2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

Fast predictive control of linear systems combining Nesterov's gradient method and the method of multipliers

Markus Kögel and Rolf Findeisen

***Abstract*— The fast, tailored solution of linear, predictive control problems is important, yet challenging. We present an algorithm based on the fast gradient method and the method of multipliers for model predictive control of linear, discrete-time, time-invariant, systems with box constraints. The algorithm uses the augmented Lagrangian method to handle equality constraints, so it can takes advantage of the sparsity of the problem. We present different schemes to update the multipliers. An example illustrates the performance of the algorithm, which is competitive with other tailored solution methods.**

***Index Terms*— Model predictive control, Augmented Lagrangian, Online optimization, Fast gradient method.**

I. Introduction

Predictive control, also denoted model predictive control (MPC) or receding horizon control, is frequently used to control systems subject to constraints, see [7], [8], [11], [17]. Predictive control allows to control systems with a high performance and such that constraints are satisfied. In predictive control at each time a new measurement becomes available the input is determined by solving a finite horizon optimal control problem of which the resulting input is applied until the next time. Solving the required optimization problem fast and with a limited memory footprint, e.g. for embedded control problems, is challenging. We present in this paper a suitable method for discrete-time, linear, time-invariant systems subject to a quadratic cost criterion and box constraints.

In principle there are two main solution approaches to solve the resulting quadratic program appearing in such predictive control problems: online and offline (explicitly). In the so-called online-optimization at each step the problem is solved online in real-time. The resulting computational demand depends on the problem and can be challenging. For this reason online-optimization is usually limited by the available computation speed. In the latter approach, often called explicit MPC, the solution for all possible states is calculated offline and stored in for example a table, see [1]. Unfortunately, this table grows in general exponentially in the number of states, inputs and horizon. So, the explicit solution is often limited by its memory demand and often restricted to small-scale systems.

M. Kögel and R. Findeisen are with the Institute for Automation Engineering, Otto-von-Guericke-University Magdeburg, Germany. M. K. is also a member of the International Max Planck Research School for Analysis, Design and Optimization in Chemical and Biochemical Process Engineering, Magdeburg, Germany. {markus.koegel, rolf.findeisen}@ovgu.de.

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. Active set methods tailored for model predictive control problems are presented in [5], [14]. Unfortunately, in theory the worst case time complexity of these algorithm is exponential in the number of constraints. However, in practice they are very efficient and can be used e.g. to control diesel engines with sampling times down to milliseconds, cf. [6].

In [18], [22] interior-points methods taking the special structure of the problem into account are considered. Moreover, [21] discusses inexact interior points approaches.

The works [19], [20] present online-optimization for systems with input constraints using the Nesterov's gradient method [15], [16]. In [10] we described a method to determine the gradient efficiently exploiting the underlying structure of the problems. This approach is extended in [9] using the method of augmented Lagrangians to handle constraints on the states. This result is based on the condensed problem i.e. with eliminated equality constraints.

In this paper we use the method of augmented Lagrangians to handle the equality constraints due to the dynamic and a possible terminal constraint. We employ the fast gradient method to solve the sparse subproblem arising from the multiplier method and discuss methods to update the multipliers. As shown, the memory demand and the computational time is only linearly increasing in the length of the horizon for most update schemes and the conditioning of the arising subproblems is bounded above for any horizon length.

The remainder of the paper is structured as follows. First we outline the problem setup in Section II. Afterwards we present the used optimization methods in Section III. In Section IV we present as the main result the proposed algorithm for the considered MPC problem. We outline an efficient implementation of the algorithm, analyze properties of the arising subproblems and discuss the computational demand. In Section V, we illustrate the performance of the approach by an example. Finally, we draw conclusions and outline future research directions in Section V.

II. Problem statement

We consider linear, discrete-time, time-invariant systems of the form

$$x_{k+1} = Ax_k + Bu_k + \tilde{d}, \tag{1}$$

where $x_k \in \mathbb{R}^n$ is the state, $u_k \in \mathbb{R}^p$ is the input and $\tilde{d}$ is a constant disturbance or reference. All matrices have the

978-1-61284-799-3/11/$26.00 ©2011 IEEE

appropriate dimensions. We assume that the state and input are constrained to compact boxes,

$$\underline{x} \leq x_k \leq \overline{x}, \qquad \underline{u} \leq u_k \leq \overline{u}. \tag{2}$$

where $\leq$ holds entry-wise.

