Proceedings of the 18th World Congress
The International Federation of Automatic Control
Milano (Italy) August 28 - September 2, 2011

A Fast Gradient method for embedded linear predictive control

Markus Kögel [*,1] **Rolf Findeisen** [*]

[*] *Institute for Automation Engineering, Otto-von-Guericke-University Magdeburg, Germany. e-mail: {markus.koegel, rolf.findeisen}@ovgu.de.*

Abstract: This work considers the fast solution of model predictive control problems for linear systems with input constraints and a quadratic cost criterion. If the resulting optimization problem arising from the model predictive control is solved online using the Fast Gradient method one needs to determine the gradient of the cost function. We propose a method, tailored for embedded control purposes, that efficiently calculates the gradient taking the underlying structure of the system into account. Moreover, we discuss how the stability of the plant influences the required number of iterations to obtain a solution within a prescribed accuracy.

Keywords: Model predictive control, Fast Gradient method, Computational methods, Online optimization, Embedded Systems.

1. INTRODUCTION

Predictive control, also denoted model predictive control (MPC), is a modern technique frequently used to control systems subject to constraints, see Maciejowski (2002); Qin and Badgwell (2003); Garcia et al. (1989). In predictive control each time a new measurement becomes available the input is determined by solving a finite horizon optimal control problem. Predictive control allows to control systems with a high performance and such that constraints are satisfied. However solving the required optimization problem is computational challenging. Therefore, in this paper we derive a method to reduce the computational effort for linear, discrete time, time-invariant systems subject to input constraints and a quadratic cost criterion.

There are two main approaches to solve the quadratic programs appearing in predictive control: online and offline. In the so-called online-optimization the problem is solved online in each step. The computational demand depends on the problem is challenging, so online-optimization is usually limited by the available computation speed. In the latter approach called explicit MPC the solution for all possible states is calculated offline and stored in for example a table, see Bemporad et al. (2002). Since, this table might grow exponentially in the number of states, explicit solution is often limited by its memory demand and restricted to small-scale systems.

Therefore we present in this paper an online-optimization method tailored for MPC. With respect to the efficient online solution of such predictive control problems many tailored approaches exist by now. Rao et al. (1998) and Wang and Boyd (2008) consider interior-points method taking the special structure of the problem into account. Moreover, Shahzad et al. (2010) discusses inexact interior points methods. Milman and Davison (2008); Ferreau et al. (2008) consider tailored fast active set methods for predictive control. In theory the worst case computational effort of the algorithm increases exponential in the number of constraints. In practice they are quite efficient and can be used e.g. to control diesel engines using sampling times down to milliseconds, see Ferreau et al. (2007).

Richter et al. (2009, 2010) report an online-optimization for systems with input constraints using Nesterov's method or so-called Fast Gradient method , see Nesterov (1983), and in addition pre-conditioning or warm-starting. The ideas utilized in this paper are along similar lines.

Since the effort of the considered Fast Gradient method is governed by the effort to determine the gradient, we show how to determine the gradient efficiently exploiting the underlying structure of the MPC problem. The memory demand and the computation time of the gradient of the proposed method increases linearly in the number of inputs, states and the horizon length. In contrast, the memory demand and the computation time of the gradient the standard method is quadratically increasing in the number of inputs and the horizon length. Moreover we discuss the influence the plant stability onto the required number of iterations.

The remainder of the paper is structured as follows. First we will discuss the problem setup and review Nesterov's method in Section 2. Section 3 presents the main result of this paper: an efficient way to determine the gradient. In Section 4, we analyze the computational effort of this method. Section 5 illustrates the results by an example. Finally, we discuss the results.

2. PROBLEM STATEMENT

In this work, we consider linear, time-invariant, discrete-time plants given by

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k, \qquad \mathbf{x}_0 = \mathbf{x}^0, \tag{1}$$

with the state $\mathbf{x}_k \in \mathbb{R}^n$, input $\mathbf{u}_k \in \mathbb{R}^p$ and the system matrices $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, where (A, B) is stabilizable.

[1] The researcher was supported in part by the International Max Plank Research School, Magdeburg, Germany.

In addition, the input $\mathbf{u}_k$ is constrained in each step to a closed, convex set $\mathcal{U}$ containing the origin. In this paper, we focus on so-called box-constraints

$$\mathcal{U} = \{l[i] \leq u_k[i] \leq r[i], \forall i = 1, \ldots, p\},$$

where $u_k[i]$ denotes the ith entry of $\mathbf{u}_k$. Note that the results can easily be extended to more general constraints such as a simple polytopes, spheres, see Nesterov (2004). Let us define the set U as

$$U = \{\mathbf{u}_k \in \mathcal{U}, k \geq 0\}. \tag{2}$$

We want to control the system (1) satisfying constraints (2) subject to minimizing the quadratic cost criterion

$$\begin{aligned} J = &\sum_{i=0}^{N-1} \frac{1}{2}\mathbf{x}_{i+k}^T Q \mathbf{x}_{i+k} + \mathbf{u}_{i+k}^T S \mathbf{x}_{i+k} \\ &+ \sum_{i=0}^{N-1} \frac{1}{2}\mathbf{u}_{i+k}^T R \mathbf{u}_{i+k} + \frac{1}{2}\mathbf{x}_{N+k}^T P \mathbf{x}_{N+k}, \end{aligned} \tag{3}$$

with the prediction and control horizon $N \geq 1$ and the weighting matrices $P \in \mathbb{R}^{n\times n}$, $Q \in \mathbb{R}^{n\times n}$, $S \in \mathbb{R}^{p\times n}$ and $R \in \mathbb{R}^{p\times p}$ chosen such that

