ELSEVIER

Journal of Process Control 12 (2002) 577–585

JOURNAL OF PROCESS CONTROL

www.elsevier.com/locate/jprocont

Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations

Moritz Diehl[a,*], H. Georg Bock[a], Johannes P. Schlöder[a], Rolf Findeisen[b], Zoltan Nagy[c], Frank Allgöwer[b]

[a]*Interdisciplinary Center for Scientific Computing (IWR), University of Heidelberg, Im Neuenheimer Feld 368, D-69120 Heidelberg, Germany*
[b]*Institute for Systems Theory in Engineering, University of Stuttgart, Pfaffenwaldring 9, D-70550, Stuttgart, Germany*
[c]*Faculty of Chemistry and Chemical Engineering, "Babes-Bolyai" University, Cluj, Romania*

Abstract

Optimization problems in chemical engineering often involve complex systems of nonlinear DAE as the model equations. The direct multiple shooting method has been known for a while as a fast off-line method for optimization problems in ODE and later in DAE. Some factors crucial for its fast performance are briefly reviewed. The direct multiple shooting approach has been successfully adapted to the specific requirements of real-time optimization. Special strategies have been developed to effectively minimize the on-line computational effort, in which the progress of the optimization iterations is nested with the progress of the process. They use precalculated information as far as possible (e.g. Hessians, gradients and QP presolves for iterated reference trajectories) to minimize response time in case of perturbations. In typical real-time problems they have proven much faster than fast off-line strategies. Compared with an optimal feedback control computable upper bounds for the loss of optimality can be established that are small in practice. Numerical results for the Nonlinear Model Predictive Control (NMPC) of a high-purity distillation column subject to parameter disturbances are presented. © 2002 Published by Elsevier Science Ltd.

Keywords: Predictive control; Nonlinear control systems; Differential algebraic equations; Numerical methods; Optimal control; Distillation columns

1. Introduction

Real-time control based on the optimization of nonlinear dynamic process models has attracted increasing attention over the past decade, e.g. in chemical engineering [1–2]. Among the advantages of this approach are the flexibility provided in formulating the objective and the process model, the capability to directly handle equality and inequality constraints, and the possibility to treat disturbances fast.

One important precondition, however, is the availability of reliable and efficient numerical optimal control algorithms.

Direct methods reformulate the original infinite dimensional optimization problem as a finite nonlinear programming (NLP) problem by a parameterization of the controls and states. Such a method is called a *simultaneous* solution strategy, if the NLPs are solved by an infeasible point method such as sequential quadratic programming (SQP) or generalized Gauss–Newton.

Typical parameterizations are collocation [5], finite differences, or direct multiple shooting [6,7]. The latter is the basic method we treat in this paper.

Direct multiple shooting offers the following advantages in the context of real-time process optimization:

- As a simultaneous strategy, it allows to exploit solution information in controls, states and derivatives in subsequent optimization problems by suitable embedding techniques.
- Efficient state-of-the-art DAE solvers are employed to calculate the function values and derivatives quickly and accurately.
- Since the integrations are decoupled on different multiple shooting intervals, the method is well suited for parallel computation.
- The approach allows a natural treatment of control and path constraints as well as boundary conditions.

The efficiency of the approach, which has been observed in many practical applications, has several reasons. One of the most important is the possibility to incorporate information about the behaviour of the state trajectory into the initial guess for the iterative

* Corresponding author.
E-mail addresses: moritz.diehl@iwr.uni-heidelberg.de (M. Diehl), scicom@iwr.uni-heidelberg.de (H.G. Bock).

0959-1524/02/$ - see front matter © 2002 Published by Elsevier Science Ltd.
PII: S0959-1524(01)00023-3

solution procedure; this can damp the influence of poor initial guesses for the controls (which are usually much less known). In the context of NMPC, where a sequence of neighbouring optimization problems is treated, solution information of the previous problem can be exploited on several levels.

The paper reviews results presented in Bock et al. [8]; Diehl et al. [3]; Bock et al. [4]; Nagy et al. [9], and presents a new view on real-time optimization in NMPC. It is organized as follows:

- In Section 2, we introduce a general class of optimal control problems that can be treated by the current implementation of the real-time direct multiple shooting method.
- We sketch the direct multiple shooting method in Section 3, with emphasis on the SQP method specially tailored to the solution of such highly structured NLP problems. In Subsection 3.4 we look in detail at the computations during one SQP iteration to prepare the real-time strategies of Section 4.
- In Section 4 we describe our real-time embedding strategy for the efficient solution of subsequent optimization problems. This strategy allows to dovetail the iterative solution procedure with the process development in order to compute fast approximate closed-loop controls.
- The NMPC of a high-purity distillation column model of 164 states is treated in Section 5. As a scenario, disturbances in the feed stream are considered, which result in changes of the desired operating point.

