Real-Time Optimization for Economic Model Predictive Control

Sokoler, Leo Emil; Edlund, Kristian; Frison, Gianluca; Skajaa, Anders; Jørgensen, John Bagterp

Published in:
The 10th European Workshop on Advanced Control and Diagnosis (ACD 2012)

Publication date:
2012

Citation (APA):
Real-Time Optimization for Economic Model Predictive Control

Leo Emil Sokoler∗,** Kristian Edlund** Gianluca Frison∗
Anders Skajaa∗ John Bagterp Jørgensen∗

∗ DTU Informatics, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark {leso, giaf, andsk, jbj}@imm.dtu.dk
** DONG Energy, DK-2820 Gentofte, Denmark (kried@dongenergy.dk)

Abstract: In this paper, we develop an efficient homogeneous and self-dual interior-point method for the linear programs arising in economic model predictive control. To exploit structure in the optimization problems, the algorithm employs a highly specialized Riccati iteration procedure. Simulations show that in comparison to conventional interior-point methods, our solver is a) significantly faster per. iteration and b) converges in a smaller and less fluctuating number of iterations.

1. INTRODUCTION

The main computational task in economic model predictive control (MPC) is real-time minimization of an economic objective function subject to system dynamics, input limits, input-rate limits and soft output limits. To make large-scale linear programs of this type solvable in real-time, we present a variant of Mehrotra’s predictor-corrector interior-point method (IPM) (Mehrotra, 1992; Wright, 1987; Czyzyk et al., 1999) that combines a homogeneous and self-dual model (Andersen et al., 1996; Xu et al., 1996; Ye et al., 1994; Jansen et al., 1993) with a Riccati iteration procedure (Rao et al., 1998; Jørgensen et al., 2004; Wang and Boyd, 2010; Jørgensen et al., 2012). The algorithm is tested in a small conceptual example where economic MPC is applied to minimize operational costs of a power system. In addition, we compare our algorithm against state of the art general purpose solvers and a conventional structure-exploiting IPM.

1.1 Related Work

Structure-exploiting IPMs for set-point based MPC with ℓ2-penalty have been reported in (Rao et al., 1998; Jørgensen et al., 2004; Wang and Boyd, 2010; Jørgensen et al., 2012), and similar work for ℓ1-penalty in (Edlund et al., 2009; Vandenberghhe et al., 2002). For general small and medium dimensional convex optimal control problems, custom code generators such as CVXGEN (Mattingley et al., 2010) facilitate MPC for systems with dynamics even in the kHz range. First order methods aimed at MPC for embedded systems have been described by (Be-mporad and Patrinos, 2012; Hans-Bernd and Ebenbauer, 2012; Jones et al., 2012). Another branch of emerging algorithms for fast MPC utilizes parametric programming (Alessio and Bemporad, 2009; Kvasnica and Fikar, 2010). Traditionally, such algorithms have been limited to small systems due to an exponential growth in complexity. However, work inspired by this field such as qpOASES (Potschka et al., 2010; Ferreau et al., 2008, 2012), partial enumeration (Pannocchia et al., 2007) and the multiresolution approximation method (Summers et al., 2011) can be applied in real-time to larger systems. For parallelization of the online algorithm, iterative methods such as (Hartley et al., 2012; Kerrigan et al., 2012) as well as the alternating direction multiplier method (Markus and Findeisen, 2012) have been proposed.

2. PAPER ORGANIZATION

This paper is organized as follows. In Section 3, we formulate the linear program solved in economic MPC and optimality conditions are derived. Section 4 presents a Riccati iteration procedure for specializing a homogeneous and self-dual IPM to economic MPC. A case study of economic MPC and benchmarks are provided in Section 5. We give concluding remarks in section 6.