In this work we consider model predictive control of the system (1) subject to the constraints (2) with a control and prediction horizon of length N and a quadratic cost criterion [7], [11], [12]. The quadratic cost consists of a stage cost J_j and final cost J_f given by

$$\begin{aligned} J =& \sum_{j=k}^{k+N-1} J_j + J_f \qquad (3)\\ J_j =& \frac{1}{2} x_j^T Q x_j + \frac{1}{2} u_j^T R u_j + u_j^T S x_j + x_j^T \tilde{q} + u_j^T \tilde{r} \\ & \begin{pmatrix} Q & S^T \\ S & R \end{pmatrix} = W = W^T > 0 \\ J_f =& \frac{1}{2} x_{k+N}^T T x_{k+N} + x_{k+N}^T \tilde{t},\ T = T^T > 0. \end{aligned}$$

Moreover the terminal state can be constrained to a box and there can be equality constraints to enforce stability

$$\underline{x}_f \leq x_{k+N} \leq \overline{x}_f \tag{4}$$

$$F x_{k+N} = \tilde{f}, \tag{5}$$

where $F \in R^{m \times n}, m \leq n, m < Np$.

Note that this framework is rather general and contains several special cases. In particular classical stabilization i.e. the control towards the origin is given by $\tilde{d} = \tilde{r} = \tilde{q} = \tilde{t} = 0$.

We focus in this work on fast solutions of the MPC problems, stability results and conditions can be found in, e.g., [12] and the references therein.

Note that the $N(p+n)$ optimization variables are

$$z = \begin{pmatrix} u_k^T & x_{k+1}^T & u_{k+1}^T & \dots & u_{k+N-1}^T & x_{k+N}^T \end{pmatrix}^T. \tag{6}$$

Moreover, to denote the $2N(p+n)$ inequality constraints on the optimization variables due to the terminal constraint (4), state and input constraints (2), we define

$$\underline{z} \leq z \leq \overline{z} \ \Leftrightarrow \begin{cases} \underline{x} \leq x_{k+j} \leq \overline{x}, \forall j = 1, \dots, N-1 \\ \underline{u} \leq u_{k+j} \leq \overline{u}, \forall j = 0, \dots, N-1 \\ \underline{x}_f \leq x_{k+N} \leq \overline{x}_f. \end{cases} \tag{7}$$

Using these definitions the MPC optimization problem, a quadratic program (QP), is shortly given by

$$\begin{aligned} \min_z & \frac{1}{2} z^T H z + z^T g \qquad (8)\\ s.t.\ & \underline{z} \leq z \leq \overline{z} \\ & Cz = e, \end{aligned}$$

where $H \in R^{N(p+n) \times N(p+n)}$, $g \in R^{N(p+n) \times 1}$, $C \in R^{Nn+m \times N(p+n)}$, $e \in R^{Nn+m \times 1}$ are given by

$$C = \begin{pmatrix} B & -I & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & A & B & -I & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & A & B & -I \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & F \end{pmatrix} \tag{9}$$

$$H = \begin{pmatrix} R & 0 & \dots & 0 & 0 \\ 0 & W & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & W & 0 \\ 0 & 0 & \dots & 0 & T \end{pmatrix}, \tag{10}$$

$$e = \begin{pmatrix} -Ax_k + \tilde{d} \\ \tilde{d} \\ \vdots \\ \tilde{d} \\ \tilde{f} \end{pmatrix},\ g = \begin{pmatrix} Sx_k + \tilde{r} \\ \tilde{q} \\ \tilde{r} \\ \vdots \\ \tilde{r} \\ \tilde{t} \end{pmatrix}. \tag{11}$$

Note that there are $Nn + m$ equality constraints resulting from the dynamic (1) and the terminal condition (5).

III. Review of optimization methods

Several approaches to solve (8) efficiently exist [5], [9], [14], [18], [21], [22]. In this work we consider approaches based on the fast gradient method and the method of multipliers to solve (8). We briefly review these two methods.

A. Fast Gradient method methods

We will use Nesterov's gradient method also known as *Fast Gradient* method, cf. [15], [16]. Therefore we shortly recap this method as well as the required background information *gradient projection*.

Let us consider the unconstrained convex problem

$$\min_x \quad f(x), \tag{12}$$

where $f(x) : \mathbb{R}^N \to \mathbb{R}$ is a smooth, convex function with a globally Lipschitz continuous gradient. Using the well-known gradient descent method, see e.g. [16], we can determine a solution to (12) using an initial guess x^0 and

$$x^{i+1} = x^i - \frac{1}{L} \nabla f(x) = \mathcal{G}(x^i), \tag{13}$$

where $\nabla f(x)$ is the gradient. Here $\frac{1}{L}$ is used as step-size, where L is a Lipschitz constant of the gradient $\nabla f(x)$

$$\forall x, y \in \mathbb{R}^N, \|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|,$$

to guarantee $f(x^{i+1}) < f(x^i)$. Although, L is non-unique, using L as small as possible yields in general the best results.

