# Multiple Shooting in a Microsecond

**Rien Quirynen, Milan Vukov, and Moritz Diehl**

**Abstract** Nonlinear Model Predictive Control (NMPC) is a feedback control technique that uses the most current state estimate of a nonlinear system to compute an optimal plan for the future system behavior. This plan is recomputed at every sampling time, creating feedback. Thus, NMPC needs to repeatedly solve a nonlinear optimal control problem (OCP). Direct multiple shooting is since long known as a reliable approach for discretization of OCPs. This is mainly due to the fact that the approach shows good contraction properties within the NMPC framework. Moreover, the procedure is easy to initialize and parallelize. In the context of real-time NMPC, the multiple shooting method was tailored to the Real-Time Iteration (RTI) scheme. This scheme uses a strategy known as Initial Value Embedding to deal efficiently with the transition from one optimization problem to the next. It performs two algorithmic steps in each sampling time, a long preparation phase and a short feedback phase to minimize the feedback time to the system. The two phases respectively prepare and solve a convex Quadratic Program (QP) that depends parametrically on the estimated system state. The solution of this QP delivers quickly a generalized tangential predictor to the solution of the nonlinear problem. Recent algorithmic progress makes the solution of NMPC optimization problems possible at sampling times in the milli- or even microsecond range on modern computational hardware. An essential part is the simulation of the nonlinear model together with the propagation of its derivative information. This article describes the developments and their efficient software implementations that made it possible to solve a classical NMPC benchmark problem within 1 $\mu$s sampling time.

R. Quirynen (✉) • M. Vukov
KU Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium
e-mail: rien.quirynen@esat.kuleuven.be; milan.vukov@esat.kuleuven.be

M. Diehl
KU Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium

University of Freiburg, Georges-Koehler-Allee 102, 79110 Freiburg, Germany
e-mail: moritz.diehl@imtek.uni-freiburg.de



T. Carraro et al. (eds.), *Multiple Shooting and Time Domain Decomposition Methods*, Contributions in Mathematical and Computational Sciences 9,
DOI 10.1007/978-3-319-23321-5_7

## 1 Introduction

Model Predictive Control (MPC) needs to solve an Optimal Control Problem (OCP) at each sampling instant using the current system state $\bar{x}_0$ as initial value. This optimization task is almost exclusively executed using a *direct approach* which first discretizes the continuous time system to obtain a discrete time OCP formulation. Multiple Shooting (MS) will be motivated and shown to be such a time discretization with nice contraction properties and extra advantages, e.g. it is more flexible both to initialize and parallelize. One crucial dividing line in MPC is between convex and non-convex problems. In the case that the OCP is convex, e.g. for linear MPC, algorithms exist that find a global solution in a fast and reliable way. This paper will focus on the problems that use a nonlinear model i.e. the resulting OCP is non-convex and one generally has to be satisfied with approximations of locally optimal solutions. Nonlinear MPC (NMPC) has become a popular approach for real-time optimal control since it can explicitly handle constraints and nonlinear dynamics. Recent algorithmic progress [1, 2] allows to consider NMPC also for systems having rather fast dynamics. Among the available online algorithms, the Real-Time Iteration (RTI) scheme has been proposed as a highly competitive approach [3]. It is an SQP-type algorithm that uses a shooting discretization and a Gauss-Newton Hessian approximation.

It is important to use the right algorithmic tools to be able to meet the hard timing constraints of real-time applications. This paper focuses on the RTI scheme as an online algorithm to handle nonlinear OCPs using a multiple shooting discretization. It divides the computations at each sampling time into a preparation and a feedback phase [4]. The *preparation phase* takes care of the linearization and condensing resulting in a small scale Quadratic Program (QP). Reducing the computation time of this phase is crucial for achieving a certain sampling frequency since it will often dominate the total execution time. The prepared QP cannot be solved yet before the current state estimate is available. Once it becomes available, the *feedback phase* will quickly solve the subproblem to obtain an approximate solution to the original OCP. The faster this QP is solved, the faster the next control input can be fed back to the real process. Condensing is still a rather popular technique, because it leaves us with a dense but small scale QP to be solved in the feedback phase [5, 6]. This can be done by any embedded QP solver such as e.g. qpOASES [7] which uses an online active set strategy [8]. The alternative to this condensing approach would be to directly solve the multi-stage QP, using a structure exploiting convex solver such as FORCES [9], qpDUNES [10] or HPMPC [11]. An important disadvantage could be that the solution of the full QP then becomes part of the feedback phase in the context of the RTI framework.

Apart from using suitable algorithms, efficient implementations are needed to run them in real-time on embedded control hardware. One way to achieve this is by automatic code generation, i.e. by exporting a fully customized solver. Significant improvements in the computation time can be obtained by removing unnecessary computations, by optimizing the memory access and cache usage, and by exploiting the structure in the problem. This idea is already rather popular for convex optimization, examples of this are CVXGEN [12] and FORCES [9] which both generate tailored Interior Point (IP) convex solvers. In the context of NMPC, an important computational step is that of the integration and sensitivity generation for the nonlinear model. The export of tailored Explicit Runge-Kutta (ERK) integrators using the Variational Differential Equations (VDE) for sensitivity propagation has been presented and been experimentally validated in [13, 14]. This idea has been strongly extended in the work on automatic code generation for Implicit RK (IRK) methods with a tailored approach for computing their sensitivities [15, 16]. Embedded, implicit solvers allow their natural extension to Differential Algebraic Equations (DAE) and an efficient computation of continuous outputs [17]. The ACADO code generation tool pursues to export efficient C-code for the complete RTI scheme, assembled from the different necessary components [18]. It is part of the open-source ACADO Toolkit [19] with interfaces to multiple convex solvers [5, 10].

This paper is organized as follows. Section 2 presents the parametric optimization problem that needs to be solved at each sampling time together with its shooting discretization. This direct approach requires efficient integration and sensitivity generation for the nonlinear model, which is the topic of interest in Sect. 3. The optimization details are discussed in Sect. 4, focusing on Sequential Quadratic Programming (SQP) in a RTI framework. Finally, Sect. 5 shows us that NMPC can be done within $1\,\mu s$ for a benchmark problem taken from the literature and this using the tools provided in this paper.

