Fast Direct Multiple Shooting Algorithms for Optimal Robot Control

Moritz Diehl, Hans Georg Bock, Holger Diedam, Pierre-Brice Wieber

Fast Direct Multiple Shooting Algorithms for Optimal Robot Control

Moritz Diehl[1], Hans Georg Bock[1], Holger Diedam[1], and Pierre-Brice Wieber[2]

[1] Interdisciplinary Center for Scientific Computing, University of Heidelberg, Im Neuenheimer Feld 368, D-69120 Heidelberg, Germany
m.diehl@iwr.uni-heidelberg.de
[2] INRIA Rhône-Alpes, Projet BIPOP, 38334 St Ismier Cedex, France

Summary. In this overview paper, we first survey numerical approaches to solve nonlinear optimal control problems, and second, we present our most recent algorithmic developments for real-time optimization in nonlinear model predictive control.

In the survey part, we discuss three direct optimal control approaches in detail: (i) single shooting, (ii) collocation, and (iii) multiple shooting, and we specify why we believe the direct multiple shooting method to be the method of choice for nonlinear optimal control problems in robotics. We couple it with an efficient robot model generator and show the performance of the algorithm at the example of a five link robot arm. In the real-time optimization part, we outline the idea of nonlinear model predictive control and the real-time challenge it poses to numerical optimization. As one solution approach, we discuss the *real-time iteration scheme*.

1 Introduction

In this paper, we treat the numerical solution of optimal control problems. We consider the following simplified optimal control problem in ordinary differential equations (ODE).

$$\underset{x(\cdot),\,u(\cdot),\,T}{\text{minimize}} \qquad \int_0^T L(x(t),u(t))\,dt \;+\; E\left(x(T)\right) \tag{1}$$

subject to

$$\begin{aligned}
x(0) - x_0 &= 0, && && \text{(fixed initial value)}\\
\dot{x}(t) - f(x(t),u(t)) &= 0, && t\in[0,T], && \text{(ODE model)}\\
h(x(t),u(t)) &\geq 0, && t\in[0,T], && \text{(path constraints)}\\
r\left(x(T)\right) &= 0 && && \text{(terminal constraints).}
\end{aligned}$$

The problem is visualized in Fig. 1. We may or may not leave the horizon length T free for optimization. As an example we may think of a robot that shall move

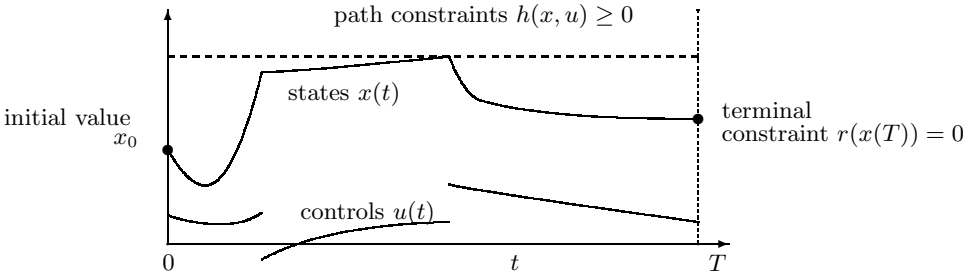

Fig. 1. Simplified Optimal Control Problem

in minimal time from its current state to some desired terminal position, and must respect limits on torques and joint angles. We point out that the above formulation is by far not the most general, but that we try to avoid unneccessary notational overhead by omitting e.g. differential algebraic equations (DAE), multi-phase motions, or coupled multipoint constraints, which are, however, all treatable by the direct optimal control methods to be presented in this paper.

1.1 Approaches to Optimal Control

Generally speaking, there are three basic approaches to address optimal control problems, (a) dynamic programming, (b) indirect, and (c) direct approaches, cf. the top row of Fig. 2.

(a) Dynamic Programming [5, 6] uses the principle of optimality of subarcs to compute recursively a feedback control for all times t and all x_0. In the continuous time case, as here, this leads to the Hamilton-Jacobi-Bellman (HJB) equation, a partial differential equation (PDE) in state space. Methods to numerically compute solution approximations exist, e.g. [34] but the approach severely suffers from Bellman's "curse of dimensionality" and is restricted to small state dimensions.

(b) Indirect Methods use the necessary conditions of optimality of the infinite problem to derive a boundary value problem (BVP) in ordinary differential equations (ODE), as e.g. described in [13]. This BVP must numerically be solved, and the approach is often sketched as "first optimize, then discretize". The class of indirect methods encompasses also the well known calculus of variations and the Euler-Lagrange differential equations, and the Pontryagin Maximum Principle [40]. The numerical solution of the BVP is mostly performed by shooting techniques or by collocation. The two major drawbacks are that the underlying differential equations are often difficult to solve due to strong nonlinearity and instability, and that changes in the control structure, i.e. the sequence of arcs where different constraints are active, are difficult to handle: they usually require a completely new problem setup. Moreover, on so called *singular arcs*, higher index DAE arise which necessitate specialized solution techniques.

(c) Direct methods transform the original infinite optimal control problem into a finite dimensional *nonlinear programming problem (NLP)*. This NLP is then solved by variants of state-of-the-art numerical optimization methods, and the approach is therefore often sketched as "first discretize, then optimize". One of the most important advantages of direct compared to indirect methods is that they can easily treat inequality constraints, like the inequality path constraints in the formulation above. This is because structural changes in the active constraints during the optimization procedure are treated by well developed NLP methods that can deal with inequality constraints and active set changes. All direct methods are based on a finite dimensional parameterization of the control trajectory, but differ in the way the state trajectory is handled, cf. the bottom row of Fig. 2.

For solution of constrained optimal control problems in real world applications, direct methods are nowadays by far the most widespread and successfully used techniques, and we will focus on them in the first part of this paper.

1.2 Nonlinear Model Predictive Control

The optimization based feedback control technique "Nonlinear Model Predictive Control (NMPC)" has attracted much attention in recent years [1, 36], in particular in the proceess industries. Its idea is, simply speaking, to use an open-loop optimal control formulation to generate a feedback control for a closed-loop system. The current system state is continuously observed, and NMPC solves repeatedly an optimal control problem of the form (1), each time with the most current state observation as initial value x_0. Assuming that the optimal control trajectory can be computed in negligible time, we can apply the first bit of our optimal plan to the real world system, for some short duration δ. Then, the state is observed again, a new optimization problem is solved, the control again applied to the real system, and so on. In this way, feedback is generated that can reject unforeseen disturbances and errors due to model-plant-mismatch.

Among the advantages of NMPC when compared to other feedback control techniques are the flexibility provided in formulating the control objective, the capability to directly handle equality and inequality constraints, and the possibility to treat unforeseen disturbances fast. Most important, NMPC allows to make use of reliable nonlinear process models $\dot{x} = f(x, u)$ so that the control performance can profit from this important knowledge, which is particularly important for transient, or periodic processes. It is this last point that makes it particularly appealing for use in robotics.

One essential problem, however, is the high on-line computational load that is often associated with NMPC, since at each sampling instant a nonlinear optimal control problem of the form (1) must be solved. The algorithm must predict and optimize again and again, in a high frequency, while the real process advances in time. Therefore, the question of fast *real-time optimization* has been intensively investigated [4, 28, 51, 44, 9]. We refer to Binder et al. [10] for an overview of existing methods. One reason why most applications of NMPC have so far been in the process industries [42] is that there, time scales are typically in the range of minutes so that the real-time requirements are less severe than in mechanics. However, we believe that it is only a matter of time until NMPC becomes an important feedback

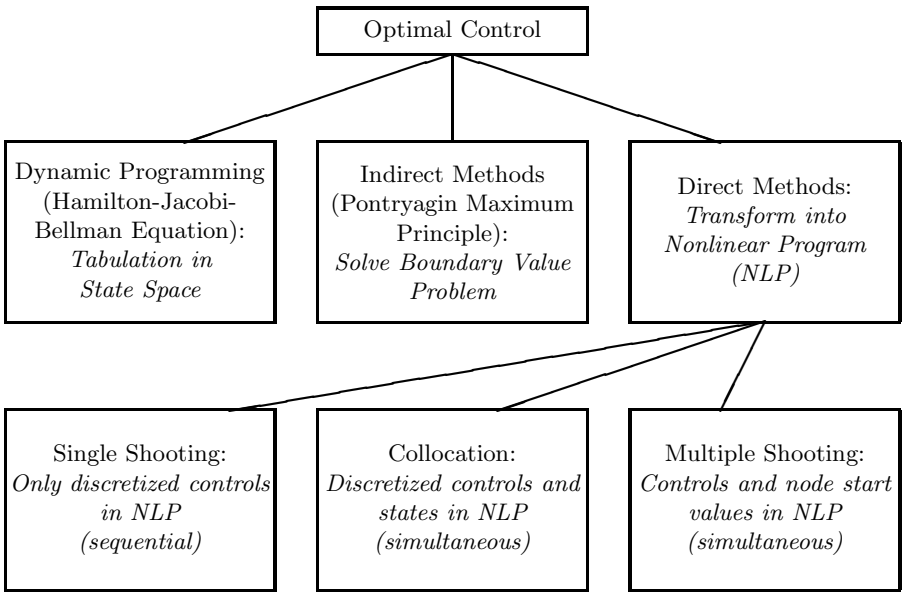

Fig. 2. Overview of numerical methods for optimal control

technique in robotics, too. The second scope of this paper is therefore to present some of our latest ideas regarding the fast real-time optimization for NMPC, which are based on direct optimal control methods.