If optimization variables are constrained to a convex, closed set $\mathcal{X}$, then one can solve the problem using gradient projection, see [16]. In particular we consider the projected gradient step $\mathcal{G}_{\mathcal{X}}(x)$ instead of (13)

$$\mathcal{G}_{\mathcal{X}}(x) = \arg\min_{z \in \mathcal{X}} \|x - z\|^2, \quad z = x - \frac{1}{L} \nabla f(x) = \mathcal{G}(x).$$

In summary,

$$x^{i+1} = \mathcal{G}_{\mathcal{X}}(x^i),$$

delivers feasible iterates and converges to the minimizer. Note that, the projected gradient step $\mathcal{G}_{\mathcal{X}}(x)$ requires an Euclidean projection into $\mathcal{X}$. This projection is in general nontrivial. However, it can be done analytically for simple sets such as boxes. Clearly, if $\mathcal{X} = \mathbb{R}^N$, then $\mathcal{G}_{\mathcal{X}}(x) = \mathcal{G}(x)$.

Next, Nesterov's gradient method is outlined in Algorithm 1. It uses the projected gradient step $\mathcal{G}_{\mathcal{X}}(x)$ and an additional step that leads to faster convergence than the gradient projection method. The scalar sequence c^i needs to satisfy certain

Algorithm 1 Fast Gradient method

Require: Initial guess $x^0 \in \mathcal{X}$, Number of iterations i_{max}, Parameter L, Sequence c^i

1: Set $x^{-1} = y^0 = x^0$
2: **for** $i = 1, \ldots, i_{max}$ **do**
3: Compute $x^i = \mathcal{G}_{\mathcal{X}}(y^{i-1})$
4: Compute $y^i = x^i + c^i(x^i - x^{i-1})$
5: **end for**
6: **return** $x^{i_{max}}$

conditions [16]. For example, it is always possible to choose $c^i = \frac{i-1}{i+2}$. However, if we know a *strong convexity constant* $\phi > 0$, i.e.

$$\forall x, y \in \mathbb{R}^N, \|\nabla f(x) - \nabla f(y)\| \geq \phi \|x - y\|,$$

then c^i can be constant: $c^i = \frac{\sqrt{L}-\sqrt{\phi}}{\sqrt{L}+\sqrt{\phi}}$. Clearly, ϕ is non-unique and $L \geq \phi$. Note that it is advantageous to use a ϕ as large as possible. As above $\mathcal{X} = \mathbb{R}^N$ is possible.

B. Method of multipliers for equality constraints

In this work we use the *method of augmented Lagrangians*, also called *method of multipliers*, cf. [2], [3], to minimize a convex function $f(s)$ subject to linear equality constraints $\Psi s = \psi$ and simple constraints on $s \in \mathcal{S}$

$$\min_s f(s), \text{ s.t. } \Psi s = \psi, \; s \in \mathcal{S}. \tag{14}$$

For such convex problems the method can be interpreted as maximizing a *proximal approximation* $\zeta(\lambda)_\mu$ of the dual function $\zeta(\xi)$ for some *penalty parameter* $\mu > 0$

$$\zeta(\lambda)_\mu = \sup_\xi \left(\zeta(\xi) - \frac{1}{2\mu} \|\xi - \lambda\|^2 \right), \tag{15}$$

where λ is a so-called augmented Lagrange multiplier or just multiplier. The proximal approximation $\zeta(\lambda)_\mu$ is concave ($-\zeta(\lambda)_\mu$ is convex) and has a Lipschitz constant of $\frac{1}{\mu}$. Moreover the gradient of the proximal approximation is

$$\nabla \zeta(\lambda)_\mu = \Psi \hat{s} - \psi, \tag{16}$$

where $\hat{s}$ is the minimizer of

$$\min_{s \in \mathcal{S}} f(s) + \frac{\mu}{2} \|\Psi s - \psi - \frac{1}{\mu}\lambda\|^2. \tag{17}$$

The augmented Lagrangian method solves the problem (14) employing (16) and (17). So the method consists of an outer iteration called multiplier update and an inner subproblem, as outlined in Algorithm 2. The standard method to update the multipliers is a gradient method given by

$$\lambda^{j+1} = \lambda^j - \mu \nabla \zeta(\lambda^j) = \lambda^j - \mu(\Psi \hat{s}^j - \psi), \tag{18}$$

where $\hat{s}^j$ is an (approximate) minimizer of the subproblem (17). The second possibility is to use the fast gradient method for the multiplier update. At the start of the algorithm we set $\lambda_v^0 = \lambda^{-1} = \lambda^0$ and then solve the subproblem (17) using the multiplier λ_v^j. Afterwards we update via

$$\begin{aligned} \lambda^{j+1} &= \lambda_v^j - \mu \nabla \zeta(\lambda_v^j) = \lambda_v^j - \mu(\Psi \hat{s}^j - \psi) \\ \lambda_v^{j+1} &= \lambda^{j+1} + \frac{j-1}{j+2}(\lambda^{j+1} - \lambda^j). \end{aligned} \tag{19}$$