## 2 Nonlinear Model Predictive Control

NMPC is an approach of increasing popularity for real-time control due to the ability to explicitly handle constraints and nonlinear dynamics that characterize the system of interest. Section 2.1 presents the optimization problem that needs to be solved at each sampling time. Section 2.2 then describes multiple shooting as a reliable way of reformulating this as an approximate but tractable problem.

## 2.1 Parametric Optimization Problem

In what follows, the OCP that needs to be solved at each time point is assumed to be of the following form

$$\begin{aligned}
\underset{x(\cdot),\,u(\cdot)}{\text{minimize}} \quad & \int_0^T \|F(t,x(t),u(t))\|_2^2 \,\mathrm{d}t + \|F_N(x(T))\|_2^2 && \text{(1a)}\\
\text{subject to} \quad & x(0) = \bar{x}_0, && \text{(1b)}\\
& \boldsymbol{\dot{x}(t) = f(t,x(t),u(t))}, \qquad \forall t \in [0,T], && \text{(1c)}\\
& 0 \geq h(x(t),u(t)), \qquad \forall t \in [0,T], && \text{(1d)}\\
& 0 \geq r(x(T)), && \text{(1e)}
\end{aligned}$$

where $x(t) \in \mathbb{R}^{n_x}$ denotes the differential states at time $t$, $u(t) \in \mathbb{R}^{n_u}$ are the control inputs and Eq. (1a) defines the NMPC objective while Eqs. (1d) and (1e) are respectively the path and terminal constraints. The nonlinear dynamics in Eq. (1c) are described by an explicit system of Ordinary Differential Equations (ODE), although this could be generalized to e.g. an implicit DAE system of index 1. Note that $\bar{x}_0 \in \mathbb{R}^{n_x}$ is a parameter on which the OCP depends through the initial value constraint in Eq. (1b). What is mainly of interest is $u^*(\bar{x}_0)$, which denotes a locally optimal control trajectory to be applied as a function of the current system state $\bar{x}_0$.

## 2.2 Multiple Shooting Discretization

The continuous time OCP formulation from (1) leaves us with an infinite dimensional optimization problem which is impossible to solve in a general case. This problem is often discretized and subsequently optimized which is characteristic for any *direct approach* to optimal control. An important separator here is whether such a direct approach is either *sequential* or *simultaneous*. Variants of the latter approach are *direct discretization* [20] and *direct multiple shooting* [21], the focus of this paper. A shooting discretization of the problem in Eq. (1) results in the structured Nonlinear Program (NLP)

$$\begin{aligned}
\underset{X,\,U}{\text{minimize}} \quad & \frac{1}{2}\sum_{i=0}^{N-1} \|F_i(x_i,u_i)\|_2^2 + \frac{1}{2}\,\|F_N(x_N)\|_2^2 && \text{(2a)}\\
\text{subject to} \quad & 0 = x_0 - \bar{x}_0, && \text{(2b)}\\
& \boldsymbol{0 = x_{i+1} - \Phi_i(x_i,u_i)}, \qquad i = 0,\ldots,N-1, && \text{(2c)}\\
& 0 \geq h_i(x_i,u_i), \qquad i = 0,\ldots,N-1, && \text{(2d)}\\
& 0 \geq r(x_N), && \text{(2e)}
\end{aligned}$$

with state trajectory $X = [x_0^\top, \ldots, x_N^\top]^\top$ where $x_i \in \mathbb{R}^{n_x}$ and control trajectory $U = [u_0^\top, \ldots, u_{N-1}^\top]^\top$ where $u_i \in \mathbb{R}^{n_u}$. Note that the function $\Phi_i(x_i, u_i)$ here denotes the simulation of the nonlinear dynamics over one shooting interval, starting from the states $x_i$ and using the control values $u_i$. This component is essential for any shooting method and will be the topic of interest in Sect. 3. When one addresses this optimization problem directly in a Newton-type framework, the variables in $X$ generally represent a feasible state trajectory only at convergence. In direct multiple shooting, simulation and optimization are therefore performed simultaneously.

On the other hand, a sequential approach carries out the simulation task separately from solving the optimization problem. A reduced OCP formulation is obtained after replacing the variables $x_i$ by the results $X_{\text{sim}}(\bar{x}_0, U)$ of a forward simulation starting from the initial states $\bar{x}_0$ using the control trajectory $U$. This technique is also known as *single shooting*. The equality constraints from Eq. (2c) are now automatically satisfied and can therefore be eliminated. Since the variable space of this problem is strongly reduced in dimension from $(N + 1)n_x + Nn_u$ to only $Nn_u$, the task of solving this NLP appears to be simplified. But it has been shown that the cost per Newton iteration can be made equal for both approaches because of the sparsity structure in (2). Advantages of multiple shooting over single shooting are also the stronger flexibility in initializing the problem and parallelizing the algorithm, and the improved convergence properties especially in case of an unstable system [22].

## 3 Auto Generated Integrators

This section presents auto generated RK methods with efficient sensitivity generation. As one-step methods, they are particularly suited for simulation over relatively short shooting intervals such as needed in Eq. (2c). Their implementation is discussed in Sect. 3.1, for ODE as well as DAE systems of index 1. Extending these methods with an efficient computation of continuous outputs and first order sensitivity information is respectively described in Sects. 3.2 and 3.3. In Sect. 3.4, a common three stage model formulation is briefly introduced.

### *3.1 Runge-Kutta Methods*

An integration method in general needs to solve the following Initial Value Problem (IVP) over a certain time interval $t \in [0, T_s]$:

$$\begin{aligned} 0 &= g(t, \dot{x}(t), x(t), z(t), u(t)), \\ x(0) &= x_0, \end{aligned} \tag{3}$$

with $x(t)$ a vector of $n_x$ differential states, $\dot{x}(t)$ the corresponding time derivatives and $z(t)$ a vector of $n_z$ algebraic states. This covers models ranging from explicit ODE to fully implicit DAE systems, which can be dealt with by IRK methods [15]. The only assumption is that the Jacobian matrix $\frac{\partial g(\cdot)}{\partial (z,\dot{x})}$ is invertible, i.e. the DAE system is of index 1. Mainly because of their good stability properties, the focus is often on A-stable schemes such as the Gauss methods [23].

