Inexact Newton based Lifted Implicit Integrators for fast Nonlinear MPC

Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 48-23 (2015) 032–038

Inexact Newton based Lifted Implicit Integrators for fast Nonlinear MPC ⋆

Rien Quirynen [*,***] **Sébastien Gros** [**] **Moritz Diehl** [*,***]

[*] *Department ESAT, KU Leuven University, Kasteelpark Arenberg 10, 3001 Leuven, Belgium (e-mail: rien.quirynen@esat.kuleuven.be).*
[**] *Signals and Systems, Chalmers University of Technology, Sweden.*
[***] *Department IMTEK, University of Freiburg, Germany.*

Abstract: Nonlinear Model Predictive Control (NMPC) requires the online solution of an Optimal Control Problem (OCP) at every sampling instant. In the context of multiple shooting, a numerical integration is needed to discretize the continuous time dynamics. For stiff, implicitly defined or differential-algebraic systems, implicit schemes are preferred to carry out the integration. The Newton-type optimization method and the implicit integrator then form a nested Newton scheme, solving the optimization and integration problem on two different levels. In recent research, an exact lifting technique was proposed to improve the computational efficiency of the latter framework. Inspired by that work, this paper presents a novel class of lifted implicit integrators, using an inexact Newton method. An additional iterative scheme for computing the sensitivities is proposed, which provides similar properties as the exact lifted integrator at considerably reduced computational costs. Using the example of an industrial robot, computational speedups of up to factor 8 are reported. The proposed methods have been implemented in the open-source ACADO code generation software.

Keywords: Numerical algorithms, Optimal control, Nonlinear Predictive control

1. INTRODUCTION

Nonlinear Model Predictive Control (NMPC) is a popular control approach, thanks to its ability to directly treat nonlinear dynamics and process constraints. At each sampling time t_0, the following continuous time Optimal Control Problem (OCP) needs to be solved:

$$\min_{x(\cdot),\,u(\cdot)} \int_{t_0}^{t_0+T} \|F(x(t),u(t))\|_2^2\,\mathrm{d}t + \|F_N(x(t_0+T))\|_2^2 \quad (1a)$$

$$\text{s.t.} \quad 0 = x(t_0) - \hat{x}_0, \quad (1b)$$

$$0 = f(\dot{x}(t), x(t), z(t), u(t)), \ \forall t \in [t_0, t_0+T], \quad (1c)$$

$$0 \geq h(x(t), u(t)), \ \forall t \in [t_0, t_0+T], \quad (1d)$$

$$0 \geq r(x(t_0+T)), \quad (1e)$$

where T is the horizon length, $x(t) \in \mathbb{R}^{n_x}$ denotes the differential states, $z(t) \in \mathbb{R}^{n_z}$ the algebraic variables and $u(t) \in \mathbb{R}^{n_u}$ are the control inputs. This nonlinear OCP depends on the current state estimate $\hat{x}_0 \in \mathbb{R}^{n_x}$ only through the initial value condition (1b). The objective function in Eq. (1a) is of the least squares type, while path and terminal constraints are defined by Eqs. (1d) and (1e). The nonlinear dynamics in Eq. (1c) are formulated as an implicit system of Differential Algebraic Equations (DAE), for which the Jacobian matrix $\frac{\partial f(\cdot)}{\partial(z,\dot{x})}$ is assumed invertible.

Algorithmic progress (Diehl et al., 2009; Kirches et al., 2010) allows for real-time implementations of NMPC, even for systems having fast dynamics. In this paper, the Real-Time Iteration (RTI) scheme is used as an efficient online algorithm (Diehl et al., 2002). The discretization of the continuous time OCP is based on direct multiple shooting (Bock and Plitt, 1984) and Sequential Quadratic Programming (SQP) is used to solve the resulting Nonlinear Program (NLP). Efficient integrators with tailored sensitivity propagation are required to discretize and linearize the nonlinear dynamics, as discussed in (Quirynen et al., 2014). When the considered system has either stiff or implicitly defined dynamics, the use of an implicit integration scheme is recommended. Collocation methods form an important class of Implicit Runge-Kutta (IRK) methods, which offer good stability properties (Hairer and Wanner, 1991) and can be deployed efficiently within a multiple shooting method (Quirynen et al., 2014).