Note that using the fast gradient method for the multiplier update in combination with an inexact solution of the subproblem might be problematic due to the inherent error accumulation of the fast gradient method, see [4]. However if we solve the subproblem precise enough, then the fast gradient based update might be faster as the gradient update.

Finally, we consider the second order update [2], [3], [13]

$$\begin{aligned} \lambda^{j+1} &= \lambda^j - (M_I^j)^{-1} \nabla \zeta(\lambda^j) = \lambda^j - (M_I^j)^{-1}(\Psi \hat{s}^j - \psi), \\ M_I^j &= \frac{\partial \nabla \zeta(\lambda^j)}{\partial s}^T (P_a^j)^{-1} \frac{\partial \nabla \zeta(\lambda^j)}{\partial s} = \Psi^T (P_a^j)^{-1} \Psi \\ P_a^j &= \frac{\partial^2 f}{\partial s^2}(\hat{s}^j) + \mu \Psi \Psi^T. \end{aligned} \tag{20}$$

Note that P_a^j is the Hessian matrix of (17), cf. [3], [13].

Remark 1: (Choice of μ and Convergence) Since $f(s)$ is convex any value of $\mu > 0$ guarantees convergence of this method, see [3], for first order updates (18), (19), if the subproblem is solved exact. Larger values of μ lead to an improved convergence, but can lead to a more difficult subproblem, see Section IV-B. In contrast for the second order update convergence cannot be guaranteed globally.

Algorithm 2 Augmented Lagrangian method

Require: Initial guess λ^0, number of iterations j_{max}

1: **for** $j = 1, \ldots, j_{max}$ **do**
2: Solve subproblem (17) and obtain the minimizer $\hat{s}^j$
3: Multiplier update: obtain λ^{j+1}, (λ_v^{j+1}) using (18), (19) or (20)
4: **end for**
5: **return** $\lambda^{j_{max}}$, $\hat{s}^{j_{max}}$

IV. Main result: Application to MPC problem

In this Section we present the proposed algorithm tailored for MPC. We first consider the outer iteration with the multiplier update and than the subproblem.

A. Outer iteration and multiplier updates

We apply the outlined augmented Lagrangian method to the MPC problem. First, observe that the gradient of the proximal approximation $\nabla \zeta(v)_\mu$ (16) for (8) is

$$\nabla \zeta(v)_\mu = C\hat{z} - e, \tag{21}$$

where C and e are as in (9), (11) and where $\hat{z}$ is the minimizer of the subproblem (17). For the QP underlying the MPC problem this subproblem is

$$\begin{aligned} \hat{z} = \arg\min_z \frac{1}{2} z^T H z + z^T g + \frac{\mu}{2} \|Cz - e - \frac{1}{\mu} v\|^2 \\ s.t. \; \underline{z} \leq z \leq \overline{z}, \end{aligned} \tag{22}$$

where H, g, C and e are as in (9), (10), (11) and the inequality constraints are defined as in (7). For the second order update (20) we need the matrix P_a^j, which is given by

$$P_a^j = H_a = H + \mu C^T C, \tag{23}$$

is the Hessian matrix of (22).

B. Solving the subproblem

Let us now consider the QP arising as the subproblem from the augmented Lagrangian method above, which is given by (22). Note that, this problem is always feasible, e.g. $z = \overline{z}$ is feasible. If we define g_a as

$$g_a(x_k, v) = g(x_k) - \mu C^T e(x_k) - \mu C^T v, \tag{24}$$

and use H_a as defined in (23) and drop the constant term, then we can rewrite the problem as

$$\begin{aligned} \min_z \frac{1}{2} z^T H_a z + z^T g_a(x_k, v) \\ s.t.\ \underline{z} \leq z \leq \overline{z}. \end{aligned} \tag{25}$$