$$\begin{aligned} &Q = Q^T \geq 0,\ R = R^T > 0,\ P = P^T \geq 0, P \neq 0, \\ &\begin{pmatrix} Q & S^T \\ S & cR \end{pmatrix} \geq 0,\ c < 1, \end{aligned}$$

and $(A, Q^{\frac{1}{2}})$ is detectable. As usual in predictive control, the applied input is given as the first part of the optimal input resulting from the quadratic program

$$\begin{aligned} \underline{\mathbf{v}} = \arg\min_{\underline{\mathbf{z}},\underline{\mathbf{v}}}\ & J(\underline{\mathbf{z}}, \underline{\mathbf{v}}) \\ s.t.\quad & \mathbf{z}_{i+1} = A\mathbf{z}_i + B\mathbf{v}_i \\ & i = 0, \ldots, N-1 \\ & \mathbf{z}_0 = \mathbf{x}_k \\ & \underline{\mathbf{v}} \in U. \end{aligned} \tag{4}$$

where

$$\underline{\mathbf{z}} = \begin{pmatrix} \mathbf{z}_1 \\ \ldots \\ \mathbf{z}_N \end{pmatrix} \in \mathbb{R}^{Nn},\ \underline{\mathbf{v}} = \begin{pmatrix} \mathbf{v}_0 \\ \ldots \\ \mathbf{v}_{N-1} \end{pmatrix} \in \mathbb{R}^{Np}$$

denotes the predicted state trajectory $\underline{\mathbf{z}}$ and the input sequence $\underline{\mathbf{v}}$ to be determined using the measured state $\mathbf{x}_k$. So, the first part of the optimal input is here $\mathbf{u}_k = \mathbf{v}_0$.

There are different methods to solve the optimization problem (4), which has $N(n+p)$ optimization variables, $2Np$ inequality constraints and Nn linear equality constraints. If we eliminate the linear equality constraints we obtain the *condensed* problem

$$\begin{aligned} \underline{\mathbf{v}} = \arg\min_{\underline{\mathbf{v}}}\ & \left(\frac{1}{2}\underline{\mathbf{v}}^T H \underline{\mathbf{v}} + \underline{\mathbf{v}}^T F \mathbf{x}_k\right) \\ s.t.\quad & \underline{\mathbf{v}} \in U, \end{aligned} \tag{5}$$

where $H = H^T \in \mathbb{R}^{pN\times pN}$ and $F \in \mathbb{R}^{pN\times n}$ are matrices which depend on the system dynamics (1) and the cost criterion (3). The condensed problem has still $2Np$ inequality constraints, but only Np optimization variables.

The method utilized here is based on Nesterov's method also known as the Fast Gradient method, see Nesterov (1983, 2004). We recap now this method as well as the required background, gradient projection.

If we consider the unconstrained optimization problem

$$\underline{\mathbf{v}} = \arg\min_{\underline{\mathbf{v}}} \left(\frac{1}{2}\underline{\mathbf{v}}^T H \underline{\mathbf{v}} + \underline{\mathbf{v}}^T F \mathbf{x}_k\right), \tag{6}$$

then this problem can be solved using the well-known gradient descent method, cf. Boyd and Vandenberghe (2004). We determine a solution to (6) starting from an initial guess $\underline{\mathbf{v}}^0$ and using the iteration

$$\underline{\mathbf{v}}^{i+1} = \underline{\mathbf{v}}^i - h^i \nabla J(\underline{\mathbf{v}}), \tag{7}$$

where h^i is the step size and $\nabla J(\underline{\mathbf{v}})$ the gradient given by

$$\nabla J(\underline{\mathbf{v}}) = \begin{pmatrix} \frac{\partial}{\partial \mathbf{v}_0} \\ \vdots \\ \frac{\partial}{\partial \mathbf{v}_{N-1}} \end{pmatrix} J(\underline{\mathbf{v}}) = H\underline{\mathbf{v}} + F\mathbf{x}_k. \tag{8}$$

We choose the step-size h^i such that $J(\underline{\mathbf{v}}^{i+1}) < J(\underline{\mathbf{v}}^i)$ and to guarantee convergence. Here using e.g. $h^i = \frac{1}{L}$ is possible due the convexity of the problem, where L denotes the maximum eigenvalue of H.

The constrained problem can be solved in a similar fashion using gradient projection, see Nesterov (2004). In particular we use the projected gradient step $P_U(\underline{\mathbf{v}}, h)$ instead of the gradient step (7) i.e.

$$\begin{aligned} P_U(\underline{\mathbf{v}}, h) = &\arg\min_{\underline{\mathbf{q}}\in U} \left\|\underline{\mathbf{q}} - \underline{\mathbf{w}}\right\|_2^2 \\ &\underline{\mathbf{w}} = \underline{\mathbf{v}} - h\nabla J(\underline{\mathbf{v}}). \end{aligned} \tag{9}$$

Note that, the projected gradient step $P_U(\underline{\mathbf{v}}, h)$ is an Euclidean projection of $\underline{\mathbf{w}}$ resulting from the gradient step (7) into the feasible set U. This is equal, by definition of the set U, to projections of $\mathbf{v}_j$, $j = 0, \ldots, N-1$ onto $\mathcal{U}$

$$\begin{aligned} P_U(\underline{\mathbf{v}}, h) = &\underline{\mathbf{y}} \\ &y_j = \arg\min_{\mathbf{q}_j\in U} \|\mathbf{q}_j - \mathbf{w}_j\|_2^2, j = 0, \ldots, N-1 \\ &\underline{\mathbf{w}} = \underline{\mathbf{v}} - h\nabla J(\underline{\mathbf{v}}). \end{aligned}$$

Fortunately, $P_U(\underline{\mathbf{v}}, h)$ can easily be computed in the case of box-constraints by an entry-wise saturation

$$\begin{aligned} P_U(\underline{\mathbf{v}}, h) &= \underline{\mathbf{y}} \\ y_j[i] &= \begin{cases} l[i], & \text{if } w_j[i] < l[i] \\ r[i], & \text{if } w_j[i] > r[i] \\ w_j[i], & \text{else} \end{cases}, \\ &\quad i = 1, \ldots, p, j = 0, \ldots, N-1 \\ \underline{\mathbf{w}} &= \underline{\mathbf{v}} - h\nabla J(\underline{\mathbf{v}}). \end{aligned} \tag{10}$$

Note that the gradient projection method

$$\underline{\mathbf{v}}^{i+1} = P_U(\underline{\mathbf{v}}^i, h^i),$$

always delivers feasible iterates.

