Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Fast Model Predictive Control Using Online Optimization

Yang Wang * **Stephen Boyd** **

* *Stanford University, Stanford, CA 94305*
(e-mail: yw224@stanford.edu)
** *Stanford University, Stanford, CA 94305*
(e-mail: boyd@stanford.edu)

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 5), 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 1260 constraints) in around 5msec, allowing MPC to be carried out at 200Hz.

1. 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 (Bemporad et al. [2002], ndel et al. [2001]); 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 5).

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 (Boyd and Vandenberghe [2004], Wright [1997]). It has already been noted that with an appropriate variable re-ordering, 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.

Another important technique that can be used in online MPC is *warm-starting* (Potra and Wright [2000], Yildirim and Wright [2002]), 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 5 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.

For a mechanical control system example with $n = 12$ states, $m = 3$ controls, and a horizon $T = 30$; each MPC step requires the solution of a QP with 450 variables, and 1260 constraints. A simple, non-optimized C implementation of our method allows each MPC step to be carried out in around 5msec on a 3GHz PC, which would allow MPC to be carried out at around 200Hz. For a larger example, with $n = 30$ states, $m = 8$ controls, and horizon

$T = 30$, our method can be compute each MPC step in under 25msec, allowing an update rate of 40Hz.

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 (Maciejowski [2002], Camacho and Bordons [2004], Kwon and Han [2005]) control of queuing systems (Meyn [2006]), supply chain management (Powell [2007]) stochastic control problems in economics and finance, (Herzog [2005]), dynamic hedging (Primbs [2007]), and revenue management.

2. TIME-INVARIANT STOCHASTIC CONTROL

2.1 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

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

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 $x(0), \ldots, x(t)$.

2.2 Objective and constraints

At each time instant t we define the stage cost:

$$s(x(t), u(t)) = \begin{bmatrix} x(t) \\ u(t) \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} \begin{bmatrix} x(t) \\ u(t) \end{bmatrix},$$

and the following objective:

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

Here $Q = Q^T \in \mathbf{R}^{n\times n}$, $S \in \mathbf{R}^{n\times m}$ and $R = R^T \in \mathbf{R}^{m\times m}$ 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,

$$F_1 x(t) + F_2 u(t) \leq f, \quad t = 0, 1, \ldots, \tag{3}$$

where $F_1 \in \mathbf{R}^{l\times n}$, $F_2 \in \mathbf{R}^{l\times m}$, and $f \in \mathbf{R}^l$ are problem data, and $\leq$ denotes vector (componentwise) inequality.

The stochastic control problem is to find a control policy that satisfies the state and control constraints (3), and minimizes the objective J (2). It can be shown that there is an optimal policy which has the form of a time-invariant static state feedback control, *i.e.*, $u(t) = \psi_{\text{opt}}(x(t))$. (See, *e.g.*, Bertsekas [2005] for more on the technical aspects of linear stochastic control.)

2.3 Model predictive control

Model predictive control 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 QP

$$\begin{array}{ll} \text{minimize} & \dfrac{1}{T} \displaystyle\sum_{\tau=t}^{t+T-1} s(x(\tau), u(\tau)) \\ \text{subject to} & F_1 x(\tau) + F_2 u(\tau) \leq f, \quad \tau = t, \ldots, t+T-1, \\ & x(\tau+1) = Ax(\tau) + Bu(\tau) + \bar{w}, \\ & \qquad \tau = t, \ldots, t+T-2, \\ & 0 = Ax(t+T-1) + Bu(t+T-1) + \bar{w}, \end{array} \tag{4}$$

with variables

$$x(t+1), \ldots, x(t+T-1), \qquad u(t), \ldots, u(t+T-1),$$

and with problem data

$$x(t),\ A,\ B,\ Q,\ S,\ R,\ F_1,\ F_2,\ f,\ \bar{w}.$$

Here T is a parameter in MPC, called the (planning) horizon. Let $u^\star(t), \ldots, u^\star(t+T-1)$, $x^\star(t+1), \ldots, x^\star(t+T-1)$ be optimal for the QP (4). At each time instant t, the MPC policy takes $u(t) = u^\star(t)$.

3. 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 §4, we describe variations on the method described here, which give even faster methods.