1.3 Paper Outline

The paper is organized as follows. In the next section we will describe three popular direct optimal control methods, single shooting, collocation, and multiple shooting. We will argue why we believe the last method to be the method of choice for nonlinear optimal control problems in robotics, and in Section 3 we will present its coupling to an efficient robot model generator and show its application to the time optimal point to point maneuver of a five link robot arm. In Section 4 we will discuss nonlinear model predictive control (NMPC) and show how the challenge of fast online optimization can be addressed by the so called *real-time iteration* scheme, in order to make NMPC of fast robot motions possible. Finally, in Section 5, we conclude the paper with a short summary and an outlook.

2 Direct Optimal Control Methods

Direct methods reformulate the infinite optimal control problem (1) into a finite dimensional *nonlinear programming problem (NLP)* of the form

$$\min_w a(w) \quad \text{subject to} \quad b(w) = 0, \quad c(w) \geq 0, \tag{2}$$

with a finite dimensional vector w representing the optimization degrees of freedom, and with differentiable functions a (scalar), b, and c (both vector valued). As said above, all direct methods start by a parameterization of the control trajectory, but they differ in the way how the state trajectory is handled. Generally, they can be divided into *sequential* and *simultaneous* approaches.

In *sequential* approaches, the state trajectory $x(t)$ is regarded as an implicit function of the controls $u(t)$ (and of the initial value x_0), e.g. by a forward simulation with the help of an ODE solver in *direct single shooting* [45, 31]. Thus, simulation and optimization iterations proceed sequentially, one after the other, and the NLP has only the discretized control as optimization degrees of freedom.

In contrast to this, *simultaneous* approaches keep a parameterization of the state trajectory as optimization variables within the NLP, and add suitable equality constraints representing the ODE model. Thus, simulation and optimization proceed simultaneously, and only at the solution of the NLP do the states actually represent a valid ODE solution corresponding to the control trajectory. The two most popular variants of the simultaneous approach are *direct collocation* [8] and *direct multiple shooting*[12].

We will present in detail the mentioned three direct approaches. As all direct methods make use of advanced NLP solvers, we also very briefly sketch one of the most widespread NLP solution methods, Sequential Quadratic Programming (SQP), which is also at the core of the real-time iteration scheme to be presented in the second part.

A tutorial example

For illustration of the different behaviour of sequential and simultaenous approaches, we will use the following tutorial example with only one state and one control dimension. The ODE $\dot{x} = f(x, u)$ is slightly unstable and nonlinear.

$$\begin{aligned}
&\underset{x(\cdot),\,u(\cdot)}{\text{minimize}} && \int_0^3 x(t)^2 + u(t)^2 \, dt \\
&\text{subject to} \\
&\qquad x(0) = x_0, && \text{(initial value)} \\
&\qquad \dot{x} = (1+x)x + u, \quad t \in [0,3], && \text{(ODE model)} \\
&\begin{bmatrix} 1 - x(t) \\ 1 + x(t) \\ 1 - u(t) \\ 1 + u(t) \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, && t \in [0,3], \text{ (bounds)} \\
&\qquad x(3) = 0. && \text{(zero terminal constraint).}
\end{aligned}$$

We remark that due to the bounds $|u| \leq 1$, we have uncontrollable growth for any $x \geq 0.618$ because then $(1 + x)x \geq 1$. We set the inital value to $x_0 = 0.05$. For the

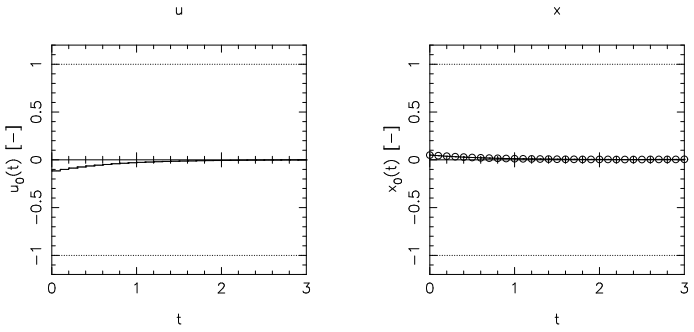

Fig. 3. Solution of the tutorial example.

control discretization we will choose $N = 30$ control intervals of equal length. The solution of this problem is shown in Figure 3.

2.1 Sequential Quadratic Programming (SQP)

To solve any NLP of the form (2), we will work within an iterative *Sequential Quadratic Programming (SQP)*, or *Newton-type* framework. We omit all details here, and refer to excellent numerical optimization textbooks instead, e.g. [39]. We need to introduce, however, the *Lagrangian function*

$$\mathcal{L}(w, \lambda, \mu) = a(w) - \lambda^T b(w) - \mu^T c(w),$$

with so called *Lagrange multipliers* λ and μ, that plays a preeminent role in optimization. The necessary conditions for a point w^* to be a local optimum of the NLP (2) are that there exist multipliers λ^* and μ^*, such that

$$\nabla_w \mathcal{L}(w^*, \lambda^*, \mu^*) = 0, \tag{3}$$

$$b(w^*) = 0, \tag{4}$$

$$c(w^*) \geq 0, \quad \mu^* \geq 0, \quad c(w^*)^T \mu^* = 0. \tag{5}$$

In order to approximately find such a triple (w^*, λ^*, μ^*) we proceed iteratively. Starting with an initial guess (w_0, λ_0, μ_0), a standard full step SQP iteration for the NLP is

$$w_{k+1} = w_k + \Delta w_k, \tag{6}$$

$$\lambda_{k+1} = \lambda_k^{\mathrm{QP}}, \quad \mu_{k+1} = \mu_k^{\mathrm{QP}}, \tag{7}$$

where $(\Delta w_k, \lambda_k^{\mathrm{QP}}, \mu_k^{\mathrm{QP}})$ is the solution of a quadratic program (QP). In the classical Newton-type or SQP approaches, this QP has the form

$$\begin{array}{ll} \min\limits_{\Delta w \,\in\, \mathbb{R}^{n_w}} & \frac{1}{2}\ \Delta w^T\ A_k\ \Delta w \quad + \quad \nabla_w a(w_k)^T \Delta w \\ \text{subject to} & \begin{cases} b(w_k) + \nabla_w b(w_k)^T \Delta w = 0 \\ c(w_k) + \nabla_w c(w_k)^T \Delta w \geq 0 \end{cases} \end{array} \tag{8}$$

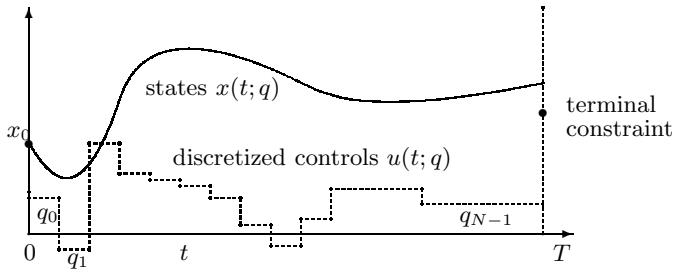

Fig. 4. Illustration of single shooting.

where A_k is an approximation of the Hessian of the Lagrangian,

$$A_k \approx \nabla_w^2 \mathcal{L}(w_k, \lambda_k, \mu_k),$$

and $\nabla_w b(w_k)^T$ and $\nabla_w c(w_k)^T$ are the constraint Jacobians. Depending on the quality of the Hessian approximation we may expect linear, super-linear or even quadratic convergence. Practical SQP methods differ e.g. in the type of globalisation strategy, in the type of QP solver used, or in the way the Hessian is approximated – e.g. by BFGS updates or by a Gauss-Newton Hessian. This last approach is favourable for least squares problems, as e.g in tracking or estimation problems. When the objective is given as $a(w) = \|r(w)\|_2^2$, the Gauss-Newton Hessian is given by $A_k = 2\nabla_w r(w_k)\nabla_w r(w_k)^T$. It is a good approximation of the exact Hessian $\nabla_w^2 \mathcal{L}(w_k, \lambda_k, \mu_k)$ if the residual $\|r(w)\|_2^2$ is small or if the problem is only mildly nonlinear.

2.2 Direct Single Shooting

The single shooting approach starts by discretizing the controls. We might for example choose grid points on the unit interval, $0 = \tau_0 < \tau_1 < \ldots < \tau_N = 1$,, and then rescale these gridpoints to the possibly variable time horizon of the optimal control problem, $[0, T]$, by defining $t_i = T\tau_i$ for $i = 0, 1, \ldots, N$. On this grid we discretize the controls $u(t)$, for example piecewise constant, $u(t) = q_i$ for $t \in [t_i, t_{i+1}]$, so that $u(t)$ only depends on the the finitely many control parameters $q = (q_0, q_1, \ldots, q_{N-1}, T)$ and can be denoted by $u(t;q)$. If the problem has a fixed horizon length T, the last component of q disappears as it is no optimization variable. Using a numerical simulation routine for solving the initial value problem

$$x(0) = x_0, \quad \dot{x}(t) = f(x(t), u(t;q)), \quad t \in [0, T],$$

we can now regard the states $x(t)$ on $[0, T]$ as dependent variables, cf. Fig. 4. We denote them by $x(t;q)$. The question which simulation routine should be chosen is crucial to the success of any shooting method and depends on the type of ODE model. It is essential to use an ODE solver that also delivers sensitivities, as they are needed within the optimization. We also discretize the path constraints to avoid

a semi-infinite problem, for example by requiring $h(x(t), u(t)) \geq 0$ only at the grid points t_i, but we point out that also a finer grid could be chosen without any problem. Thus, we obtain the following finite dimensional nonlinear programming problem (NLP):

$$\underset{q}{\text{minimize}} \qquad \int_0^T L(x(t;q), u(t;q))\, dt \;+\; E\left(x(T;q)\right) \tag{9}$$

subject to

$$\begin{aligned} h(x(t_i;q), u(t_i;q)) \geq 0, \qquad & i = 0, \ldots, N, \text{ (discretized path constraints)} \\ r\left(x(T;q)\right) = 0. \qquad & \text{(terminal constraints)} \end{aligned}$$

This problem is solved by a finite dimensional optimization solver, e.g. Sequential Quadratic Programming (SQP), as described above.

