Numerical Methods for Efficient and Fast Nonlinear Model Predictive Control

Hans Georg Bock, Moritz Diehl, Peter Kühl, Ekaterina Kostina,
Johannes P. Schlöder, and Leonard Wirsching

Interdisciplinary Center for Scientific Computing (IWR), University of Heidelberg,
Im Neuenheimer Feld 368, 69120 Heidelberg, Germany
{bock,m.diehl}@iwr.uni-heidelberg.de

Summary. The paper reports on recent progress in the real-time computation of constrained closed-loop optimal control, in particular the special case of nonlinear model predictive control, of large differential algebraic equations (DAE) systems arising e.g. from a MoL discretization of instationary PDE. Through a combination of a direct multiple shooting approach and an initial value embedding, a so-called "real-time iteration" approach has been developed in the last few years. One of the basic features is that in each iteration of the optimization process, new process data are being used. Through precomputation - as far as possible - of Hessian, gradients and QP factorizations the response time to perturbations of states and system parameters is minimized. We present and discuss new real-time algorithms for fast feasibility and optimality improvement that do not need to evaluate Jacobians online.

1 Introduction

Feedback control based on an online optimization of nonlinear dynamic process models subject to constraints, and its special case, nonlinear model predictive control (NMPC) [1], is an emerging optimal control technique, mainly applied to problems in chemical engineering [17]. Currently, NMPC is also transferred to new fields of application such as automotive engineering, where the principal dynamics are much faster. Among the advantages of NMPC are the capability to directly handle equality and inequality constraints as well as the flexibility provided in formulating the objective function and the process model. Lately, a major aim has become to develop algorithms that are able to treat *large-scale* nonlinear first principle models without further need of re-modeling or model reduction.

In this paper we present and investigate several variants of the "real-time iteration" approach to online computation of constrained optimal feedback control laws. The present realization of this approach is based on the direct multiple shooting method [5] for DAE models [15]. Some of the ideas in this paper, and details on the standard variant of the real-time iteration approach can e.g. be found in [3, 8, 10]. The main contribution of this paper is to bring together three

R. Findeisen et al. (Eds.): Assessment and Future Directions, LNCIS 358, pp. 163–179, 2007.
springerlink.com

real-time iteration levels (first introduced in [2]) that lead to cheap feasibility or optimality refinements with very fast response times for feedback. They are based on approximating both the Hessian of the optimization problem as well as the constraint Jacobians. The overall idea behind the presented ideas is to move away from the paradigm of solving the optimal control problem inherent to NMPC to convergence during each sampling period. Instead, different levels of real-time iterations are proposed that always use the most current information from the evolving process and allow to stay close to the – also evolving – optimal NMPC solution. It is important to note that the presented refinement strategies work in the presence of inequality constraints and active set changes.

A dynamic mechanical model is used to demonstrate that NMPC based on the proposed real-time iteration variants enters new time scales in computation time in the range of milliseconds.

Overview

In Section 2 we give some background material on the classical "direct" multiple shooting method for optimization problems subject to instationary differential equations, and outline in Section 3 the major ideas of the standard "real-time iteration" scheme. In Section 4 three real-time iteration variants are presented that largely avoid costly approximations of Jacobians and Hessians as well as decompositions during the runtime of the online algorithm, and therefore are particularly suitable for large scale models:

- In the linearized optimal feedback control variant presented in Section 4.2, only a matrix vector multiplication and a solution of a small scale quadratic program (QP) are necessary online.
- In the "feasibility improving" suboptimal feedback control variant presented in Section 4.3, one additional forward simulation of the nonlinear DAE system is necessary. In the limit, this variant yields feasible, but only approximately optimal controls.
- In the online approach in Section 4.4 also the gradient of the Lagrangian is needed. In the limit this method yields both feasible and optimal controls, at still lower costs than the standard real-time iteration.

In Section 5, the last variant is used to control the motion of a chain of balls connected by springs. This example demonstrates the real-time character of the presented schemes. The paper concludes with a summary and final remarks in Section 6.

2 Direct Multiple Shooting to Solve NMPC Problems

In this section we prepare the ground for the different variants of the *real-time iteration scheme*. First, the model class is presented and the open-loop optimal control problem for NMPC is set up. Then, we briefly review the direct multiple shooting approach which forms the basis of the real-time iteration variants to be discussed.

2.1 Differential Algebraic Equation Systems

Throughout this paper, we consider DAE models of index one in the following form

$$B(x(t), z(t), u(t), p)\, \dot{x}(t) = f(x(t), z(t), u(t), p) \tag{1}$$

$$0 = g(x(t), z(t), u(t), p) \tag{2}$$

Here, x and z denote the differential and the algebraic state vectors, respectively, u is the vector valued control function, whereas p is a vector of system parameters. This equation type covers many problems in practical engineering applications, from systems of ODE to reactive flow problems, e.g. the Navier-Stokes equation with chemical reactions. For the sake of simplicity we restrict ourselves in this paper to DAE of index 1, however, the generalization to higher index problems by reduction to index 1 problems with invariants can be derived following well-known techniques [18]. For notational simplicity, we will omit the parameters p in the following.

2.2 Nonlinear Model Predictive Control

Given a (possibly estimated) system state x_0, a Nonlinear Model Predictive Control (NMPC) scheme obtains a feedback control $\bar{u}(x_0)$ from the solution of an open-loop optimal control problem on a prediction and control horizon $[0, T_p]$ with length T_p:

$$\min_{u(\cdot),x(\cdot),z(\cdot)} \int_0^{T_p} L(x(t), z(t), u(t))\, dt + E(x(T_p)) \tag{3a}$$

$$\text{subject to} \quad x(0) = x_0 \tag{3b}$$

$$B(\cdot)\dot{x}(t) = f(x(t), z(t), u(t)), \quad \forall t \in [0, T_p], \tag{3c}$$

$$0 = g(x(t), z(t), u(t)), \quad \forall t \in [0, T_p], \tag{3d}$$

$$0 \leq h(x(t), z(t), u(t)), \quad \forall t \in [0, T_p], \tag{3e}$$

$$0 \leq r\left(x\left(T_p\right)\right). \tag{3f}$$

Here, (3b) denotes the initial value constraint and (3c,3d) the DAE system. Additional state and control inequality constraints are expressed in (3e), and (3f) are terminal constraints that have to be satisfied.