Finally, Nesterov's method (often called the Fast Gradient method) is given in Algorithm 1, where $\mu > 9$ is the minimum eigenvalue of H. Note that due to the choice of the cost criterion, the matrix H is positive definite i.e. $\mu > 0$. Nesterov's method uses the projected gradient step $P_U(\underline{\mathbf{v}}, h)$ and an additional "step" that leads to faster convergence than the gradient projection method. For $\mu > 0$ the gradient projection method converges with a rate of $O(1 - \frac{\mu}{L})$, whereas the Fast Gradient method converges with $O(1 - \sqrt{\frac{\mu}{L}})$, see also Section 4.

Clearly, the optimization problem (5) is always feasible and we can use e.g. as a feasible initial guess the zero-vector i.e. $\underline{\mathbf{v}} = \underline{\mathbf{0}}$, which is called *cold-starting*. However using an initial guess based on the solution of the previous optimal control problem, called warm-starting usually

Algorithm 1 Fast Gradient method

Require: State $\mathbf{x}_k$, Initial guess $\underline{\mathbf{y}} \in U$, Number of iterations i_{max}, maximum and minimum eigenvalues of H: L, μ

1: Set $\underline{\mathbf{v}}^{old} = \underline{\mathbf{y}}$, $\underline{\mathbf{w}} = \underline{\mathbf{y}}$
2: **for** $i = 1, \dots, i_{max}$ **do**
3: Compute $\underline{\mathbf{v}} = P_U(\underline{\mathbf{w}}, \frac{1}{L})$
4: Compute $\underline{\mathbf{w}} = \underline{\mathbf{v}} + \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}(\underline{\mathbf{v}} - \underline{\mathbf{v}}^{old})$
5: Set $\underline{\mathbf{v}}^{old} = \underline{\mathbf{v}}$
6: **end for**
7: **return** $\underline{\mathbf{v}}$

decreases the computational effort. If the solution to the optimal control problem at step k is $\underline{\mathbf{w}}$, then an often useful choice of an initial guess $\mathbf{v}^0$ is

$$\mathbf{v}_j^0 = \mathbf{w}_{j+1}, \forall j = 0, \dots, N-2; \mathbf{v}_{N-1}^0 = \mathbf{0}.$$

3. GRADIENT DETERMINATION

As mentioned in the previous section the Fast Gradient method requires the gradient of

$$\frac{1}{2}\underline{\mathbf{v}}^T H \underline{\mathbf{v}} + \underline{\mathbf{v}}^T F \mathbf{x}_k, \tag{11}$$

which is given by

$$\nabla J(\underline{\mathbf{v}}) = H\underline{\mathbf{v}} + F\mathbf{x}_k \tag{12}$$

where $H \in \mathbb{R}^{Np \times Np}$, $F \in \mathbb{R}^{Np \times n}$. Obviously, we can compute the gradient by first determining H and F offline and afterwards evaluating (12). In particular, at each time step k we need to calculate $F\mathbf{x}_k$ once. Moreover, in each iteration step of the optimization problem we need to evaluate $H\underline{\mathbf{v}}$ and $H\underline{\mathbf{v}} + F\mathbf{x}_k$, which requires $O((Np)^2)$ calculations, because H is not sparse. We will denote this method as *standard method*.

However this is not the only possibility. We derive in this section a method to determine $\nabla J(\underline{\mathbf{v}})$, which requires only $O(Nnp)$ calculations.

By definition of the gradient (8) we need to determine $\frac{\partial J(\underline{\mathbf{v}})}{\partial \mathbf{v}_i}$ for all $i = 0, \dots, N-1$. For the MPC problem considered here

$$\begin{aligned}\frac{\partial J(\underline{\mathbf{v}})}{\partial \mathbf{v}_i} =& \frac{\partial}{\partial \mathbf{v}_i}\left(\frac{1}{2}\mathbf{z}_N^T P \mathbf{z}_N + \sum_{b=0}^{b=N-1} \frac{1}{2}\mathbf{z}_b^T Q \mathbf{z}_b \right.\\ &\left. + \sum_{a=0}^{N-1} \mathbf{z}_a^T S \mathbf{v}_a + \frac{1}{2}\mathbf{v}_a^T R \mathbf{v}_a)\right)\end{aligned}$$

where

$$\mathbf{z}_{b+1} = A\mathbf{z}_b + B\mathbf{v}_b,\ b = 0, \dots, N-1, \qquad \mathbf{z}_0 = \mathbf{x}_k.$$

Note that $\mathbf{v}_i$ influences only future states and does not influence $\mathbf{v}_j$, $i \neq j$. This yields

$$\begin{aligned}\frac{\partial J(\underline{\mathbf{v}})}{\partial \mathbf{v}_i} =& \frac{\partial}{\partial \mathbf{v}_i}\left(\frac{1}{2}\mathbf{z}_N^T P \mathbf{z}_N + \frac{1}{2}\mathbf{v}_i^T R \mathbf{v}_i \right.\\ &\left. + \sum_{a=i}^{N-1} \mathbf{z}_a^T S \mathbf{v}_a + \frac{1}{2}\mathbf{z}_a^T Q \mathbf{z}_a\right)\end{aligned} \tag{13}$$

where

$$\mathbf{z}_{b+1} = A\mathbf{z}_b + B\mathbf{v}_b,\ b = 0, \dots, N-1, \qquad \mathbf{z}_0 = \mathbf{x}_k$$

We will show in the next theorem that the above condition can be evaluated for $i = 0, \dots, N-1$ in a very efficiently.