3. PROBLEM DEFINITION

The optimization problem solved in economic MPC may be stated as

\[
\min_{u,x,y,w} \phi(u, x, w) = \sum_{k=0}^{N-1} p_k^T u_k + q_{k+1}^T y_{k+1},
\]

s.t. \( x_{k+1} = Ax_k + Bu_k, \)
\( y_{k+1} = Cx_{k+1}, \)
\( u_k \leq u_k \leq \pi_k, \)
\( \Delta u_k \leq u_k - D x_k \leq \Delta \pi_k, \)
\( y_{k+1} - w_{k+1} \leq y_{k+1} \leq \bar{y}_{k+1} + w_{k+1}, \)
\( w_{k+1} \geq 0, \)
for \( k \in \mathbb{N} := \{0, 1, \ldots, N - 1\}. \) The problem data are the state space matrices \( (A, B, C), \) the initial state \( x_0, \) the input limits \( u_k \) and \( \pi_k, \) the input-rate limits \( \Delta u_k \) and \( \Delta \pi_k, \) the output limits \( y_k \) and \( \bar{y}_k, \) the input prices \( p_k, \) the prices for violating the output limits \( q_k \) and the prediction horizon \( N. \) We have labelled the inputs as \( u_k \in \mathbb{R}^n_u, \) the states as \( x_k \in \mathbb{R}^n_x, \) the outputs as \( y_k \in \mathbb{R}^n_y \) and the soft variables as \( w_k \in \mathbb{R}^n_w. \) It is assumed that the state space system has been augmented such that \( D x_k = u_{k-1}. \)
3.1 Linear Program Formulation

In a simple form, we can write (1) as

\[
\begin{aligned}
\min_t \quad & g^T t, \\
\text{s.t.} \quad & Ft = b, \\
& Ht \leq c.
\end{aligned}
\]

E.g. for \( N = 2 \) the structures are

\[
t := [u_0^T x^T u_0^T u_1^T x_2^T w^T]^T,
\]

\[
g := [p_0^T 0 q_1^T p_1^T 0 q_2^T]^T,
\]

and

\[
[F][b] := \\
\begin{bmatrix}
B & -I & 0 & 0 & 0 & 0 & -Ax_0 \\
0 & A & B & -I & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
I & 0 & 0 & 0 & 0 & -u_0 \\
0 & 0 & 0 & -I & 0 & 0 \\
I & 0 & 0 & 0 & 0 & -\Delta \tilde{u}_0 \\
0 & -D & 0 & I & 0 & 0 \\
0 & D & 0 & I & 0 & 0 \\
0 & C & -I & 0 & 0 & -\Delta \tilde{u}_1 \\
0 & 0 & 0 & C & -I & 0 \\
0 & 0 & 0 & -C & -I & 0 \\
0 & 0 & 0 & 0 & 0 & -I \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}
\]

\[
[H][c] := \\
\begin{bmatrix}
\Delta \tilde{u}_0 := \Delta \tilde{u}_0 + Dx_0, \\
\Delta \tilde{u}_0 := \Delta \tilde{u}_0 + Dx_0.
\end{bmatrix}
\]

The optimization problem is a highly structured linear program with \( n := N(n_u + n_s + n_y) \) variables, \( m_E := N n_x \) equality constraints and \( m_I := N (4n_u + 3n_y) \) inequality constraints.

3.2 Optimality Conditions

The Lagrangian for (2) is given by

\[
\mathcal{L}(t, p, z) := g^T t - p^T (b - Ft) - z^T (c - Ht).
\]

Hence, the first order necessary and sufficient optimality conditions may be stated as

\[
\begin{aligned}
g + F^T p + H^T z &= 0, \\
b - Ft &= 0, \\
c - Ht - s &= 0, \\
z_i s_i &= 0, \\
z_i s_i &= 0
\end{aligned}
\]

in which we have introduced the dual variables \( p \in \mathbb{R}^{m_E} \) and \( z \in \mathbb{R}^{m_I} \), and the slack variables \( s := c - Ht \). Now define \( Z \) as the diagonal matrix with diagonal elements \( z_1, \ldots, z_{m_I} \) and similarly for \( S \). Let \( 1_{m_I} \) be the vector of all ones of size \( m_I \). We can then state the optimality conditions as

\[
V(t, p, z, s) := \\
\begin{bmatrix}
g + F^T p + H^T z \\
b - Ft \\
c - Ht - s \\
ZS 1_{m_I}
\end{bmatrix} = \\
\begin{bmatrix}
0 \\
0 \\
0 \\
0
\end{bmatrix},
\]

subject to \((z, s) \geq 0\).
Here $e_E$, $e_I$, $c_D$ and $c_G$ are small user-defined tolerances representing accuracy in primal and dual feasibility, as well as accuracy in the duality gap.