Solving this problem for a given initial value x_0, we obtain an open-loop optimal control $u^*(t; x_0)$ and corresponding state trajectories $x^*(t; x_0), z^*(t; x_0)$. Based on this solution, a constrained nonlinear feedback control law is given by

$$\bar{u}(x_0) := u^*(0; x_0). \tag{4}$$

Due to its origin from an optimal control formulation, the NMPC feedback law has several appealing properties: among them are the possibility to base the feedback on economic criteria, to make use of important process knowledge in the form of nonlinear first principle models, and to include constraints (3e)

in a straightforward way. Given suitable choices of the objective function and the final state constraint (3f), stability of the nominal NMPC dynamics can be proven [6, 7, 16].

The present article is concerned with efficient ways to calculate the feedback control $\bar{u}(x_0)$ or a suitable approximation *in real-time* while the considered process moves on.

2.3 Direct Multiple Shooting for DAE

Our approaches to the online solution of the optimal control problem (3a)–(3f) – the real-time iteration schemes – are based on the direct multiple shooting method [5] for DAE models [15], which is briefly reviewed in this section.

Parameterization of the Infinite Optimization Problem

The parameterization of the infinite optimization problem consists of two steps. For a suitable partition of the time horizon $[0, T_p]$ into N subintervals $[t_i, t_{i+1}]$,, $0 = t_0 < t_1 < \ldots < t_N = T_p$, not necessarily equidistant, we first parameterize the control function u as $u(t) = \phi_i(t, u_i)$ for $t \in [t_i, t_{i+1}]$.

Note that any parameterization ϕ_i with local support can be used without changing the structure of the problem as analyzed in the next sections.

In a second step, the DAE solutions are parameterized by *multiple shooting*. For simplicity of presentation we choose the same grid points here as for the controls. The DAE solution is decoupled on the N intervals $[t_i, t_{i+1}]$ by introducing the initial values s_i^x and s_i^z of differential and algebraic states at times t_i as additional optimization variables.

On each subinterval $[t_i, t_{i+1}]$ independently, the trajectories $x_i(t)$ and $z_i(t)$ can be computed as solutions of an initial value problem:

$$B(\cdot)\dot{x}_i(t) = f(x_i(t), z_i(t), \phi_i(t, u_i)) \tag{5a}$$

$$0 = g(x_i(t), z_i(t), \phi_i(t, u_i)) - \alpha_i(t) g(s_i^x, s_i^z, \phi_i(t_i, u_i)) \tag{5b}$$

$$x_i(t_i) = s_i^x, \; z_i(t_i) = s_i^z \tag{5c}$$

Here, the subtrahend in (5b) is deliberately introduced to relax the DAE and allow an efficient solution for initial values and controls s_i^x, s_i^z, u_i that may violate temporarily the consistency conditions (3d). This allows to avoid consistency iterations at the start of the integration. The scalar damping factor $\alpha_i(t)$ is chosen such that $\alpha_i(t_i) = 1$, and $\alpha_i(t) > 0$ is non-increasing on $t \in [t_i, t_{i+1}]$. Consistency of the algebraic states is ensured by adding consistency conditions (7c) for each multiple shooting node at t_i in the overall NLP defined in the next section. For more details on the relaxation of the DAE the reader is referred, e.g. to [4, 15, 18].

Since the trajectories $x_i(t)$ and $z_i(t)$ on the interval $[t_i, t_{i+1}]$ are functions of the initial values $s_i := (s_i^x, s_i^z)$ and control parameters u_i only, they will be referred to as $x_i(t; s_i, u_i)$ and $z_i(t; s_i, u_i)$ in the following. The integral part of the cost function is evaluated on each interval independently:

$$L_i(s_i, u_i) := \int_{t_i}^{t_{i+1}} L(x_i(t), z_i(t), \phi_i(t, u_i))\, dt. \tag{6}$$

Note that up to now the multiple shooting parameterization does not involve any discretization of differential operators f, g, but is exact.

Structured Nonlinear Programming Problem

The parameterization of problem (3a)–(3f) using multiple shooting and a suitable control representation leads to the following structured nonlinear programming (NLP) problem :

$$\begin{aligned}
\min_{u,s} \quad & \sum_{i=0}^{N-1} L_i(s_i, u_i) \quad + \quad E(s_N^x) && \text{(7a)}\\
\text{subject to} \quad & s_0^x = x_0, && \text{(7b)}\\
& 0 = g(s_i^x, s_i^z, \phi_i(t_i, u_i)), \qquad i = 0, 1, \ldots N-1, && \text{(7c)}\\
& s_{i+1}^x = x_i(t_{i+1}; s_i, u_i), \qquad i = 0, 1, \ldots N-1, && \text{(7d)}\\
& r(s_N^x) \geq 0, && \text{(7e)}
\end{aligned}$$

with initial condition (7b), consistency conditions (7c), continuity conditions (7d), and terminal constraint (7e). Additional control and path constraints are supposed to be imposed pointwise for a suitable discretization (at n_i points τ_{ij} on each interval, $\tau_{ij} \in [t_i, t_{i+1})$, $j = 0, \ldots, n_i - 1$)

$$h(x_i(\tau_{ij}; s_i, u_i), z_i(\tau_{ij}; s_i, u_i), u_i) \geq 0, \quad j = 0, \ldots, n_i - 1, \quad i = 0, \ldots N-1. \tag{7f}$$

The NLP (7a)–(7e) can be summarized as

$$P(x_0): \qquad \min_{w} \quad a(w) \quad \text{subject to} \quad \begin{cases} b_{x_0}(w) = 0 \\ c(w) \geq 0, \end{cases} \tag{8}$$

where w contains all the multiple shooting state variables and controls:

$$w = (s_0^x, s_0^z, u_0, s_1^x, s_1^z, u_1, \ldots, u_{N-1}, s_N^x) \in \mathbb{R}^{n_w}.$$

The function $a(w)$ is the objective (7a), the vector valued equation $b_{x_0}(w) = 0$ summarizes all equalities from (7b)-(7d), and the vector valued $c(w) \geq 0$ contains the inequality constraints (7f) and (7e).