The behaviour of single shooting (with full step SQP and Gauss-Newton Hessian) applied to the tutorial example is illustrated in Fig. 5. The initialization – at the zero control trajectory, $u(t) = 0$ – and the first iteration are shown. Note that the state path and terminal constraints are not yet satisfied in the first iteration, due to their strong nonlinearity. The solution (up to an accuracy of 10^{-5}) is obtained after seven iterations. The strong points of single shooting are (i) that it can use fully adaptive, error controlled state-of-the-art ODE or DAE solvers, (ii) that it has only few optimization degrees of freedom even for large ODE or DAE systems, and (iii) that only initial guesses for the control degrees of freedom are needed. The weak points are (i) that we cannot use knowledge of the state trajectory x in the initialization (e.g. in tracking problems), (ii) that the ODE solution $x(t;q)$ can depend very nonlinearly on q, as in the example, and (iii) that unstable systems are difficult to treat.

However, due to its simplicity, the single shooting approach is very often used in engineering applications e.g. in the commercial package gOPT [41].

2.3 Collocation

We only very briefly sketch here the idea of the second direct approach, collocation. We start by discretizing both, the controls *and* the states on a fine grid. Typically, the controls are chosen to be piecewise constant, with values q_i on each interval $[t_i, t_{i+1}]$. The value of the states at the grid points will be denoted by $s_i \approx x(t_i)$. In order to avoid notational overhead, we will in the remainder of this section assume that the length of the time horizon, T, is constant, but point out that the generalization to variable horizon problems by the above mentioned time transformation is straightforward. In collocation, the infinite ODE

$$\dot{x}(t) - f(x(t), u(t)) = 0, \quad t \in [0, T]$$

is replaced by finitely many equality constraints

$$c_i(q_i, s_i, s_i', s_{i+1}) = 0, \quad i = 0, \ldots, N-1,$$

where the additional variables s_i' might represent the state trajectory on intermediate "collocation points" within the interval $[t_i, t_{i+1}]$. By a suitable choice of the location

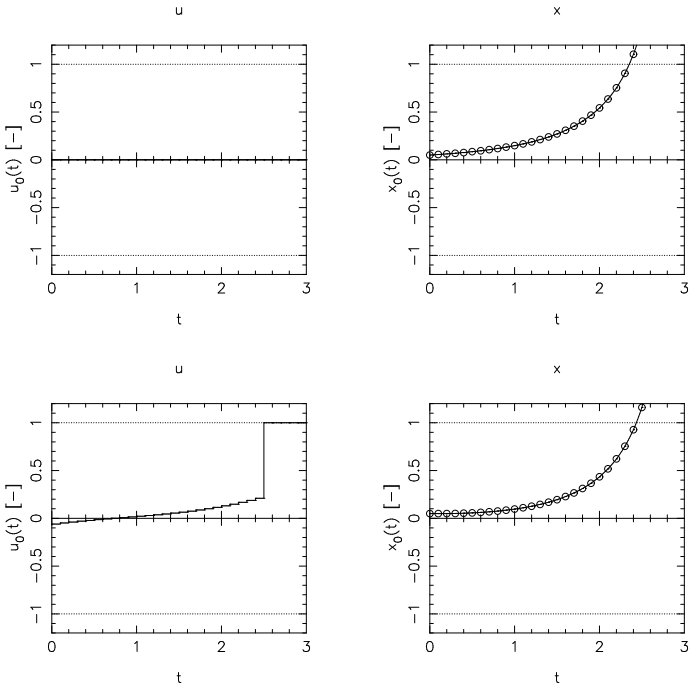

Fig. 5. Single shooting applied to the tutorial example: Initialization and first iteration.

of these points a high approximation order can be achieved, and typically they are chosen to be the zeros of orthogonal polynomials. But we sketch here only a simplified tutorial case, where no intermediate variables s_i' are present, to give a flavour of the idea of collocation. Here, the additional equalities are given by

$$c_i(q_i, s_i, s_{i+1}) := \frac{s_{i+1} - s_i}{t_{i+1} - t_i} - f\left(\frac{s_i + s_{i+1}}{2}, q_i\right).$$

Then, we will also approximate the integrals on the collocation intervals, e.g. by

$$l_i(q_i, s_i, s_{i+1}) := L\left(\frac{s_i + s_{i+1}}{2}, q_i\right)(t_{i+1} - t_i) \approx \int_{t_i}^{t_{i+1}} L(x(t), u(t))dt$$

After discretization we obtain a large scale, but sparse NLP:

$$\underset{s,\,q}{\text{minimize}} \qquad \sum_{i=0}^{N-1} l_i(q_i, s_i, s_{i+1}) \;+\; E(s_N)$$

subject to

$$
\begin{aligned}
s_0 - x_0 &= 0, && && \text{(fixed initial value)} \\
c_i(q_i, s_i, s_{i+1}) &= 0, && i = 0, \ldots, N-1, && \text{(discretized ODE model)} \\
h(s_i, q_i) &\geq 0, && i = 0, \ldots, N, && \text{(discretized path constraints)} \\
r\left(s_N\right) &= 0. && && \text{(terminal constraints)}
\end{aligned}
$$

This problem is then solved e.g. by a reduced SQP method for sparse problems [8, 48], or by an interior-point method [7]. Efficient NLP methods typically do not keep the iterates feasible, so the discretized ODE model equations are only satisfied at the NLP solution, i.e., simulation and optimization proceed simultaneously. The advantages of collocation methods are (i) that a very sparse NLP is obtained (ii) that we can use knowledge of the state trajectory x in the initialization (iii) that it shows fast local convergence (iv) that it can treat unstable systems well, and (v) that it can easily cope with state and terminal constraints. Its major disadvantage is that adaptive discretization error control needs regridding and thus changes the NLP dimensions. Therefore, applications of collocation do often not address the question of proper discretization error control. Nevertheless, it is successfully used for many practical optimal control problems [3, 50, 47, 14, 54].

2.4 Direct Multiple Shooting

The direct multiple shooting method (that is due to Bock and Plitt [12]) tries to combine the advantages of a simultaneous method like collocation with the major advantage of single shooting, namely the possibility to use adaptive, error controlled ODE solvers. In direct multiple shooting, we proceed as follows. First, we again discretize the controls piecewise on a coarse grid

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

where the intervals can be as large as in single shooting. But second, we solve the ODE on each interval $[t_i, t_{i+1}]$ independently, starting with an artificial initial value s_i:

$$
\begin{aligned}
\dot{x}_i(t) &= f(x_i(t), q_i), \quad t \in [t_i, t_{i+1}], \\
x_i(t_i) &= s_i.
\end{aligned}
$$

By numerical solution of these initial value problems, we obtain trajectory pieces $x_i(t; s_i, q_i)$, where the extra arguments after the semicolon are introduced to denote the dependence on the interval's initial values and controls. Simultaneously with the decoupled ODE solution, we also numerically compute the integrals

$$l_i(s_i, q_i) := \int_{t_i}^{t_{i+1}} L(x_i(t_i; s_i, q_i), q_i) dt.$$

In order to constrain the artificial degrees of freedom s_i to physically meaningful values, we impose continuity conditions $s_{i+1} = x_i(t_{i+1}; s_i, q_i)$. Thus, we arrive at the following NLP formulation that is completely equivalent to the single shooting NLP, but contains the extra variables s_i, and has a block sparse structure.

$$\underset{s,\,q}{\text{minimize}} \qquad \sum_{i=0}^{N-1} l_i(s_i, q_i) \;+\; E\left(s_N\right) \tag{10}$$

subject to

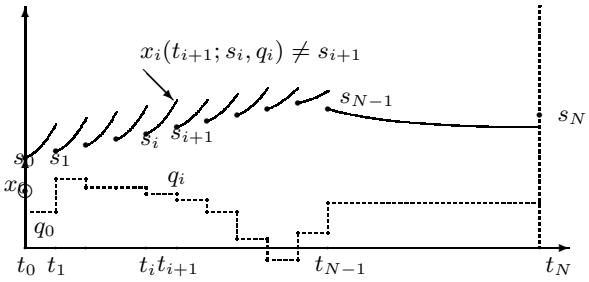

Fig. 6. Illustration of multiple shooting.

$$
\begin{aligned}
s_0 - x_0 &= 0, && && \text{(initial value)}\\
s_{i+1} - x_i(t_{i+1}; s_i, q_i) &= 0, && i = 0, \ldots, N-1, && \text{(continuity)}\\
h(s_i, q_i) &\geq 0, && i = 0, \ldots, N, && \text{(discretized path constraints)}\\
r(s_N) &= 0. && && \text{(terminal constraints)}
\end{aligned}
$$

If we summarize all variables as $w := (s_0, q_0, s_1, q_1, \ldots, s_N)$ we obtain an NLP in the form (2). The block sparse Jacobian $\nabla b(w^k)^T$ contains the linearized dynamic model equations, and the Hessian $\nabla^2_w \mathcal{L}(w_k, \lambda_k, \mu_k)$ is block diagonal, which can both be exploited in the tailored SQP solution procedure [12]. Because direct multiple shooting only delivers a valid (numerical) ODE solution when also the optimization iterations terminate, it is usually considered a simultaneous method, as collocation. But sometimes it is also called a *hybrid method*, as it combines features from both, a pure sequential, and a pure simultaneous method. Its advantages are mostly the same as for collocation, namely that knowledge of the state trajectory can be used in the initialization, and that it robustly handles unstable systems and path state and terminal constraints.