In detail, the matrix H_a is sparse and has the structure

$$H_a = \begin{pmatrix} \phi_1 & \phi_2 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ \phi_2^T & \phi_3 & \phi_4 & \phi_5 & 0 & \dots & 0 & 0 & 0 \\ 0 & \phi_4^T & \phi_1 & \phi_2 & 0 & \dots & 0 & 0 & 0 \\ 0 & \phi_5^T & \phi_2^T & \phi_3 & \phi_4 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & \phi_3 & \phi_4 & \phi_5 \\ 0 & 0 & 0 & 0 & 0 & \dots & \phi_4^T & \phi_1 & \phi_2 \\ 0 & 0 & 0 & 0 & 0 & \dots & \phi_5^T & \phi_2^T & \phi_6 \end{pmatrix}.$$

where,

$$\phi_1 = R + \mu B^T B, \qquad \phi_2 = -\mu B^T \tag{26}$$

$$\phi_3 = Q + \mu(I + A^T A), \quad \phi_4 = S^T + \mu A^T B \tag{27}$$

$$\phi_5 = -\mu A^T, \qquad \phi_6 = P + \mu(I + F^T F). \tag{28}$$

These matrices have the dimensions $\phi_1 \in \mathbb{R}^{p\times p}$, $\phi_2 \in \mathbb{R}^{p\times n}$, $\phi_3 \in \mathbb{R}^{n\times n}$, $\phi_4 \in \mathbb{R}^{n\times p}$, $\phi_5 \in \mathbb{R}^{n\times n}$ and $\phi_6 \in \mathbb{R}^{n\times n}$. Note that H_a has at most $N(n^2 + p^2 + 2np) + (N-1)(2n^2 + 2np)$ nonzero entries, because ϕ_1, ϕ_2, ϕ_2^T appear N times in H_a and $\phi_3, \phi_4, \phi_4^T, \phi_5, \phi_5^T$ appear $N-1$ times and ϕ_6 appears once in H_a. Moreover, the block-diagonal matrix H_a depends neither on the multiplier v nor on the state x_k, which simplifies the following analysis.

In order to solve the QP (25) we use the Fast Gradient method, presented in Section III-A. Note that this method employs two parameters, the Lipschitz constant of the gradient L and the strong convexity parameter, ϕ. Since the cost function is quadratic, these parameters can easily be obtained from the matrix H_a. In particular $L = \lambda_{Max}(H_a)$ is the largest eigenvalue of H_a and $\phi = \lambda_{Min}(H_a)$ is the smallest eigenvalue of H_a. Due to our assumptions the smallest eigenvalue of H_a is positive, see also Proposition 1.

Note that the condition number of the matrix H_a is $\mathrm{cond}(H_a) = \frac{\lambda_{Max}(H_a)}{\lambda_{Min}(H_a)} = \frac{L}{\phi}$ and thus governs the convergence rate of the fast gradient method, cf. Section III-A. Therefore we are able to analyze the influence of the horizon length N as well as the penalty parameter μ onto the conditioning of H_a.

First we discuss the impact of μ onto the eigenvalues of H_a. Note that $C^T C$ is singular, due to the assumption $m < Np$, thus

$$\lambda_{Min}(H_a) \geq \lambda_{Min}(H) \tag{29}$$

$$\lambda_{Max}(H_a) \leq \lambda_{Max}(H) + \mu\lambda_{Max}(C^T C), \tag{30}$$

which yields the following proposition.

Proposition 1: (Influence of μ on the conditioning of H_a) The condition number of the matrix H_a (23) is bounded above by an affine, increasing function in μ. In particular,

$$\mathrm{cond}(H_a) \leq \frac{\lambda_{Max}(H) + \mu\lambda_{Max}(C^T C)}{\lambda_{Min}(H)}. \tag{31}$$

The proof follows directly from the discussion above.

From this proposition it follows that the inner problem might get harder to solve for increasing μ.

Now let us investigate the influence of the horizon length N onto the condition number.

Proposition 2: (Influence of N on the conditioning of H_a) For a given system (1), weighting (3) and penalty parameter μ there exists, independent of N, an upper bound $\overline{K}$ and lower bound $\underline{K}$ on the eigenvalues of H_a, and thus an upper bound on the condition number $\mathrm{cond}(H_a) \leq \overline{K}\underline{K}^{-1}$.

Proof: The lower bound follows from the fact that H_a is given as in (23) by the sum of a positive definite matrices H and the positive semi-definite $\mu C^T C$ (we assume $m < Np$). Finally from the structure of H (10) we obtain

$$\lambda_{Min}(H_a) \geq \min(\lambda_{Min}(W), \lambda_{Min}(R), \lambda_{Min}(T)) = \underline{K}.$$

Now let us determine the upper bound on the eigenvalues. First due to the spectral radius property of the 1-norm we have $\lambda_{Max}(H_a) \leq ||H_a||_1$. Finally, due to basic properties of the 1-norm and the structure of H_a we obtain

$$\begin{aligned} ||H_a||_1 = \max(&|| \begin{pmatrix} \phi_5^T & \phi_2^T & \phi_3^T & \phi_4 & \phi_5 \end{pmatrix}^T ||_1, \\ &|| \begin{pmatrix} \phi_4^T & \phi_1^T & \phi_2 \end{pmatrix}^T ||_1, || \begin{pmatrix} \phi_5^T & \phi_2^T & \phi_6^T \end{pmatrix}^T ||_1), \end{aligned}$$

i.e. $||H_a||_1 = \overline{K}$ is independent of N and thus $\lambda_{Max}(H_a)$ is bounded above for any N by $\overline{K}$. ∎

Therefore increasing N has only a limited impact on the convergence rate of the fast gradient method, cf. [16].