It is important to note that the initial condition (7b) is a linear constraint among the equality constraints, with the varying parameter x_0 entering linearly only in this constraint, so that

$$b_{x_0}(w) = \begin{bmatrix} s_0^x - x_0 \\ g(s_0^x, s_0^z, \phi_0(t_0, u_0)) \\ s_1^x - x_0(t_1; s_0^x, s_0^z, u_0) \\ \vdots \end{bmatrix} = b_0(w) + L x_0 \quad \text{with} \quad L := \begin{bmatrix} -\mathbb{I}_{n_x} \\ 0 \\ 0 \\ \vdots \end{bmatrix}. \tag{9}$$

Structure of the NLP

Due to the deliberate choice of state and control parameterizations, in [5] it was observed that the NLP problem (8) has a particular structure: its Lagrangian $\mathcal{L}_{x_0}(w,\lambda,\mu) = a(w) - \lambda^T b_{x_0}(w) - \mu^T c(w)$ (with Lagrange multipliers λ and μ) is *partially separable* so that its Hessian $\nabla_w^2 \mathcal{L}(w,\lambda,\mu)$ is *block diagonal*, and obviously independent of x_0 (such that we drop the index x_0 in $\nabla_w^2 \mathcal{L}_{x_0}$). Similarly, the multiple shooting parameterization introduces a characteristic *block sparse structure* of the constraint Jacobian that is also independent of x_0, e.g.[1]

$$\nabla_w b(w)^T = \begin{pmatrix} \mathbb{I} & & & & \\ Z_0^x & Z_0^z & Z_0^u & & \\ -X_0^x & -X_0^z & -X_0^u & \mathbb{I} & \\ & & & \ddots & \ddots & \ddots & \ddots \end{pmatrix}. \tag{10}$$

Furthermore, if the variables w are split into the *state trajectory* $s := (s_0^x,\ s_0^z,\ s_1^x,\ s_1^z,\ \ldots,\ s_N^x)$ and the *control trajectory* $u := (u_0,\ u_1,\ \ldots,\ u_{N-1})$, it is easily seen that $\nabla_s b(w)^T$ is nonsingular, a property to be exploited in the QP linear algebra.

A Newton-Type Method Solution Framework

Throughout the paper, we will work within a *Newton-type method* framework for the solution of the NLP (8). Starting with an initial guess (w_0, λ_0, μ_0), a standard full step iteration for the NLP is

$$w_{k+1} = w_k + \Delta w_k, \tag{11}$$

$$\lambda_{k+1} = \lambda_k^{\mathrm{QP}}, \quad \mu_{k+1} = \mu_k^{\mathrm{QP}}, \tag{12}$$

where $(\Delta w_k, \lambda_k^{\mathrm{QP}}, \mu_k^{\mathrm{QP}})$ is the solution of a quadratic program (QP). Other than in the classical Gauss-Newton or SQP approaches, we will rather use the more convenient form of the QP

$$\begin{array}{ll} \min\limits_{\Delta w \in \mathbb{R}^{n_w}} & \frac{1}{2}\ \Delta w^T\ A_k\ \Delta w \quad + \quad a_k^T \Delta w \\ \text{subject to} & \begin{cases} b_{x_0}(w_k) + B_k \Delta w = 0 \\ \quad c(w_k) + C_k \Delta w \geq 0 \end{cases} \end{array} \tag{13}$$

where $A_k \approx \nabla_w^2 \mathcal{L}(w_k, \lambda_k, \mu_k)$ is an approximation of the Hessian of the Lagrangian, $a_k = \nabla_w \mathcal{L}(w_k, \lambda_k, \mu_k) + B_k^T \lambda_k + C_k^T \mu_k$, and B_k and C_k are approximations of the constraint Jacobians. Depending on the errors of these approximations we may expect linear or even super-linear convergence (see [13] for more information on quasi-Newton Jacobian updates, where superlinear convergence is proved under mild assumptions). These errors however do not influence

[1] We use the definition $\{\nabla_x f\}_{ij} := \frac{\partial f_j}{\partial x_i}$ throughout the paper.

the accuracy of the solution of the NLP which only depends on the (discretization) errors made in the evaluation of $\nabla_w \mathcal{L}$, b_{x_0} and c. In this paper we restrict ourselves to the mentioned full step iteration. In the case of NMPC where a sequence of neighboring problems is solved, this turns out to be sufficiently robust and offers the advantage of fast convergence.

3 Initial Value Embedding and Real-Time Iterations

In theoretical approaches towards constrained feedback control, including NMPC, optimal control problems have to be solved online for *varying* initial values x_0. To emphasize the dependence on a varying x_0 we write the problems (7a)–(7e) resp. (8) as $P(x_0)$. An obvious question then is how to determine an initial guess w_0 for the Newton-type iterations in each problem $P(x_0)$.

3.1 Initial Value Embedding

From previous optimization steps a solution $w^*(x_0')$ of a neighboring optimization problem $P(x_0')$ is known, including multipliers $\lambda^*(x_0')$ and $\mu^*(x_0')$. A conventional approach hence would be to use the latest information available, namely to use the old control trajectory, and to compute new state trajectory by integrating the DAE over the whole horizon using the old control trajectory and the new initial state x_0.