2. Real-time optimal control problems

Throughout this paper, we consider optimal control problems of the following simplified type:

$$\min_{u(\cdot),x(\cdot),z(\cdot),p} \int_{t_0}^{t_f} L(x(t),z(t),u(t),p)\,\mathrm{d}t + E\big(x(t_f)p\big) \tag{1}$$

subject to a system of differential algebraic equations (DAE) of index one

$$B(\cdot)\dot{x}(t) = f(x(t),z(t),u(t),p) \tag{2}$$

$$0 = g(x(t),z(t),u(t),p) \tag{3}$$

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. Matrix $B(x(t),z(t),u(t),p)$ is assumed to be invertible, so that the DAE is of semi-explicit type. Initial values for the differential states and values for the system parameters are prescribed:

$$x(t_0) = x_0 \tag{4}$$

$$p = p_0 \tag{5}$$

In addition, terminal constraints

$$r\big(x(t_f),p\big)\left\{\begin{matrix} = \\ \geqslant \end{matrix}\right\}0 \tag{6}$$

as well as state and control inequality constraints

$$h(x(t),z(t),u(t)p)\geqslant 0 \tag{7}$$

have to be satisfied.

Remark. *The reason to introduce the parameters p as a variable subject to the equality constraint (5) has certain algorithmic advantages, as will become apparent in Section 4.*

Solving this problem we obtain an open-loop optimal control and corresponding state trajectories, that we may implement to control a plant. However, during operation of the real process, both state variables and system parameters are most likely subject to disturbances, e.g. due to model plant mismatch. Hence, an optimal closed-loop or feedback control law

$$\tilde{u}\big(x_0,p_0,t_f-t_0\big),$$

would be preferable, that gives us the optimal control for a sufficiently large range of time points t_0 and initial values x_0 and parameters p_0. One computationally expensive and storage consuming possibility would be to precalculate such a feedback control law off-line on a sufficiently fine grid. In contrast to this, the present paper is concerned with efficient ways to calculate this feedback control *in real-time* for progressing t_0.

One important variant of the optimization problem (1)–(7) arises in NMPC, where the final time t_f progresses with t_0, i.e.

$$t_f - t_0 = T_\mathrm{p}.$$

The constant T_p is called the prediction horizon. In this case the closed-loop control $\tilde{u}$ does no longer depend on time:

$$\tilde{u}(x_0,p_0).$$

3. Direct multiple shooting for optimal control

The solution of the real-time optimal control problem is based on the direct multiple shooting method, which is reviewed briefly in this section. This review prepares the presentation of the real-time embedding strategies in

Section 4. For a more detailed description see e.g. Leineweber [7].

3.1. Parametrization of the infinite optimization problem

The parameterization of the infinite optimization problem consists of two steps. For a suitable partition of the time horizon $[t_0, t_f]$ into N subintervals $[t_i, t_{i+1}]$ with

$$t_0 < t_1 < \ldots < t_N = t_f$$

we first discretize the control function $u(\cdot)$. For simplicity, we assume here that it is parametrized as a piecewise constant vector function

$$u(t) = u_i, \quad \text{for} \quad t \in [t_i, t_{i+1}],$$

but every parameterization with local support could be used without changing the structure of the problem.

In a second step the DAE are parametrized by *multiple shooting*. We decouple the DAE solution on the N intervals $[t_i, t_{i+1}]$ by introducing the initial values s_i^x and s_i^z (combined: s_i) of differential and algebraic states at times t_i as additional optimization variables.

On each subinterval $[t_i, t_{i+1}]$ we compute the trajectories $x_i(t)$ and $z_i(t)$ as the solution of an initial value problem:

$$B(\cdot)\dot{x}(t) \quad = \quad f(x_i(t), z_i(t), u_i, p) \tag{8}$$

$$0 = g(x_i(t), z_i(t), u_i, p) - \alpha_i(t) g\left(s_i^x, s_i^z, u_i, p\right) \tag{9}$$

$$x_i\left(t_i = s_i^x\right) \tag{10}$$

Here the subtrahend in (9) is deliberately introduced to allow an efficient DAE solution for initial values and controls s_i^x, s_i^z, u_i that may violate temporarily the consistency conditions (3). Therefore, we require for the scalar damping factor α that $\alpha_i(t_i) = 1$. For more details on the relaxation of the DAE the reader is referred, e.g. to Leineweber [7] or Schulz et al. [10]. Note that the trajectories $x_i(t)$ and $z_i(t)$ on the interval $[t_i, t_{i+1}]$ are functions of the initial values, controls, and parameters s_i^x, s_i^z, u_i, p only.