With real-time applications in mind, a code generation tool exports a tailored integrator with a deterministic runtime. Applying an $s$-stage IRK method to the model in (3) results in a nonlinear system that can be solved using a Newton-type method. The step size, the order of the method and the number of Newton iterations have to be fixed such that there is no adaptivity. In the context of shooting methods, a good initialization of the variables using the previous solution is available so that a small amount of iterations is typically sufficient. Even a custom linear solver can be exported to perform the iterations, e.g. based on an LU decomposition. Note that an ERK method can be used in case of a model described by an explicit ODE system. Its code generation implementation is relatively trivial and these methods can also be more efficient when their use is restricted to non-stiff models [13].

### *3.2 Continuous Output*

Some promising possibilities of auto generated integration methods with continuous output have been illustrated in [13, 17]. The general idea is to define some output function $y = \psi(t, \dot{x}(t), x(t), z(t))$ that can be evaluated efficiently on an arbitrarily fine grid, independent of the integration grid. These output functions can then be used to define the objective function or some constraint functions in the NMPC formulation. In case of collocation methods which are a specific family of IRK methods, this continuous extension comes rather naturally. But it is also possible to define continuous extensions for explicit or semi-implicit RK methods.

### *3.3 Sensitivity Generation*

In the context of dynamic optimization, at least first order sensitivities with respect to the variables are needed in addition to the simulated values of states and outputs. A thorough discussion on techniques of forward sensitivity propagation for IRK methods can be found in [24]. The conclusion from that work is that the most efficient way is to apply the Implicit Function Theorem (IFT) to the nonlinear system of the IRK method. This direct approach also provides very accurate derivatives which is important for optimization. Note that for an explicit method, it is efficient to compute the first order derivatives by simulating the system extended

with the Variational Differential Equations (VDE):

$$\begin{aligned}\dot{x}(t) &= f(t, x(t), u(t)),\\ \dot{S}_x(t) &= \frac{\partial f(t, x(t), u(t))}{\partial x} S_x(t),\\ \dot{S}_u(t) &= \frac{\partial f(t, x(t), u(t))}{\partial x} S_u(t) + \frac{\partial f(t, x(t), u(t))}{\partial u},\end{aligned} \tag{4}$$

with $x(0) = x_0$ and the sensitivity matrices are defined as $S_x(t) = \partial x(t)/\partial x_0$ and $S_u(t) = \partial x(t)/\partial u$ for which it holds that $[\, S_x(0) \,|\, S_u(0) \,] = [\, \mathbb{1} \,|\, \mathbb{0} \,]$. Since a fixed step size is assumed for the auto generated integrators, this approach is equivalent to performing algorithmic differentiation in forward mode [18].

### 3.4 Linear Subsystems

When modeling a system, the result is typically a set of nonlinear differential equations with possibly some algebraic states but one would often recognize one or more of the following three subsystems in this specific order:

$$C_1 \dot{x}_{[1]} = A_1 x_{[1]} + B_1 u, \tag{5a}$$

$$0 = f_2(\dot{x}_{[1]}, x_{[1]}, \dot{x}_{[2]}, x_{[2]}, z, u), \tag{5b}$$

$$C_3 \dot{x}_{[3]} = A_3 x_{[3]} + f_3(\dot{x}_{[1]}, x_{[1]}, \dot{x}_{[2]}, x_{[2]}, z, u), \tag{5c}$$

with matrices $A_1$, $B_1$, $A_3$ and invertible matrices $C_1$ and $C_3$ and the nonlinear functions $f_2$ and $f_3$. The main assumption is that the Jacobian matrix $\frac{\partial f_2(\cdot)}{\partial (z, \dot{x}_{[2]})}$ is invertible, i.e. the second subsystem represents a DAE of index 1. In the case that $A_3 = 0$ and $C_3$ is an identity matrix, Eq. (5c) reduces to

$$\dot{x}_{[3]} = f_3(\dot{x}_{[1]}, x_{[1]}, \dot{x}_{[2]}, x_{[2]}, z, u) \tag{6}$$

which are better known as quadrature states [25]. They are typically used to formulate objective and constraint functions, similar to how the more general linear input and output states can be used respectively from Eqs. (5a) and (5c). The exploitation of this three-stage model formulation in auto generated integration methods has been presented and illustrated in [16].

## 4 Sequential Quadratic Programming

A Newton-type algorithm to solve the NLP in (2) is dedicated to find a locally optimal solution by solving the nonlinear Karush-Kuhn-Tucker (KKT) conditions. There are different options to treat the inequality constraints in this system. One way

which is done by nonlinear IP methods is to smoothen the corresponding KKT conditions, a popular software implementation of this is the code IPOPT [26]. The focus in this article is on Sequential Quadratic Programming (SQP), which defines another family of algorithms. The QP subproblem to be solved is described in Sect. 4.1 together with the popular Gauss-Newton Hessian approximation. Section 4.2 then presents two alternative ways to deal with this structured, multi-stage QP. Some important implementation details of the RTI scheme are discussed eventually in Sect. 4.3.

## *4.1 Generalized Gauss-Newton*

After linearizing the nonlinear functions in the KKT system, it becomes equivalent to solving the following Quadratic Program (QP)

$$\underset{X,\,U}{\text{minimize}} \qquad \Phi_{\text{quad}}(X, U; X^{[k]}, U^{[k]}, Y^{[k]}, \lambda^{[k]}) \tag{7a}$$

$$\text{subject to} \qquad G_{\text{eq,lin}}(\cdot) = \begin{bmatrix} x_0 - \bar{x}_0 \\ x_1 - \phi_0(x_0^{[k]}, u_0^{[k]}) - \begin{bmatrix} A_0^{[k]} & B_0^{[k]} \end{bmatrix} \begin{bmatrix} x_0 - x_0^{[k]} \\ u_0 - u_0^{[k]} \end{bmatrix} \\ \vdots \end{bmatrix} = 0, \tag{7b}$$

$$G_{\text{ineq,lin}}(\cdot) = \begin{bmatrix} h_0(x_0^{[k]}, u_0^{[k]}) + \begin{bmatrix} C_0^{[k]} & D_0^{[k]} \end{bmatrix} \begin{bmatrix} x_0 - x_0^{[k]} \\ u_0 - u_0^{[k]} \end{bmatrix} \\ \vdots \\ r(x_N^{[k]}) + C_N^{[k]}(x_N - x_N^{[k]}) \end{bmatrix} \leq 0, \tag{7c}$$

where $Y^{[k]}$ and $\lambda^{[k]}$ are the Lagrange multipliers for respectively the equality and inequality constraints and $A_i^{[k]}, B_i^{[k]}, C_i^{[k]}, D_i^{[k]}$ denote the Jacobian matrices of the corresponding functions evaluated at the current iterate $k$. Note that $\phi_i(\cdot)$, $A_i$ and $B_i$ for $i = 0, \ldots, N-1$ form the information that needs to be provided by the integration methods from Sect. 3. The result is a sequence of QPs of which the solution provides a sequence of iterations, converging to a local solution of the original, nonlinear optimal control problem under the assumptions as discussed in [27].