Instead, the principle of the initial value embedding suggests *not* to make use of x_0, but to use the solution of the previous problem $P(x_0')$ *without any modification*. This initialization for the current problem $P(x_0)$ results, of course, in a violation of the initial value constraint (7b) in the NLP (7a)–(7e), because $s_0^x = x_0' \neq x_0$. However, the constraint is already perfectly satisfied after the first full step Newton-type iteration, due to its linearity. The formulation of the initial value constraint (7b) in the NLP (7a)–(7e) can be considered a linear embedding of each optimization problem into the manifold of perturbed problems, therefore the name "initial value embedding". It allows for an efficient transition from one optimization problem to the next.

In practical applications one observes that the first iteration already yields an excellent approximation of the solution. Indeed, one can show as an obvious application of the implicit function theorem (assuming the classical regularity properties) in the case considered (exact Jacobian and Hessian, initialization at solution of $P(x_0')$), that the first QP solution delivers a tangential predictor w_1 to the solution $w^*(x_0)$ of $P(x_0)$

$$\|w_1 - w^*(x_0)\| = O\left(\|x_0' - x_0\|^2\right).$$

It is remarkable that this property even holds if the change from x_0' to x_0 requires a change of the active set, which can be proven under mild conditions [8].

Also note that the solution of the first QP not only gives us directional sensitivity feedback in a small neighborhood of x_0' where the active set does not

change anymore, but in an even larger neighborhood where the linearization is still valid, see the illustration in Figure 1. In the case that only approximations of Jacobian and Hessian are used within the QP, we still obtain

$$\|w_1 - w^*(x_0)\| \leq \kappa \|x_0' - x_0\|,$$

with κ being small if the quality of the approximations is good.

It is interesting to note that the good prediction properties are independent from the class of optimization problems. Therefore, the scheme can be applied to solve both tracking problems and problems with an economic objective. An example for the latter can be found in [14].

3.2 Standard Real-Time Iteration Scheme

Let us now consider the full real-time scenario, where we want to solve a sequence of optimization problems $P(x(t))$ where $x(t)$ is the system state that changes continuously with time and which is used as initial value x_0 in problem (8).

In the standard real-time iteration scheme [8, 10] we proceed as follows:

Start with an initial guess (w_0, λ_0, μ_0), and perform the following steps for $k = 0, 1, \ldots$:

1. Preparation: Based on the current solution guess (w_k, λ_k, μ_k), compute all functions and their exact Jacobians that are necessary to build the QP (13), and prepare the QP solution as far as possible without knowledge of x_0 (see Section 4.1 for more details). This corresponds to the initialization needed for the initial value embedding stated above.
2. Feedback Response: at time t_k, obtain the initial value $x_0 := x(t_k)$ from the real system state; solve the QP (13) to obtain the step $\Delta w_k = (\Delta s^x_{0\,k}, \Delta s^z_{0\,k}, \Delta u_{0k}, \ldots)$, and give the approximation $\tilde{u}(x(t_k)) := u_{0k} + \Delta u_{0k}$ immediately to the real system.
3. Transition: Set the next solution guess as

$$w_{k+1} := w_k + \Delta w_k, \quad \lambda_{k+1} := \lambda_k^{\mathrm{QP}}, \quad \text{and} \quad \mu_{k+1} := \mu_k^{\mathrm{QP}}.$$

3.3 Nominal Stability of the Real-Time Iteration Scheme

A central question in NMPC is nominal stability of the closed loop. For the real-time iteration scheme, the state vector of the closed loop consists of the real system state $x(t_k)$ and the content (w_k, λ_k, μ_k) of the prediction horizon in the optimizer. Due to the close connection of system and optimizer, stability of the closed loop system can only be addressed by combining concepts from both, NMPC stability theory and convergence analysis of Newton-type optimization methods. For the standard real-time iteration scheme this analysis has been carried out in [11], and for a related scheme with shift in [12]. In these papers, proofs of nominal stability of the closed loop are given under reasonable assumptions. The class of feedback controls for shrinking horizon problems is treated in [9].

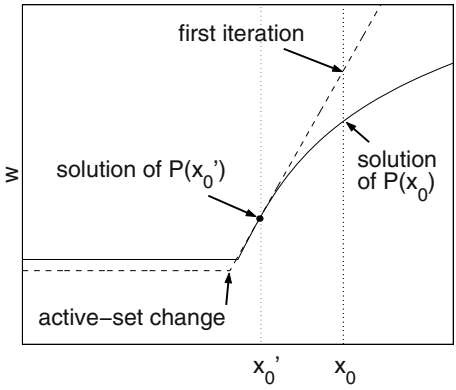

Fig. 1. Solution manifold (solid line) and tangential predictor after initial value embedding (dashed line), when initialized with the solution of $P(x_0')$. The first iteration already delivers a good predictor for the exact solution of $P(x_0)$.

4 Real-Time Iteration Variants

In the standard version of the real-time iteration scheme the time for each cycle corresponds to the time of one SQP iteration. In this article, however, we discuss four different levels of the real-time iteration scheme that differ in their need to evaluate Jacobians online. The basic idea for all these variants is to replace the QP (13) in the standard real-time iteration scheme by a generic approximated QP. This QP leaves the Hessian A as well as the Jacobians B and C constant and contains only parts of new information; a possible choice for A, B, C is $A := \nabla_w^2 \mathcal{L}(\bar{w}, \bar{\lambda}, \bar{\mu})$, $B := \nabla_w b(\bar{w})^T$, and $C := \nabla_w c(\bar{w})^T$ at some reference solution $(\bar{w}, \bar{\lambda}, \bar{\mu})$. In the following, $x_k := x(t_k)$ is the current system state at time t_k:

$$\min_{\Delta w \in \mathbb{R}^{n_w}} \quad \frac{1}{2}\, \Delta w^T\, A\, \Delta w + a_k^T \Delta w \tag{14a}$$

$$\text{subject to} \quad L x_k + b_k + B \Delta w = 0 \tag{14b}$$

$$c_k + C \Delta w \geq 0 \tag{14c}$$

The methods, that differ by the choices of a_k, b_k, c_k, proceed by performing the same three steps as in the standard real-time iteration scheme presented in Section 3.2, with the only difference that now the approximated version (14) is prepared and solved in each iteration instead of a QP (13) with exact Jacobians. As the matrices A, B, C are constant, a large share of the computations for preparation and solution of the QP can for all variants already be performed offline, leading to a considerably shorter preparation phase.