Analogously, the integral part of the cost function is evaluated on each interval independently:

$$L_i\left(s_i^x, s_i^z, u_i, p\right) = \int_{t_i}^{t_{i+1}} L(x_i(t), z_i(t), u_i, p)\mathrm{d}t.$$

3.2. The structured nonlinear programming problem

The parameterization of problem (1)–(7) using multiple shooting and a piecewise constant control representation leads to the following structured nonlinear programming (NLP) problem

$$\min_{u,s,p} \sum_{i=0}^{N-1} L_i\left(s_i^x, s_i^z, u_i, p\right) + E\left(S_N^x, p\right) \tag{11}$$

subject to the initial value and parameter constraint

$$s_0^x = x_0, \tag{12}$$

$$p = p_0, \tag{13}$$

the continuity conditions

$$s_{i+1}^x = x_i(t_{i+1}) \quad i = 0, 1, \ldots, N-1\,, \tag{14}$$

and the consistency conditions

$$0 = g\left(s_i^x, s_i^z, u_i, p\right) \quad i = 0, 1, \ldots, N. \tag{15}$$

Control and path constraints are imposed pointwise at the multiple shooting nodes

$$h\left(s_i^x, s_i^z, u_i, p\right) \geqslant 0 \quad i = 0, 1, \ldots, N \tag{16}$$

as well as at the terminal point

$$r\left(s_N^x, p\right) \left\{ \begin{matrix} = \\ \geqslant \end{matrix} \right\} 0. \tag{17}$$

3.3. SQP for multiple shooting

The above NLP problem (11)–(17) is solved by a *sequential quadratic programming (SQP)* method tailored to the multiple shooting structure.

The NLP can be summarized as

$$\min_{w} F(w) \quad \text{subject to} \quad \begin{cases} G(w) = 0 \\ H(w) \geqslant 0, \end{cases} \tag{18}$$

where w contains all the multiple shooting state variables and controls as well as the model parameters:

$$w = \left(s_0^x, s_0^z, u_0, s_1^x, s_1^z, u_1, \ldots, s_N^x, s_N^z, p\right).$$

The discretized dynamic model is included in the equality constraints $G(w) = 0$.

Starting from an initial guess w^0, an SQP method for the solution of (18) iterates

$$w^{k+1} = w^k + \alpha^k \Delta w^k, \qquad k = 0, 1, \ldots, \tag{19}$$

where $\alpha^k \in [0,1]$ is a relaxation factor, and the search direction Δw^k is the solution of the quadratic programming (QP) subproblem

$$\min_{\Delta w \in \Omega^k} \nabla F(w^k)^T \Delta w + \frac{1}{2} \Delta w^T A^k \Delta w \tag{20}$$

subject to

$$G(w^k) + \nabla G(w^k)^T \Delta w = 0$$
$$H(w^k) + \nabla H(w^k)^T \Delta w \geqslant 0.$$

A^k denotes an approximation of the Hessian $\nabla_w^2 \ell$ of the *Lagrangian function* ℓ, $\ell(w, \lambda, \mu) = F(w) - \lambda^T G(w) - \mu^T H(w)$, where λ and μ are the Lagrange multipliers.

Some remarks are in order on how to exploit the multiple shooting structure in the construction of a tailored SQP method.

Due to our choice of state and control parameterizations the NLP problem and the resulting QP problems have a particular structure: the Lagrangian function ℓ is *partially separable*, i.e. it can be written in the form

$$\ell(w, \lambda, \mu) = \sum_{i=0}^{N} \ell_i(w_i, \lambda, \mu)$$

where $w_i := (s_i, u_i, p)$ are the components of the primal variables w which are effective on interval $[t_i, t_{i+1}]$ only. This separation is possible if we simply interpret the parameters p as piecewise constant continuous controls.

As a consequence, the Hessian of ℓ has a *block diagonal structure* with blocks $\nabla_{w_i}^2 \ell_i(w_i, \lambda, \mu)$. Similarly, the multiple shooting parameterization introduces a characteristic *block sparse structure* of the Jacobian matrices $\nabla G(w)^T$ and $\nabla H(w)^T$.

It is of crucial importance for performance and numerical stability of the direct multiple shooting method that these structures of (18) are fully exploited. A number of specific algorithmic developments contribute to this purpose:

- For the exploitation of the block diagonal structure of the Hessian, three versions are recommended for different purposes:
  (a) A numerical calculation of the exact Hessian corresponds to Newton's method. This version is recommended if the computation of the Hessian is cheap, or in the case of neighbouring feedback control, where the Hessian can be computed and stored in advance. The use of the "exact" Hessian has excellent local convergence properties. For globalisation, techniques based on trust regions are needed, since the Hessian may become indefinite far from the optimal solution.
  (b) Partitioned high rank updates as introduced by Block and Plitt [6] speed up local convergence with negligible computational effort for the Hessian approximation.
  (c) A third approach to obtain a cheap Hessian approximation — the constrained Gauss–Newton method — is recommended in the special case of a least squares type cost function $F(w) = \frac{1}{2} \|C(w)\|_2^2$. The matrix $\{\nabla_w C \nabla_w C^T$ is already available from the gradient computation and provides an excellent approximation of the Hessian, if the residual $C(w)$ of the cost function is sufficiently small, as it can easily be shown that

$$\|\nabla_w C \nabla_w C^T - \nabla_w^2 \ell\| = O(\|C(w)\|).$$

This method is especially recommended for tracking problems that often occur in NMPC. However, the involved least squares terms may arise in integral form:

$$\int_{t_i}^{t_{i+1}} \|l(x, z, u, p)\|_2^2 \mathrm{d}t.$$

Specially adapted integrators that are able to compute a numerical approximation of $\nabla_w C \nabla_w C^T$ for this type of least squares term have been developed [11]. This method was used to compute the Hessian approximation in the numerical calculations presented in this paper.

