help with the examples.

References

[1] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, Jan. 2002.

[2] P. Tøndel, T. A. Johansen, and A. Bemporad, "An algorithm for multi-parametric quadratic programming and explicit MFC solutions," in *Proc. IEEE Conf. Dec. Control*, 2001, pp. 1199–1204.

[3] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge University Press, 2004.

[4] S. J. Wright, "Applying new optimization algorithms to model predictive control," *Chemical Process Control-V.*, vol. 93, no. 316, pp. 147–155, 1997.

[5] J. M. Maciejowski, *Predictive Control with Constraints*. Englewood Cliffs, NJ: Prentice-Hall, 2002.

[6] R. Cagienard, P. Grieder, E. C. Kerrigan, and M. Morari, "Move blocking strategies in receding horizon control," in *Proc. 43rd IEEE Conf. Dec. Control*, Dec. 2004, pp. 2023–2028.

[7] F. A. Potra and S. J. Wright, "Interior-point methods," *J. Comput. Appl. Math.*, vol. 124, no. 1–2, pp. 281–302, 2000.

[8] E. A. Yildirim and S. J. Wright, "Warm-start strategies in interior-point methods for linear programming," *SIAM J. Opt.*, vol. 12, no. 3, pp. 782–810, 2002.

[9] R. Soeterboek, *Predictive Control: A Unified Approach*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[10] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Eng. Practice*, vol. 11, no. 7, pp. 733–764, 2003.

[11] E. Camacho and C. Bordons, *Model Predictive Control*. New York: Springer-Verlag, 2004.

[12] W. Wang, D. E. Rivera, and K. Kempf, "Model predictive control strategies for supply chain management in semiconductor manufacturing," *Int. J. Production Economics*, vol. 107, pp. 57–77, 2007.

[13] S. P. Meyn, *Control Techniques for Complex Networks*. Cambridge, U.K.: Cambridge University Press, 2006.

[14] E. G. Cho, K. A. Thoney, T. J. Hodgson, and R. E. King, "Supply chain planning: Rolling horizon scheduling of multi-factory supply chains," in *Proc. 35th Conf. Winter Simulation: Driving Innovation*, 2003, pp. 1409–1416.

[15] W. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. New York: Wiley, 2007.

[16] H. Dawid, "Long horizon versus short horizon planning in dynamic optimization problems with incomplete information," *Economic Theory*, vol. 25, no. 3, pp. 575–597, Apr. 2005.

[17] F. Herzog, "Strategic portfolio management for long-term investments: An optimal control approach," Ph.D. dissertation, ETH, Zurich, The Netherlands, 2005.

[18] J. Primbs, "Dynamic hedging of basket options under proportional transaction costs using receding horizon control," in *Int. J. Control*, 2009. [Online]. Available: http://www.Stanford.edu/japrimbs/Auto-Submit20070813.pdf

[19] K. T. Talluri and G. J. V. Ryzin, *The Theory and Practice of Revenue Management*. New York: Springer, 2004.

[20] D. Bertsimas and I. Popescu, "Revenue management in a dynamic network environment," *Transportation Sci.*, vol. 37, no. 3, pp. 257–277, 2003.

[21] W. H. Kwon and S. Han, *Receding Horizon Control*. New York: Springer-Verlag, 2005.

[22] P. Whittle, *Optimization Over Time*. New York: Wiley, 1982.

[23] G. C. Goodwin, M. M. Seron, and J. A. De Doná, *Constrained Control and Estimation*. New York: Springer, 2005.

[24] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, Jun. 2000.

[25] G. Pannocchia, J. B. Rawlings, and S. J. Wright, "Fast, large-scale model predictive control by partial enumeration," *Automatica*, vol. 43, no. 5, pp. 852–860, May 2006.

[26] J. T. Belts, *Practical Methods for Optimal Control Using Nonlinear Programming*. Warrendale, PA: SIAM, 2001.

[27] A. Hansson and S. Boyd, "Robust optimal control of linear discrete-time systems using primal-dual interior-point methods," in *Proc. Amer. Control Conf.*, 1998, vol. 1, pp. 183–187.