The performance of direct multiple shooting – and of any other simultaneous method – is for the tutorial example illustrated in Figure 7. The figure shows first the initialization by a forward simulation, using zero controls. This is one particularly intuitive, but by far not the best possibility for initialization of a simultaneous method: it is important to note that the state trajectory is by no means constrained to match the controls, but can be chosen point for point if desired. In this example, the forward simulation is at least reset to the nearest bound whenever the state bounds are violated at the end of an interval, in order to avoid simulating the system in areas where we know it will never be at the optimal solution. This leads to the discontinuous state trajectory shown in the top row of Figure 7. The result of the first iteration is shown in the bottom row, and it can be seen that it is already much closer to the solution than single shooting, cf. Fig. 5. The solution, cf. Fig. 3, is obtained after two more iterations. It is interesting to note that the terminal constraint is already satisfied in the first iteration, due to its linearity. The nonlinear effects of the continuity conditions are distributed over the whole horizon, which is seen in the discontinuities. This is in contrast to single shooting, where the nonlinearity of the system is accumulated until the end of the horizon, and the terminal constraint becomes much more nonlinear than necessary. Any simultaneous

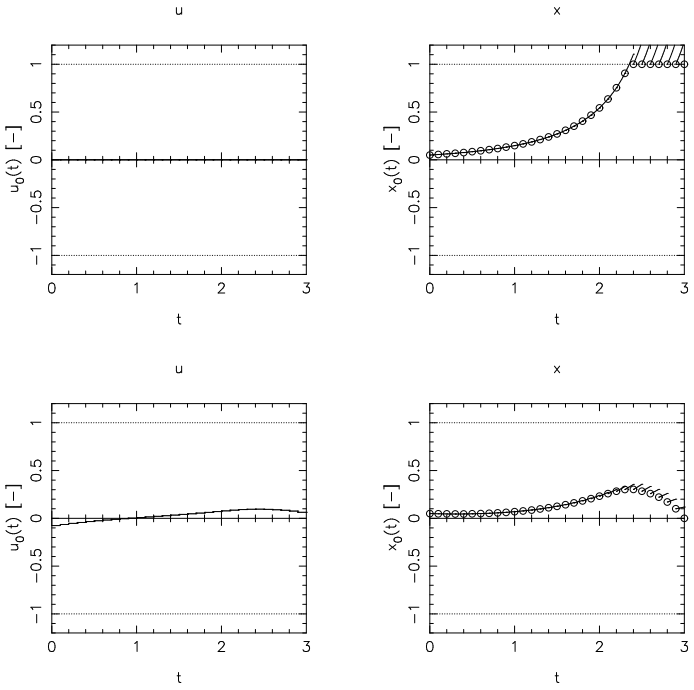

Fig. 7. Multiple shooting applied to the tutorial example: Initialization and first iteration.

method, e.g. collocation, would show the same favourable performance as direct multiple shooting here.

As said above, in contrast to collocation, direct multiple shooting can combine adaptivity with fixed NLP dimensions, by the use of adaptive ODE/DAE solvers. Within each SQP iteration, the ODE solution is often the most costly part, that is easy to parallelize. Compared to collocation the NLP is of smaller dimension but less sparse. This loss of sparsity, together with the cost of the underlying ODE solution leads to theoretically higher costs per SQP iteration than in collocation. On the other hand, the possibility to use efficient state-of-the-art ODE/DAE solvers and their inbuilt adaptivity makes direct multiple shooting a strong competitor to direct collocation in terms of CPU time per iteration. From a practical point of view it offers the advantage that the user does not have to decide on the grid of the ODE discretization, but only on the control grid. Direct multiple shooting was used to solve practical offline optimal control problems e.g. in [24, 33], and it is also used for the calculations in this paper. It is also widely used in online optimization and NMPC applications e.g. in [44, 43, 52, 53, 18, 25].

3 Time Optimal Control of a Five Link Robot Arm

We consider the time optimal point to point motion of a robot arm with five degrees of freedom. Figure 8 shows the robot and its possible movements. To provide a better visualization the last link and the manipulator in the images are shorter and simplified compared to the assumed model parameters.

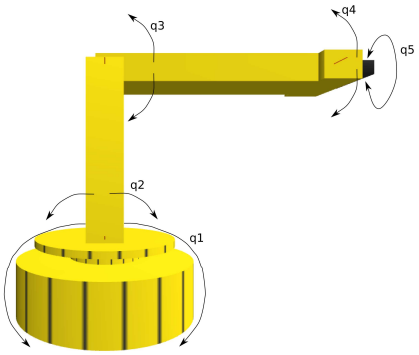

Fig. 8. Robot appearance with simplified last link and manipulator.

The robot is modelled as a kinematic chain of rigid bodies, i.e.,the robot is assumed to just consist of joints and links between them. The robot has a rotational base joint with two degrees of freedom, followed by two links with rotary joints, and finally one rotational joint at the "hand" of the arm. Each of the five joints contains a motor to apply a torque $u_i(t)$. The geometric description of the robot uses the notation of Denavit and Hartenberg [16]. To provide the data for the dynamic calculation each link is associated with an inertia tensor, the mass and the position of the center of mass. This approach leads to a set of five generalized coordinates $(q_1(t), \ldots, q_5(t))$ each representing a rotation in the corresponding joint. We have chosen parameters that correspond to a small toy robot arm, and which are listed in Table 1 using the conventional Denavit-Hartenberg notation. The corresponding equations of motion can then be generated automatically by a script from the HuMAnS Toolbox [29].

3.1 Fast computations of the dynamics of robots

The dynamics of a robot is most usually presented in its Lagrangian form

$$M(q(t))\,\ddot{q}(t) + N(q(t), \dot{q}(t)) = u(t),$$

which gives a compact description of all the nonlinear phenomena and can be manipulated easily in various ways. Since the mass matrix $M(q(t))$ is Symmetric Definite

Table 1. Dynamic data of the example robot, and Denavit-Hartenberg parameters.

Joint i	mass m_i	c.o.m. r_i	inertia tensor I_i	α_i	a_i	θ_i	d_i
1	0.1	$(0,0,0)^{\mathrm{T}}$	$\mathrm{diag}(23,23,20)\cdot 10^{-6}$	0	0	$q_1(t)$	0
2	0.02	$(0.06,0,0)^{\mathrm{T}}$	$\mathrm{diag}(7,118,113)\cdot 10^{-6}$	$-\frac{\pi}{2}$	0	$-\frac{\pi}{2}+q_2(t)$	0
3	0.1	$(0.06,0,0)^{\mathrm{T}}$	$\mathrm{diag}(20,616,602)\cdot 10^{-6}$	0	0.12	$\frac{\pi}{2}+q_3(t)$	0
4	0.03	$(0,-0.04,0)^{\mathrm{T}}$	$\mathrm{diag}(-51,-7,-46)\cdot 10^{-6}$	0	0.12	$\frac{\pi}{2}+q_4(t)$	0
5	0.06	$(0,0,0.1)^{\mathrm{T}}$	$\mathrm{diag}(650,640,26)\cdot 10^{-6}$	$\frac{\pi}{2}$	0	$q_5(t)$	0

Positive, it is invertible and the acceleration of the system can be related with the controls $u(t)$ either in the way

$$u(t) = M(q(t))\,\ddot{q}(t) + N(q(t),\dot{q}(t)) \tag{11}$$

or in the way

$$\ddot{q}(t) = M(q(t))^{-1}\big(u(t) - N(q(t),\dot{q}(t))\big), \tag{12}$$

corresponding respectively to the inverse and direct dynamics of the system. Very helpful from the point of view of analytical manipulations [56], this way of describing the dynamics of a robot is far from being efficient from the point of view of numerical computations, neither in the form (11) nor (12). Especially the presence of a matrix-vector multiplication of $O(N^2)$ complexity in both (11) and (12), and of a matrix inversion of $O(N^3)$ complexity in (12) can be avoided: recursive algorithms for computing both (11) and (12) with only an $O(N)$ complexity are well known today.

The first algorithm that has been investigated historically for the fast computation of the dynamics of robots is the Recursive Newton-Euler Algorithm that allows computing directly the controls related to given accelerations exactly as in (11). Extensions have been devised also for cases when not all of the acceleration vector $\ddot{q}$ is known, in the case of underactuated systems such as robots executing aerial maneuvers [49]. This recursive algorithm is the fastest way to compute the complete dynamics of a robotic system and should be preferred therefore as long as one is not strictly bound to using the direct dynamics (12). This is the case for collocation methods but unfortunately not for shooting methods.

The Recursive Newton-Euler Algorithm has been adapted then in the form of the Composite Rigid Body Algorithm in order to compute quickly the mass matrix that needs to be inverted in the direct dynamics (12), but we still have to face then a matrix inversion which can be highly inefficient for "large" systems. The computation of this mass matrix and its inversion can be necessary though for systems with unilateral contacts, when some internal forces are defined through implicit laws [55].