We want to point out that for strongly nonlinear processes it might be necessary to update the matrices A, B, C from time to time to ensure sufficient

contractivity of the real-time iterates, but that we leave this rare updating unconsidered here for simplicity of presentation. For mildly nonlinear systems, however, the matrices A, B, C might really be kept constant without any updates, for example evaluated once for all at a reference solution.

In what follows, three different variants of the real-time iteration scheme will be shown in detail, differing in the choice of a_k, b_k, and c_k. While variant A is nothing else than linear MPC, variant B converges to nonlinearly feasible (but suboptimal) MPC solutions. Variant C will even converge to the true nonlinear MPC feedback - without the need to evaluate any derivative matrix online. But before we proceed, a preliminary remark on condensing of a QP is necessary.

4.1 A Prerequisite: Offline Condensing

In all approaches we use fixed approximations of the Jacobians B and C, e.g. by evaluating offline $\nabla_w b(\bar{w})^T$ and $\nabla_w c(\bar{w})^T$ for a reference trajectory $\bar{w}$ that may be an exact or approximate solution of an NLP $P(\bar{x})$ for some state $\bar{x}$. We also use a fixed approximation A of the Hessian, that may be based on the reference solution of $P(\bar{x})$ and computed as $\nabla_w^2 \mathcal{L}(\bar{w}, \bar{\lambda}, \bar{\mu})$, or be chosen otherwise. Online, we use these fixed components A, B, C to formulate a QP of the form (14), where only the vectors a_k, b_k, c_k and the initial value x_k are changing online. It is well known that because $\nabla_s b(\bar{w})$ is invertible, the online QP solution can be prepared by a *condensing* of the QP [5, 8]: We divide Δw into its state and control components Δs and Δu, and resolve the equality constraints (14b) to obtain Δs as a linear function of Δu (and x_k), such that we can substitute

$$\Delta w = m(b_k) + \tilde{L} x_k + M \Delta u. \tag{15}$$

Note that the matrices $\tilde{L}$ and M are independent of a_k, b_k, c_k, x_k and can in all variants be precomputed offline, exploiting the structure of B in Eq. (10). In Eq. (17) below, we show how $m(b_k)$ can be computed efficiently online. We use expression (15) to substitute Δw wherever it appears in the QP, to yield the condensed QP:

$$\begin{array}{ll} \min\limits_{\Delta u} & \frac{1}{2}\ \Delta u^T\, A^{\mathrm{c}}\, \Delta u \quad + \quad \left(a^{\mathrm{c}}(a_k, b_k) + \tilde{A} x_k\right)^T \Delta u \\ \text{subject to} & \left(c^{\mathrm{c}}(c_k, b_k) + \tilde{C} x_k\right) + C^{\mathrm{c}} \Delta u \geq 0 \end{array}. \tag{16}$$

All matrices, $A^{\mathrm{c}} := M^T A M, \tilde{A} := M^T A \tilde{L}, \tilde{C} := C\tilde{L}, C^{\mathrm{c}} := CM$ of this condensed QP are precomputed offline. Online, only the vectors $a^{\mathrm{c}}(a_k, b_k) + \tilde{A} x_k$ and $c^{\mathrm{c}}(c_k, b_k) + \tilde{C} x_k$ need to be computed, as shown in the following.

4.2 Variant A: Linear MPC Based on a Reference Trajectory

In the first approach [3, 8], we compute offline the *fixed* vectors $b := b_0(\bar{w}), c := c(\bar{w})$ and $a := \nabla_w a(\bar{w})$, and set $a_k := a, b_k := b, c_k := c$ in all real-time iterations. We can therefore precompute $m := m(b)$ and also $a^{\mathrm{c}} := a^{\mathrm{c}}(a, b) = M^T(Am + a)$ and $c^{\mathrm{c}} := c^{\mathrm{c}}(c, b) = c + Cm$.

Online, once x_k becomes known, only two sparse matrix-vector products and two vector additions are needed for computation of $a^c + \tilde{A}x_k$ and $c^c + \tilde{C}x_k$, and the condensed QP (16) in variables $\Delta u \in \mathbb{R}^{n_u \times N}$ must be solved. The solution of a QP of this size is standard in linear MPC applications and can usually be achieved quickly, in particular if an online active set strategy is used. Note that the dimension of the condensed QP (16) does *not* depend on the dimensions n_x and n_z of the state vectors, and that the cost of the matrix-vector products grows linearly with n_x and is independent of n_z.

4.3 Variant B: Online Feasibility Improvement

In the second variant of the real-time iteration scheme (originally proposed in [3]), we extend the online computational burden by one additional evaluation of $b_0(w_k)$ and $c(w_k)$, i.e. we set $b_k := b_0(w_k)$ and $c_k := c(w_k)$ in the QP (14). This allows to yield a feasibility improvement for nonlinear constraints. Offline, in addition to A, B, C we also compute a fixed objective gradient, e.g. $a = \nabla_w a(\bar{w})$, and then simply set $a_k := a + A(w_k - \bar{w})$ in each iteration.

Recalling the precomputed form of the condensed QP (16), only the vectors $a^c(a_k, b_k)$ and $c^c(c_k, b_k)$ have to be computed online, during the preparation phase.