A novel exact lifting technique for implicit integrators in the context of multiple shooting, has been recently proposed in (Quirynen et al., 2015). The technique is inspired from the Lifted Newton method (Albersmeyer and Diehl, 2010) and improves the computational efficiency and convergence properties of deploying the implicit integration scheme within an SQP method. It only requires a minor additional implementation effort. This paper continues that work by extending it to inexact Newton methods, constructing new lifted schemes which can significantly outperform the state-of-the-art algorithms.

The paper is organized as follows. Section 2 briefly introduces direct multiple shooting, SQP and collocation.

⋆ This research was supported by Research Council KUL: PFV/10/002 Optimization in Engineering Center OPTEC; Eurostars SMART; Belgian Federal Science Policy Office: IUAP P7 (DYSCO, Dynamical systems, control and optimization, 2012-2017); EU: FP7-TEMPO (MCITN-607957), ERC HIGHWIND (259166), H2020-ITN AWESCO (642682). R. Quirynen holds a PhD fellowship of the Research Foundation – Flanders (FWO).

2405-8963 © 2015, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2015.11.259

Section 3 then summarizes the lifting scheme for implicit integrators. Subsequently, Section 4 proposes the novel inexact lifting technique and multiple efficient variants. The open-source software implementation is briefly discussed in Section 5. Based on a real-world control example, the numerical case study in Section 6 illustrates the performance of the presented schemes.

Contribution: we propose a novel inexact lifting scheme for implicit integrators in a multiple shooting method for real-time optimal control, which can significantly reduce its computational cost. Also, we provide an open-source implementation of these techniques in the ACADO code generation software package.

2. DIRECT OPTIMAL CONTROL METHODS

In direct optimal control (Bock and Plitt, 1984), one tackles the continuous time OCP (1) by forming a discrete approximation and solving the resulting NLP. For the sake of simplicity, we consider here an equidistant grid over the control horizon consisting in the collection of time points t_i, where $t_{i+1} - t_i = \frac{T}{N} := T_s$ for $i = 0, \dots, N-1$. Additionally, we consider a piecewise constant control parametrization $u(\tau) = u_i$ for $\tau \in [t_i, t_{i+1})$.

2.1 Multiple Shooting Discretization

A direct *multiple shooting* discretization (Bock and Plitt, 1984) of the OCP in (1) results in following NLP:

$$\min_{X,U} \quad \frac{1}{2} \sum_{i=0}^{N-1} \|F(x_i, u_i)\|_2^2 + \frac{1}{2} \|F_N(x_N)\|_2^2 \tag{2a}$$

$$\text{s.t.} \quad 0 = x_0 - \hat{x}_0, \tag{2b}$$

$$0 = \phi(x_i, u_i) - x_{i+1}, \quad i = 0, \dots, N-1, \tag{2c}$$

$$0 \geq h(x_i, u_i), \quad i = 0, \dots, N-1, \tag{2d}$$

$$0 \geq r(x_N), \tag{2e}$$

with state trajectory $X = [x_0^\top, \dots, x_N^\top]^\top$ and control trajectory $U = [u_0^\top, \dots, u_{N-1}^\top]^\top$. Function $\phi(\cdot)$ represents a numerical approximation $\phi(x_i, u_i) \approx x(t_{i+1})$ for the solution of the following initial value problem:

$$0 = f(\dot{x}(\tau), x(\tau), z(\tau), u_i), \quad \tau \in [t_i, t_{i+1}], \tag{3}$$

where $x(t_i) = x_i$. This function typically needs to be evaluated numerically using an implicit integration method for DAE systems (Hairer and Wanner, 1991).