**Algorithm 1** Interior-point algorithm for (2)

**Require:**
- DATA $(y, F, b, H, c)$
- INITIAL POINT $(t, p, z, s)$
- PARAMETERS $\nu$

```plaintext
// initialize
\mu \leftarrow \frac{z^T s}{m_1}

while CONVERGED do
    // affine step
    \Delta t \leftarrow -J_Y(t, p, z, s)^{-1}V(t^k, p^k, z^k, s^k)
    \Delta \alpha \leftarrow \max \{ \bar{a} \in [0, 1] | z + \bar{a} \Delta \bar{z} \geq 0 \}
    \Delta \beta \leftarrow \max \{ \bar{b} \in [0, 1] | s + b \Delta s \geq 0 \}
    \Delta \bar{s} \leftarrow s + \bar{a} \Delta \bar{s}
    \Delta \bar{z} \leftarrow z + \bar{\beta} \Delta \bar{z}
    \Delta \bar{\mu} \leftarrow \frac{\Delta \bar{s}}{m_1}
    \Delta \bar{z} \leftarrow \Delta \bar{s}/m_1
    \Delta \bar{\mu} \leftarrow \Delta \bar{s}/m_1
```

**4.1 Riccati Iteration Procedure**

The main operations in Algorithm 1 consist of solving linear systems in the form

\[ \begin{bmatrix} 0 & F^T & H^T & 0 \\ -F & 0 & 0 & 0 \\ -H & 0 & 0 & -I \\ 0 & 0 & S & Z \end{bmatrix} \begin{bmatrix} \Delta t \\ \Delta p \\ \Delta s \end{bmatrix} = \begin{bmatrix} r_D \\ r_E \\ r_f \end{bmatrix} \]  

By eliminating $\Delta s$ using the last row, we obtain the reduced system

\[ \begin{bmatrix} 0 & F^T & H^T \\ -F & 0 & 0 \\ -H & 0 & -Z^{-1} S \end{bmatrix} \begin{bmatrix} \Delta t \\ \Delta p \\ \Delta z \end{bmatrix} = \begin{bmatrix} r_D \\ r_E \end{bmatrix}, \]  

in which $r_f$ := $r_f + Z^{-1} r_C$. We now show that this system can be solved efficiently for the optimization problem (1). The Lagrange multipliers associated with the inequality constraints (1d-lg) are labelled $\Delta \eta$, $\Delta \lambda$, $\Delta \nu$, $\Delta \omega$, $\Delta \gamma$, $\Delta \rho$ and $\Delta \xi$ where

\[ \Delta \eta := \left[ \Delta \eta_0^T \Delta \eta_1^T \ldots \Delta \eta_{N-1}^T \right]^T \]

and similarly for $\Delta \lambda$, $\Delta \nu$, $\Delta \omega$, $\Delta \gamma$, $\Delta \rho$ and $\Delta \xi$. Here $\Delta \eta$ and $\Delta \lambda$ are multipliers for the input limits (1d), $\Delta \nu$ and $\Delta \omega$ are multipliers for the input-rate limits (1e), $\Delta \gamma$ and $\Delta \rho$ are multipliers for the output limits (1f) and $\Delta \xi$ is the vector of multipliers for the non-negative constraints (1g). Based on this notation, the optimization variables in the system (7) can be expressed as

\[ \Delta t = \left[ \Delta u_0^T \Delta x_1^T \Delta u_1^T \ldots \Delta u_{N-1}^T \Delta x_N^T \Delta w_N^T \right]^T, \]

\[ \Delta \rho = \left[ \Delta p_0^T \Delta p_1^T \ldots \Delta p_{N-1}^T \right]^T, \]

\[ \Delta \xi = \left[ \Delta \eta^T \Delta \lambda^T \Delta \nu^T \Delta \omega^T \Delta \gamma^T \Delta \rho^T \Delta \xi^T \right]^T. \]

Accordingly, we partition the right hand side such that

\[ r_D = \left[ \begin{bmatrix} r_D^T & r_D^T & 0 & 0 \end{bmatrix} \right]^T, \]

\[ r_E = \left[ \begin{bmatrix} r_E^T & r_E^T & 0 & 0 \end{bmatrix} \right]^T, \]

and write the diagonal matrix $Z^{-1} S$ in terms of diagonal submatrices

\[ Z^{-1} S = \text{diag} \left( \Sigma_{\eta, \eta}, \Sigma_{\lambda, \lambda}, \Sigma_{\nu, \nu}, \Sigma_{\omega, \omega}, \Sigma_{\gamma, \gamma}, \Sigma_{\rho, \rho}, \Sigma_{\xi, \xi} \right). \]