The Articulated Body Algorithm has been designed then to propose a recursive method of $O(N)$ complexity for computing directly the accelerations related to given torques as in (12) but without resorting to a matrix inversion. Even though generating a slightly higher overhead, this algorithm has been proved to be more efficient than the Composite Rigid Body Algorithm for robots with as few as 6 degrees of freedom [26]. Moreover, avoiding the matrix inversion allows producing a less noisy numerical result, what can greatly enhance the efficiency of any adaptive

ODE solver to which it is connected [2]. For these reasons, this recursive algorithm should be preferred as soon as one needs to compute the direct dynamics (12), what is the case for shooting methods.

Now, one important detail when designing fast methods to compute numerically the dynamics of a robot is to generate offline the computer code corresponding to the previous algorithms. Doing so, not only is it possible to get rid of constants such as 0 and 1 with all their consequences on subsequent computations, but it is also possible to get rid of whole series of computations which may appear to be completely unnecessary, depending on the specific structure of the robot. Such an offline optimization leads to computations which can be as much as twice faster than the strict execution of the same original algorithms.

The HuMAnS toolbox [29], used to compute the dynamics of the robot for the numerical experiment in the next section, proposes only the Composite Rigid Body Algorithm, so far, so even faster computations should be expected when using an Articulated Body Algorithm. Still, this toolbox produces faster computations than other generally available robotics toolboxes thanks to its offline optimization of the generated computer code (a feature also present in the SYMORO software [30]).

3.2 Optimization Problem Formulation

In order to solve the problem to minimize a point to point motion of the robot arm, we consider the following example maneuver: the robot shall pick up an object at the ground and put it as fast as possible into a shelf, requiring a base rotation of ninety degrees. We formulate an optimal control problem of the form (1), with the following definitions:

$$
\begin{aligned}
x(t) &= (q_1(t), \ldots, q_5(t), \dot{q_1}(t), \ldots, \dot{q_5}(t))^{\mathrm{T}} \\
u(t) &= (u_1(t), \ldots, u_5(t))^{\mathrm{T}} \\
L(x(t), u(t)) &= 1 \\
E(x(T)) &= 0 \\
f(x(t), u(t)) &= \begin{pmatrix} (\dot{q_1}(t), \ldots, \dot{q_5}(t))^{\mathrm{T}} \\ M(x(t))^{-1} \cdot (u(t) - N(x(t))) \end{pmatrix} \\
x_0 &= (-0.78, 0.78, 0, 0.78, 0, 0, 0, 0, 0, 0)^{\mathrm{T}} \\
r(x(T)) &= x(T) - (0.78, 0, -0.78, 0.78, 0, 0, 0, 0, 0, 0)^{\mathrm{T}} \\
h(x(t), u(t)) &= \begin{pmatrix} x_{\max} - x \\ x - x_{\min} \\ u_{\max} - u \\ u - u_{\min} \\ (1,0,0,1) \cdot T_5^0(x(t)) \cdot (0,0,l,1)^{\mathrm{T}} - 0.05 \\ (0,0,1,1) \cdot T_5^0(x(t)) \cdot (0,0,l,1)^{\mathrm{T}} + 0.15 \end{pmatrix}
\end{aligned}
$$

The controls $u(t)$ are the torques acting in the joints. The cost functional $\int_0^T L(x,u)dt + E(x(T))$ is the overall maneuver time, T. Within the dynamic model $\dot{x} = f(x,u)$, the matrix $M(x(t))$ is the mass matrix which is calculated in each evaluation of $f(x(t), u(t))$ and inverted using a cholesky algorithm. The vector $N(x(t))$ describes the combined centrifugal, Coriolis and gravitational force. The initial and terminal

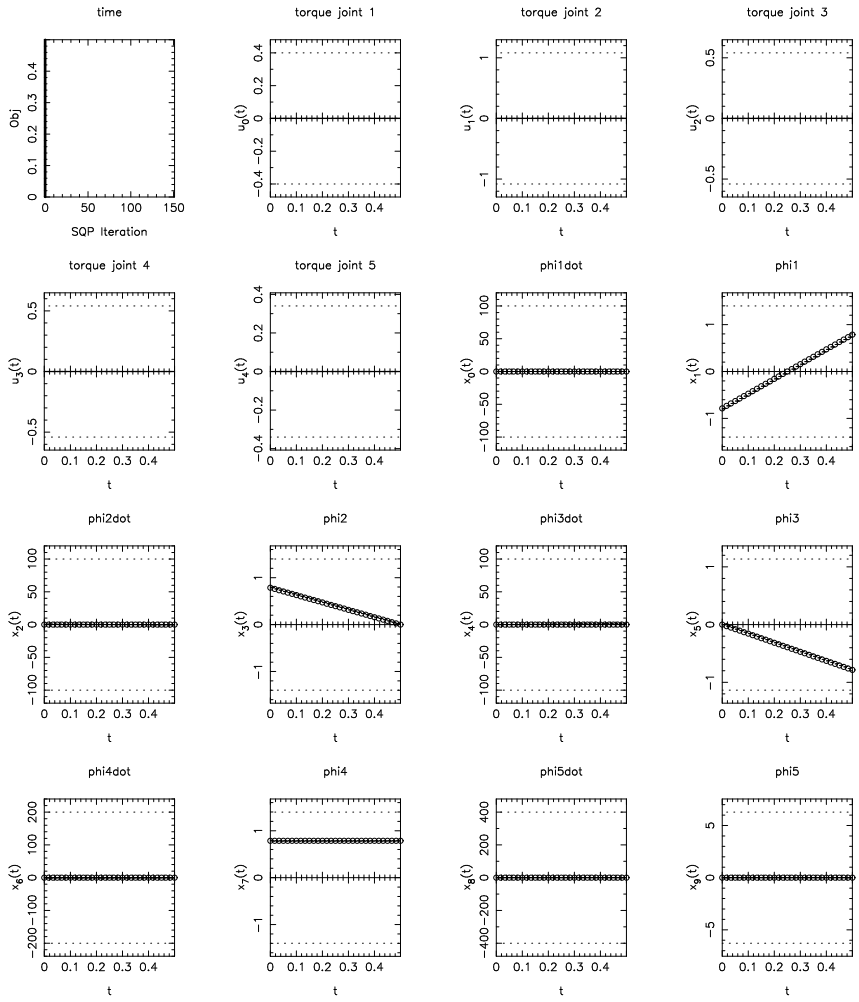

Fig. 9. Initalization of the optimization problem by linear interpolation.

constraints $x(0) = x_0$ and $r(x(T)) = 0$ describe the desired point to point maneuver. As the states and controls have lower and upper bounds, and as the robot hand shall avoid hitting the the ground as well as its own base, we add the path constraints $h(x, u) \geq 0$. Here, the matrix $T_5^0(x(t)$ describes the transformation that leads from the local end effector position $(0, 0, l, 1)^{\mathrm{T}}$ in the last frame to the absolute coordinates in the base frame.

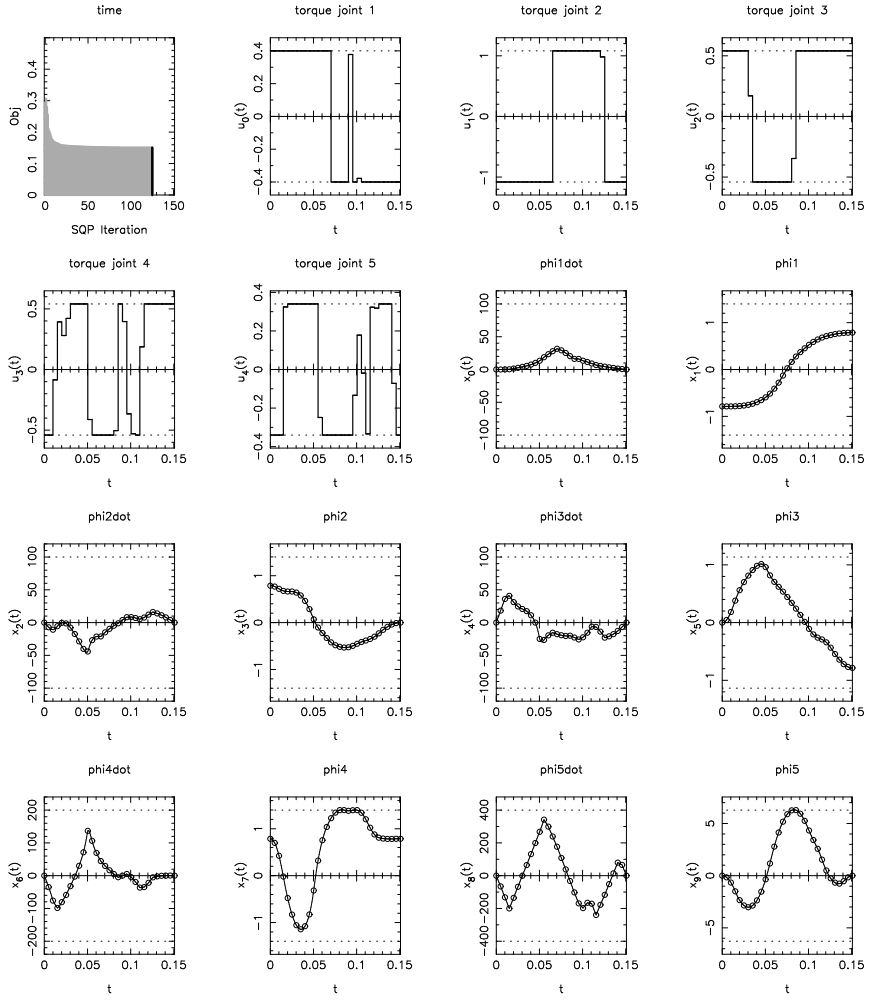

Fig. 10. Solution of the optimization problem, obtained after 130 SQP iterations and 20 CPU seconds