2.2 Sequential Quadratic Programming

This paper considers the use of an SQP method to solve the NLP (2). Since the OCP formulation consists of a least squares type objective, a popular approach is to use the Generalized Gauss-Newton (GGN) method (Bock and Plitt, 1984). It iterates by sequentially solving the following Quadratic Program (QP):

$$\min_{\Delta W} \quad \frac{1}{2} \sum_{i=0}^{N-1} \|F(\bar{w}_i) + J_i \, \Delta w_i\|_2^2 + \frac{1}{2} \|F_N(\bar{x}_N) + J_N \, \Delta x_N\|_2^2 \tag{4a}$$

$$\text{s.t.} \quad 0 = \bar{x}_0 - \hat{x}_0 + \Delta x_0, \tag{4b}$$

$$0 = \phi(\bar{w}_i) - \bar{x}_{i+1} + C_i \Delta w_i - \Delta x_{i+1}, \tag{4c}$$

$$0 \geq h(\bar{w}_i) + D_i \Delta w_i, \quad i = 0, \dots, N-1, \tag{4d}$$

$$0 \geq r(\bar{x}_N) + D_N \Delta x_N, \tag{4e}$$

where $\Delta W := (\Delta w_0, \dots, \Delta w_N)$, $w_i := (x_i, u_i)$, $\Delta w_i := w_i - \bar{w}_i$ for $i = 0, \dots, N-1$ and $\Delta w_N := \Delta x_N$. The Jacobian matrices are defined as $C_i = \frac{d\phi(\bar{w}_i)}{dw_i}$, $D_i = \frac{\partial h(\bar{w}_i)}{\partial w_i}$ and $J_i = \frac{\partial F(\bar{w}_i)}{\partial w_i}$. The notation $\bar{w}_i := (\bar{x}_i, \bar{u}_i)$ is used to denote the current optimization values, which are updated in each SQP iteration by solving the QP subproblem (4), i.e. $\bar{W}^+ = \bar{W} + \Delta W$.

2.3 Standard Collocation Integrator

Let us deploy a collocation method (Biegler, 2010) for the evaluation of function $\phi(x_i, u_i)$ in Eq. (2c), using a fixed number of steps N_s. In that context, one needs to solve the following system of collocation equations for K_i:

$$G_i(w_i, K_i) = \begin{bmatrix} g_i^1(w_i, K_i^1) \\ \vdots \\ g_i^{N_s}(w_i, K_i^1, \dots, K_i^{N_s}) \end{bmatrix} = 0, \tag{5}$$

$$\text{where} \quad g_i^j(\cdot) = \begin{bmatrix} f(k_{i,1}^j, x_i^{j-1} + T_s \sum_{r=1}^{S} a_{1r} k_{i,r}^j, Z_{i,1}^j, u_i) \\ \vdots \\ f(k_{i,S}^j, x_i^{j-1} + T_s \sum_{r=1}^{S} a_{Sr} k_{i,r}^j, Z_{i,S}^j, u_i) \end{bmatrix},$$

where S denotes the number of nodes and the matrix $[A]_{ij} := a_{ij}$ contains the internal coefficients of the IRK method (Hairer and Wanner, 1991). The collocations variables $k_{i,r}^j \in \mathbb{R}^{n_x}$ and $Z_{i,r}^j \in \mathbb{R}^{n_z}$ for a given shooting interval i and integration step j, are collected in the variable $K_i := (K_i^1, \dots, K_i^{N_s})$ with $K_i^j := (k_{i,1}^j, Z_{i,1}^j, \dots, k_{i,S}^j, Z_{i,S}^j)$ for $i = 0, \dots, N-1$ and $j = 1, \dots, N_s$. In this formulation, the k-variables denote values of the time derivative of the differential states while the Z-variables denote values of the algebraic states (Biegler, 2010). The intermediate state values x_i^j are defined by the IRK weights b_r:

$$x_i^j = x_i^{j-1} + T_s \sum_{r=1}^{S} b_r k_{i,r}^j, \quad j = 1, \dots, N_s, \tag{6}$$

where $x_i^0 := x_i$ such that $\phi(x_i, u_i) := x_i^{N_s} = x_i + T_s \, B \, K_i$ in which B is a linear operator.