The linear system of equations (7) may now be stated in the form

\[ \Delta \eta_\alpha - \Delta \lambda_i + \Delta v_i - \Delta \omega_i + B^T \Delta p_i = r_{u,i}, \quad i \in N, \]

\[ -\Delta u_i + \Delta \eta_i + \Delta \nu_i = r_{\nu,i}, \quad i \in N, \]

\[ \Delta u_i + \lambda_i + \Delta \gamma_i = r_{\lambda,i}, \quad i \in N, \]

\[ -\Delta u_i + D \Delta x_i + \Delta \omega_i = r_{\omega,i}, \quad i \in N, \]

\[ -\Delta u_i - \Delta \lambda_i + \Delta x_i = r_{\lambda,i}, \quad i \in N, \]

\[ \Delta w_{i+1} - \Delta \xi_{i+1} - \Delta \xi_{i+1} = r_{\xi,i+1}; \]

\[ \Delta w_{i+1} + \Delta \xi_{i+1} - \Delta \xi_{i+1} = r_{\xi,i+1}, \quad i \in N, \]

\[ -\Delta v_i + C^T (\Delta \gamma_{i+1} - \Delta \rho_{i+1}) + A^T \Delta p_i + D^T (\Delta u_i - \Delta \gamma_{i+1}) = r_{x,i}, \quad i \in N, \]

with $N := N \setminus 0$, and the special cases

\[ -\Delta u_0 + \Delta \xi_0 = r_{u,0}, \]

\[ \Delta w_0 - \Delta \xi_0 = r_{\xi,0}, \]

\[ -\Delta v_0 + \Delta \omega_0 = r_{\omega,0}, \]

\[ -\Delta v_0 - \Delta \omega_0 = r_{\omega,0}, \]

\[ -\Delta p_{N-1} + C^T (\Delta \gamma_N - \Delta \rho_N) = r_{x,N}. \]

By eliminating the Lagrange multipliers for the inequality constraints $\Delta \eta$, $\Delta \lambda$, $\Delta \nu$, $\Delta \omega$, $\Delta \gamma$, $\Delta \rho$ and $\Delta \xi$ we get the reduced set of equations

\[ B^T \Delta p_0 + U_0 \Delta u_0 = R_{u,0}, \]

\[ B^T \Delta p_i + U_i \Delta u_i + G_i \Delta x_i = R_{u,i}, \quad i \in N, \]

\[ -\Delta r_i + B^T \Delta u_0 = R_{r,0}, \]

\[ -\Delta r_i + D \Delta x_i + B \Delta u_i = R_{r,i}, \quad i \in N, \]

\[ W_{i+1} \Delta w_{i+1} + M_{i+1} \Delta x_{i+1} = R_{x,i+1}, \quad i \in N, \]

\[ -\Delta p_{i-1} + M_i \Delta w_i + X_{i} \Delta x_i = G_i \Delta u_i + A^T \Delta p_i = R_{x,i}, \quad i \in N, \]

\[ -\Delta p_{N-1} + M_N \Delta w_{N-1} + X_N \Delta x_N = R_{x,N}. \]

Here we have defined

\[ U_i := \Sigma_{\eta,i} + \Sigma_{\lambda,i} + \Sigma_{\nu,i} + \Sigma_{\omega,i}, \quad i \in N, \]

\[ W_{i+1} := \Sigma_{\rho,i,1} + \Sigma_{\xi,i,1} + \Sigma_{\omega,i,1}, \quad i \in N, \]

\[ X_i := C^T (\Sigma_{\gamma,i} + \Sigma_{\rho,i}) D + D^T (\Sigma_{\gamma,i} + \Sigma_{\rho,i}) D, \quad i \in N, \]

\[ X_N := C^T (\Sigma_{\gamma,N} + \Sigma_{\rho,N}) D, \quad i \in N, \]

\[ M_{i+1} := C^T (\Sigma_{\gamma,i+1} - \Sigma_{\rho,i+1}), \quad i \in N, \]