Theorem 1. (Structure of gradient) Assume that the system (1) and a cost criterion (3) are given. Then for every $\mathbf{x}_k$, $i \in \{0, \dots, N-1\}$ and every $\underline{\mathbf{v}}$ there exist V_i, T_i and W_i such that

$$\begin{aligned}\frac{\partial J(\underline{\mathbf{v}})}{\partial \mathbf{v}_i} =& V_i \mathbf{v}_i + T_i \mathbf{z}_i + W_i(\mathbf{v}_{i+1}, \dots, \mathbf{v}_{N-1}) \\ &\text{where} \\ \mathbf{z}_{b+1} =& A\mathbf{z}_b + B\mathbf{v}_b,\ b = 0, \dots, i-1, \qquad \mathbf{z}_0 = \mathbf{x}_k.\end{aligned} \tag{14}$$

Moreover V_i, T_i are independent of $\mathbf{x}_k$ and $\underline{\mathbf{v}}$. Furthermore if $p \leq n$, then the computational effort to determine $\{V_j\}$, $\{T_j\}$ is $O(Nn^2p)$ and for given $\{T_j\}$ and $\underline{\mathbf{v}}$ the computational effort to determine $\{W_j\}$ is $O(Nnp)$.

The proof of Theorem 1 is in the Appendix A. Note that we can compute $\{V_i\}$, $\{T_i\}$ offline.

Now we present a corollary, which describes the computational effort to determine the gradient $\nabla J(\underline{\mathbf{v}})$ and the projected gradient step $P_U(\underline{\mathbf{v}}, h)$ employing $\{V_j, T_j\}$.

Corollary 2. (Effort to determine gradient) The computational effort for determining the gradient $\nabla J(\underline{\mathbf{v}})$ for given $\{V_j, T_j\}$, $\underline{\mathbf{v}}$ is $O(Nnp)$, if $p \leq n$. Furthermore, if we have box-constraints, then also the computational effort to compute the projected gradient step $P_U(\underline{\mathbf{v}}, h)$ is $O(Npn)$.

Proof. We assume without loss of generality (cf. proof of Theorem 1 in the Appendix), that A contains less than $2.5n$ non-zero entries. To determine the gradient we need to evaluate (14) for all $i = 0, \dots, N-1$. We first need to determine $\{\mathbf{z}_j\}$ and $\{W_j\}$. We have shown above that calculating $\{W_j\}$ is of the order $O(Nnp)$. We can easily determine $\{\mathbf{z}_j\}$ by using the system dynamics (1), which requires $O(Nnp)$ calculation, because we need to determine matrix-vector products of the sparse A matrix and the B matrix and add two vectors.

Computing $V_i\mathbf{v}_i$ and $T_i\mathbf{z}_i$ have a computational effort of $O(p^2)$ or $O(np)$, respectively. Calculating $V_i\mathbf{v}_i + T_i\mathbf{z}_i + Z_i$ is only of order $O(p)$. So determining $V_i\mathbf{v}_i + T_i\mathbf{z}_i + W_i$ for all $i = 0, \dots N-1$ requires $O(Nnp)$ operations.

For box-constraints the projected gradient step $P_U(\underline{\mathbf{v}}, h)$ can be determined from the gradient and (10), which has a computational effort of $O(Np)$. □

The proposed method is illustrated in Algorithm 2. Since the tlines (1), (2) and (3) are independent, they can be computed simultaneously, which might allow a speedups.

Algorithm 2 Gradient determination

Require: $\mathbf{x}_k$, $\underline{\mathbf{v}}$, $\{T_j\}$, $\{V_j\}$

1: Compute $\mathbf{z}_{j+1}$, $T_j\mathbf{z}_j$, $j = 1, \dots, N$
2: Compute $V_j\mathbf{v}_j, j = 1, \dots, N-1$
3: Compute W_j, $Z_j\mathbf{v}_j, j = N-1, \dots, 0$
4: **return** $\nabla J(\underline{\mathbf{v}}) = \begin{pmatrix} (T_1\mathbf{z}_1 + V_1\mathbf{v}_1 + W_1)^T & \dots \end{pmatrix}^T$

Remark 3. (Pre-conditioning) If we want to use pre-conditioning as mentioned in e.g. Richter et al. (2010) in order to improve the convergence behavior, then we solve the optimization problem using a different coordinate system i.e. $\tilde{\underline{\mathbf{v}}} = D\underline{\mathbf{v}}$, $\det D \neq 0$. So, we have B_j, R_j, S_j, which depend on the index j. It the proposed method can be extended to this case by keeping track of the required

indexes.Still we can determine $\{T_j\}$ and $\{V_j\}$ and L, μ offline.

4. MEMORY AND COMPUTATIONAL DEMAND

In this section we consider the computational effort and memory demand of the overall Fast Gradient method.

4.1 Memory demand

Let us compare the memory required for the standard method, i.e. direct evaluation of (12), and the proposed method (Algorithm 2). We distinguish between dynamic data, which is updated during the algorithm, and static data, which remains unchanged during the run of the algorithm.

Fist let us consider the static data. Using the standard method, we need to store H and F, which have using symmetry $pN(0.5(pN+1)+n)$ elements. In contrast, our proposed method needs A, B, $\{V_i\}$ and $\{T_i\}$, which have together about $pN(0.5(p+1)+n)+2.5np+np$ elements using symmetry and assuming without loss of generality a sparse A.