We first re-write (4) in a more compact form. We define an overall optimization variable $z \in \mathbf{R}^{Tm+T(n-1)}$ as follows,

$$z = (u(t), x(t+1), u(t+1), \ldots, x(t+T-1), u(t+T-1)),$$

and express the QP as

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

where

$$H = \begin{bmatrix} R & 0 & 0 & \ldots & 0 & 0 \\ 0 & Q & S & \ldots & 0 & 0 \\ 0 & S^T & R & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & Q & S \\ 0 & 0 & 0 & \ldots & S^T & R \end{bmatrix}, \ g = \begin{bmatrix} 2S^T x(t) \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix},$$

$$P = \begin{bmatrix} F_1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & F_1 & F_2 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & F_1 & F_2 \end{bmatrix}, \ h = \begin{bmatrix} f - F_1 x(t) \\ f \\ \vdots \\ f \end{bmatrix},$$

$$C = \begin{bmatrix} -B & I & 0 & 0 & \ldots & 0 & 0 \\ 0 & -A & -B & I & \ldots & 0 & 0 \\ 0 & 0 & 0 & -A & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \ldots & I & 0 \\ 0 & 0 & 0 & 0 & \ldots & -A & -B \end{bmatrix}, \ b = \begin{bmatrix} Ax(t) + \bar{w} \\ \bar{w} \\ \bar{w} \\ \vdots \\ \bar{w} \\ \bar{w} \end{bmatrix}.$$

3.1 Primal barrier method

We will use an infeasible start primal barrier method to solve the QP ([Boyd and Vandenberghe, 2004, Chap. 11] Nocedal and Wright [1999]). We replace the inequality constraints in the QP (5) with a barrier term in the objective, to get the approximate problem

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

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

$$\phi(z) = \sum_{i=1}^{lT} -\log(h_i - p_i^T z),$$

where $p_1^T, \ldots, p_{lT}^T$ 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 κlT suboptimal for the QP (5) (see, *e.g.*, [Boyd and Vandenberghe, 2004§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 (for rigorous upper bounds see ([Boyd and Vandenberghe, 2004§11.5], Nesterov and Nemirovsky [1994]).

3.2 Infeasible start Newton method

We now focus on solving the problem (6) using an infeasible start Newton method [Boyd and Vandenberghe, 2004§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

$$r_d = 2Hz + g + \kappa P^T d + C^T\nu = 0, \qquad r_p = Cz - b = 0, \tag{7}$$

where $d_i = 1/(h_i - p_i^T z)$, and p_i^T denotes the ith row of P. The term $\kappa P^T d$ 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 $r = [r_d^T \; r_p^T]^T$ 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 Δz, $\Delta\nu$ 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

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

(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.*, [Boyd and Vandenberghe, 2004§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 within a finite number of steps. 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.

3.3 Fast computation of the Newton step

If we do not exploit the structure of the equations (8), and solve it using a dense $\mathrm{LDL}^{\mathrm{T}}$ 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 [Boyd and Vandenberghe, 2004, App. C]. Before we proceed, let us define $\Phi = 2H + \kappa P^T \mathbf{diag}(d)^2 P$, which is block diagonal, with first block $m \times m$, and the remaining $T-1$ blocks $(n+m)\times(n+m)$. Its inverse is also block diagonal, with the same block structure as H.

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 $\Phi\Delta z = -r_d - C^T\nu$.

It is not difficult to show that the Shur complement Y is actually block tridiagonal, with T (block) rows and $n \times n$ blocks. Forming Y in the first step requires on the order of $T(n+m)^3$ flops.

Step 2 is carried out by Cholesky factorization of Y, followed by backward and forward-substitution. Y can be factored efficiently using a specialized method described below for block tridiagonal matrices (which is related to the Riccati recursion in control theory) (kerblad and Hansson [2004], Rao et al. [2004], Vandenberghe et al. [2002])or by treating it as a banded matrix, with bandwidth $3n$. Both of these methods require order Tn^3 flops (Wright [1997], Boyd and Vandenberghe [2004], kerblad and Hansson [2004], Bartlett et al. [2002], Rao et al. [2004], Vandenberghe et al. [2002], Biegler [2000]). Step 2 therefore requires order Tn^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 equations (8) are solved using a generic method.

3.4 Warm start

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 ν computed during the previous time step. Suppose at time $t-1$ the computed trajectory is

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

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

$$z^{\mathrm{init}} = (\tilde{u}(t), \tilde{x}(t+1), \ldots, \tilde{x}(t+T-2), \tilde{u}(t+T-2), 0, 0) .$$

Assuming $\tilde{z}$ satisfies the equality constraints, and satisfies the inequality constraints strictly, then z^{init} will also satisfy these constraints, except possibly at the first time step. In this case we can modify the initial $u(t)$ so that it satisfies $F_1 u(t) + F_2 x(t) < f$.

4. 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 above.

4.1 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. 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. 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 (Wills and Heath [2004]) 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 idea of fixing a barrier parameter to speed up the approximate solution of a convex problem was described in (Boyd and Wegbreit [2007]), 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.

4.2 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. 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 $F_1 x(t) + F_2 u(t) \leq f$ (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.