Based on the block sparse structure of B shown in Eq. (10), the vector $m(b_k) = (m_0^x, m_0^z, m_0^u, \ldots, m_N^x)$ in (15) is for given $b_k = (b_0^x, b_0^z, b_1^x, b_1^z, \ldots, b_N^x)$ efficiently computed by a recursion. Starting with $m_0^x := b_0^x$, we compute for $i = 0, \ldots, N-$ 1[2]:

$$m_i^u := 0, \quad m_i^z := -\left(Z_i^z\right)^{-1}\left(b_i^z + Z_i^x m_i^x\right), \quad m_{i+1}^x := b_{i+1}^x + X_i^x m_i^x + X_i^z m_i^z. \tag{17}$$

Based on $m(b_k)$, we can quickly compute $a^c(a_k, b_k) = M^T(Am(b_k) + a_k)$ and $c^c(a_k, b_k) = c_k + Cm(b_k)$ (with $a_k = a + A(w_k - \bar{w})$). This computation involves only structured matrix vector products. This is the end of step 1, the preparation phase. Once x_k is known, the condensed QP (16) is solved in the feedback step 2, as in level A.

However, we do have to perform a transition step 3, i.e., update $w_{k+1} = w_k + \Delta w_k$, to meet nonlinear constraints. This means that the matrix vector multiplication $M\Delta u_k$ and the additions $\Delta w_k = m(b_k) + Lx_k + M\Delta u_k$ need to be done online.

This online algorithm does not make use of λ_k and μ_k and therefore does not need to update them during online computations.

4.4 Variant C: Online Optimality Improvement

In the third variant of the real-time iteration scheme, we further extend the online computational burden by one additional evaluation of the gradient of the Lagrangian $\nabla_w \mathcal{L}(w_k, \lambda_k, \mu_k)$. In the online QP (14), we set $a_k := \nabla_w \mathcal{L}(w_k, \lambda_k, \mu_k) + B^T\lambda_k + C^T\mu_k$, as well as $b_k := b_0(w_k)$ and $c_k := c(w_k)$. This approach allows

[2] The matrices Z_i^z can be pre-factored offline.

to yield not only an improvement of feasibility, but also of optimality for the original NLP (8).

The remaining online computations are slightly more expensive than for levels B and C, as we need to recover the multipliers $\lambda_k^{\rm QP}, \mu_k^{\rm QP}$ of the uncondensed QP (14) for the transition step 3, as follows:

First, the inequality multipliers $\mu_k^{\rm QP}$ are directly obtained as the multipliers $\mu_k^{\rm cQP}$ of the condensed QP (16): $\mu_k^{\rm QP} := \mu_k^{\rm cQP}$. Second, the equality multipliers $\lambda_k^{\rm QP}$ can be computed as $\lambda_k^{\rm QP} := (BS^T)^{-T} S(A\Delta w_k + a_k - C^T \mu_k^{\rm QP})$ where S is a projection matrix that maps w to its subvector s.

The matrix BS^T contains only those columns of B that correspond to the variables s, cf Eq. (10), and is thus invertible. Abbreviating $a := S(A\Delta w_k + a_k - C^T \mu_k^{\rm QP})$, $a = (a_0^x, a_0^z, \ldots, a_N^x)$, we can compute $\lambda_k^{\rm QP} = (\lambda_0^x, \lambda_0^z, \ldots, \lambda_N^x)$ recursively backwards: Starting with $\lambda_N^x := a_N^x$, we compute, for $i = N-1, N-2, \ldots, 0$:

$$\lambda_i^z = (Z_i^z)^{-T} \left(a_i^z + (X_i^z)^T \lambda_{i+1}^x \right), \quad \lambda_i^x = a_i^x + (X_i^x)^T \lambda_{i+1}^x - (Z_i^x)^T \lambda_i^z.$$

where we employ the submatrix notation of Eq. (10) for the matrix B, respectively BS^T. The proof of nominal stability of NMPC based on this variant follows the lines of the proof for the standard scheme mentioned in section 3.3.

5 A Real-Time NMPC Example

To demonstrate the real-time applicability of the NMPC schemes discussed above, a simulation experiment has been set up. In this experiment, a chain of massive balls connected by springs is perturbed at one end, and the control task is to bring the system back to steady state.

An ODE Model for a Chain of Spring Connected Masses

Consider the following nonlinear system of coupled ODEs

$$\ddot{x}_i + \beta \dot{x}_i - \frac{1}{m} (F_{i+\frac{1}{2}} - F_{i-\frac{1}{2}}) - g = 0, \qquad i = 1, 2, \ldots, N-1 \quad \text{(18a)}$$

$$F_{i+\frac{1}{2}} \triangleq S \left(1 - \frac{L}{\|x_{i+1} - x_i\|} \right) (x_{i+1} - x_i), \qquad i = 0, 1, \ldots, N-1. \quad \text{(18b)}$$

$$x_0(t) \equiv 0, \qquad \dot{x}_N(t) = u(t), \quad \text{(18c)}$$

for the ball positions $x_0(t), \ldots, x_N(t) \in \mathbb{R}^3$ with boundary conditions (18c) and a prescribed control function $u(t) \in \mathbb{R}^3$. Equations (18a)–(18c) describe the motion of a chain that consists of eleven balls (i.e. $N = 10$), numerated from 0 to 10, that are connected by springs. At one end, the first ball is fixed in the origin, the velocity of the other end (the "free" end) is prescribed by the function u. The motion of the chain is affected by both laminar friction and gravity (gravitational acceleration $g = 9.81\ \mathrm{m/s^2}$) as an external force. The model parameters are: mass $m = 0.03$ kg, spring constant $S = 1$ N/m, rest length of a spring $L = 0.033$ m,

and friction coefficient $\beta = 0.1\ \mathrm{s}^{-1}$. The chain movement can be controlled by adjusting the velocity of ball no. 10 at the "free" end (for the sake of simplicity we assume that it is possible to directly adjust this velocity). Figure 2 illustrates the example.