There are different variants of the SQP algorithm that use certain expressions for the QP objective in (7a). In case of an *exact Hessian* variant, the sparse Hessian of the Lagrangian $\nabla^2 \mathscr{L}$ needs to be evaluated and used in the QP subproblem leading to a locally quadratic convergence rate in solving the NLP. The Hessian is often approximated resulting in a trade-off between computational complexity per step and convergence speed. The special case of using a least squares objective such as

in Eq. (2a) allows a popular Hessian approximation that is known as the Generalized Gauss-Newton (GGN) method [28]. The objective function to be used in Eq. (7a) then reads as

$$\begin{aligned}\Phi_{\text{quad}} \;=\; & \frac{1}{2}\sum_{i=0}^{N-1}\begin{bmatrix} x_i - x_i^{[k]} \\ u_i - u_i^{[k]} \end{bmatrix}^{\top} J_i^{[k]\top} J_i^{[k]} \begin{bmatrix} x_i - x_i^{[k]} \\ u_i - u_i^{[k]} \end{bmatrix} + \sum_{i=0}^{N-1} F_i^{[k]\top} J_i^{[k]} \begin{bmatrix} x_i - x_i^{[k]} \\ u_i - u_i^{[k]} \end{bmatrix} \\ & + \frac{1}{2}\,(x_N - x_N^{[k]})^{\top} J_N^{[k]\top} J_N^{[k]}\,(x_N - x_N^{[k]}) \;+\; F_N^{[k]\top} J_N^{[k]}\,(x_N - x_N^{[k]})\end{aligned} \tag{8}$$

where $F_i^{[k]} = F_i(x_i^{[k]}, u_i^{[k]})$ and $F_N^{[k]} = F_N(x_N^{[k]})$ are evaluations of the residual functions from Eq. (2a) and $J_i^{[k]}$, $J_N^{[k]}$ are respectively their Jacobians $\frac{\partial F_i(x_i^{[k]}, u_i^{[k]})}{\partial (x_i, u_i)}$ and $\frac{\partial F_N(x_N^{[k]})}{\partial x_N}$. The GGN method is based on the observation that $J_i^{[k]\top} J_i^{[k]}$ for each $i = 0, \ldots, N$ forms a good approximation for the Hessian result $\nabla^2_{w_i} \mathscr{L}$ where $w_i = (x_i, u_i)$, as long as the residual evaluations $F_i(\cdot)$ remain small [27]. The convergence rate of the GGN method is only linear but its implementation is rather simple and works very well in practice for such small residual problems. A special case of this is when the original objective function $\Phi(X, U)$ was already convex quadratic, meaning that it can be used directly in the QP. The method can also be further generalized to Sequential Convex Programming e.g. in case of NMPC problems with elliptic terminal regions [29].

## *4.2 Sparsity Exploitation*

The QP subproblem presented in the previous Subsection shows a specific sparsity structure which should be exploited. One option is to reduce the variable space by a procedure called *condensing*, and then to solve the smaller, condensed QP using a suitable solver [5]. Another option is to directly use a tailored QP solver that can efficiently exploit this structure.

### 4.2.1 The Condensed Problem

To simplify notation, let us define the trajectories $\Delta X = X - X^{[k]}$ and $\Delta U = U - U^{[k]}$. The constraints in (7b) can be used to eliminate $\Delta X$ from the QP using

$$\Delta X = d + C\Delta \bar{x}_0 + E\Delta U \quad \text{with} \quad \Delta \bar{x}_0 = \bar{x}_0 - x_0^{[k]}, \tag{9}$$

and

$$d = \begin{bmatrix} \Delta\phi_0 \\ \Delta\phi_1 + A_1\Delta\phi_0 \\ \Delta\phi_2 + A_2\Delta\phi_1 + A_2A_1\Delta\phi_0 \\ \vdots \end{bmatrix}, \quad C = \begin{bmatrix} A_0 \\ A_1A_0 \\ A_2A_1A_0 \\ \vdots \end{bmatrix} \text{ and}$$
$$E = \begin{bmatrix} B_0 & & & \\ A_1B_0 & B_1 & & \\ A_2A_1B_0 & A_2B_1 & B_2 & \\ \vdots & & & \ddots \end{bmatrix}, \tag{10}$$

where $\Delta\phi_i = \phi_i(x_i^{[k]}, u_i^{[k]}) - x_{i+1}^{[k]}$. Note that a compact notation $A_i, B_i$ has been used here for respectively the matrices $A_i^{[k]}, B_i^{[k]}$ at the current iteration. Insertion of the expression $\Delta X = d + C\Delta\bar{x}_0 + E\Delta U$ into (7c) and (8) yields an equivalent, but smaller scale QP of the following form:

$$\underset{\Delta U}{\text{minimize}} \quad \frac{1}{2}\Delta U^\top H_\text{c}\, \Delta U + \Delta U^\top g_\text{c} \tag{11a}$$

$$\text{subject to} \quad w + K\Delta U \leq 0. \tag{11b}$$

Let us omit the lengthy explicit expressions for the matrices $H_\text{c}, K$ and the vectors $g_\text{c}$ and $w$. For a simplified setting of a quadratic objective and simple bound constraints, these expressions can be found in [5]. Although the QP subproblem can now be solved in the reduced variable space $\Delta U \in \mathbb{R}^{Nn_u}$, the variables in $X$ are still updated using an expansion step based on Eq. (9). The fact that the iterations are still performed in the full variable space, is the crucial difference with using a single shooting formulation. The bottleneck in an implementation of condensing is the computation of the condensed Hessian $H_c$, which has been shown to be of complexity $O(N^2)$ [6, 30]. It is hereby important to exploit the lower block triangular structure of matrix $E$ from (10), the separability of the objective function in (8) and the symmetry of the Hessian matrix [5, 31]. Note that the small scale QP can be solved by an efficient, dense linear algebra solver such as qpOASES. This significantly reduces the feedback delay time between receiving the new state estimate $\bar{x}_0$ and applying the next control input $u_0^{[k+1]} = u_0^{[k]} + \Delta u_0^\star$.