- Special recursive QP solvers are used for problem (20) that exploit the block sparse structure of (18). Both active set strategies (as used in this paper) and interior point methods are available for the treatment of large systems of inequality constraints [6,7,12].
- Leineweber [7] developed a reduction technique for DAE systems with a large share of algebraic variables, which is also employed for the computations in this paper. He exploits the linearized algebraic consistency conditions for a reduction in variable space, so that only reduced gradients and Hessian blocks need to be calculated, which correspond to the differential variables, controls and parameters only [13,14].
- The solution of the DAE initial value problems and the corresponding derivatives are computed simultaneously by specially designed integrators which use the principle of *internal numerical differentiation*. In particular, the integrator DAESOL [15,16], which is based on the backward-differentiation-formulae (BDF), was used in the numerical calculations presented in this paper.
- The DAE solution and derivative generation can be performed in parallel on the different multiple shooting intervals. The latest parallel implementa-

tion of the direct multiple shooting method for DAE shows considerable speedups. For the numerical example presented in this paper, processor efficiencies in the range of 80% for 8 nodes have been observed.

Important for the use of the above methods in the real-time context is their excellent local convergence behaviour. By proper strategies to select stepsizes α^k or trust regions Ω^k or both, global convergence can be theoretically proven. Reassuring as this property is, it is of lesser importance in real-time optimization, as generally no runtime bounds can be established. For a detailed description of globalisation strategies available in the latest version of direct multiple shooting (MUSCOD2) the reader is referred to Leineweber [7].

3.4. A close look at one full SQP iteration

During each SQP iteration a variety of computations have to be performed. We will state them here for the direct multiple shooting variant that is the basis for the real-time algorithm described in Section 4. We will describe in detail how to compute the direction Δw^k that is needed to proceed from iterate w^k to the next iterate $w^{k+1} = w^k + \Delta w^k$ cf. Eq. (19) with $\alpha^k = 1$). For notational convenience we will not employ the index k for the subvectors of Δw^k and write

$$\Delta w^k = \left(\Delta s_0^x, \Delta s_0^z, \Delta u_0, \dots, \Delta p\right).$$

The computations that are needed to *formulate* the quadratic programming subproblem (20), i.e. the calculation of ∇F, A, G, ∇G, H, ∇H, and those that are needed to actually *solve* it, are intertwined. The algorithm proceeds as follows:

1. Reduction: Linearize the consistency conditions (15) and resolve the linear system to eliminate the Δs_i^z as a linear function of Δs_i^x, Δu_i and Δp.
2. DAE solution and derivative generation: linearize the continuity conditions (14) by solving the relaxed initial value problems (8)–(10) and computing directional derivatives with respect to Δs_i^x, Δu_i and $\Delta\ p$. Simultaneously, compute the gradient ∇F of the objective function (11), and the Hessian approximation A according to the Gauss–Newton approach. Linearize also the remaining constraints (16) and (17).
3. Condensing: using the linearized continuity conditions (14), eliminate the variables Δs_1^x, $\dots$ Δs_N^x. Project the objective gradient ∇F onto the space of the remaining variables Δs_0^x, Δu_0, $bm9$, Δu_{N-1} and Δp, and also the Hessian A and the linearized constraints (16) and (17). This step generates the so called *condensed QP* in the variables Δs_0^x, Δu_0, $\dots$, Δu_{N-1} and Δp only.
4. Step generation: solve the condensed QP with an efficient dense QP solver using an active set strategy. The solution yields the final values of Δs_0^x, Δu_0, $\dots$, Δu_{N-1} and of Δp. (Note that due to the linear constraints (12) and (13) $\Delta s_0^x = x_0 - s_0^x$ and $\Delta p = p_0 - p$.)
5. Expansion: expand the solution to yield final values for Δs_1^x, $\dots$, Δs_N^x, and for Δs_0^z, $\dots \Delta s_N^z$.

The main computational burden lies in step 2. Note that all steps before step 4 can be performed *without knowledge* of x_0 and p_0 — this will be exploited in the real-time embedding strategy in the following section.

4. A real-time embedding strategy

In a real-time scenario we aim at solving a sequence of optimal control problems. At each time point t_0 a different optimization problem (1)–(7) is treated, with an initial value x_0 that we do not know in advance. We must also expect that some of the parameters p_0, which are assumed to be constant in the model, are subject to disturbances.