3.3 Numerical Solution by Direct Multiple Shooting

We have coupled the automatic robot model generator HuMAnS [29] with an efficient implementation of the direct multiple shooting method, the optimal control package MUSCOD-II [32, 33]. This coupling allows us to use the highly optimized C-code delivered by HuMAnS within the model equations $\dot{x} = f(x, u)$ required by MUSCOD-II in an automated fashion. In the following computations, we choose an error controlled Runge-Kutta-Fehlberg integrator of order four/five. We use 30

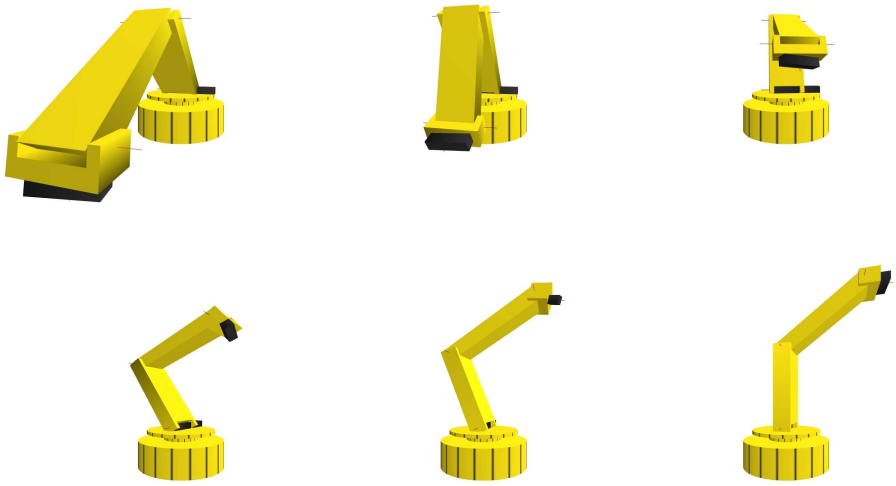

Fig. 11. Visualization of the time optimal point to point motion from Figure 10.

multiple shooting nodes with piecewise constant controls. Within the SQP method, a BFGS Hessian update and watchdog line search globalisation is used.

For initialization, the differential states on the multiple shooting nodes are interpolated linearly between desired initial and terminal state, as shown in Figure 9. The maneuver time for initialization was set to 0.3 seconds, and the controls to zero. Starting with this infeasible initialization, the overall optimization with MUSCOD-II took about 130 SQP iterations, altogether requiring about 20 CPU seconds on a standard LINUX machine with a 3 GHz Pentium IV processor. The solution is shown in Figure 10. The calculated time optimal robot movement of 0.15 seconds duration is illustrated in Figure 11 with screenshots from an OpenGL visualization.

4 Nonlinear Model Predictive Control

As mentioned in the introduction, Nonlinear Model Predictive Control (NMPC) is a feedback control technique based on the online solution of open-loop optimal control problems of the form (1). The optimization is repeated again and again, at intervals of length δ, each sampling time $t_k = k\delta$ for the most currently observed system state $\bar{x}(t_k)$, which serves as initial value $x_0 := \bar{x}(t_k)$ in (1). We have introduced the bar to distinguish the observed system states $\bar{x}(t)$ from the predicted states $x(t)$ within the

optimal control problem. Note that the time t_k from now on is the physical time, and no longer the time at a discretization point within the optimal control problem, as in Section 2. We stress that for autonomous systems, as treated in this paper, the NMPC optimization problems differ by the varying initial values only, and that the time coordinate used within the optimal control problem (1) can be assumed to always start with $t = 0$ even though this does not reflect the physical time. From now on, we will denote the time coordinate within the optimal control problem with τ in this section to avoid confusion.

To be specific, we denote the optimal solution of the optimal control problem (1) by $u^*(\tau; \bar{x}(t_k))$, $\tau \in [0, T^*(\bar{x}(t_k))]$, to express its parametric dependence on the initial value $\bar{x}(t_k)$. The feedback control implemented during the following sampling interval, i.e. for $t \in [t_k, t_{k+1}]$, is simply given by $u_0^*(\bar{x}(t_k)) := u^*(0; \bar{x}(t_k))$.[3] Thus, NMPC is a sampled data feedback control technique. It is closely related to *optimal feedback control* which would apply the continuous, non-sampled feedback law $u_0^*(\bar{x}(t))$ for all t, which can be called the limit of NMPC for infinitely small sampling times δ. Note that the optimal predicted maneuver time $T^*(x_k)$ would typically be shrinking for an optimal point to motion. In this case we speak of *shrinking horizon NMPC* [10]. If a large disturbance occurs, the horizon might also be enlarged as the future plan is changed. In the other case, when the horizon length is fixed to $T = T_p$, where the constant T_p is the *prediction horizon* length, we speak of *moving*, or *receding horizon control (RHC)* [37]. The moving horizon approach is applicable to continuous processes and so widely employed that the term NMPC is often used as synonymous to RHC. When a given trajectory shall be tracked, this is often expressed by the choice of the cost function in form of an integrated least squares deviation on a fixed prediction horizon. In fast robot motions, however, we believe that a variable time horizon for point to point maneuvers will be a crucial ingredient to successful NMPC implementations. A shrinking horizon NMPC approach for robot point to point motions that avoids that $T^*(x)$ shrinks below a certain positive threshold was presented by Zhao et al. [57]. For setpoint tracking problems, extensive literature exists on the stability of the closed loop system. Given suitable choices of the objective functional defined via L and E and a terminal constraint of the form $r(x(T)) = 0$ or $r(x(T)) \geq 0$, stability of the nominal NMPC dynamics can be proven even for strongly nonlinear systems [37, 15, 38, 35].

One important precondition for successful NMPC applications, however, is the availability of reliable and efficient numerical optimal control algorithms. Given an efficient offline optimization algorithm – e.g. one of the three SQP based direct methods described in Section 2 – we might be tempted to restart it again and again for each new problem and to solve each problem until a prespecified convergence criterion is satisfied. If we are lucky, the computation time is negligible; if we are not, we have to enter the field of real-time optimization.

[3] Sometimes, instead of the optimal initial control value $u^*(0; \bar{x}(t_k))$, the whole first control interval of length δ, i.e., $u^*(\tau; \bar{x}(t_k)), \quad \tau \in [0, \delta]$, is applied to the real process. This is more appealing in theory, and stability proofs are based on such an NMPC formulation. When a control discretization with interval lengths not smaller than the sampling time is used, however, both formulations coincide.

The Online Dilemma

Assuming that the computational time for one SQP iteration is more or less constant, we have to address the following dilemma: If we want to obtain a sufficiently exact solution for a given initial value $\bar{x}(t_k)$, we have to perform several SQP iterations until a prespecified convergence criterion is satisfied. We can suppose that for achieving this we have to perform n iterations, and that each iteration takes a time ϵ. This means that we obtain the optimal feedback control $u_0^*(\bar{x}(t_k))$ only at a time $t_k + n\epsilon$, i.e., with a considerable delay. However at time $t_k + n\epsilon$ the system state has already moved to some system state $\bar{x}(t_k + n\epsilon) \neq \bar{x}(t_k)$, and $u_0^*(x(t_k))$ is *not* the exact NMPC feedback, $u_0^*(\bar{x}(t_k + n\delta))$. In the best case the system state has not changed much in the meantime and it is a good approximation of the exact NMPC feedback. Also, one might think of predicting the most probable system state $\bar{x}(t_k + n\epsilon)$ and starting to work on this problem already at time t_k. The question of which controls have to be applied in the meantime is still unsolved: a possible choice would be to use previously optimized controls in an open-loop manner. Note that with this approach we can realize an NMPC recalculation rate with intervals of length $\delta = n\epsilon$, under the assumption that *each* problem needs at most n iterations and that each SQP iteration requires at most a CPU time of ϵ. Note also that feedback to a disturbance comes with a delay δ_d of one full sampling time. Summarizing, we would have $\delta_d = \delta = n\epsilon$.

4.1 Real-Time Iteration Scheme

We will now present a specific answer to the online dilemma, the real-time iteration scheme [17, 20]. The approach is based on two observations.

- Due to the online dilemma, we will never be able to compute the *exact* NMPC feedback control $u_0^*(\bar{x}(t_k))$ without delay. Therefore, it might be better to compute only an *approximation* $\tilde{u}_0(\bar{x}(t_k))$ of $u_0^*(\bar{x}(t_k))$, if this approximation can be computed much faster.
- Second, we can divide the computation time of each cycle into a a short feedback phase (FP) and a possibly much longer preparation phase (PP). While the feedback phase is only used to evaluate the approximation $\tilde{u}_0(\bar{x}(t_k))$, the following preparation phase is used to *prepare* the next feedback, i.e., to compute $\tilde{u}_0(\bar{x}(t_{k+1}))$ as much as possible without knowledge of $\bar{x}(t_{k+1})$.

This division of the computation time within each sampling interval allows to achieve delays δ_d that are smaller than the sampling interval δ, see Figure 12. The crucial question is, of course, which approximation $\tilde{u}_0(\bar{x}(t_k))$ should be used, and how it can be made similar to the exact NMPC feedback $u_0^*(\bar{x}(t_k))$.