### 4.2.2 Solving the Sparse Problem

Using a condensing approach, the corresponding cost per iteration is of order $O(N^2)$ including the factorization cost as discussed in [30, 32]. Alternatively, one would directly solve the sparse QP problem from (7) in the full variable space

and exploiting the sparsity structure then becomes essential. Both for active set and Interior Point (IP) algorithms to solve this multi-stage QP, the cost per iteration of the solver can be made of order $O(N)$ [33]. Code generation tools exist that export tailored convex IP solvers, popular examples are CVXGEN [34] and FORCES [9]. An efficient implementation of a structure exploiting, primal-barrier IP method can be found in [35]. Employing the condensing technique is known to perform very well for relatively short horizon lengths $N$ but can be outperformed by using a structure exploiting convex solver for longer horizons. Comparative simulation results can be found in [5]. Classical condensing is generally a good approach in case of many state variables $n_x > n_u$, while complementary condensing [36] was proposed as a competitive alternative in case of many controls $n_u > n_x$. A known issue with IP methods in a real-time framework is that it is difficult to warm-start them efficiently. There is ongoing research on combining the beneficial properties of both an active set method and a structure exploiting IP solver. A promising example based on a dual Newton strategy to solve structured multi-stage QPs is the open-source software qpDUNES, presented in [10].

## *4.3 Real-Time Iterations*

The RTI scheme has already been mentioned multiple times in this paper, but it is important to elaborate on some of its properties since it is the key idea that allows nonlinear optimal control with microsecond execution times.

### 4.3.1 Initial Value Embedding

In NMPC, a sequence of optimal control problems with different initial values $\bar{x}_0^{[0]}, \bar{x}_0^{[1]}, \ldots$ needs to be solved. For the transition from one problem to the next, it is beneficial to take into account the fact that the optimal solution $U^*(\bar{x}_0)$ depends almost everywhere differentiably on $\bar{x}_0$ which is the idea behind a continuation method. The solution manifold has smooth parts whenever the active set does not change, but non-differentiable points occur where the active set changes. After linearizing at such a point in the context of a nonlinear IP method, a simple *tangential predictor* would lead to a rather bad approximation. One remedy would be to increase the path parameter $\tau$, which decreases the nonlinearity but it comes at the expense of generally less accurate solutions.

One can deal with active set changes naturally in an SQP type framework by the following procedure proposed and analysed in [4, 29, 37]: first of all, the parameter $\bar{x}_0$ needs to enter the NLP linearly, which is automatically the case for a simultaneous OCP formulation such as in Eq. (2b). The problem needs to be addressed using an exact Hessian SQP method. Finally, the solution trajectories $X^{[k]}$ and $U^{[k]}$ for the current problem in $\bar{x}_0^{[k]}$ are used as initial guess to solve the

OCP for the new parameter value $\bar{x}_0^{[k+1]}$. In the context of NMPC with a quadratic cost, this continuation technique can also be used with a Gauss-Newton Hessian approximation. This is done in the RTI algorithm and yields a multiplier free, *generalized* tangential predictor i.e. one that works across active set changes [2].

### 4.3.2 Reducing the Computational Delay

Ideally, the solution to a new optimal control problem is obtained instantly which is however impossible due to computational delays. Several ingredients of the RTI scheme can help us in dealing with this issue. A first and important one is to divide the computations at each sampling time into a preparation and a feedback phase [4]. The typically more CPU intensive *preparation phase* is performed with a predicted state, before the state estimate is even available. Once the new value $\bar{x}_0$ becomes available, the *feedback phase* quickly delivers an *approximate* solution to the original problem by solving the prepared, convex subproblem. The idea is to always work with the most current information in each iteration, i.e. not to iterate until convergence for an MPC problem that is only getting older. It can also be seen as a distributed-in-time optimization procedure.

Another important ingredient is to transfer solution information from one OCP to the next one, i.e. to efficiently warm-start each solution procedure. This can be done by using the previously optimal trajectories as an initial guess at the next time step and this either directly or in a shifted version. A last technique concerns code generation which has become a quite popular way to do code optimizations based on a high-level description of the problem to be solved. Multiple tools already exist to automatically generate custom solvers in a low-level language [34, 38]. Also for NMPC, the consecutive optimal control problems are similar and many computations can be done offline before the controller starts. The auto generated code then exploits problem dimensions and sparsity structures, it avoids dynamic memory allocation and has a nearly deterministic runtime. The latter is important to be able to satisfy the hard timing constraints in real-time applications. A tailored RTI algorithm for nonlinear optimal control can be generated as plain C-code by the open-source software ACADO Toolkit [18].

## 5 A Classical Benchmark Problem

This section presents numerical results that allow for interesting comparisons to be made between different NMPC formulations, algorithms and their implementation. The problem formulation is first presented in Sect. 5.1, followed by a description in Sect. 5.2 of the three test cases that are used in simulation. Some results of the corresponding numerical experiments are eventually shown and discussed in Sect. 5.3. The simulations presented in this section are performed using the ACADO

code generation tool on a modern computer equipped with Intel i7-3720QM processor, running a 64-bit version of Ubuntu 12.04. All programs are compiled using the Clang 3.0 compiler.