[28] A. Hansson, "A primal-dual interior-point method for robust optimal control of linear discrete-time systems," *IEEE Trans. Autom. Control*, vol. 45, no. 9, pp. 1639–1655, Sep. 2000.

[29] L. Vandenberghe, S. Boyd, and M. Nouralishahi, "Robust linear programming and optimal control," presented at the 15th IFAC World Congr. Autom. Control, Barcelona, Spain, Jul. 2002.

[30] R. A. Bartlett, L. T. Biegler, J. Backstrom, and V. Gopal, "Quadratic programming algorithms for large-scale model predictive control," *J. Process Control*, vol. 12, no. 7, pp. 775–795, 2002.

[31] L. T. Biegler, "Efficient solution of dynamic optimization and NMPC problems," in *Nonlinear Model Predictive Control*, F. Allgöwer and A. Zheng, Eds. Cambridge, MA: Birkhauser, 2000, pp. 119–243.

[32] J. Albuquerque, V. Gopal, G. Staus, L. T. Biegler, and B. E. Ydstie, "Interior point SQP strategies for large-scale, structured process optimization problems," *Comput. Chem. Eng.*, vol. 23, no. 4–5, pp. 543–554, 1999.

[33] P. J. Goulart, E. C. Kerrigan, and D. Ralph, "Efficient robust optimization for robust control with constraints," *Math. Program.*, vol. Series A, pp. 1–33, 2007.

[34] M. Cannon, W. Liao, and B. Kouvaritakis, "Efficient MFC optimization using pontryagin's minimum principle," *Int. J. Robust Nonlinear Control*, vol. 18, no. 8, pp. 831–844, 2008.

[35] V. M. Savala, C. D. Laird, and L. T. Biegler, "Fast implementations and rigorous models: Can both be accommodated in NMPC?," *Int. J. Robust Nonlinear Control*, vol. 18, no. 8, pp. 800–815, 2008.

[36] C. G. Economou, M. Morari, and B. O. Palsson, "Internal model control: Extension to nonlinear system," *Ind. Eng. Chem. Process Des. Development*, vol. 25, no. 2, pp. 403–411, 1986.

[37] W. C. Li, L. T. Biegler, C. G. Economou, and M. Morari, "A constrained Pseudo-Newton control strategy for nonlinear systems," *Comput. Chem. Eng.*, vol. 14, no. 4, pp. 451–468, 1990.

[38] R. Bitmead, V. Wertz, and M. Gevers, *Adaptive Optimal Control: The Thinking Man's GPC.*. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[39] A. Bemporad, "Model predictive control design: New trends and tools," in *Proc. 45th IEEE Conf. Dec. Control*, 2006, pp. 6678–6683.

[40] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[41] D. Q. Mayne and H. Michalska, "Receding horizon control of nonlinear systems," *IEEE Trans. Autom. Control*, vol. 35, no. 7, pp. 814–824, 1990.

[42] S. M. Estill, "Real-time receding horizon control: Application programmer interface employing LSSOL," Dept. Mech. Eng., Univ. California, Berkeley, Dec. 2003.

[43] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Belmont, MA: Athena Scientific, 2005.

[44] Y. Wang and S. Boyd, "Performance bounds for linear stochastic control," *Syst. Control Lett.*, vol. 53, no. 3, pp. 178–182, Mar. 2009.