C. Implementation

We consider first the implementation of the outer iteration (Algorithm 1) and second we discuss the implementation of the fast gradient method for the subproblem (Step 4).

1) Multiplier updates: For all multiplier updates we need to evaluate (21). We can compute $C\hat{z}$ efficiently by exploiting the sparsity structure of C, cf. (9). This requires only $O(Nn^2)$ calculations, assuming $p \leq n$. Secondly, only the first n entries of $e(x_k)$ depend on x_k, i.e. those need to be updated once per MPC step.

For the gradient based update (18) and fast gradient based update (19) we can avoid the online multiplication of C, $e(x_k)$ by scaling the data by μ offline appropriately.

The second order update (20) can be implemented straightforward: Since $M^j = M = C(H_a)^{-1}C^T$ is constant, we can determine $(M^j)^{-1}$ offline.

2) Subproblem: First note that the gradient of the subproblem (25) is

$$\frac{\partial J_a}{\partial z} = H_a z + g_a(x_k, v). \tag{32}$$

We can use the sparse structure of H_a to compute $H_a z$. This requires only $O(Nn^2)$ computations assuming $p \leq n$. Moreover, $g_a(x_k, v)$ consists of three different parts

$$g_a(x_k, v) = \tilde{g}_a + g_a(x_k) + g_a(v). \tag{33}$$

First the constant term $\tilde{g}_a$, which can be zero, is

$$\tilde{g}_a = \begin{pmatrix} \tilde{r} - \mu B^T \tilde{d} \\ \tilde{q} - \mu(A^T \tilde{d} - \tilde{d}) \\ \vdots \\ \tilde{r} - \mu B^T \tilde{d} \\ \tilde{t} - \mu(F^T \tilde{f} - \tilde{d})) \end{pmatrix}. \tag{34}$$

Second there are terms, which depend on x_k or on v

$$g_a(x_k) = \begin{pmatrix} Sx_k + \mu B^T A x_k \\ -\mu A x_k \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{35}$$

$$g_a(v) = \begin{pmatrix} -\mu B^T v_1 \\ \mu(v_1 - A^T v_2) \\ -\mu B^T v_2 \\ \vdots \\ -\mu B^T v_N \\ \mu(v_N - F^T v_{N+1}) \end{pmatrix}. \tag{36}$$

In summary, we need to recalculate $g_a(x_k, v)$ each time v or x_k have changed, but $\tilde{g}_a$ is constant and $g_a(x_k)$ need to be updated only after x_k has changed.

The fast gradient algorithm requires also the projection of z into the set $\underline{z} \leq z \leq \overline{z}$, i.e. the box constraints. Since these constraints are separable, this is only a entry-wise saturation and thus $O(N(n+p))$. We have

$$z[i] = \begin{cases} \overline{z}[i], & \text{if } z[i] > \overline{z}[i] \\ \underline{z}[i], & \text{if } z[i] < \underline{z}[i] \\ z[i], & \text{else,} \end{cases} \tag{37}$$

where $z[i]$ denotes the ith entry of z.

3) Warmstarting : Note that, we need initial guesses for z to solve the subproblem (25) and initial guesses for the multipliers λ. We use use warmstarting to improve the performance, i.e., the initial guess are based on previous solution. There are two different cases for warmstarting. First, if we need to solve the subproblem after a multiplier update, then we just use the last obtained z as new initial guess. Second, if we need to do a new MPC step, i.e., x_k changes, then we need initial guesses for the optimization variables z and the multipliers λ. Here we use basically a shifted solution appended by zeros. For the multipliers λ this can be done similarly.

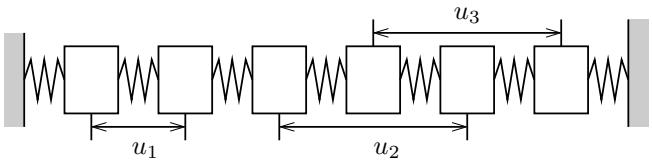

Fig. 1. Example: Chain of masses connected by springs.

D. Computational effort

In this section we discuss the memory demand and summarize the time complexity.