and
\[\begin{align*}
R_{u,i} &:= r_{u,i} + \sum_{\lambda=1}^{n-1} r_{\lambda,i} + \sum_{\nu=1}^{n-1} r_{\nu,i} - \sum_{\nu=1}^{n-1} r_{\nu,i} + \sum_{\nu=1}^{n-1} r_{\nu,i}, \quad i \in \mathcal{N}, \\
R_{p,i} &:= -r_{p,i}, \quad i \in \mathcal{N}, \\
R_{w,i+1} &:= r_{w,i} + \sum_{\nu=1}^{n-1} r_{\nu,i} + \sum_{\nu=1}^{n-1} r_{\nu,i} + \sum_{\nu=1}^{n-1} r_{\nu,i} + \sum_{\nu=1}^{n-1} r_{\nu,i}, \quad i \in \mathcal{N}, \\
R_{x,i} &:= r_{x,i} + \sum_{\nu=1}^{n-1} r_{\nu,i} + \sum_{\nu=1}^{n-1} r_{\nu,i} + \sum_{\nu=1}^{n-1} r_{\nu,i} + \sum_{\nu=1}^{n-1} r_{\nu,i}, \quad i \in \mathcal{N}, \\
R_{x,N} &:= r_{x,N} + \sum_{\nu=1}^{n-1} r_{\nu,N} + \sum_{\nu=1}^{n-1} r_{\nu,N} + \sum_{\nu=1}^{n-1} r_{\nu,N} + \sum_{\nu=1}^{n-1} r_{\nu,N}.
\end{align*}\]

Solving (8e) for \(\Delta w\) gives

\[\Delta w_{i+1} = W_{i+1}^{-1}(R_{w,i+1} - M_{i+1}^{T} \Delta x_{i+1}), \quad i \in \mathcal{N}. \quad (9)\]

Substituting back into (8) results in the equations

\[\begin{align*}
B^{T} \Delta p_{0} + U_{0} \Delta u_{0} &= R_{u,0}, \\
B^{T} \Delta p_{i} + U_{i} \Delta u_{i} + G_{i} \Delta x_{i} &= R_{u,i}, \quad i \in \mathcal{N}, \\
&-\Delta x_{i+1} + B_{u} \Delta u_{i} = R_{p,i}, \\
&-\Delta x_{i+1} + \Delta x_{i} + B_{u} \Delta u_{i} = R_{p,i}, \\
&-\Delta p_{i-1} + X_{i} \Delta x_{i} + A_{i} \Delta p_{i} = R_{x,i}, \quad i \in \mathcal{N}, \\
&-\Delta p_{N-1} + X_{N} \Delta x_{N} = R_{x,N},
\end{align*}\]

where

\[\begin{align*}
X_{i+1} &:= \dot{X}_{i+1} = M_{i+1}^{-1} W_{i+1}^{-1} \Delta x_{i+1}, \quad i \in \mathcal{N}, \\
R_{x,i+1} &:= \dot{R}_{x,i+1} = M_{i+1}^{-1} W_{i+1}^{-1} \Delta x_{i+1}, \quad i \in \mathcal{N}.
\end{align*}\]

As described in (Rao et al., 1998; Jørgensen et al., 2004; Wang and Boyd, 2010; Czyzyk et al., 1999), the equations above can be solved efficiently by a Riccati iteration procedure. With this approach the overall complexity of Algorithm 1 becomes of order \(O(N(n_{u}^2 + n_{v}^2 + n_{y}^2))\) per iteration. In comparison, solving the system (6) by a general purpose method has order of complexity \(O(N^3(n_{u} + n_{v} + n_{y}))\). Thus, the computational cost per iteration is reduced by orders of magnitude in \(N\). Furthermore, a considerable improvement for systems of growing dimensions \((n_{u}, n_{v}, n_{y})\) is obtained as well. Therefore, a Riccati-based IPM is expected to run significantly faster than a conventional method and scale in a favourable way for both increasing system size and particularly for an increasing prediction horizon.