[45] M. Corless and G. Leitmann, "Controller design for uncertain system via Lyapunov functions," in *Proc. Amer. Control Conf.*, 1988, vol. 3, pp. 2019–2025.
[46] R. A. Freeman and J. A. Primbs, "Control Lyapunov functions, new ideas from an old source," in *Proc. 35th IEEE Conf. Dec. Control*, 1996, vol. 4, pp. 3926–3931.
[47] E. D. Sontag, "A Lyapunov-like characterization of asymptotic controllability," *SIAM J. Control Opt.*, vol. 21, no. 3, pp. 462–471, 1983.
[48] M. Sznaier, R. Suarez, and J. Cloutier, "Suboptimal control of constrained nonlinear systems via receding horizon constrained control Lyapunov functions," *Int. J. Robust Nonlinear Control*, vol. 13, no. 3–4, pp. 247–259, 2003.
[49] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MFC," *Int. J. Robust Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.
[50] P. Tøndel and T. A. Johansen, "Complexity reduction in explicit linear model predictive control," presented at the 15th IFAC World Congr. Autom. Control, Barcelona, Spain, Jul. 2002.
[51] A. Bemporad and C. Filippi, "Suboptimal explicit receding horizon control via approximate multiparametric quadratic programming," *J. Opt. Theory Appl.*, vol. 117, no. 1, pp. 9–38, Nov. 2004.
[52] A. Magnani and S. Boyd, "Convex piecewise-linear fitting," *Opt. Eng.* Mar. 2008. [Online]. Available: http://www.stanford.edu/~boyd/cvx_pwl_fitting.html
[53] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York: Springer, 1999.
[54] Y. Nesterov and A. Nemirovsky, *Interior-Point Polynomial Methods in Convex Programming*. Warrendale, PA: SIAM, 1994.
[55] M. A. Kerblad and A. Hansson, "Efficient solution of second order cone program for model predictive control," *Int. J. Control*, vol. 77, no. 1, pp. 55–77, Jan. 2004.
[56] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of interior point methods to model predictive control," *J. Opt. Theory Appl.*, vol. 99, no. 3, pp. 723–757, Nov. 2004.
[57] A. G. Wills and W. P. Heath, "Barrier function based model predictive control," *Automatica*, vol. 40, no. 8, pp. 1415–1422, Aug. 2004.
[58] S. Boyd and B. Wegbreit, "Fast computation of optimal contact forces," *IEEE Trans. Robot.*, vol. 23, no. 6, pp. 1117–1132, Dec. 2007.
[59] E. Anderson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. D. Croz, S. Hammarling, J. Demmel, C. Bischof, and D. Sorensen, "LAPACK: A portable linear algebra library for high-performance computers," in *Proc. Supercomput.*, 1990, pp. 2–11.
[60] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*. Warrendale, PA: SIAM, 1999.
[61] K. C. Toh, M. J. Todd, and R. H. Tütüncü, "SDPT3: A matlab software package for semidefinite programming," *Opt. Methods Softw.*, vol. 11, no. 12, pp. 545–581, 1999.
[62] M. Grant, S. Boyd, and Y. Ye, "CVX: Matlab software for disciplined convex programming," 2006. [Online]. Available: http://www.stanford.edu/~boyd/cvx
[63] K. Ling, B. Wu, and J. Maciejowski, "Embedded model predictive control (MPC) using a FPGA," in *Proc. 17th IFAC World Congr.*, Jul. 2008, pp. 15250–15255.

Yang Wang received the B.A. and M.Eng. degrees in electrical and information engineering from Cambridge University (Magdalene College), Cambridge, U.K., in 2006. He is currently pursuing the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA.

His current research interests include convex optimization with applications to control, signal processing, and machine learning. He is generously supported by a Rambus Corporation Stanford Graduate Fellowship.

Stephen P. Boyd (S'82–M'85–SM'97–F'99) received the A.B. degree in mathematics (*summa cum laude*) from Harvard University, Cambridge, MA, in 1980, and the Ph.D. degree in electrical engineering and computer science from the University of California, Berkeley, in 1985.

He is the Samsung Professor of Engineering with the Information Systems Laboratory, Electrical Engineering Department, Stanford University, Stanford, CA. His current research interests include convex programming applications in control, signal processing, and circuit design. He is the author of *Linear Controller Design: Limits of Performance* (Prentice-Hall, 1991), *Linear Matrix Inequalities in System and Control Theory* (SIAM, 1994), and *Convex Optimization* (Cambridge University Press, 2004).

Dr. Boyd was a recipient of an ONR Young Investigator Award, a Presidential Young Investigator Award, and the 1992 AACC Donald P. Eckman Award. He has received the Perrin Award for Outstanding Undergraduate Teaching in the School of Engineering, and an ASSU Graduate Teaching Award. In 2003, he received the AACC Ragazzini Education award. He is a Distinguished Lecturer of the IEEE Control Systems Society.