The time for the solution of each optimization problem must be short enough to guarantee a sufficiently fast reaction to disturbances. Fortunately, we can assume that we have to solve a sequence of neighbouring optimization problems. Let us assume that a solution of the optimization problem for values t_0, x_0, p_0 is available, including function values, gradients and a Hessian approximation, but that at time t_0 the real values of the process are the deviated values $(x_0', p_0') = (x_0, p_0) + \epsilon$. How to obtain an updated value for the feedback control $\tilde{u}$ $(x_0', p_0', t_f - t_0)$ (resp. $\tilde{u}$ (x_0', p_0') in the NMPC case)? A conventional approach would be

- to start the SQP procedure as described above from the deviated values x_0', p_0' and to use the *old* control values u_i for an integration over the complete interval $t_f - t_0$, and
- to iterate until a given (strict) convergence criterion is satisfied.

Note that in the meantime the old control variables will be used, so that no response to the disturbed values x_0', p_0' is applied so far.

In time critical processes this may take much too long to be able to cope with the nonlinear dynamics.

In contrast to this, the authors suggest an algorithm which differs from this approach in two important aspects:

First, we propose to start the SQP iterations from the solution for the reference values x_0, p_0 instead of the deviated values, accepting an initial violation of the constraints (12), (13). Due to the linearity of these constraints their violation is immediately corrected after the first (full) SQP iteration. It turns out that the formulation of these constraints, that can be considered as an initial value embedding of the optimal control problem into the manifold of perturbed problems, is crucial for the real-time performance: an examination of the algorithm of Section 3.4 shows that steps 1, 2 and 3 can be performed *without* knowledge of the actual values x'_0, p'_0, thus allowing to perform them in advance and enabling a fast response at the moment when the disturbance occurs. In our approach, the first iteration is available in a small fraction of the time of a whole SQP iteration. This is in sharp contrast to the conventional approach, where all steps of the first SQP iteration have to be performed *after* x'_0, p'_0 are known.

Moreover, for a Newton's method, it is easy to show that the error of this first SQP iteration– compared to the solution of the full nonlinear problem — is only second order in the size of the perturbation ϵ. This property — that the first iterate is already close to the solution for small ϵ — holds also for the generalized Gauss-Newton method. Based on this observation, we secondly propose not to iterate the SQP iterations to convergence, but rather use the following real-time iteration scheme, that repeats the following cycle:

(I) Feedback response: After observation of the current values x'_0, p'_0 perform step 4 and apply the result — a first correction of the controls — immediately to the real process. Maintain these control values during some process duration δ which is sufficiently long to perform all calculations of one cycle.

(II) Preparation phase: During this period δ first expand the outcome of step 4 to the full QP solution (expansion step 5), then calculate a new (full step) iterate $w^{k+1} = w^k + \Delta w^k$, and based on this *new* iterate, perform the steps 1, 2 and 3 to prepare the feedback response for the following step. Go back to I.

In each cycle the same steps as for one classical SQP iteration are performed, but in a rotated order. Note, however, that in the middle of the preparation phase, the transition to a *new* optimization problem is performed. For shrinking horizon problems– e.g. for batch processes — this new problem will be on the remaining time interval $[t_0 + \delta, \ t_f]$ only. The steps 1, 2, and 3 will then only be performed on this shrunk horizon.

Note that the algorithm is prepared to react to a further disturbance after each cycle time δ, taking the outcome of the last iteration on the shrunk horizon as a reference solution.

Remark 1. *For moving horizon problems, as they arise in NMPC, the horizon length T_p is constant. There exist two possibilities to perform the transition from the old horizon $[t_0, \ t_0 + T_p]$ to the new horizon $[t_0 + \delta, \ t_0 + T_p + \delta \ ]$, before steps 1, 2 and 3 are performed:*

- *we either shift all problem variables by a time δ to account for the progressing time horizon, or*
- *we take the iterate w^{k+1} without a shift for a warm start.*

For short sampling times δ, both strategies have shown nearly identical performance [3]. In the numerical simulations presented in Section 5.4 we have adopted the second alternative.

Remark 2. *Compared to conventional SQP methods, the solution procedures for the real-time iterations have to be modified considerably. First, steps 1, 2 and 3, and step 5 need to be clearly separated from step 4. The crucial step 4 can even be further subdivided into parts that can be solved without knowledge of the unknown values x'_0, p'_0; these parts should actually become part of the preparation phase to make the feedback response as fast as possible.*

Remark 3. *The feedback phase itself is typically orders of magnitude shorter than the preparation phase. Thus, our algorithm can be interpreted as the successive generation of immediate feedback laws that take state and control inequality constraints on the complete horizon into account. Experience shows that the active set does not change much from one cycle to the next so that the computation time is bounded in practice.*

Remark 4. *The time δ required for a full cycle depends on the complexity of the model and the optimization problem, the numerical solution algorithms involved and the available computer. If δ is not sufficiently small, a parallelization of the expensive step 2 may be a remedy.*

Remark 5. *As the described real-time iterations correspond each to a different optimization problem, general convergence results are difficult to obtain. However, it can be shown under reasonable assumptions that the correction steps Δw^k will become smaller from cycle to cycle, if, after an initial disturbance ϵ, the process behaves as predicted by the model. In the case of shrinking horizon problems, the value of the objective function on the complete horizon $[t_0, \ t_f]$ that is obtained by using the the real-time iterations can be compared to that of an optimal feedback control. It turns out that for an exact Newton's method the loss of optimality is of fourth order in the size of the initial disturbance ϵ. A proof that covers also the Gauss-Newton method will appear in a forthcoming paper [17].*

5. NMPC of a high-purity distillation column

As a realistic application example we consider the control of a high purity binary distillation column with

40 trays for the separation of Methanol and n-Propanol. The column is modelled by a DAE with 42 differential states and 122 algebraic states, that is described in [9]. The model assumes constant molar hold up on the trays and ideal thermodynamics, but takes enthalpy balances into account to determine the mass flows from tray to tray.