### 4.2 Homogeneous and Self-Dual Model

A drawback related to infeasible primal-dual IPMs such as Algorithm 1, is that the number of iterations can be very sensitive to the choice of initial point (Andersen et al., 1996). For MPC applications this presents a problem since reliability of the online solver is critical. A possible way to overcome the issue is to employ the homogeneous and self-dual model described in (Andersen et al., 1996; Xu et al., 1996; Ye et al., 1994; Jansen et al., 1993). Other than making it simple to find a suitable initial point, this approach facilitates easy detection of infeasibility, as well as strategies for warm starting (Skajaa et al., 2012). In the following, we show that the Riccati iteration procedure described above can be used for IPMs based on the homogeneous and self-dual model as well. For this class of algorithms, the optimality conditions are
\[
\begin{bmatrix}
0 & F^T & H^T \\
-F & 0 & 0 \\
-H & 0 & Z^{-1}S
\end{bmatrix}
\begin{bmatrix}
h_1 \\
h_2 \\
h_3
\end{bmatrix}
= \begin{bmatrix}
-g \\
-h \\
-c
\end{bmatrix},
\]  
(16)
such that a set of auxiliary variables \( f \) and \( h \) satisfy
\[
\begin{bmatrix}
\Delta t \\
\Delta p \\
\Delta z
\end{bmatrix}
= \begin{bmatrix}
f_1 \\
f_2 \\
f_3
\end{bmatrix}
+ \begin{bmatrix}
h_1 \\
h_2 \\
h_3
\end{bmatrix}
\Delta \tau,
\]  
(17)
which implies
\[
\begin{bmatrix}
g^T \Delta t \\
b^T \Delta p \\
c^T \Delta z
\end{bmatrix}
= \begin{bmatrix}
g^T f_1 \\
b^T f_2 \\
c^T f_3
\end{bmatrix}
+ \begin{bmatrix}
g^T h_1 \\
b^T h_2 \\
c^T h_3
\end{bmatrix}
\Delta \tau.
\]
Substituting into (14) yields
\[
\Delta \tau = \frac{\dot{\rho}_M - \tau (g^T f_1 + b^T f_2 + c^T f_3)}{\kappa + \tau (g^T h_1 + b^T h_2 + c^T h_3)}.
\]
Having determined \( f \), \( g \) and \( \Delta \tau \), the remaining variables can be recovered from (12) and (17). The major operations involved in solving (10) are therefore reduced to computing \( f \) and \( g \) from (15-16). This can be done efficiently by the Riccati iteration procedure described in Section 4.1. To determine both the affine step and the predictor-corrector step, 4 linear systems have to be solved in each iteration of our homogeneous and self-dual variant of Algorithm 1. In comparison, conventional IPMs only require solving 2 linear systems per iteration. Since the system matrix is constant in each iteration however, all the major computations are only involved in solving the first system. We therefore expect the additional cost per iteration in the homogeneous and self-dual algorithm to be insignificant compared to the overall reduction in the number of iterations. To detect if a solution is optimal or infeasible, we use the following measures (Andersen et al., 2003)
\[
\begin{align*}
\rho_E &= \frac{||b^T - Ft||_\infty}{\max(1, ||F||_\infty)}, \\
\rho_I &= \frac{||c^T - Ht - s||_\infty}{\max(1, ||H||_\infty)}, \\
\rho_D &= \frac{||g^T + Ftp + Ht^2z||_\infty}{\max(1, ||H^T||_\infty)}, \\
\rho_G &= \frac{||g^T t - (-b^T p - c^T z) - k_i||_\infty}{\max(1, ||g^T b^T c^T||_\infty)}, \\
\rho_O &= \frac{||g^T t - (-b^T p - c^T z) - \kappa_i||_\infty}{\tau + ||-b^T p - c^T z||_\infty}.
\end{align*}
\]
An iterate \((t^k, p^k, z^k, s^k, t^k, s^k)\) is classified as optimal if
\[
\rho_E^k \leq \epsilon_E, \quad \rho_I^k \leq \epsilon_I, \quad \rho_D^k \leq \epsilon_D, \quad \rho_O^k \leq \epsilon_O,
\]
and infeasible if \(\tau^k \leq \epsilon_T, \max(1, \kappa^k)\) and
\[
\rho_E^k \leq \epsilon_E, \quad \rho_I^k \leq \epsilon_I, \quad \rho_D^k \leq \epsilon_D, \quad \rho_O^k \leq \epsilon_O.
\]
Again \(\epsilon_T, \epsilon_E, \epsilon_I, \epsilon_D, \epsilon_O\) and \(\epsilon_G\) are small user-defined tolerances.