Second, let us investigate the size of the dynamic data. If we use the standard method to determine the gradient, we need to store $F\mathbf{x}_k$, $\underline{\mathbf{w}}$ and the gradient $\nabla J(\underline{\mathbf{v}})$ i.e. in summary $3Np$ elements. In contrast, the proposed method requires memory for $\underline{\mathbf{w}}$, the gradient $\nabla J(\underline{\mathbf{v}})$ and additionally W_j, $\mathbf{z}_j$, but only for a single value of j and either W_j or $\mathbf{z}_j$. So, we need only memory for $2Np+n$ elements.

4.2 Computational effort

Let us assume we want to obtain a solution $\underline{\mathbf{v}}$ such that for a given $\epsilon > 0$ we have $J(\underline{\mathbf{v}}) - J^\star \leq \epsilon$, where $J^\star$ denotes the optimal solution. As shown in Nesterov (2004); Richter et al. (2009) for the case of box constraints and using $\frac{r[i]+l[i]}{2}$ as initial guess this requires at most I_{max} iterations, where

$$I_{Max} \geq \min\left(\frac{\ln 2\epsilon - \ln Ld^2}{\ln(1-\sqrt{\frac{\mu}{L}})}, \sqrt{\frac{2Ld^2}{\epsilon}} - 2\right) \qquad (15)$$

$$d^2 = N\sum_{i=1}^{p}\frac{(r[i]-l[i])^2}{2}.$$

As defined earlier the constants L, μ are the minimum and maximum eigenvalues of H. Note that the matrix $H \in \mathbb{R}^{Np\times Np}$ is given by

$$H = \begin{pmatrix} V_0 & M_{0,1} & \dots & M_{0,N-1} \\ M_{1,0} & V_1 & \dots & M_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ M_{N-1,0} & M_{N-1,1} & \dots & V_{N-1} \end{pmatrix}, \qquad (16)$$

where we already have determined V_j (A.2) and we know that $M_{j,m} = M_{m,j}^T$, because H is symmetric. Moreover, if we consider (A.1), then the only term, which depends on $\mathbf{v}_i$ and $\mathbf{z}_j$ is $\mathbf{v}_i^T T_i \mathbf{z}_j$. Therefore for $j > i$ we get

$$M_{i,j} = T_j A^{j-1} B.$$

Clearly, we can always determine L, μ from H. However, let us now analyze the influence of the horizon N and the stability of A, onto L, μ and therefore the number of iterations. Let us denote the spectral radius of A by $\rho(A)$.

Theorem 4. (Condition of the MPC problem) The following statements are true for the system (1), the cost criterion (3) and the matrix H as in (16).
1) If $\rho(A) < 1$, then L is bounded above by a function linear increasing in Np.
2) If $\rho(A) = 1$, then L is not exponentially increasing in N.
3) If $\rho(A) > 1$, (A, B) is stabilizable and $(A, Q^{\frac{1}{2}})$ is detectable, then L is exponentially increasing in N for large N.
4) $\mu \geq (1-c)\lambda_{Min}(R)$.
5) μ is not increasing in N.

The proof of this theorem is given in the Appendix A.

So, if A is asymptotically stable, then L is $O(Np)$ and therefore I_{Max} is $O((Np)^{\frac{1}{2}}\ln\frac{1}{\epsilon})$. So the Fast Gradient method has a time complexity of $O(N^{\frac{3}{2}}p^{\frac{3}{2}}n\ln\frac{1}{\epsilon})$ using the proposed gradient determination (Algorithm 2), if box-constraints are used. In contrast, if $\rho(A) > 1$ i.e. the system is (exponentially) unstable, then the computational effort to solve the optimization problem (4) using the Fast Gradient method increases exponentially in N. Note that, we only consider the worst case complexity and did ot consider warm-starting or pre-conditioning as in Richter et al. (2009, 2010).

4.3 Comparission with backward-propagation

Bertsekas (1995) reports also a way to determine the gradient using the underlying structure consisting of two coupled iterations, called backward-propagation. It has in general a time complexity of $O(Nn^2)$ and a larger dynamic memory demand. The static memory demand is constant (unless pre-conditioning is used).

5. EXAMPLE

Let us illustrate the results by an example. The algorithms are implemented in the Embedded Matlab subset of Matlab (comes with Simulink), so it possible to generate native code using the *emlmex* command. We use a 2.4 GHz Intel CPU to evaluate the algorithms.

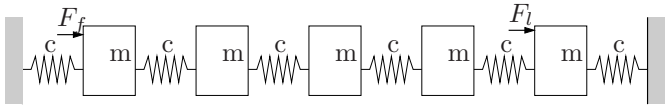

Fig. 1. Chain of five masses.

As example we use a chain of 5 masses connected to each other and to two walls by springs as illustrated in Figure 1. We assume that each mass has a value of $m = 1$ and the spring constants are also $c = 1$. The system has $n = 10$ states and there are $p = 2$ inputs: forces F_f, $|F_f| \leq 1$ and F_l, $|F_l| \leq 1$ acting on the first and the last mass. We use a sampling time of 0.2 seconds and a zero-order hold. Since there is no damping all eigenvalues are on the unit disc, complex and single. Hence, the real-block Jordan form of A has 20 non-zero entries.

We regulate the chain using MPC with different horizon length and $Q = R = P = I$, $S = 0$. We choose the number

Table 1. Example 2: Time [μs] of one Fast Gradient method call.