In a standard real-time implementation, one performs a fixed amount L of Newton-type iterations for the nonlinear system (5) to evaluate $K_i^{[L]}(\bar{w}_i)$:

$$K_i^{[j]} = K_i^{[j-1]} - M_i^{-1} \, G_i(\bar{w}_i, K_i^{[j-1]}), \tag{7}$$

where $K_i^{[j]}$ denotes the values of the collocation variables at iteration j (Quirynen et al., 2014). In this context, the Jacobian matrix $M_i := \frac{\partial G_i(\bar{w}_i, K_i^{[0]})}{\partial K}$ is evaluated only once, such that its LU factorization can be reused throughout the iterations (Hairer and Wanner, 1991). The latter method converges linearly to the solution K_i^* of the nonlinear system (Dennis, 1968).

Remark 1. Because of the particular structure of Eq. (5) for N_s integration steps, the variables K_i^j can be computed sequentially for $j = 1, \dots, N_s$, to reduce the computational burden of the scheme. Each integration step then provides a good initialization for the collocation variables in the

next step. For the sake of brevity, we omit this detail and consider all collocation equations and variables together.

When deploying an SQP method to solve the OCP problem, linearizations of the continuity condition (4c) with respect to the decision variables w_i are needed. The sensitivities of the collocation variables $K_i^{[L]}$ are therefore needed, which can be obtained by using the implicit function theorem for $0 = G_i(\bar{w}_i, K_i^{[L]})$:

$$\frac{dK_i^{[L]}}{dw_i} = -\frac{\partial G_i(\bar{w}_i, K_i^{[L]})}{\partial K}^{-1} \frac{\partial G_i(\bar{w}_i, K_i^{[L]})}{\partial w}. \quad (8)$$

The latter requires an evaluation of the Jacobian $\frac{\partial G_i(\bar{w}_i, K_i^{[L]})}{\partial K}$ and of its LU factorization, which can also be reused throughout the integration steps as discussed in (Quirynen et al., 2014). The resulting implementation of a standard *collocation integrator* is detailed in Algorithm 1.

Algorithm 1: Standard collocation integrator.

Input: The vector $\bar{w}_i = (\bar{x}_i, \bar{u}_i)$ and collocation initialization $K_i^{[0]}$.
Algorithm:

(1) Perform L Newton iterations:
Compute new values $K_i^{[L]}$ from Eq. (7).

(2) Compute the derivatives:
$\frac{dK_i^{[L]}}{dw_i} = -\frac{\partial G_i(\bar{w}_i, K_i^{[L]})}{\partial K}^{-1} \frac{\partial G_i(\bar{w}_i, K_i^{[L]})}{\partial w}$.

(3) Evaluate integrator results:
$\phi_i \leftarrow \bar{x}_i + T_s\, B\, K_i^{[L]}$.
$\frac{d\phi_i}{dw_i} \leftarrow \left[\mathbb{1}\ \mathbb{0}\right] + T_s\, B\, \frac{dK_i^{[L]}}{dw_i}$.

Output: The new values $K_i^{[L]}$ and the results $\left(\phi_i, \frac{d\phi_i}{dw_i}\right)$.

3. LIFTED IMPLICIT INTEGRATORS

This section briefly recalls the lifting technique for an implicit integrator, as presented in (Quirynen et al., 2015).

3.1 Lifted Implicit Integrator

At each SQP iteration, the standard collocation integrator from Algorithm 1 performs a fixed amount of Newton-type iterations L to compute new values for the collocation variables satisfying $0 = G_i(\bar{w}_i, K_i)$ up to a certain accuracy, given the current optimization values $\bar{w}_i$. The lifting technique for this implicit integrator can be summarized in the key concepts detailed next.