## 5.1 Problem Formulation

Throughout this section, the simple NMPC problem from [39] will be used as a benchmark example. Its corresponding continuous time OCP reads

$$\min_{x(\cdot),u(\cdot)} \quad \int_t^{t+T} (\|x(\tau)\|_Q^2 + \|u(\tau)\|_R^2)\,\mathrm{d}\tau \;+\; \|x(t+T)\|_P^2 \tag{12a}$$

$$\text{s.t.} \quad x(t) \;=\; \bar{x}_t, \tag{12b}$$

$$\begin{aligned} \dot{x}_1(\tau) &= x_2(\tau) + u(\tau)\,(\mu + (1-\mu)\,x_1(\tau))\,, \\ \dot{x}_2(\tau) &= x_1(\tau) + u(\tau)\,(\mu - 4(1-\mu)\,x_2(\tau))\,, \end{aligned} \tag{12c}$$

$$-2 \;\le\; u(\tau) \;\le\; 2, \qquad \forall \tau \in [t, t+T], \tag{12d}$$

$$\|x(t+T)\|_P^2 \;\le\; \alpha, \tag{12e}$$

where Eq. (12c) defines the simple but unstable ODE system with two differential states $x_1$ and $x_2$, a control input $u$ and constant value $\mu = 0.5$. The parameters $P \succ 0$ and $\alpha \geq 0$ from Eqs. (12a) and (12e) define the terminal penalty and the terminal region $\Omega_\alpha$ of which the latter is preferably as large as possible, while still leading to closed-loop stability for the resulting NMPC approach. The following parameter values will be used in simulation:

$$T_s = 0.1s \qquad N = 15 \qquad T = 1.5s$$

$$Q = \begin{pmatrix} 2.0 & 0.0 \\ 0.0 & 2.0 \end{pmatrix} \qquad R = 0.1$$

$$P = \begin{pmatrix} 10.605 & -9.395 \\ -9.395 & 10.605 \end{pmatrix} \qquad \alpha = 0.7$$

w here $T_s$ and $N$ respectively define the size and number of shooting intervals over the horizon $[0, T]$ i.e. they define the shooting discretization of the OCP.

## 5.2 Simulation Test Cases

### 5.2.1 Case A: Original Formulation and QP

Starting from the formulation in (12), a first NMPC scheme is to solve this OCP until convergence at every time point and this will be further called *case A*. Note that the resulting NMPC controller is the same as case A in the paper from [39].

### 5.2.2 Case B: Tuned Formulation and QP

It is interesting to have a look at the tuning possibilities to achieve faster sampling times when necessary. First of all, the RTI scheme can be used instead of iterating the procedure until convergence. Then it is important to use a suitable integration method with efficient sensitivity generation for the shooting discretization, as discussed in Sect. 3. The ODE system in (12c) is rather simple and non-stiff, thus an explicit Euler discretization with a step size of 0.1s already suffices in this case. To further improve the computation time of one RTI iteration, the number of optimization variables can be reduced. This means that the number of shooting intervals $N$ will be reduced while keeping the horizon length long enough for a good NMPC performance. To achieve this, a non equidistant control grid is used as depicted in Fig. 1. Eventually, the quadratic terminal constraint in (12e) is also removed and the resulting scheme will be referred to as *case B*.

To obtain results that are comparable to the original scheme (case A), a few details must be addressed. One is the terminal cost matrix $P$ that needs to be altered since the new horizon is shorter. By integrating the differential Riccati equation backwards over 0.5s, a new terminal cost matrix $P(1.0)$ can be found:

$$P(1.5) = \begin{pmatrix} 10.605 & -9.395 \\ -9.395 & 10.605 \end{pmatrix} \quad \Rightarrow \quad P(1.0) = \begin{pmatrix} 4.432 & -3.558 \\ -3.558 & 4.432 \end{pmatrix}.$$

Because of the varying interval size in the alternative control horizon, the weighting matrices $Q$ and $R$ are scaled using a factor relative to this interval size. Also standard shifting approaches are not applicable anymore and therefore abandoned.

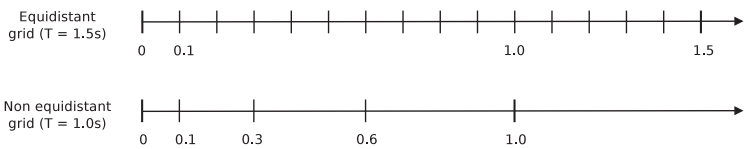


**Fig. 1** Illustration of a non equidistant control grid of only four shooting intervals over ones

### 5.2.3 Case C: Original Formulation and QCQP

It is important to note that all presented techniques can be extended further towards Sequential Convex Programming (SCP) [29]. In Sect. 4.2.2, FORCES has been presented as a sparsity exploiting QP solver although the most general convex problem that it targets is a quadratically constrained QP (QCQP). For our benchmark example, the terminal inequality constraint from (12e) can be kept in the convex subproblem such that it becomes of this QCQP form. This will be referred to as *case C* and it uses the same, original OCP formulation as case A.

## *5.3 Numerical Results*

### 5.3.1 Comparison of Single and Multiple Shooting

First of all, let us illustrate multiple shooting by comparing it with a single shooting discretization both on the OCP formulation of case A using initial value embedding. The performance of the NMPC controller will be measured using the Karush-Kuhn-Tucker (KKT) tolerance, computed as in [31]. Figure 2 illustrates the resulting convergence rate for both solvers in a simulation over a time period of 5s. It can be seen that the convergence is slightly better using the multiple shooting discretization. In both cases, the solver starts from the exact same initial guess which is the reference trajectory. Note that also the computational complexity is the same for both.

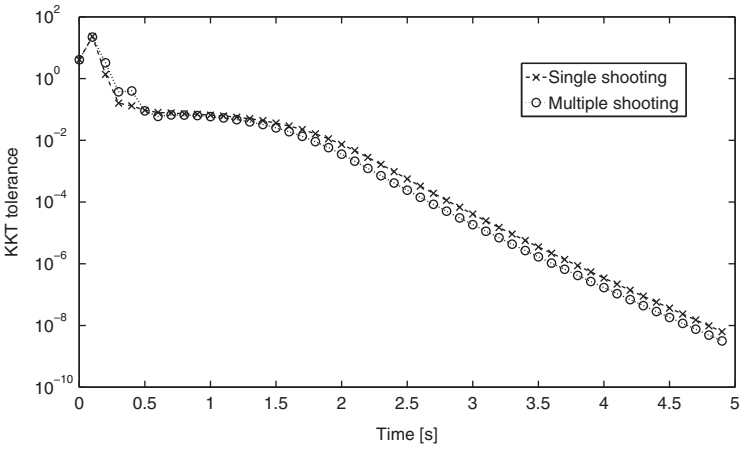


**Fig. 2** Closed-loop NMPC performance using both single and multiple shooting

**Table 1** Average computation times for NMPC using the RTI scheme

| NMPC | Case A (μs) | Case B (μs) | Case C (μs) |
|---|---|---|---|
| Integration method | 0.222 | 0.142 | 0.221 |
| Condensing | 3.370 | 0.088 | – |
| QP solution | 4.340 | 0.633 | 29.300 |
| Remaining operations | 1.608 | 0.087 | 0.679 |
| **One real-time iteration** | **9.540** | **0.950** | **30.200** |