Horizon N	5	10	20	40	60	80
Iterations	5	9	19	34	53	72
Standard Alg. with BLAS	0.98	6.19	28.2	168	505	1137
Standard Alg. without BLAS	0.97	6.76	50.7	370	1255	4255
Proposed Alg.	2.73	10.4	41.3	146	341	650
Speed up factor (worst case)	0.36	0.59	0.68	1.15	1.48	1.75

of iterations such that (15) holds for $\epsilon = 10^{-3}$, i.e. the obtained solution differs less than ϵ from the optimum

Table 2 shows the time of one call of the Fast Gradient method - i.e. one evaluation of the MPC - obtained by averaging over one million calls. Note that Embedded Matlab uses by default a BLAS library, optimized implementations of numerical linear algebra and the standard method (12) to obtain the gradient is a matrix-vector product. In contrast, we do not tune the implementation of the proposed algorithm much. Therefore, for the sake of comparison we also evaluated the performance of the standard method without BLAS. The standard algorithm performs better for lower N.

Note that only the worst case was considered and did not consider warm starting or pre-conditioning, see Richter et al. (2009, 2010).

6. CONCLUSIONS

In this work we consider fast model predictive control of linear, time-invariant systems with input-constraints using the Fast Gradient method. In particular, we present a method to efficiently determine the gradient required for the Fast Gradient method exploiting the problem structure, which requires less memory and is faster as the standard method for large horizons. Additionally, we investigate the influence of the stability of the plant onto the number of iterations. An example illustrates the results.

REFERENCES

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

Bertsekas, D.P. (1995). *Nonlinear Programming*. Athena Scientific.

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

Ferreau, H.J., Bock, H.G., and Diehl, M. (2008). An online active set strategy to overcome the limitations of explicit mpc. *International Journal of Robust and Nonlinear Control*, 18, 816–830.

Ferreau, H.J., Ortner, P., Langthaler, P., del Re, L., and Diehl, M. (2007). Predictive control of a real-world diesel engine using extended online active set strategy. *Annual Reviews in Control*, 31, 293–301.

Garcia, C., Prett, D., and Morari, M. (1989). Model predictive control: Theory and practice - A survey. *Automatica*, 25(3), 335–348.

Maciejowski, J.M. (2002). *Predictive Control with Constraints*. Prentice Hall, Upper Saddle River, New Jersey.

Milman, R. and Davison, E.J. (2008). A fast mpc algorithm using nonfeasible active set methods. *Journal of Optimization Theory and Applications*, 139, 591–616.

Nesterov, Y. (1983). A method for solving a convex programming problem with convergence rate $1/k^2$. *Soviet Mathematics Doklady*, 27(2), 372–376.

Nesterov, Y. (2004). *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Acad. Publ., Berlin.

Qin, S. and Badgwell, T. (2003). A survey of industrial model predictive control technology. *Control engineering practice*, 11(7), 733–764.

Rao, C., Wright, S., and Rawlings, J. (1998). Application of interior-methods to model predictive control. *Journal of Optimization Theory and Applications*, 99, 723–757.

Richter, S., Jones, C.N., and Morari, M. (2009). Real-time input-constrained mpc using fast gradient methods. In *48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, 7387–7393.

Richter, S., Mariethoz, S., and Morari, M. (2010). High-speed online mpc based on a fast gradient method applied to power converter control. In *Proceedings of the 2010 American Control Conference*, 4737–4743.

Shahzad, A., Kerrigan, E.C., and Constantinides, G.A. (2010). Preconditioners for inexact interior point methods for predictive control. In *Proceedings of the 2010 American Control Conference*, 5714–5719.

Wang, Y. and Boyd, S. (2008). Fast model predictive control using online optimization. In *17th World Congress The International Federation of Automatic Control*, 6974–6979.

Appendix A. PROOF OF THEOREM 1 AND 7

A.1 Proof of Theorem 1

Let us define the cost $J^i(\underline{\mathbf{v}})$ as

$$J^i(\underline{\mathbf{v}}) = \frac{1}{2}\mathbf{z}_N^T P \mathbf{z}_N + \sum_{b=i}^{N-1} \frac{1}{2}\mathbf{v}_b^T R \mathbf{v}_b + \mathbf{z}_b^T S \mathbf{v}_b + \frac{1}{2}\mathbf{z}_b^T Q \mathbf{z}_b.$$

Comparing the cost $J^i(\underline{\mathbf{v}})$ with (3) yields

$$J(\underline{\mathbf{v}}) = J(\underline{\mathbf{v}})^i + \sum_{b=0}^{i-1} \frac{1}{2}\mathbf{v}_b^T R \mathbf{v}_b + \mathbf{z}_b^T S \mathbf{v}_b + \frac{1}{2}\mathbf{z}_b^T Q \mathbf{z}_b.$$

Since the system (1) is causal, $\mathbf{v}_i$ influences only $J(\underline{\mathbf{v}})^i$.

Now, we will show, that for any $i = 0, \ldots, N-1$

$$\begin{aligned} J^i(\underline{\mathbf{v}}) =& \frac{1}{2}\mathbf{v}_i^T V_i \mathbf{v}_i + \mathbf{v}_i^T T_i \mathbf{z}_i + \mathbf{v}_i^T W_i \\ &+ \frac{1}{2}\mathbf{z}_i^T Y_i \mathbf{z}_i + X_i + \mathbf{z}_i^T Z_i \end{aligned} \tag{A.1}$$

holds, where for $j = 1, \ldots, N$

$$\begin{aligned} V_{j-1} &= B^T Y_j B + R, \quad & T_{j-1} &= B^T Y_j A + S \\ W_{j-1} &= B^T (Z_j + T_j^T \mathbf{v}_j), & Y_{j-1} &= A^T Y_j A + Q, \end{aligned} \tag{A.2}$$