Newton step of the implicit integrator: each call to the integrator updates the currently stored values of the collocation variables $\bar{K}_i$ for $i = 0, \ldots, N-1$, by taking a single Newton iteration on the system $0 = G_i(w_i, K_i)$. In addition, the derivative information K_i^w of the form (8) is computed and stored. This reads as follows:

$$\tilde{K}_i = \bar{K}_i - \frac{\partial G_i(\bar{w}_i, \bar{K}_i)}{\partial K}^{-1} G_i(\bar{w}_i, \bar{K}_i), \quad (9a)$$

$$\tilde{K}_i^w = -\frac{\partial G_i(\bar{w}_i, \bar{K}_i)}{\partial K}^{-1} \frac{\partial G_i(\bar{w}_i, \bar{K}_i)}{\partial w}, \quad (9b)$$

where $\bar{K}_i := (\bar{K}_i^1, ..., \bar{K}_i^{N_s})$ denote the current collocation values. Note that the same factorization of the Jacobian $\frac{\partial G_i}{\partial K}$ can be used to compute both $\tilde{K}_i$ and $\tilde{K}_i^w$.

Linearized continuity condition: the lifted collocation variables K_i are *hidden* from the SQP solver by forming the linearized continuity condition $\Delta x_{i+1} = \phi_i - \bar{x}_{i+1} + \frac{d\phi_i}{dw_i}\Delta w_i$, for which ϕ_i and $\frac{d\phi_i}{dw_i}$ are defined as:

$$\begin{aligned} \phi_i &= \bar{x}_i + T_s\, B\, \tilde{K}_i, \\ \frac{d\phi_i}{dw_i} &= [\mathbb{1}\ \mathbb{0}] + T_s\, B\, \tilde{K}_i^w, \end{aligned} \quad (10)$$

where $\mathbb{1}$ and $\mathbb{0}$ denote respectively an identity matrix and a matrix of zeros. This step corresponds to a numerical elimination of the collocation variables, such that the lifted integrator only needs to output the results ϕ_i and $\frac{d\phi_i}{dw_i}$.

Solving the QP subproblem: the solver computes the new step ΔW as the solution of the QP in Eq. (4).

Update of the collocation variables: given the SQP step ΔW, the collocation variables can be updated based on the stored values $\tilde{K}_i$ and the sensitivities $\tilde{K}_i^w$:

$$\bar{K}_i^+ = \tilde{K}_i + \tilde{K}_i^w \Delta w_i, \quad (11)$$

where $\bar{K}_i^+$ refers to the values in the next SQP iteration.

The update of the collocation variables is performed after solving the QP subproblem and before the next Newton step of the lifted integrator. The implementation is detailed in Algorithm 2 (`Exact`), and starts with the update based on the previous SQP step Δw_i and the stored values $\tilde{K}_i$, $\tilde{K}_i^w$. As discussed in (Quirynen et al., 2015), it is possible to approximate the update step in order to avoid the need to store all derivative information $\tilde{K}_i^w$ for $i = 0, \ldots, N-1$. Only the exact scheme for the lifted implicit integrator, as presented above, will however be considered further in this paper.

3.2 Key properties of the Lifted Implicit Integrator

We briefly recall here the key points discussed in (Quirynen et al., 2015):

- The SQP steps obtained when deploying the exact lifted scheme from Algorithm 2 within a multiple shooting method are strictly equivalent to the steps from a direct collocation method (Quirynen et al., 2015). As a result, the convergence of the scheme detailed in Section 3.1 inherits the one of the direct collocation method, see e.g. (Biegler, 2010).
- The lifted implicit integrator in Algorithm 2 can be computationally much cheaper than the standard scheme from Algorithm 1. Computational complexity is detailed further in this paper.
- Implementing the proposed lifting technique requires a fairly small coding effort when starting from the standard implicit integrator scheme.

4. INEXACT LIFTED IMPLICIT INTEGRATORS

We present next a novel scheme based on the exact lifting technique for implicit integrators, but which avoids the evaluation and factorization of the Jacobian matrix in each Newton iteration. In the context of Newton-type methods (Dennis, 1968), one instead relies on a Jacobian approximation $M_i \approx \frac{\partial G_i(\bar{w}_i, \bar{K}_i)}{\partial K}$.