5.1. The distillation column

The binary mixture is fed in the column with flow rate F and molar feed composition x_F. Products are removed at the top and bottom of the column with concentrations x_B and x_D. The column is considered in L/V configuration, i.e. the liquid reflux rate L and the vapor flow rate V (resp. the heating power Q) are the control inputs. The control problem is to maintain the specifications on the product concentrations x_B and x_D despite disturbances in the feed concentration x_F.

As usual in distillation control, the product purities x_B and x_D at reboiler and condenser are not controlled directly — instead, an inferential control scheme which controls the deviation of the concentrations on trays 14 and 28 from a given setpoint is used. These two concentrations are much more sensitive to changes in the inputs of the system than the product concentrations; if they are kept constant, the product purities are safely maintained for a large range of process conditions. As concentrations are difficult to measure, we consider instead the tray *temperatures*, which correspond directly to the concentrations via the Antoine equation.

5.2. State estimation

To obtain an estimate of the 42 differential system states and of the model parameter x_F by measurements of the three temperatures T_{14}, T_{21} and T_{28}, we have implemented a variant of an Extended Kalman Filter (EKF).

An EKF is based on subsequent linearizations of the system model at each current estimate; each measurement is compared with the prediction of the nonlinear model, and the estimated state is corrected according to the deviation. The weight of past measurement information is kept in a weighting matrix, which is updated according to the current system linearization.

In contrast to a standard EKF our estimator can incorporate additional knowledge about the possible range of states and parameters in form of inequality constraints. This is especially useful as the tray concentrations need to be constrained to be in the interval [0,1] to make a reasonable simulation possible. The performance of the estimator was such that step disturbances in the model parameter x_F were completely detected after 600 seconds, as can be seen in the second last graph of Fig. 1 for an example disturbance scenario.

5.3. Controller design

Given an estimate of the system parameters p_0 (here x_F), our controller first determines an appropriate desired steady-state for states and controls x_s, z_s, and u_s. This is done by formulating a steady-state constraint

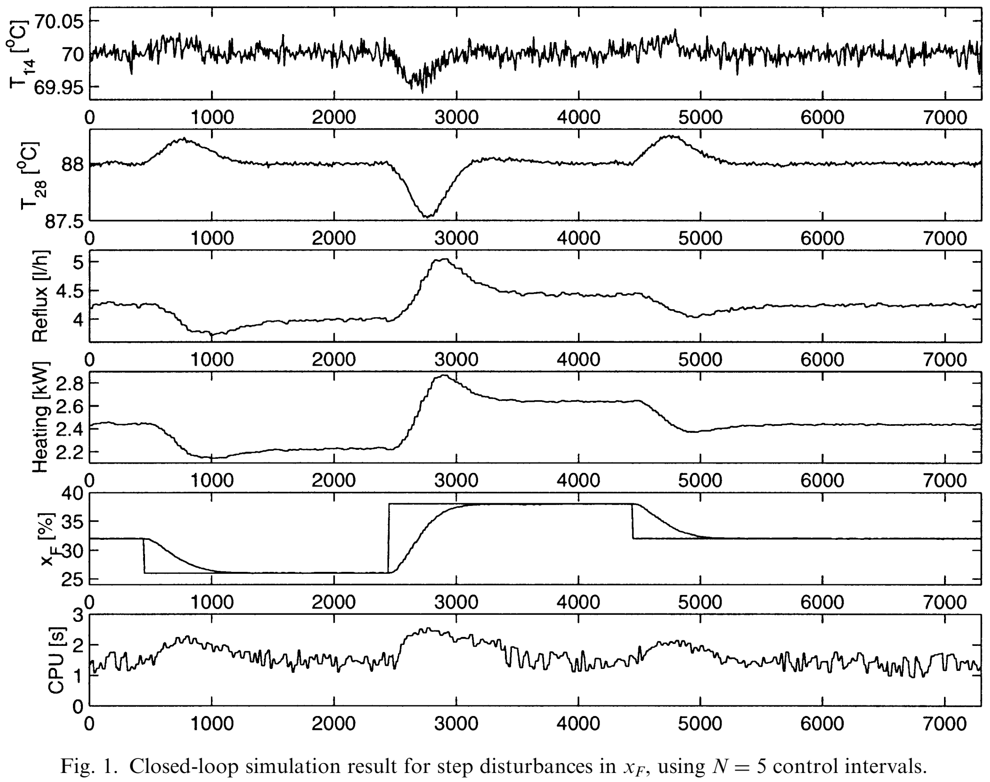

Fig. 1. Closed-loop simulation result for step disturbances in x_F, using $N = 5$ control intervals.