### 5.3.2 Execution Times

The average computational times for the different components in one RTI iteration are shown in Table 1 and this for the three different cases. Using ACADO code generation, one iteration for case A, B and C on average takes respectively 9.54, 0.95 and 30.2 μs. Note that the formulation used in case B has been tuned precisely to result in a total execution time that is below 1 μs. The scheme which uses FORCES to solve a QCQP subproblem (case C) appears not to be competitive with the condensing based approach (case A) for this example. The reason is that the used horizon is relatively small as discussed more detailed in [5]. An important advantage of case C is that the terminal inequality becomes part of the subproblem to be solved and it is therefore guaranteed to be satisfied for a feasible trajectory. This is not necessarily true when linearizing that same constraint. Exploiting convexity as much as possible can therefore be a rather powerful tool.

### 5.3.3 Tracking Performance

Figure 3 compares the closed-loop tracking performance of the three NMPC schemes for five different initial values, also used in [39]. As a reference, the closed-loop behavior of the corresponding LQR scheme with control saturation is shown in the same figure. The latter controller appears to be unstable for one of these initial values while the NMPC schemes all exhibit a performance that is similar to one another. According to the RTI scheme, only one SQP iteration is performed per time step for cases B and C while the NMPC results for case A are iterated until convergence using a rather strict stopping criterion. Note that the feedback delay has not been taken into account in these closed-loop simulations.

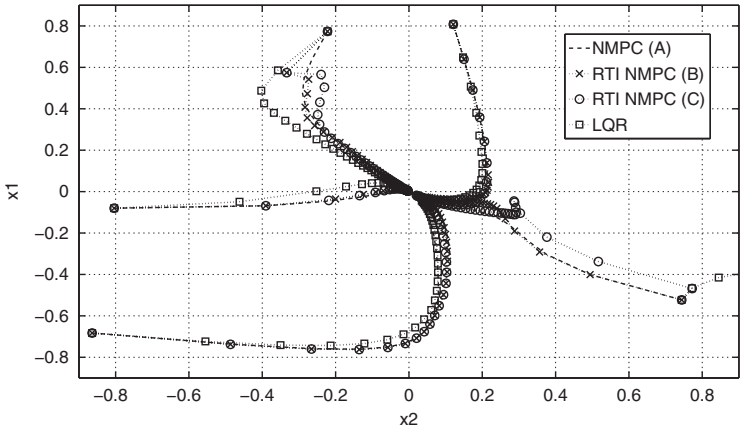


**Fig. 3** Closed-loop performance of three different NMPC schemes and the LQR controller and this for five different initial values. This figure is comparable to one presented originally in [39]

## 6 Conclusions

This paper handled a general parametric optimization problem, which arises naturally in NMPC where one has to solve a nonlinear OCP at every sampling instant. It gave an outline of the different algorithmic developments that made these techniques real-time feasible today even for nonlinear systems with fast dynamics. Auto generated integration methods have been presented as an essential part to efficiently linearize the problem and compute derivative information. The RTI scheme has been described as an online algorithm that allows to perform real-time NMPC while having a fast control feedback to the real process. A simple example taken from the literature, was used to illustrate the performance of the presented tools. The average execution time per time step for this nonlinear problem was eventually shown to be below 1 $\mu$s.

**Acknowledgements** The authors would like to thank all reviewers for their valuable comments, which helped to improve this article. This research was supported by Research Council KUL: PFV/10/002 Optimization in Engineering Center OPTEC, GOA/10/09 MaNet and GOA/10/11 Global real-time optimal control of autonomous robots and mechatronic systems, KUL-BOF OT/10/035. Flemish Government: FWO: PhD/postdoc grants, FWO KAN2013 1.5.189.13, FWO-G.0930.13; IWT: PhD Grants, projects: Eurostars SMART; Belgian Federal Science Policy Office: IUAP P7 (DYSCO, Dynamical systems, control and optimization, 2012–2017); EU: FP7-SADCO (MC ITN-264735), FP7-TEMPO (MC ITN-607957), H2020-AWESCO (MC ITN-642682), ERC HIGHWIND (259 166). R. Quirynen holds a PhD fellowship of the Research Foundation– Flanders (FWO).