In addition to the lifted collocation variables K_i, one needs to iteratively update the current values for the derivatives K_i^w such that they converge to the exact Jacobian (9b). For this purpose, we consider a Newton-type iteration on (9b) using the Jacobian approximation M_i. The resulting Newton-type step reads as:

$$\tilde{K}_i = \bar{K}_i - M_i^{-1} G_i(\bar{w}_i, \bar{K}_i), \tag{12a}$$

$$\tilde{K}_i^w = \bar{K}_i^w - M_i^{-1} \left(\frac{\partial G_i(\bar{w}_i, \bar{K}_i)}{\partial w} + \frac{\partial G_i(\bar{w}_i, \bar{K}_i)}{\partial K} \bar{K}_i^w \right), \tag{12b}$$

where $\bar{K}_i$ denote the current lifted collocation values and $\bar{K}_i^w$ are the current derivative values.

The use of the latter iteration instead of the exact Newton step from Eq. (9), will be referred to as the *inexact lifted* implicit integrator. Note that the remaining steps detailed in Section 3.1 are still identical, as illustrated in Algorithm 2. The Newton-type iteration from Eq. (12) can be implemented rather efficiently, given the structure of the collocation equations in (5). The right-hand side in the Newton step for the sensitivities (12b) can e.g. be computed using forward Algorithmic Differentiation (AD).

If the matrix M_i is the exact Jacobian matrix, i.e. $M_i = \frac{\partial G_i(\bar{w}_i, \bar{K}_i)}{\partial K}$, the resulting scheme becomes equivalent to the exact lifted implicit integrator. In the following, we present three Newton-type schemes of the form in (12) using specific choices of matrix M_i. A detailed convergence analysis for this inexact lifting technique is however part of ongoing research.

Algorithm 2: Lifted collocation integrator.

Input: The new values $\bar{w}_i := (\bar{x}_i, \bar{u}_i)$ and the SQP step Δw_i.
Algorithm:

(1) Update collocation variables:
$\bar{K}_i \leftarrow \tilde{K}_i + \tilde{K}_i^w \Delta w_i$.

(2) Perform one Newton iteration:
(`Exact`) Compute values $\tilde{K}_i$, $\tilde{K}_i^w$ from Eq. (9).
(`Inexact`) Compute values $\tilde{K}_i$, $\tilde{K}_i^w$ from Eq. (12).

(3) Evaluate integrator results:
$\phi_i \leftarrow \bar{x}_i + T_s\, B\, \tilde{K}_i$.
$\frac{d\phi_i}{dw_i} \leftarrow \begin{bmatrix} \mathbb{1} & \mathbb{0} \end{bmatrix} + T_s\, B\, \tilde{K}_i^w$.

(4) Memory of the integrator:
Store $\tilde{K}_i$ and $\tilde{K}_i^w$.

Output: The linearization results $\left(\phi_i, \frac{d\phi_i}{dw_i}\right)$.

4.1 Scheme I: fixed Jacobian over integrator steps

The most natural choice for the Jacobian approximation M_i is to reuse the evaluation and factorization from the previous step of the integrator. Given the notation for the collocation equations from (5), one can consider the following Newton-type iteration for the lifted collocation variables in case of N_s integration steps:

$$\begin{aligned} m_i &= \frac{\partial g_i^1(\bar{w}_i, \bar{K}_i^1)}{\partial K}, \\ \tilde{K}_i^1 &= \bar{K}_i^1 - m_i^{-1} g_i^1(\bar{w}_i, \bar{K}_i^1), \\ \tilde{K}_i^j &= \bar{K}_i^j - m_i^{-1} g_i^j(\bar{w}_i, \bar{K}_i^j), \qquad \text{for } j = 2, \ldots, N_s, \end{aligned} \tag{13}$$

where the exact Jacobian and its factorization is only evaluated for the first integration step $m_i = \frac{\partial g_i^1}{\partial K}$. Reusing the Jacobian factorization over multiple steps can result in a considerable speedup when using many integration steps per shooting interval.