### 4.3 Special Operators

To avoid forming \( F \) and \( G \) explicitly, operations involving these matrices are implemented as special operations. For \( N = 2 \) the optimization variables may be written as
\[
t = [u_0^T u_1^T w_1^T u_1^T x_2^T w_2^T]^T,
\]
\[
p = [p_1^T p_2^T]^T,
\]
\[
z = [\eta_0^T \eta_1^T \lambda_1^T \lambda_1^T v_1^T v_1^T]^T.
\]
In this case, the special operations are
\[
F^T p = [p_1^T p_2^T A - p_1^T 0 p_2^T B - p_2^T 0]^T,
\]
\[
H^T z = \begin{bmatrix}
\eta_0 - \lambda_0 + v_0 - \omega_0 \\
\lambda_1 + v_1 - \omega_1 \\
C^T(\gamma_2 - \rho_2) \\
-\gamma_2 - \rho_2 - \eta_2
\end{bmatrix},
\]
and
\[
H t = \begin{bmatrix}
[u_0^T - u_0^T (u_1^T - Dx_1)^T - u_0^T (Dx_1 - u_1)^T] \\
(Cx_1 - w_1)^T (Cx_2 - w_2)^T - (Cx_1 - w_1)^T \\
(-Cx_2 - w_2)^T - \omega t)^T \\
((Bu_0 - x_1)^T (Ax_1 + Bu_1 - x_2)^T)^T
\end{bmatrix}.
\]
Multiplications involving \( F \) and \( G \) can thus be implemented very cheaply.

### 5. RESULTS

In this section, economic MPC is applied to a simple power system. Moreover, the algorithm developed in this paper is compared to state of the art general purpose solvers. For this purpose, we introduce a mass-spring system which has been previously used in e.g. (Shahzad et al., 2010; Wang and Boyd, 2010) for evaluating performance of IPMs.

#### 5.1 Case Study - Energy System

To illustrate economic MPC of energy systems, we consider a collection of power plants in the form
\[
Y_i(s) = \frac{1}{(\tau_i s + 1)^3} U_i(s), \quad i = 1, 2, \ldots, M,
\]
where \(U_i(s)\) is the units of fuel supplied to power plant \(i\), and \(Y_i(s)\) is power produced by power plant \(i\). This third order model has been validated against measurement data in (Edlund et al., 2010). The total power production is given by \(Y_T(s) := \sum_{i=1}^M Y_i(s)\). In the following simulation, 4 units with different operational features are controlled. We have listed the controller parameters in Table 1. The objective is to keep the total power production within a certain range, while minimizing input costs. The output limits are time varying and therefore not included in the table. To formulate a controller based on (1), the system is realized in a discrete state space form. We use a sampling time of \(T_s = 5\) seconds, which is adequate to capture the system dynamics. The optimal open-loop solution for a \(N = 120\) time step prediction horizon is depicted in Fig. 1. As desired, the total power generation tracks a predefined interval. The cheapest plant accounts for a majority of the load, whereas more expensive and flexible units are used only whenever faster dynamics are required.
5.2 Benchmark - Mass-Spring System

To evaluate the performance of our numerical algorithm, we set up and solve a number of growing economic MPC problems. The study is performed on an Intel(R) Core(TM) i5-2520M CPU @ 2.50GHz with 4 GB RAM running a 64-bit Ubuntu 12.04.1 LTS operating system.

For comparison, we use the following solvers:

- **SeDuMi**: Software package written in MATLAB and C for solving optimization problems over symmetric cones. The default solver is an implementation of a homogeneous and self-dual IPM (Sturm, 1999).
- **LIPSOL**: IPM for large-scale linear programming available through MATLAB’s `linprog` interface (Zhang, 1996).
- **LPRsol**: Riccati-based MATLAB implementation of Algorithm 1 (the Riccati iteration procedure is implemented in a separate MEX-file).
- **LPRHsol**: Riccati-based MATLAB implementation of the homogeneous and self-dual variant of Algorithm 1 (the Riccati iteration procedure is implemented in a separate MEX-file).

Our parameter setting for LPRsol and LPRHsol is listed in Table 2. The initial point is \((p^0, q^0, s^0, z^0, \tau^0, \kappa^0) = (0.995, 10^{-8}, 10^{-8}, 10^{-8}, 10^{-8}, 10^{-10})\).

### Table 1. The generators have limited input and input-rate. Fuel costs increase inversely with the time constants (flexibility is expensive).