1) Memory demand: We first estimate the demand for dynamic data and later on the demand of static data, which is often crucial in embedded applications. For the outer iteration we need to store x_k and the first n entries of $e(x_k)$. Next we need to store $\hat{z}^j$, which has $N(n+p)$ entries. For the gradient update (18) the gradient to be determined, the old and the new multiplier vector have $3(Nn+m)$ elements. The other three updates have an increased memory effort of $4(Nn+m)$. Moreover solving the subproblems requires additional dynamic data. First, $g_a(x_k)$ has only $n+p$ nonzero entries. Second the fast gradient algorithm itself have a memory demand of $5N(n+p)$. In combination with $g_a(xk, v)$ and the gradient the subproblem has a memory demand of $(5N+1)(n+p)$. In summary the demand for dynamic data increases linearly in the horizon length N, the number of states n and the number of inputs p.

Let us consider now the static data, which stays constant during the run of the algorithm. For the (fast) gradient update (18) or (19), we need only C and the constant part of e. Using their structure, we need only to store a constant amount of data. The second order update 20) needs in addition M^{-1} and therefore has a memory demand of $O(Nn^2)$.

Also for the subproblem one can exploits the underlying structure of the problem data: $\phi_1, \ldots, \phi_6$ and S, A, B and F: have less as $10n^2$ elements. In addition, $\tilde{g}_a$ consists of three blocks of size n, p or m. Finally, the upper and lower bound of $\overline{z}$ and $\underline{z}$ have a memory demand of about $4n+2p$. Thus, the size of the static data is independent of N for the subproblem.

2) Time complexity: If we assume $p \leq n$, then as discusses above an iteration of the subproblem has a time complexity of order $O(Nn^2)$. The second order update (20) has a time complexity of $O((Nn)^2)$ due to the matrix-vector product with M^{-1}. All other multiplier updates have as discussed above a complexity of $O(Nn^2)$.

V. Example

We implemented the proposed algorithms in Matlab. In particular, we use a single thread of a 2.4 GHz Intel Core 2 Quad Q6600 Intel CPU to evaluate the algorithms and we utilize Embedded Matlab to generate native code.

As example we use the chain of mass example from [22] illustrated in Figure 1. This system has $n = 12$ states and

Algorithm	μ	j_{max}	i_{max}
Gradient update, (18),	50	4	14
Fast Gradient update, (19),	40	4	14
Second order update, (20),	3.5	8	5

TABLE I

CHOICE OF PENALTY PARAMETER μ, NUMBER OF MULTIPLIER UPDATES j_{max} AND ITERATIONS OF INNER PROBLEM i_{max}

Algorithm	ΔJ	T_{avg}	T_{max}
Gradient update, (18),	1.10%	2ms	2ms
Fast Gradient update, (19),	1.09%	2ms	2ms
Second order update, (20),	0.04%	2ms	2.1ms
Consed approach [9],	0.05%	1.9ms	1.9ms
qpOASES [5], (max. 20 its)	0.43%	2.4ms	5ms
qpOASES [5], (exact)	0%	2.5ms	8ms
quadprog (exact)	0%	46.6ms	153ms

TABLE II

COMPARISON OF ALGORITHMS FOR CHAIN OF MASSES EXAMPLE. T_{max} MAXIMUM COMPUTATION TIME OF AN MPC STEP, T_{avg} AVERAGE COMPUTATION TIME PER MPC STEP, ΔJ PERFORMANCE DECREASE (MONTE CARLO SIMULATIONS).

three actuators ($p = 3$), each acting on two masses. Moreover we use the same parameters: the spring constants are 1, each mass has a value of 1 and there is no damping. The actuators are limited to ± 0.5 and all states are limited to ± 4. The system is discretized using a sampling time of 0.5. In addition, there is as disturbance a random force $w(t) \in \mathbb{R}^n$, component-wise uniformly distributed on $[-0.5, 0.5]$, acting on each mass. Finally, the cost matrices are $N = 30$, $Q = R = T = I$ and $F = S = 0$.

In order to compare the different update schemes (18), (19) or (20) we first determined the possible number of multiplier updates, and iterations of the fast gradient method solving the subproblem (25) s.t. the MPC has a computational delay of 2ms. Next we tuned the penalty parameter μ using one Monte Carlo simulation by comparing the performance with exact MPC obtained using the QP solver *quadprog* of Matlab. Note that the Monte Carlo simulation are 20000 steps long and averaging is done over the last 90 % of the steps.

Table II shows the average performance decrease compared to exact MPC for the different update formulas obtained using 20 Monte Carlo simulations of the same size as above. Here, the second order update, yields the best results and is competitive with other algorithms as shown in Table II. The fast gradient and the gradient update deliver worse and similar results. For the same example the [22] report a computation time of about 5ms, but on a slower CPU (we cannot run the provided code).