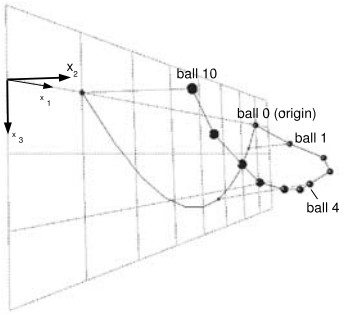

Fig. 2. A chain of 11 balls connected by springs. The first ball (in the back) is fixed, the last one can be moved freely; its Cartesian velocities serve as controls.

Optimal Control Problem Formulation

Aim of the controller is to bring the perturbed chain back to steady state. The open loop control problem is formulated with the following cost function:

$$L(x(t), u(t)) = \gamma \left\| x^N(t) - x^{\mathrm{end}} \right\|_2^2 + \delta \sum_{j=1}^{N-1} \left\| \dot{x}^j(t) \right\|_2^2 + \epsilon \|u(t)\|_2^2, \tag{19}$$

with the weighting factors $\gamma = 25$, $\delta = 1$, and $\epsilon = 0.01$. The chosen cost function (19) implicitly describes the steady state and allows to omit calculating the steady state position for each ball. The optimal control problem is

$$\min_{u(\cdot), x(\cdot)} \int_0^{T_p} L(x(t), u(t))\ dt \tag{20a}$$

subject to the system equations (18) and the input constraint

$$\|u(t)\|_\infty \leq 1, \quad \forall t \in [0, T_p] \tag{20b}$$

The control horizon T_p is set to 8 s, while the sampling time is only 200 ms. It is clear that the NMPC has to update the controls for every new sampling instant. If the time needed to solve the optimization problem exceeds 200 ms, then this delay will deteriorate the control performance and eventually even miss to bring the chain back to steady state. In the case of delayed control updates, the obtained velocities act as new disturbances on the process rather than as controls. This illustrates why waiting for the exact solution of the optimal control problem is not a good idea for such fast processes. The control problem is solved with the direct multiple shooting method on 40 multiple shooting intervals.

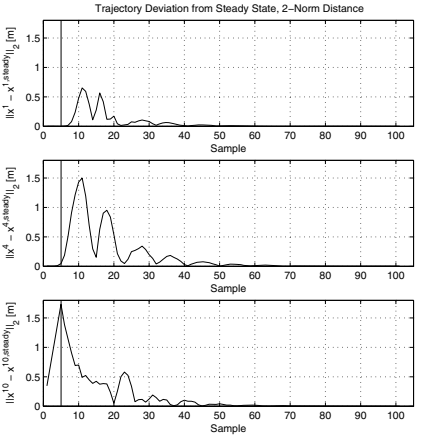

Fig. 3. Deviation of balls no. 1,4, and 10 from steady state in the controlled case. The deviation is expressed in the 2-norm of all Cartesian components.

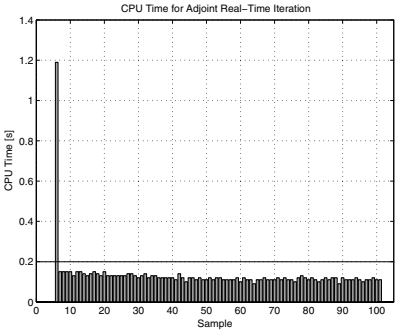

Fig. 4. CPU time for each sampling period. Note that the CPU time consists of both preparation and feedback phase. However, the feedback phase is in the range of 2 ms only! The preparation phase is then carried out while waiting for the new measurement update. The large CPU time peak in the beginning is due to the preparatory calculations that are done off-line.

In the simulation experiment, the chain initially is in steady state. Then, it is perturbed over five sampling periods by a constant velocity vector $u_{\text{pert}} = [-1\ 1\ 1]^T$ at the free end, before the controller becomes active, using the same end for disturbance rejection. The controls are calculated with variant C of the real-time iteration scheme (Section 4.4).

The resulting closed loop response for three different balls and the control moves can be seen in Figures 3 (2-norm deviation from steady state for three different balls) and 5 (absolute deviation from steady-state in y-direction). The corresponding control moves are shown in Figure 6. In all figures, x represents the vector of Cartesian coordinates of a ball. A superscript denotes the number of the ball (also see Figure 2), while a subscript picks one of the three coordinates. The simulation has been run on an Intel Pentium 4 machine with 2.8 GHz, 1024 kB L2 cache, 1 GB main memory, under Linux operating system Suse 9.3.

The computation times are depicted in Figure 4. Here, the entire CPU time of variant C needed to compute a control update has been measured. It is important to note that this comprises both the preparation and the feedback phase. The CPU time for the feedback phase only is in the range of 2 ms, meaning that the feedback delay between the new measurement update and the control update is negligible. The preparation phase is carried out *after* the new controls are given to the process in order to prepare the next optimization step. The large CPU time at the beginning of the control action is due to preparation calculations which are done off-line, before the state disturbance is measured. They correspond to the cost of a standard real-time iteration.

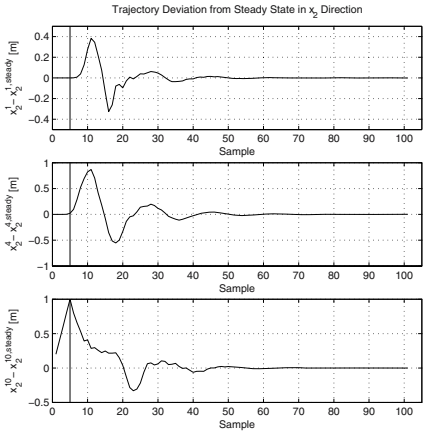

Fig. 5. Absolute deviation from steady state in y-direction of balls no. 1,4, and 10. The horizontal line at sample no. 5 marks the beginning of control action. Before, ball no. 10 was perturbed by a constant velocity.