<table>
<thead>
<tr>
<th>Plant #1</th>
<th>Plant #2</th>
<th>Plant #3</th>
<th>Plant #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\tau)</td>
<td>60</td>
<td>40</td>
<td>20</td>
</tr>
<tr>
<td>(p_k)</td>
<td>1</td>
<td>2.5</td>
<td>5</td>
</tr>
<tr>
<td>(q_k)</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>(u_k)</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\Delta u_k)</td>
<td>-7.5</td>
<td>-5</td>
<td>-2.5</td>
</tr>
<tr>
<td>(\Delta p_k)</td>
<td>7.5</td>
<td>5</td>
<td>2.5</td>
</tr>
</tbody>
</table>

### Table 2. Algorithmic parameter settings for large-scale benchmark.

<table>
<thead>
<tr>
<th>(\eta)</th>
<th>(\epsilon_k)</th>
<th>(\epsilon_t)</th>
<th>(\epsilon_d)</th>
<th>(\epsilon_G)</th>
<th>(\epsilon_O)</th>
<th>(\epsilon_p)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.995</td>
<td>10^{-8}</td>
<td>10^{-8}</td>
<td>10^{-8}</td>
<td>10^{-8}</td>
<td>10^{-8}</td>
<td>10^{-10}</td>
</tr>
</tbody>
</table>

### Table 3. Economic MPC parameters for mass-spring system.

<table>
<thead>
<tr>
<th>(p_k)</th>
<th>(q_k)</th>
<th>(u_k)</th>
<th>(y_k)</th>
<th>(\Delta u_k)</th>
<th>(\Delta y_k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>(100)</td>
<td>(-\frac{1}{20})</td>
<td>(-\frac{1}{20})</td>
<td>(0)</td>
<td>(\frac{1}{3})</td>
</tr>
<tr>
<td>(1)</td>
<td>(100)</td>
<td>(-\frac{1}{20})</td>
<td>(-\frac{1}{20})</td>
<td>(0)</td>
<td>(\frac{1}{3})</td>
</tr>
</tbody>
</table>

Fig. 1. Open-loop simulation of economic MPC applied to a simple power system.

Fig. 2. CPU timings for economic MPC of mass-spring system.

Fig. 3. CPU timings for economic MPC of mass-spring system.

\((0_n, 0_m, \eta_m, \eta_n, 1, 1, 1)\). For SeDuMi and LIPSOL the default tolerance level and parameter setting is used. It has been verified that this approximately gives the same accuracy in the solution. The system used for benchmarks is a mass-spring system (Shahzad et al., 2010; Wang and Boyd, 2010). The system consists of \(n_m\) 1 kg masses connected by springs, and walls at the end. No damping is assumed and the spring constant is 1 N/m. Manipulable actuators are attached to each of the first \(n_m\) masses. The objective of our controller is to keep the individual mass displacement within certain bounds, at a minimum cost. Initially, the mass displacement for all masses are 1 m. Costs are imposed for using the actuators and violating the displacement bounds. The actuators have a limited force and a limited rate of change. We have summarized the controller setup for \(n_m = 3\) and \(n_a = 2\) in Table 3. The parameters are held constant for all time steps. In Fig. 2, we have depicted computational results for problems of growing dimension. The figure shows that the structure-exploiting methods are about an order of magnitude faster than SeDuMi and LIPSOL. This difference increases as the problem size grows. Also notice that our homogeneous and self-dual IPM LPRHsol, outperforms the conventional IPM LPRsol. As illustrated in Fig. 3, this is due to its ability to consistently maintain a relatively low number of iterations. The same iteration pattern applies for SeDuMi, which is also based on the homogeneous and self-dual model.

6. CONCLUSIONS

In this paper, we have developed a structure-exploiting IPM, which combines a homogeneous and self-dual model with a highly specialized Riccati iteration procedure. Com-
Fig. 3. Iteration counts for economic MPC of mass-spring system.

pared to general purpose solvers, the complexity of our algorithm is reduced by two orders of magnitude in the prediction horizon $N$, and a considerable improvement for growing system dimensions is obtained as well. Furthermore, the algorithm facilitates warm-starting and have a less fluctuating iteration pattern compared to conventional IPMs. Altogether, our solver is therefore well suited to handle real-time optimization of the structured linear programs which arise in economic MPC applications.

REFERENCES