4.3 Summary and implementation results

We have developed a simple implementation of our approximate primal barrier method, written in C, using the LAPACK library (Anderson et al. [1990]) to carry out the numerical linear algebra computations. In §5 we will describe a mechanical control example in detail. We mention here the times required to compute the control laws for randomly generated examples of different sizes using the approximate primal barrier method, on a 3Ghz AMD Athlon running Linux. We compare the performance of our method against solving the QP exactly, using the generic optimization solver SDPT3 (Toh et al. [1999]), called by CVX (Grant et al. [2006]). (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

n	m	T	QP size	$K^{\max}$	Time (ms)	SDPT3 (ms)
4	2	20	100/320	3	0.63	250
10	3	30	360/1080	3	3.97	1400
16	4	30	570/1680	3	7.70	2600
30	8	30	1110/3180	5	23.39	3400

Table 1. Time taken to solve each QP for randomly generated examples.

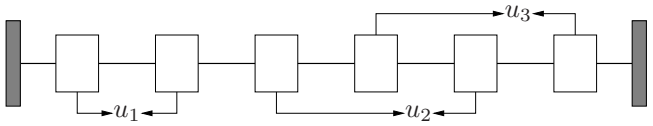

Fig. 1. Oscillating masses model. Bold lines represent springs, and the dark regions on each side represent walls.

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 2% worse (and in some cases, better).

Table 1 lists results for 4 problems of different sizes. 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 23ms, 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.

5. EXAMPLE: OSCILLATING MASSES

Our example consists of a sequence of 6 masses connected by springs to each other, and to walls on either side, as shown in figure 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 $Q = I$, $R = I$, $S = 0$. The problem dimensions are $n = 12$, $m = 3$, and $l = 18$. 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

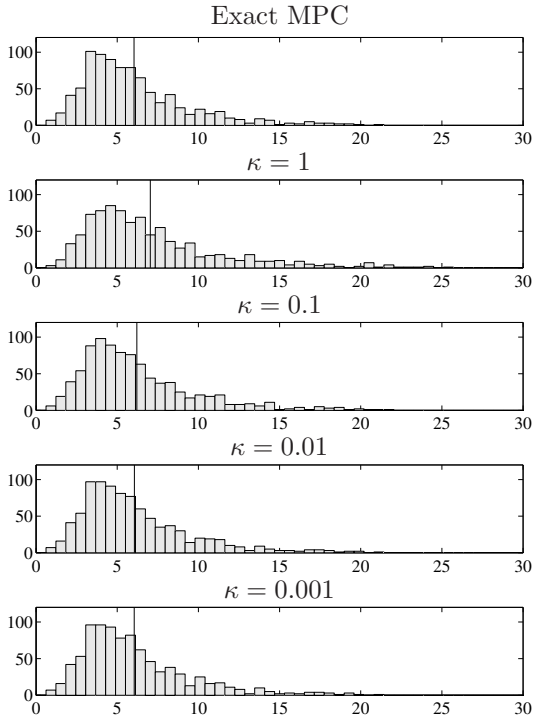

Fig. 2. Histograms of stage costs, for different values of κ, for oscillating masses example. Solid vertical line shows the mean of each distribution.

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 (Grant et al. [2006]), which relies on the solver SDPT3 (Toh et al. [1999])) with approximate MPC, computed using a fixed positive value of κ, but with no iteration limit. Exact MPC achieves an objective value $J = 6.0352$ (computed as the average stage cost over the 1000 time steps). Figure 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 2.88% larger than that obtained by exact MPC, and for $\kappa = 1$, the objective is 16.58% larger than exact MPC.

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 figure 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 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 or more 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.

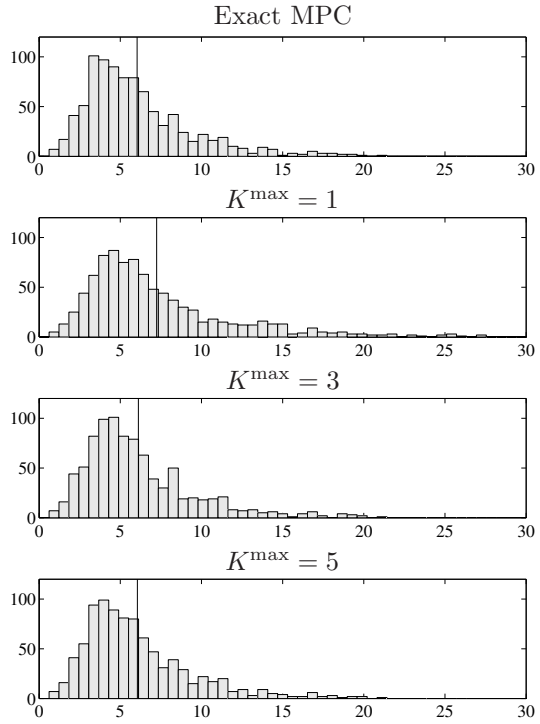

Fig. 3. Histograms of stage costs, for different values of $K^{\max}$, with $\kappa = 10^{-2}$, for oscillating masses example. The solid vertical line shows the mean of each distribution.