## References


1. Kirches, C., Wirsching, L., Sager, S., Bock, H.: Efficient numerics for nonlinear model predictive control. In: Diehl, M., Glineur, F.F., Michiels, E.J.W. (eds.) Recent Advances in Optimization and its Applications in Engineering, pp. 339–357. Springer, Berlin (2010)
2. Diehl, M., Ferreau, H.J., Haverbeke, N.: Efficient numerical methods for nonlinear MPC and moving horizon estimation. In: Nonlinear Model Predictive Control. Lecture Notes in Control and Information Sciences, vol. 384. pp. 391–417. Springer, Berlin (2009)
3. Diehl, M., Bock, H., Schlöder, J.: A real-time iteration scheme for nonlinear optimization in optimal feedback control. SIAM J. Control. Optim. **43**(5), 1714–1736 (2005)
4. Diehl, M., Bock, H., Schlöder, J., Findeisen, R. Nagy, Z., Allgöwer, F.: Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. J. Process Control **12**(4), 577–585 (2002)
5. Vukov, M., Domahidi, A., Ferreau, H.J., Morari, M., Diehl, M.: Auto-generated algorithms for nonlinear model predicitive control on long and on short horizons. In: Proceedings of the 52nd Conference on Decision and Control (CDC) (2013)
6. Andersson, J.: A general-purpose software framework for dynamic optimization. Ph.D. thesis, Arenberg Doctoral School, KU Leuven, Department of Electrical Engineering (ESAT/SCD) and Optimization in Engineering Center, Kasteelpark Arenberg 10, 3001-Heverlee, Belgium (October 2013)
7. Ferreau, H., Kirches, C., Potschka, A., Bock, H., Diehl, M.: qpOASES: A parametric active-set algorithm for quadratic programming. Math. Program. Comput. **6**(4), 327–363 (2014)
8. Ferreau, H.J., Bock, H.G., Diehl, M.: An online active set strategy to overcome the limitations of explicit MPC. Int. J. Robust Nonlinear Control **18**(8), 816–830 (2008)
9. Domahidi, A., Zgraggen, A., Zeilinger, M., Morari, M., Jones, C.: Efficient Interior point methods for multistage problems arising in receding horizon control. In: IEEE Conference on Decision and Control (CDC) (Maui, HI, USA), pp. 668–674 (December 2012)
10. Frasch, J.V., Sager, S., Diehl, M.: A parallel quadratic programming method for dynamic optimization problems. Math. Program. Comput. **7**(3) 289–329 (September 2015)
11. Frison, G., Sorensen, H., Dammann, B., Jorgensen, J.: High-performance small-scale solvers for linear model predictive control. In: Proceedings of 2014 European Control Conference (ECC), pp. 128–133 (June 2014)
12. Mattingley, J., Boyd, S.: Automatic code generation for real-time convex optimization. In: Convex Optimization in Signal Processing and Communications. Cambridge University Press, Cambridge (2009)
13. Quirynen, R., Vukov, M., Zanon, M., Diehl, M.: Autogenerating microsecond solvers for nonlinear MPC: a tutorial using ACADO integrators. Optim. Control Appl. Methods (2014). http://onlinelibrary.wiley.com/doi/10.1002/oca.2152/abstract
14. Vukov, M., Loock, W.V., Houska, B., Ferreau, H., Swevers, J., Diehl, M.: Experimental Validation of Nonlinear MPC on an Overhead Crane using Automatic Code Generation. In: The 2012 American Control Conference, Montreal, Canada (2012)
15. Quirynen, R., Vukov, M., Diehl, M.: Auto generation of implicit integrators for embedded NMPC with microsecond sampling times. In: Lazar, M., Allgöwer, F. (eds.) Proceedings of the 4th IFAC Nonlinear Model Predictive Control Conference (2012)
16. Quirynen, R., Gros, S., Diehl, M.: Efficient NMPC for nonlinear models with linear subsystems. In: Proceedings of the 52nd IEEE Conference on Decision and Control (2013)
17. Quirynen, R., Gros, S., Diehl, M.: Fast auto generated ACADO integrators and application to MHE with multi-rate measurements. In: Proceedings of the European Control Conference (2013)
18. Houska, B., Ferreau, H., Diehl, M.: An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. Automatica **47**(10), 2279–2285 (2011)
19. Houska, B., Ferreau, H., Diehl, M.: ACADO toolkit – an open source framework for automatic control and dynamic optimization. Optim. Control Appl. Methods **32**(3), 298–312 (2011)

20. Biegler, L.: An overview of simultaneous strategies for dynamic optimization. Chem. Eng. Process. **46**, 1043–1053 (2007)
21. Bock, H., Plitt, K.: A multiple shooting algorithm for direct solution of optimal control problems. In: Proceedings 9th IFAC World Congress Budapest, pp. 242–247. Pergamon Press, New York (1984)
22. Albersmeyer, J., Diehl, M.: The lifted Newton method and its application in optimization. SIAM J. Optim. **20**(3), 1655–1684 (2010)
23. Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems, 2nd edn. Springer, Berlin/Heidelberg (1991)
24. Quirynen, R.: Automatic code generation of Implicit Runge-Kutta integrators with continuous output for fast embedded optimization. Master's thesis, KU Leuven (2012)
25. Serban, R., Hindmarsh, A.: CVODES: The sensitivity-enabled ODE solver in SUNDIALS. In: Proceedings of IDETC/CIE (2005)
26. Wächter, A., Biegler, L.: On the Implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006)
27. Nocedal, J., Wright, S.: Numerical Optimization. In: Springer Series in Operations Research and Financial Engineering, 2nd edn. Springer, Berlin (2006)
28. Bock, H. Recent advances in parameter identification techniques for ODE. In: Deuflhard, P., Hairer, E. (eds.) Numerical Treatment of Inverse Problems in Differential and Integral Equations. Birkhäuser, Boston (1983)
29. Tran-Dinh, Q., Savorgnan, C., Diehl, M.: Adjoint-based predictor-corrector sequential convex programming for parametric nonlinear optimization. SIAM J. Optim. **22**(4), 1258–1284 (2012)
30. Frison, G., Jorgensen, J.: A fast condensing method for solution of linear-quadratic control problems. In: Proceedings of the 52nd IEEE Conference on Decision and Control (2013)
31. Leineweber, D. Efficient reduced SQP methods for the optimization of chemical processes described by large sparse DAE models. Fortschritt-Berichte VDI Reihe 3, Verfahrenstechnik, vol. 613. VDI Verlag, Düsseldorf (1999)
32. Axehill, D., Morari, M.: An alternative use of the riccati recursion for efficient optimization. Syst. Control Lett. **61**(1), 37–40 (2012)
33. Rao, C., Wright, S., Rawlings, J.: Application of interior-point methods to model predictive control. J. Optim. Theory Appl. **99**, 723–757 (1998)
34. Mattingley, J., Wang, Y., Boyd, S.: Code generation for receding horizon control. In: Proceedings of the IEEE International Symposium on Computer-Aided Control System Design (Yokohama, Japan) (2010)
35. Wang, Y., Boyd, S.: Fast model predictive control using online optimization. IEEE Trans. Control Syst. Technol. **18**(2), 267–278 (2010)
36. Kirches, C., Bock, H., Schlöder, J., Sager, S.: Block structured quadratic programming for the direct multiple shooting method for optimal control. Optim. Methods Softw. **26**, 239–257 (2010)
37. Zavala, V., Anitescu, M.: Real-time nonlinear optimization as a generalized equation. SIAM J. Control Optim. **48**(8), 5444–5467 (2010)
38. Ohtsuka, T., Kodama, A.: Automatic code generation system for nonlinear receding horizon control. Trans. Soc. Instrum. Control Eng. **38**(7), 617–623 (2002)
39. Chen, H., Allgöwer, F.: A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. Automatica **34**(10), 1205–1218 (1998)

Springer

http://www.springer.com/978-3-319-23320-8

Multiple Shooting and Time Domain Decomposition Methods

MuS-TDD, Heidelberg, May 6-8, 2013

Carraro, Th.; Geiger, M.; Körkel, S.; Rannacher, R. (Eds.)

2015, X, 422 p., Hardcover

ISBN: 978-3-319-23320-8