In its current realization, the real-time iteration scheme is based on the direct multiple shooting method. The online optimization task is to solve a sequence of nonlinear programming problems of the form (10), but with varying initial value constraint $s_0 - \bar{x}(t_k) = 0$. Similar to the NLP notation (2), in the online context we have to solve, as fast as possible, an NLP

$$P(\bar{x}(t_k)): \qquad \min_w a(w) \quad \text{subject to} \quad b_{\bar{x}(t_k)}(w) = 0, \quad c(w) \geq 0, \qquad (13)$$

where the index takes account of the fact that the first equality constraint $s_0 - \bar{x}(t_k) = 0$ from $b_{\bar{x}(t_k)}(w) = 0$ depends on the initial value $\bar{x}(t_k)$, and where $w =$

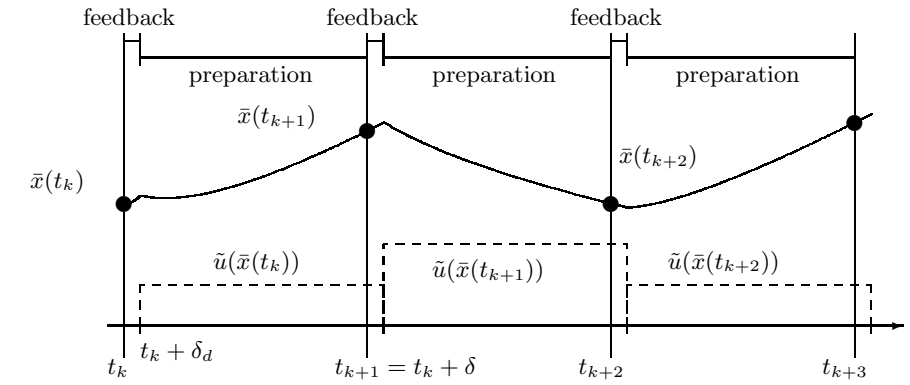

Fig. 12. Division of the computation time in the real-time iteration scheme; real system state and control trajectory, for sampling time δ and feedback delay $\delta_d \ll \delta$.

$(s_0, q_0, s_1, q_1, \ldots, s_N)$. Ideally, we would like to have the solution $w^*(\bar{x}(t_k))$ of each problem $P(\bar{x}(t_k))$ as quick as possible, and to take the NMPC feedback law to be the first control within $w^*(\bar{x}(t_k))$, i.e., to set $u_0^*(\bar{x}(t_k)) := q_0^*(\bar{x}(t_k))$. The exact solution manifold $w^*(\cdot)$ in dependence of the initial value $\bar{x}(t_k)$ is sketched as the solid line in Figure 13 – nondifferentiable points on this manifold are due to active set changes in the NLP. The exact solution, however, is not computable in finite time.

Initial Value Embedding

The major idea underlying the real-time iteration scheme is to initialize each new problem $P(\bar{x}(t_k))$ with the most current solution guess from the last problem, i.e. with the solution of $P(\bar{x}(t_{k-1}))$. In a simultaneous method like direct multiple shooting, it is no problem that the initial value constraint $s_0 - \bar{x}(t_k) = 0$ is violated. On the contrary, because this constraint is linear, it can be shown that the first SQP iteration after this "initial value embedding" is a first order predictor for the correct new solution, even in the presence of active set changes [17]. This observation is visualized in Figure 13, where the predictor delivered by the first SQP iteration is depicted as dashed line.

In the real-time iteration scheme, we use the result of the first SQP iteration directly for the approximation $\tilde{u}_0(\bar{x}(t_k))$. This would already reduce the feedback delay δ_d to the time of one SQP iteration, ϵ. Afterwards, we would need to solve the old problem to convergence in order to prepare the next feedback. In Fig. 13 also the second iterate and solution for problem $P(\bar{x}(t_k))$ are sketched. But two more considerations make the algorithm even faster.

- First, the computations for the first iteration can be largely performed *before* the initial value $\bar{x}(t_k)$ is known. Therefore, we can reduce the delay time further, if we perform all these computations before time t_k, and at time t_k we can quickly compute the feedback response $\tilde{u}_0(\bar{x}(t_k))$ to the current state. Thus, the feedback delay δ_d becomes even smaller than the cost of one SQP iteration, $\delta_d \ll \epsilon$.

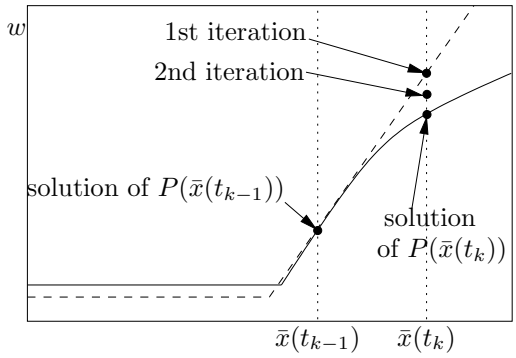

Fig. 13. Exact solution manifold (solid line) and tangential predictor after initial value embedding (dashed line), when initialized with the solution of $P(\bar{x}(t_{k-1}))$. The first iteration delivers already a good predictor for the exact solution of $P(\bar{x}(t_k))$.

- Second, taking into account that we already use an approximate solution of the optimal control problem we can ask if it is really necessary to iterate the SQP until convergence requiring a time $n\epsilon$ for n SQP iterations. Instead, we will considerably reduce the preparation time by performing just one iteration per sampling interval. This allows shorter sampling intervals that only have the duration of one single SQP iteration, i.e., $\delta = \epsilon$. A positive side-effect is that this shorter recalculation time most probably leads to smaller differences in subsequent initial states $\bar{x}(t_k)$ and $\bar{x}(t_{k+1})$, so that the initial value embedding delivers better predictors.

These two ideas are the basis of the real-time iteration scheme. It allows to realize feedback delays δ_d that are much shorter than a sampling time, and sampling times δ that are only as long as a single SQP iteration, i.e. we have $\delta_d \ll \delta = \epsilon$. Compared with the conventional approach with $\delta_d = \delta = n\epsilon$, the focus is now shifted from a sequence of optimization problems to a sequence of SQP iterates: we may regard the SQP procedure iterating uninterrupted, with the only particularity that the initial value $\bar{x}(t_k)$ is modified *during* the iterations. The generation of the feedback controls can then be regarded as a by-product of the SQP iterations. Due to the initial value embedding property, it can be expected that the iterates remain close to the exact solution manifold for each new problem. In Figure 14 four consecutive real-time iterates are sketched, where the dashed lines show the respective tangential predictors.

Applications

The real-time iteration scheme has successfully been used in both simulated and experimental NMPC applications, among them the experimental NMPC of a high purity distillation column [23] described by a 200 state DAE model with sampling times δ of 20 seconds, or simulated NMPC of a combustion engine described by 5 ODE, with sampling times of 10 milliseconds [27]. Depending on the application,

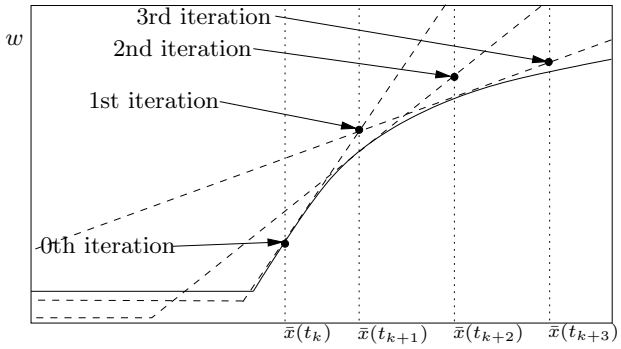

Fig. 14. Sketch of the real-time iterations that stay close to the exact solution manifold (solid line).

the feedback delay δ_d was between 0.5 and 5 percent of the sampling time. Within the studies, the approximation errors of the real-time iteration scheme compared to exact NMPC are often negligible. The scheme's theoretical contraction properties have been investigated in [21] for the variant described in this paper, and in [22, 19] for other variants. Recently, several novel variants of the real-time iteration scheme have been proposed that can either work with inexact jacobians within the SQP procedure [11], or that only evaluate the jacobians on a subspace [46]. These variants offer advantages for large scale systems where they promise to allow sampling times that are one or two orders of magnitude smaller than in the standard implementation. A numerical application of the standard real-time iteration scheme to the time optimal point to point motion of a robot arm described by 8 ODE was presented in [57], with CPU times still in the order of 100 milliseconds per sampling time. We expect that the development of real-time iteration variants that are specially tailored to robotics applications will make NMPC of fast robot motions possible within the next five years.

5 Summary and Outlook

In this tutorial paper, we have tried to give a (personal) overview over the most widely used methods for numerical optimal control, and to assess the possibility of real-time optimization of fast robot motions. We discussed in detail direct optimal control methods that are based on a problem discretization and on the subsequent use of a nonlinear programming algorithm like sequential quadratic programming (SQP). We compared three direct methods, (i) direct single shooting as a *sequential* approach, together with (ii) direct collocation and (iii) direct multiple shooting as *simultaneous* approaches. At hand of a tutorial example we have illustrated the better nonlinear convergence properties of the simultaneous over the sequential approaches that can be observed in many other applications, too. The direct multiple shooting method allows to use state-of-the-art ODE/DAE integrators with inbuilt adaptivity and error control which often shows to be an advantage in practice. At the example

of the time optimal motion of a robot arm we have demonstrated the ability of direct multiple shooting to cope even with strongly nonlinear two point boundary value optimization problems. Using the coupling of an efficient tool for generation of optimized robot model equations, HuMAnS, and a state-of-the-art implementation of the direct multiple shooting method, MUSCOD-II, computation times for a five link robot are in the order of 200 milliseconds per SQP iteration. Finally, we discussed the possibility to generate optimization based feedback by the technique of nonlinear model predictive control (NMPC), and pointed out the necessity of fast online optimization. We have presented the *real-time iteration* scheme – that is based on direct multiple shooting and SQP – as a particularly promising approach to achieve this aim. The scheme uses an *initial value embedding* for the transition from one optimization problem to the next, and performs exactly one SQP-type iteration per optimization problem to allow short sampling times. Furthermore, each iteration is divided into a preparation and a much shorter feedback phase, to allow an even shorter feedback delay. Based on the ongoing development of the presented approaches, we expect NMPC – that performs an online optimization of nonlinear first principle robot models within a few milliseconds – to become a viable technique for control of fast robot motions within the following five years.