Our simple C implementation can carry out one Newton step for the oscillating masses problem in around 1msec. With $K^{\max} = 5$, our approximate MPC control can be implemented with sample time of 5msec.

ACKNOWLEDGMENTS

This material is based on work supported by by the Precourt Institute on Energy Efficiency, by NSF award 0529426, by NASA award NNX07AEIIA, by AFOSR award FA9550-06-1-0514, by AFOSR award FA9550-06-1-0312, and by a Rambus Corporation Stanford Graduate Fellowship. The authors would like to thank Manfred Morari, Sean Meyn, and Pablo Parrilo for very helpful discussions, Kwangmoo Koh and Almir Mutapcic for advice on the C implementation, and Dimitri Gorinevski for help with the examples.

REFERENCES

E. Anderson, Z. Bai, J. Dongarra, A. Greenbaum, A. McKenney, J. Du Croz, S. Hammarling, J. Demmel, C. Bischof, and D. Sorensen. LAPACK: a portable linear algebra library for high-performance computers. In *Proceedings of Supercomputing*, pages 2–11, 1990.

R. A. Bartlett, L. T. Biegler, J. Backstrom, and V. Gopal. Quadratic programming algorithms for large-scale model predictive control. *Journal of Process Control*, 12(7):775–795, 2002.

A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, January 2002.

D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2005.

L. T. Biegler. Efficient solution of dynamic optimization and NMPC problems. In F. Allgöwer and A. Zheng, editors, *Nonlinear Model Predictive Control*, pages 119–243. Birkhauser, 2000.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

S. Boyd and B. Wegbreit. Fast computation of optimal contact forces. *IEEE Transactions on Robotics*, 23(6): 1117–1132, December 2007.

E. F. Camacho and C. Bordons. *Model Predictive Control*. Springer-Verlag, 2004.

M. Grant, S. Boyd, and Y. Ye. CVX: Matlab software for disciplined convex programming, 2006. Available at http://www. stanford. edu/ boyd/cvx.

F. Herzog. *Strategic Portfolio Management for Long-Term Investments: An Optimal Control Approach*. PhD thesis, ETH, Zurich, 2005.

M. Åkerblad and A. Hansson. Efficient solution of second order cone program for model predictive control. *International Journal of Control*, 77(1):55–77, January 2004.

W. H. Kwon and S. Han. *Receding Horizon Control*. Springer-Verlag, 2005.

J. M. Maciejowski. *Predictive Control with Constraints*. Prentice-Hall, 2002.

S. P. Meyn. *Control Techniques for Complex Networks*. Cambridge University Press, 2006.

P. Tøndel, T. A. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. In *IEEE Conference on Decision and Control*, pages 1199–1204, 2001.

Y. Nesterov and A. Nemirovsky. *Interior-Point Polynomial Methods in Convex Programming*. SIAM, 1994.

J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 1999.

F. A. Potra and S. J. Wright. Interior-point methods. *Journal of Computational and Applied Mathematics*, 124(1-2):281–302, 2000.

W. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. J. Wiley & Sons, 2007.

J. Primbs. Dynamic hedging of basket options under proportional transaction costs using receding horizon control. *Submitted*, 2007.

C. V. Rao, S. J. Wright, and J. B. Rawlings. Application of interior point methods to model predictive control. *Journal of optimization theory and applications*, 99(3): 723–757, November 2004.

K. C. Toh, M. J. Todd, and R. H. Tütüncü. SDPT3: A matlab software package for semidefinite programming. *Optimization Methods and Software*, 11(12):545–581, 1999.

L. Vandenberghe, S. Boyd, and M. Nouralishahi. Robust linear programming and optimal control. In *15th IFAC World Congress on Automatic Control*, volume 15, July 2002.

A. G. Wills and W. P. Heath. Barrier function based model predictive control. *Automatica*, 40(8):1415–1422, August 2004.

S. J. Wright. Applying new optimization algorithms to model predictive control. *Chemical Process Control-V*, 93(316):147–155, 1997. AIChE Symposium Series.

E.A. Yildirim and S. J. Wright. Warm-start strategies in interior-point methods for linear programming. *SIAM Journal on Optimization*, 12(3):782–810, 2002.