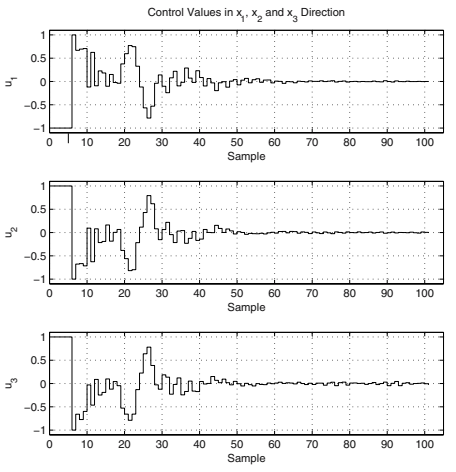

Fig. 6. Control moves calculated by the NMPC to reject the disturbance. The controls are the three Cartesian velocities of ball no. 10. Note that during the first 5 samples, these controls serve as disturbances to the chain.

6 Conclusions

We have discussed a class of methods for online computation of constrained optimal feedback controls in NMPC that are based on the direct multiple shooting method and a "real-time iteration" approach. They use an initial value embedding for efficient initialization of subsequent optimization problems, and treat in each iteration of the optimization process a different optimization problem, always with the most current system state x_k as initial value.

Three real-time iteration variants have been proposed that do not need to evaluate Jacobians during the runtime of the online algorithm and are therefore suitable for large scale DAE models with short timescales. In the presented variants, online computations with different properties are performed:

- Variant A requires only the online solution of a condensed QP. It can be interpreted as a linear MPC based on reference trajectories.
- In variant B, one additional DAE simulation is needed to evaluate the constraint functions. This variant yields a feasible but sub-optimal solution.
- Variant C requires also the gradient of the Lagrangian and is able to achieve feasibility as well as optimality for inequality constrained problems, still without evaluating Jacobians online.

The practical performance of variant C has been demonstrated in a simulation experiment, controlling the nonlinear mechanical system of a chain of balls and springs. A natural idea that arises is to combine the standard real-time

iteration scheme and its variants to a multi level real-time iteration which employs the variants A, B, C and the standard real-time iteration scheme in a hierarchical fashion. This algorithm aims at combining good local convergence properties with short sampling times and is subject of future investigation.

Acknowledgments. The authors gratefully acknowledge support by DFG grant BO864/10-1.

References

[1] F. Allgöwer, T.A. Badgwell, J.S. Qin, J.B. Rawlings, and S.J. Wright. Nonlinear predictive control and moving horizon estimation – An introductory overview. In P. M. Frank, editor, *Advances in Control, Highlights of ECC'99*, pages 391–449. Springer, 1999.

[2] H.G. Bock, M. Diehl, E.A. Kostina, and J.P. Schlöder. Constrained optimal feedback control of systems governed by large differential algebraic equations. In L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, editors, *Real-Time and Online PDE-Constrained Optimization*. SIAM, 2005. (in print).

[3] H.G. Bock, M. Diehl, D.B. Leineweber, and J.P. Schlöder. A direct multiple shooting method for real-time optimization of nonlinear DAE processes. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, volume 26 of *Progress in Systems Theory*, pages 246–267, Basel, 2000. Birkhäuser.

[4] H.G. Bock, E. Eich, and J.P. Schlöder. Numerical solution of constrained least squares boundary value problems in differential-algebraic equations. In K. Strehmel, editor, *Numerical Treatment of Differential Equations*. Teubner, Leipzig, 1988.

[5] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proc. 9th IFAC World Congress Budapest*, pages 243–247. Pergamon Press, 1984.

[6] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1218, 1998.

[7] G. De Nicolao, L. Magni, and R. Scattolini. Stability and robustness of nonlinear receding horizon control. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, volume 26 of *Progress in Systems Theory*, pages 3–23, Basel, 2000. Birkhäuser.

[8] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*, volume 920 of *Fortschr.-Ber. VDI Reihe 8, Mess-, Steuerungs- und Regelungstechnik*. VDI Verlag, Düsseldorf, 2002. http://www.ub.uni-heidelberg.de/archiv/1659/.

[9] M. Diehl, H.G. Bock, and J.P. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM J. Control Optim.*, 43(5):1714–1736, 2005.

[10] M. Diehl, H.G. Bock, J.P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *J. Proc. Contr.*, 12(4):577–585, 2002.

[11] M. Diehl, R. Findeisen, and F. Allgöwer. A stabilizing real-time implementation of nonlinear model predictive control. In L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, editors, *Real-Time and Online PDE-Constrained Optimization*. SIAM, 2004. (in print).

[12] M. Diehl, R. Findeisen, F. Allgöwer, H.G. Bock, and J.P. Schlöder. Nominal stability of the real-time iteration scheme for nonlinear model predictive control. *IEE Proc.-Control Theory Appl.*, 152(3):296–308, May 2005.

[13] M. Diehl, A. Walther, H.G. Bock, and E. Kostina. An adjoint-based SQP algorithm with quasi-Newton Jacobian updates for inequality constrained optimization. Technical Report MATH-WR 02-2005, Technical University Dresden, Germany, 2005.

[14] P. Kühl, A. Milewska, M. Diehl, E. Molga, and H.G. Bock. NMPC for runaway-safe fed-batch reactors. In *Proc. Int. Workshop on Assessment and Future Directions of NMPC*, pages 467–474, 2005.

[15] D.B. Leineweber. *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*, volume 613 of *Fortschr.-Ber. VDI Reihe 3, Verfahrenstechnik*. VDI Verlag, Düsseldorf, 1999.

[16] D.Q. Mayne. Nonlinear model predictive control: Challenges and opportunities. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, volume 26 of *Progress in Systems Theory*, pages 23–44, Basel, 2000. Birkhäuser.

[17] S.J. Qin and T.A. Badgwell. Review of nonlinear model predictive control applications. In B. Kouvaritakis and M. Cannon, editors, *Nonlinear model predictive control: theory and application*, pages 3–32, London, 2001. The Institute of Electrical Engineers.

[18] V.H. Schulz, H.G. Bock, and M.C. Steinbach. Exploiting invariants in the numerical solution of multipoint boundary value problems for DAEs. *SIAM J. Sci. Comp.*, 19:440–467, 1998.