$$f(x_s, z_s, u_s, p_0) = 0 \tag{21}$$

$$g(x_s, z_s, u_s, p_0) = 0 \tag{22}$$

$$y(x_s, z_s, u_s, p_0) = \begin{pmatrix} T_{14}^r \\ T_{28}^r \end{pmatrix} \tag{23}$$

where the function y extracts the controlled tray temperatures from the system state, so that Eq. (23) restricts the steady-state to satisfy the inferential control aim of keeping the temperatures T_{14}, T_{28} at fixed reference values T_{14}^r, T_{28}^r.

Secondly, the open-loop optimal control problem is posed in the form (1)–(7), with a quadratic Lagrange term:

$$L(x, z, u, p) = \|l(x, z, u, p)\|_2^2$$

where

$$l(x, z, u, p) := \begin{pmatrix} y(x, z, u, p) - \begin{pmatrix} T_{14}^r \\ T_{28}^r \end{pmatrix} \\ R(u - u_s) \end{pmatrix}.$$

Here, the second component is introduced for regularization, with a small diagonal weighting matrix R. The control inputs u (i.e. Q and L) are bounded by inequality constraints of the form (7) to avoid that reboiler or condenser run empty.

To ensure nominal stability of the closed-loop system, an additional prediction interval is appended to the control horizon, with the controls fixed to the setpoint values u_s determined by Eqs. (21)–(23). The objective contribution of this interval provides an upper bound of the neglected future costs that are due after the end of the control horizon, if its length is sufficiently close to infinity [18]. A length of 3600 s for this additional interval was considered to be sufficient and was used in all performed simulations.

In our numerical solution approach, the determination of the current setpoint for given parameters and the dynamic optimization problem are performed simultaneously, by adding Eqs. (21)–(23) as additional equality constraints to the dynamic optimization problem.

As system state and parameters are the (smoothed) result of an EKF estimation, they only slightly vary from one optimization problem to the next, so that favourable conditions for the real-time iterations with the initial value embedding strategy are given.

5.4. Numerical results

For a realistic test of the algorithm we have performed closed-loop simulations where the simulation model equals the optimization model and the three temperature measurements are disturbed by Gaussian noise with a standard deviation of 0.01°C.

In the disturbance scenario shown in Fig. 1 we consider three step changes in the feed concentration x_F: starting from the nominal value, x_F is first reduced by 20%, later increased by 40% and at the end reduced by 20% to reach the nominal value again. In the closed-loop simulation in Fig. 1 a prediction horizon of $600 + 3600$ s is used, with five control intervals each of 120 s length. In the first two graphs the controlled tray temperatures T_{14} and T_{28} are shown, which should be kept at the specified values $T_{14}^r = 70$°C and T_{28}^r 88°C. It can be seen that they vary only by some tenths of centigrades during the whole scenario, which implicitly ensures highest purity of the product streams. The control response in Reflux and Heating is shown in the two graphs in the middle, whereas the estimated and real value for x_F are both shown in the second last graph.

At the bottom the CPU time for each optimization problem is plotted, which is well below the 10 seconds that are used as a sampling time δ. Note that this CPU time does *not* cause any delay between new state estimate and control response as our embedding strategy delivers an immediate response requiring only the condensed QP solution of step 4 (cf. Section 4). The CPU time is essentially needed to prepare the linearizations for the *next* immediate response.

In Table 1 numerical results for the same scenario are shown for three simulations with different control horizon lengths of 600, 1200 and 2400 s, i.e. with 5, 10 and 20 control intervals. As the computation times vary from one optimization problem to the next due to integrator adaptivity, we list not only the average CPU time for one optimization, but also the maximum CPU time observed for each simulation. The values from Fig. 1 can be found in the first column. Even for the prediction horizon of $2400 + 3600$ s with 20 control intervals, the CPU time meets the requirement to be less or equal the sampling time of 10 s. All simulations were performed on a Compaq Alpha XP1000 workstation.

Currently, the presented algorithms and system model are applied to control a medium scale distillation column located at the Institute for System Dynamics and Process Control at the University of Stuttgart. Results of closed-loop experiments will be presented in a forthcoming paper [11].

Table 1
Maximum and average CPU times (in s) for the considered example scenario, using different numbers N of control intervals

$N=5$		$N=10$		$N=20$	
Maximum	Average	Maximum	Average	Maximum	Average
2.6 s	1.6 s	5.6 s	3.7 s	9.7 s	4.6 s

6. Conclusions

New real-time and NMPC schemes based on the direct multiple shooting method are described. Their features are an initial value embedding that leads to a negligible response time after disturbances, and the immediate application of the computational results after each iteration. These real-time iterations calculate an approximation of an optimal feedback control with computable upper bounds on the loss of optimality.