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. U.M. Ascher, D.K. Pai, and B.P. Cloutier. Forward dynamics, elimination methods, and formulation stiffness in robot simulation. *International Journal of Robotics Research*, 16(6):749–758, 1997.
3. V. Bär. Ein Kollokationsverfahren zur numerischen Lösung allgemeiner Mehrpunktrandwertaufgaben mit Schalt– und Sprungbedingungen mit Anwendungen in der optimalen Steuerung und der Parameteridentifizierung. Master's thesis, Universität Bonn, 1984.
4. R.A. Bartlett, A. Wächter, and L.T. Biegler. Active set vs. interior point strategies for model predictive control. In *Proc. Amer. Contr. Conf.*, pages 4229–4233, Chicago, Il, 2000.
5. R.E. Bellman. *Dynamic Programming*. University Press, Princeton, 1957.
6. D.P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, Belmont, MA, 1995.
7. L. T. Biegler, A. M. Cervantes, , and A. Waechter. Advances in simultaneous strategies for dynamic process optimization. *Chemical Engineering Science*, 4(57):575–593, 2002.
8. L.T. Biegler. Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Computers and Chemical Engineering*, 8:243–248, 1984.
9. L.T. Biegler. Efficient solution of dynamic optimization and NMPC problems. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, volume 26 of *Progress in Systems Theory*, pages 219–244, Basel Boston Berlin, 2000. Birkhäuser.

10. T. Binder, L. Blank, H.G. Bock, R. Bulirsch, W. Dahmen, M. Diehl, T. Kronseder, W. Marquardt, J.P. Schlöder, and O.v. Stryk. Introduction to model based optimization of chemical processes on moving horizons. In M. Grötschel, S.O. Krumke, and J. Rambau, editors, *Online Optimization of Large Scale Systems: State of the Art*, pages 295–340. Springer, 2001.
11. 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, 2006. (in print).
12. H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings 9th IFAC World Congress Budapest*, pages 243–247. Pergamon Press, 1984.
13. A.E. Bryson and Y.-C. Ho. *Applied Optimal Control*. Wiley, New York, 1975.
14. A. Cervantes and L.T. Biegler. Large-scale DAE optimization using a simultaneous NLP formulation. *AIChE Journal*, 44(5):1038–1050, 1998.
15. H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34(10):1205–1218, 1998.
16. J. Denavit and R.S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *ASME Journal of Applied Mechanics*, 22:215–221, 1955.
17. M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*. PhD thesis, Universität Heidelberg, 2001. http://www.ub.uni-heidelberg.de/archiv/1659/.
18. 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. Download also at: http://www.ub.uni-heidelberg.de/archiv/1659/.
19. M. Diehl, H.G. Bock, and J.P. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5):1714–1736, 2005.
20. 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.
21. 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. (submitted).
22. 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, 2005.
23. M. Diehl, R. Findeisen, S. Schwarzkopf, I. Uslu, F. Allgöwer, H.G. Bock, E.D. Gilles, and J.P. Schlöder. An efficient algorithm for nonlinear model predictive control of large-scale systems. Part II: Application to a distillation column. *Automatisierungstechnik*, 51(1):22–29, 2003.
24. M. Diehl, D.B. Leineweber, A.A.S. Schäfer, H.G. Bock, and J.P. Schlöder. Optimization of multiple-fraction batch distillation with recycled waste cuts. *AIChE Journal*, 48(12):2869–2874, 2002.

25. M. Diehl, L. Magni, and G. De Nicolao. Online NMPC of unstable periodic systems using approximate infinite horizon closed loop costing. *Annual Reviews in Control*, 28:37–45, 2004.
26. R. Featherstone and D. Orin. Robot dynamics: Equations and algorithms. In *Proceedings of the IEEE International Conference on Robotics & Automation*, 2000.
27. H. J. Ferreau, G. Lorini, and M. Diehl. Fast nonlinear model predictive control of gasoline engines. In *CCA Conference Munich*, 2006. (submitted).
28. R. Findeisen, M. Diehl, I. Uslu, S. Schwarzkopf, F. Allgöwer, H.G. Bock, J.P. Schlöder, and E.D. Gilles. Computation and performance assesment of nonlinear model predictive control. In *Proc. 42th IEEE Conf. Decision Contr.*, pages 4613–4618, Las Vegas, USA, 2002.
29. INRIA. Humans toolbox. http://www.inrialpes.fr/bipop/software/humans/, Feb 2005.
30. W. Khalil and D. Creusot. Symoro+: A system for the symbolic modelling of robots. *Robotica*, 15:153–161, 1997.
31. D. Kraft. On converting optimal control problems into nonlinear programming problems. In K. Schittkowski, editor, *Computational Mathematical Programming*, volume F15 of *NATO ASI*, pages 261–280. Springer, 1985.
32. D.B. Leineweber. *Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models*, volume 613 of *Fortschritt-Berichte VDI Reihe 3, Verfahrenstechnik*. VDI Verlag, Düsseldorf, 1999.
33. D.B. Leineweber, A.A.S. Schäfer, H.G. Bock, and J.P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part II: Software aspects and applications. *Computers and Chemical Engineering*, 27:167–174, 2003.
34. P.L. Lions. *Generalized Solutions of Hamilton-Jacobi Equations*. Pittman, 1982.
35. L. Magni, G. De Nicolao, L. Magnani, and R. Scattolini. A stabilizing model-based predictive control for nonlinear systems. *Automatica*, 37(9):1351–1362, 2001.
36. 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 Boston Berlin, 2000. Birkhäuser.
37. D.Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, 35(7):814–824, 1990.
38. 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 Boston Berlin, 2000. Birkhäuser.
39. J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, Heidelberg, 1999.
40. L.S. Pontryagin, V.G. Boltyanski, R.V. Gamkrelidze, and E.F. Miscenko. *The Mathematical Theory of Optimal Processes*. Wiley, Chichester, 1962.
41. PSE. *gPROMS Advanced User's Manual*. London, 2000.
42. 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.
43. L.O. Santos. *Multivariable Predictive Control of Nonlinear Chemical Processes*. PhD thesis, Universidade de Coimbra, 2000.

44. L.O. Santos, P. Afonso, J. Castro, N. M.C. de Oliveira, and L.T. Biegler. On-line implementation of nonlinear MPC: An experimental case study. In *AD-CHEM2000 - International Symposium on Advanced Control of Chemical Processes*, volume 2, pages 731–736, Pisa, 2000.
45. R.W.H. Sargent and G.R. Sullivan. The development of an efficient optimal control package. In J. Stoer, editor, *Proceedings of the 8th IFIP Conference on Optimization Techniques (1977), Part 2*, Heidelberg, 1978. Springer.
46. A. Schäfer, P. Kühl, M. Diehl, J.P. Schlöder, and H.G. Bock. Fast reduced multiple shooting methods for nonlinear model predictive control. *Chemical Engineering and Processing*, 2006. (submitted).
47. V.H. Schulz. *Reduced SQP methods for large-scale optimal control problems in DAE with application to path planning problems for satellite mounted robots*. PhD thesis, Universität Heidelberg, 1996.
48. V.H. Schulz. Solving discretized optimization problems by partially reduced SQP methods. *Computing and Visualization in Science*, 1:83–96, 1998.
49. G.A. Sohl. *Optimal Dynamic Motion Planning for Underactuated Robots*. PhD thesis, University of California, 2000.
50. O. von Stryk. Numerical solution of optimal control problems by direct collocation. In *Optimal Control: Calculus of Variations, Optimal Control Theory and Numerical Methods*, volume 129. Bulirsch et al., 1993.
51. M.J. Tenny, S.J. Wright, and J.B. Rawlings. Nonlinear model predictive control via feasibility-perturbed sequential quadratic programming. *Computational Optimization and Applications*, 28(1):87–121, April 2004.
52. S. Terwen, M. Back, and V. Krebs. Predictive powertrain control for heavy duty trucks. In *Proceedings of IFAC Symposium in Advances in Automotive Control*, pages 451–457, Salerno, Italy, 2004.
53. A. Toumi. *Optimaler Betrieb und Regelung von Simulated Moving Bed Prozessen*. PhD thesis, Fachbereich Bio und Chemieingenieurwesen, Universität Dortmund, 2004.
54. Oskar von Stryck. Optimal control of multibody systems in minimal coordinates. *Zeitschrift fur Angewandte Mathematik und Mechanik 78, Suppl 3*, 1998.
55. P.-B. Wieber. *Modélisation et Commande d'un Robot Marcheur Anthropomorphe*. PhD thesis, Ecole des Mines de Paris, 2000.
56. P.B. Wieber. Some comments on the structure of the dynamics of articulated motion. In *Proceedings of the Ruperto Carola Symposium on Fast Motion in Biomechanics and Robotics*, 2005.
57. J. Zhao, M. Diehl, R. Longman, H.G. Bock, and J.P. Schlöder. Nonlinear model predictive control of robots using real-time optimization. In *Proceedings of the AIAA/AAS Astrodynamics Conference*, Providence, RI, August 2004.