VI. CONCLUSIONS

In this paper we presented a numerical efficient solution approach for model predictive control of discrete time linear, time-invariant systems with box constraints. The proposed algorithm solves the underlying quadratic program using a combination of augmented Lagrangians and the fast gradient method and exploits the sparsity of the setup. We discussed different multiplier update and showed that the condition number of the subproblem is bounded above for any horizon. Finally, we illustrated that the performance is competitive with other recent methods tailored for MPC.

The convergence behavior will be analyzed as well as further detailed comparisons with other already existing algorithms will be obtained in future work.

REFERENCES

[1] BEMPORAD, A., MORARI, M., DUA, V., AND PISTIKOPOULOS, E. The explicit linear quadratic regulator for constrained systems. *Automatica 38*, 1 (2002), 3–20.

[2] BERTSEKAS, D. P. *Nonlinear Programming*. Athena Scientific, 1995.

[3] BERTSEKAS, D. P. *Constrained Optimization and Lagrange multiplier methods*. Athena Scientific, 1996.

[4] DEVOLDER, O., GLINEUR, F., AND NESTEROV, Y. First-order Methods of Smooth Convex Optimization with Inexact Oracle. Available online at http://www.optimization-online.org.

[5] FERREAU, H. J., BOCK, H. G., AND DIEHL, M. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control 18* (2008), 816–830.

[6] FERREAU, H. J., ORTNER, P., LANGTHALER, P., DEL RE, L., AND DIEHL, M. Predictive control of a real-world diesel engine using extended online active set strategy. *Annual Reviews in Control 31* (2007), 293–301.

[7] FINDEISEN, R., BIEGLER, L., AND ALLGÖWER, F. *Assessment and Future Directions of Nonlinear Model Predictive Control*. Lecture Notes in Control and Information Sciences. Springer, 2008.

[8] GARCIA, C., PRETT, D., AND MORARI, M. Model predictive control: Theory and practice - A survey. *Automatica 25*, 3 (1989), 335–348.

[9] KÖGEL, M., AND FINDEISEN, R. Fast predictive control of linear, time-invariant systems using an algorithm based on the Fast gradient method and augmented Lagrange multipliers. 2011 MSC, To appear.

[10] KÖGEL, M., AND FINDEISEN, R. A Fast Gradient method for embedded linear predictive control. In *Proceedings of the IFAC World Congress 2011* (2011), pp. 1362–1367.

[11] MACIEJOWSKI, J. M. *Predictive Control with Constraints*. Prentice Hall, Upper Saddle River, New Jersey, 2002.

[12] MAYNE, D. Q., RAWLINGS, J. B., RAO, C. V., AND SCOKAERT, P. Constrained Model Predictive Control: Stability and Optimality. *Automatica 36* (2000), 789–814.

[13] MIJANGOS, E. An implementation of newton-like methods on nonlinearly constrained networks. *Computers & Operations Research 31*, 2 (2004), 181–199.

[14] MILMAN, R., AND DAVISON, E. J. A Fast MPC Algorithm Using Nonfeasible Active Set Methods. *Journal of Optimization Theory and Applications 139* (2008), 591–616.

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

[16] NESTEROV, Y. *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Acad. Publ., 2004.

[17] QIN, S., AND BADGWELL, T. A survey of industrial model predictive control technology. *Control Engineering Practice 11*, 7 (2003), 733–764.

[18] RAO, C., WRIGHT, S., AND RAWLINGS, J. Application of Interior-Methods to Model Predictive Control. *Journal of Optimization Theory and Applications 99* (1998), 723–757.

[19] RICHTER, S., JONES, C. N., AND MORARI, M. Real-Time Input-Constrained MPC Using Fast Gradient Methods. In *48th IEEE Conference on Decision and Control and 28th Chinese Control Conference, Shanghai* (2009), pp. 7387–7393.

[20] RICHTER, S., MARIETHOZ, S., AND MORARI, M. High-Speed Online MPC Based on a Fast Gradient Method Applied to Power Converter Control. In *Proceedings of the 2010 ACC* (2010), pp. 4737–4743.

[21] SHAHZAD, A., KERRIGAN, E. C., AND CONSTANTINIDES, G. A. Preconditioners for Inexact Interior Point Methods for Predictive Control. In *Proceedings of the 2010 American Control Conference* (2010), pp. 5714–5719.

[22] WANG, Y., AND BOYD, S. Fast Model Predictive Control Using Online Optimization. In *17th IFAC World Congress* (2008), pp. 6974–6979.