An application of our approach to the NMPC of a high-purity distillation column shows excellent performance with CPU times in the range of a few seconds per optimization problem. This proves that even the use of a 164th order model with a prediction horizon of 6000 seconds and 20 control intervals is feasible for real-time control.

Acknowledgements

Financial support from *Deutsche Forschungsgemeinschaft* in the *Schwerpunktprogramm "Echtzeit-Optimierung großer Systeme"* is gratefully acknowledged. The authors are indebted to Daniel B. Leineweber for fruitful discussions on real-time problems in chemical engineering, and to Ilknur Uslu and Stefan Schwarzkopf for providing helpful insights into the working of a distillation column. We also want to thank the anonymous referees for their detailed comments that helped to improve the paper.

References

[1] C.E. García, D.M. Prett, M. Morari, Model predictive control: theory and practice — a survey, Automatica 25 (1989) 335.

[2] L.T. Biegler, J.B. Rawlings. Optimization approaches to nonlinear model predictive control, in: Y. Arkun, W.H. Ray, (Eds.), Chemical Process Control — CPC IV, The CACHE Corp., Austin, Texas, 1991.

[3] M. Diehl, H. Bock, D. Leineweber, J. Schlöoder, Efficient direct multiple shooting in nonlinear model predictive control, in: F. Keil, W. Mackens, H. Voß, J. Werther (Eds.), Scientific Computing in Chemical Engineering II, Vol. 2, Springer, Berlin, 1999.

[4] H. Bock, M. Diehl, D. Leineweber, J. Schlöder, A direct multiple shooting method for real-time optimization of nonlinear DAE processes, in F. Allgöwer, A. Zheng (Eds.), Nonlinear Predictive Control, Vol. 26 of Progress in Systems Theory, Birkhäuser, Basel, 2000.

[5] A. Cervantes, L. Biegler, Large-scale DAE optimization using a simultaneous NLP formulation, AIChE Journal 44 (5) (1998) 1038–1050.

[6] H. Bock, K. Plitt, A multiple shooting algorithm for direct solution of optimal control problems, in: Proceedings of the 9th IFAC World Congress Budapest, Pergamon, Oxford, 1984.

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

[8] H. Bock, I. Bauer, D. Leineweber, J. Schlöder, Direct multiple shooting methods for control and optimization in engineering, in: F. Keil, W. Mackens, H. Voß, J. Werther (Eds.). Scientific Computing in Chemical Engineering II, Vol. 2, Springer, Berlin, 1999.

[9] Z. Nagy, R. Findeisen, M. Diehl, F. Allgöwer, H. Bock, S. Agachi, J. Schlöder, D. Leineweber, Real-time feasibility of nonlinear predictive control for large scale processes — a case study, in: Proc. of ACC 2000, Chicago, in press.

[10] V.H. Schulz, H.G. Bock, M.C. Steinbach, Exploiting invariants in the numerical solution of multipoint boundary value problems for daes, SIAM J. Sci. Comp. 19 (1998) 440–467.

[11] M. Diehl, R. Findeisen, S. Schwarzkopf, I. Uslu, F. Allgöwer, H. Bock, T. Bürner, E. Gilles, A. Kienle, J. Schlöder, E. Stein, Real-time optimization of large scale process models: nonlinear model predictive control of a high purity distillation column, in: M. Groetschel, S. Krumke, J. Rambau, (Eds.), Online Optimization of Large Scale Systems: State of the Art, Springer, Berlin, 2001b.

[12] M. Steinbach, Fast Recursive SQP Methods for Large-scale Optimal Control Problems, Phd thesis, University of Heidelberg, 1995.

[13] H. Bock, E. Eich, J. Schlöder. Numerical solution of constrained least squares boundary value problems in differential-algebraic equations, in: K. Strehmel (Ed.), Numerical Treatment of Differential Equations, Teubner, Leipzig, 1988.

[14] J. Schlöder, Numerische Methoden zur Behandlung hochdimensionaler Aufgaben der Parameteridentifizierung, Vol. 187 of Bonner Mathematische Schriften, University of Bonn, Bonn, 1998.

[15] I. Bauer, F. Finocchi, W. Duschl, H.-P. Gail, J.P. Schlöoder, Simulation of chemical reactions and dust destruction in protoplanetary accretion discs, Astron. Astrophys. 317 (1997) 273–289.

[16] I. Bauer, Numerische Verfahren zur Losung von Anfangswertaufgaben und zur Generierung von ersten und zweiten Ableitungen mit Anwendungen bei Optimierungsaufgaben in Chemie und Verfahrenstechnik, PhD thesis, University of Heidelberg, 2000.

[17] M. Diehl, H. Bock, J. Schlöder, Newton type methods for the approximate solution of nonlinear programming problems in real-time, Technical report, University of Heidelberg, 2001a.

[18] G. De Nicolao, L. Magni, R. Scattolini, Stabilizing nonlinear receding horizon control via a non quadratic terminal state penalty, in: Symposium on Control, Optimization and Supervision, CESA'96 IMACS Multiconference, Lille, 1996.