and for $j = 1, \ldots, N-1$

$$\begin{aligned} X_{j-1} &= X_j + \frac{1}{2}\mathbf{v}_j^T V_j \mathbf{v}_j + \mathbf{v}_j^T W_j \\ Z_{j-1} &= A^T (Z_j + T_j^T \mathbf{v}_j), \end{aligned} \tag{A.3}$$

where

$$X_{N-1} = 0,\ Y_N = P,\ Z_{N-1} = 0 \tag{A.4}$$

and

$$\mathbf{z}_{b+1} = A\mathbf{z}_b + B\mathbf{v}_b,\ \mathbf{z}_0 = \mathbf{x}_k,\ b = 0, \ldots, i-1,$$

where $V_i \in \mathbb{R}^{p\times p}$, $T_i \in \mathbb{R}^{p\times n}$, $W_i \in \mathbb{R}^p$, $Y_i \in \mathbb{R}^{n\times n}$, $Z_i \in \mathbb{R}^n$. Now (A.1) can be easily verified by induction. First for $i = N-1$, we have

$$\begin{aligned} J^{N-1}(\underline{\mathbf{v}}) =& \frac{1}{2}\mathbf{z}_N^T P\mathbf{z}_N + \frac{1}{2}\mathbf{z}_{N-1}^T Q\mathbf{z}_{N-1} \\ &+ \frac{1}{2}\mathbf{v}_{N-1}^T R\mathbf{v}_{N-1} + \mathbf{v}_{N-1}^T S\mathbf{z}_{N-1} \end{aligned}$$

Using $\mathbf{z}_N = A\mathbf{z}_{N-1} + B\mathbf{v}_{N-1}$ we obtain

$$\begin{aligned} J^{N-1} =& \frac{1}{2}\mathbf{z}_{N-1}^T (A^T PA + Q)\mathbf{z}_{N-1} \\ &+ \mathbf{v}_{N-1}^T (S + B^T PA)\mathbf{z}_{N-1} \\ &+ \frac{1}{2}\mathbf{v}_{N-1}^T (R + B^T PB)\mathbf{v}_{N-1} \end{aligned}$$

comparing this with (A.2), (A.3) and (A.4) shows that (A.1) is true for $i = N-1$. Now, we assume that (A.1) holds for j, and show that it holds also for $j-1$. We have

$$\begin{aligned} J^{j-1}(\underline{\mathbf{v}}) =& \frac{1}{2}\mathbf{v}_j^T V_j\mathbf{v}_j + \mathbf{v}_j^T T_j\mathbf{z}_j + \mathbf{v}_j^T W_j + \frac{1}{2}\mathbf{z}_j^T Y_j\mathbf{z}_j \\ &+ \frac{1}{2}\mathbf{z}_{j-1} Q\mathbf{z}_{j-1} + \mathbf{v}_{j-1}^T S\mathbf{z}_{j-1} \\ &+ \frac{1}{2}\mathbf{v}_{j-1}^T R\mathbf{v}_{j-1} + X_j + \mathbf{z}_j^T Z_j \end{aligned}$$

holds. Using $\mathbf{z}_j = A\mathbf{z}_{j-1} + B\mathbf{v}_{j-1}$ yields

$$\begin{aligned} J^{j-1}(\underline{\mathbf{v}}) =& \frac{1}{2}\mathbf{v}_{j-1}^T (R + B^T Y_j B)\mathbf{v}_{j-1} \\ &+ (\mathbf{v}_{j-1}^T B + \mathbf{z}_{j-1}^T A)(Z_j + T_j^T) + \frac{1}{2}\mathbf{v}_j^T V_j\mathbf{v}_j \\ &+ \frac{1}{2}\mathbf{z}_{j-1}^T (Q + A^T Y_j A)\mathbf{z}_{j-1} + \mathbf{v}_j^T W_j \\ &+ \mathbf{v}_{j-1}^T (S + B^T Y_j A)\mathbf{z}_{j-1} + X_j. \end{aligned}$$

Again comparing this with (A.2), (A.3) and (A.4) shows that (A.1) is true for $i = j-1$. Thus, (A.1) is true for any $i = 0, \ldots, N-1$. Note that, (14) follows directly from (A.1) and the fact, that $\mathbf{v}_i$ influences only future states, but not $\mathbf{v}_j$, $i \neq j$.

Finally, we need to analyze the number of required calculations. First, note that determining $Y_{j-1}, V_{j-1}, T_{j-1}$ from Y_j requires matrix-matrix multiplication and additions of $n\times n$, $n\times p$, $p\times n$ or $p\times p$ matrices and $p \leq n$. So, we need $O(Nn^3)$ calculations to determine $\{T_j\}$, $\{V_j\}$. Note that all $n\times n$ matrix multiplications include the matrix A or A^T. In addition, calculating Z_{j-1} from Z_j, T_j, $\mathbf{v}_j$ requires basically matrix-vector multiplication featuring the $n\times n$ A^T matrix and a $p\times n$ matrix. In contrast to obtain W_{j-1} we need to do matrix-vector multiplications with an $p\times n$ matrix and an $n\times p$ matrix. If we assume that the matrix A is sufficiently sparse, then it is possible to reduce the computational cost to $O(Npn^2)$ and $O(Npn)$, respectively, because every matrix-vector or matrix-matrix multiplication features either A or $n\times p$, $p\times n$, $p\times p$ matrices. Choosing the state $\mathbf{x}_k$ such that A is in the real block Jordan form makes A sufficiently sparse, it contains less than $2.5n$ non-zero elements. □

A.2 Proof of Theorem 4

Note that the minimum and maximum eigenvalue of H are $\lambda_{Max}(H) = L$ or $\lambda_{Min}(H) = \mu$.

1) First note that $\lambda_{Max}(H) \leq \mathrm{tr}(H) \leq N\,\mathrm{tr}(V_0)$, due to the structure of H, $H = H^T \geq 0$ and $V_0 \geq V_1 \geq \ldots \geq V_{N-1} \geq 0$. Moreover $\mathrm{tr}(V_0) \leq p\,\|V_0\|_2$, because $\|V_0\|_2$ is the maximum eigenvalue of V_0. Let $\epsilon > 0$ be such that $r+\epsilon < 1$, where $r = \rho(A)$. There exists an K such that $\left\|V_0^K\right\|_2 \leq (r+\epsilon)^K$ and $\left\|A^j\right\|_2 \leq 1, \forall j \geq K$ due to the spectral radius property of the 2-norm. If $N-2 > K$, then there exists a $c_1 > 0$ such that $\|V_0\|_2 \leq c_1 + \|B\|_2^2 (\|Q\|_2 \sum_{i=K}^{N-2} \left\|A^i\right\|_2^2 + \|P\|_2 \left\|A^{N-1}\right\|_2^2)$. So $\|V_0\|_2 \leq c_1 + \|B\|_2^2 (\|Q\|_2 \sum_{i=C}^{N-2} (r+\epsilon)^{2i} + \|P\|_2) \leq c_1 + \|B\|_2^2 (\|Q\|_2 (\frac{1}{1-r-\epsilon})^2 + \|P\|_2)$. Hence, $\|V_0\|_2$ is bounded above by a constant c_3 for any N and $\lambda_{Max}(H) \leq pNc_3$.

2) Assume for the sake of contradiction $\lambda_{Max}(H)$ is exponentially increasing in N. There need to exist a $c_a > 1$ and c_b such that $\lambda_{Max}(H) > c_a^N + c_b$. Let δ be such that $c_a > \delta > 1$. Let K be such that $\left\|A^K\right\|_2^2 \leq \delta^K$. Again such a K always exists due to the spectral radius formula. If $N-2 > K$, then as in part 1 of the proof we have $\|V_0\|_2 \leq c_1 + \|B\|_2^2 (\|Q\|_2 \sum_{i=K}^{N-2} \delta^i + \|P\|_2 \delta^{N-1})$. So $\|V_0\|_2 \leq c_1 + c_2(N-K)\delta^N$, i.e. $\lambda_{Max}(H) \leq Np(c_1 + c_2(N-K)\delta^N)$, which contradicts the assumption $\lambda_{Max}(H) > c_a^N$, because $c_a > \delta$.

3) We have $Np\lambda_{Max}(H) \geq \mathrm{tr}(H) \geq \mathrm{tr}(V_0)$ and $\mathrm{tr}(V_0) \geq \mathrm{tr}(B^T(A^{N-2})^T QA^{N-2}B) \geq \lambda_{Max}(B^T(A^{N-2})^T QA^{N-2}B)$. Moreover for any $\mathbf{v} \in \mathbb{R}^p, |\mathbf{v}| = 1$ we have $\lambda_{Max}(B^T(A^{N-2})^T QA^{N-2}B) \geq \mathbf{v}^T B^T(A^{N-2})^T QA^{N-2}B\mathbf{v}$. Let $\mathbf{v}_{Max}$ be an eigen vector of A to the eigen value $\rho(A)$. So, $\mathbf{v}_{Max}^T(A^{N-2})^T QA^{N-2}\mathbf{v}_{Max} = \rho(A)^{2(N-2)}\mathbf{v}_{Max}^T Q\mathbf{v}_{Max}$ Since A is unstable, $(A, Q^{\frac{1}{2}})$ is detectable and (A, B) stabilizable, there is a $c_1 > 0$ and $\tilde{\mathbf{v}} \in \mathbb{R}^n$ such that $\mathbf{v}_{Max}^T Q\mathbf{v}_{Max} > c_1$ and $\mathbf{v}_{Max} = B\tilde{\mathbf{v}} = c \neq \mathbf{0}$. Let us choose $\tilde{\mathbf{v}}$ such that $\|\tilde{\mathbf{v}}\|_2 = 1$. There is a $c_2 > 0$ such that $\tilde{\mathbf{v}}^T(B^T(A^{N-2})^T QA^{N-2}B)\tilde{\mathbf{v}} \geq c_1c_2\rho(A)^{2(N-2)}$. Finally, $\lambda_{Max}(H) \geq \frac{c_1c_2\rho(A)^{2(N-2)}}{Np}$.

4) Note that, H is given by $\tilde{H} + (1-c)H_R$, where $\tilde{H}$ depends on Q, R, S and H_R only on R. In particular $H_Q \geq 0$ and $(1-c)H_R$ is a block diagonal matrix with N blocks consisting of the matrix $(1-c)R$. Therefore, $\lambda_{Min}(H) \geq (1-c)\lambda_{Min}(H_R) = (1-c)\lambda_{Min}(R)$.

5) First let us consider $N = 1$, so $H = B^T PB + R$. Let $\lambda^\star = \lambda_{min}(H)$ and $\mathbf{v} \in \mathbb{R}^p, |\mathbf{v}| = 1$ such that $\mathbf{v}^T(B^T PB + R)\mathbf{v} = \lambda^\star$. Now consider any $N \geq 1$ and $\mathbf{w} \in \mathbb{R}^{Np} = \begin{pmatrix}\mathbf{0}^T & \mathbf{v}^T\end{pmatrix}^T$. Clearly, $|\mathbf{w}| = 1$. Moreover $\mathbf{w}^T H\mathbf{w} = \mathbf{v}^T(B^T PB + R)\mathbf{v} = \lambda^\star$. So, $\lambda_{Min}(H)$ is for an arbitrary N not larger as for $N = 1$, because for a positive definite, symmetric matrix Z and a normalized vector z, we have $z^T Zz \geq \lambda_{Min}(Z)$. □