4.2 Scheme II: Simplified Newton

To simplify the following discussion, let us restrict to only one integration step $N_s = 1$. The next two schemes should be combined with the technique described in Sec. 4.1 when performing multiple integration steps. From (5) with $N_s = 1$ and considering the case $S = 3$, the exact Jacobian reads as:

$$\frac{\partial G_i}{\partial K} = \begin{bmatrix} H_1 + T_s\, a_{11} J_1 & T_s\, a_{12} J_1 & T_s\, a_{13} J_1 \\ T_s\, a_{21} J_2 & H_2 + T_s\, a_{22} J_2 & T_s\, a_{23} J_2 \\ T_s\, a_{31} J_3 & T_s\, a_{32} J_3 & H_3 + T_s\, a_{33} J_3 \end{bmatrix}, \tag{14}$$

where $H_j = \begin{bmatrix} \frac{df_{i,j}}{d\dot{x}} & \frac{df_{i,j}}{dz} \end{bmatrix}$ and $J_j = \begin{bmatrix} \frac{df_{i,j}}{dx} & \mathbb{0} \end{bmatrix}$. Our second lifted inexact scheme will be based on a transformation of the Simplified Newton iteration for IRK methods, which uses a specific approximation M_i of the latter Jacobian as proposed by (Bickart, 1977) and (Butcher, 1976).

Let us consider a 3-stage IRK method ($S = 3$) for which the internal coefficient matrix A is invertible and there exists a decomposition $A^{-1} = V \Lambda V^{-1}$ based on a block diagonal matrix Λ. In that case, it is typical for the matrix A^{-1} to have one real eigenvalue γ and one complex conjugate eigenvalue pair $\alpha \pm i\beta$ (Hairer and Wanner, 1991). Using again the notation for the collocation equations from (5), the exact Jacobian $\frac{\partial G_i}{\partial K}$ in (14) can be approximated by the following matrix:

$$\begin{aligned} M_i &= \mathbb{1}_3 \otimes H + T_s\, A \otimes J = \\ &\begin{bmatrix} H + T_s\, a_{11} J & T_s\, a_{12} J & T_s\, a_{13} J \\ T_s\, a_{21} J & H + T_s\, a_{22} J & T_s\, a_{23} J \\ T_s\, a_{31} J & T_s\, a_{32} J & H + T_s\, a_{33} J \end{bmatrix}, \end{aligned} \tag{15}$$

where $\otimes$ denotes the Kronecker product of matrices and we can for example choose the Jacobians $H = \begin{bmatrix} \frac{df_{i,1}}{d\dot{x}} & \frac{df_{i,1}}{dz} \end{bmatrix}$ and $J = \begin{bmatrix} \frac{df_{i,1}}{dx} & \mathbb{0} \end{bmatrix}$, evaluated at the first stage.

One can then carry out the Newton-type iteration on (5) using:

$$(\mathbb{1}_3 \otimes H + T_s\, A \otimes J) \Delta K_i = -G_i, \tag{16}$$

giving $\tilde{K}_i = \bar{K}_i + \Delta K_i$. However, premultiplying this equation on both sides by $(T_s\, A)^{-1} \otimes \mathbb{1}_{n_k}$ results in:

$$(\tilde{\Lambda} \otimes H + \mathbb{1}_3 \otimes J) \Delta \tilde{K}_i = -(\tilde{\Lambda}\, V^{-1} \otimes \mathbb{1}_{n_k}) G_i, \tag{17}$$

where $\Delta \tilde{K}_i = (V^{-1} \otimes \mathbb{1}_{n_k}) \Delta K_i$ and $\tilde{\Lambda} = \frac{1}{T_s} \Lambda$. It is then important to observe that:

$$\tilde{\Lambda} \otimes H + \mathbb{1}_3 \otimes J = \begin{bmatrix} \tilde{\gamma}\, H + J & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \tilde{\alpha}\, H + J & -\tilde{\beta}\, H \\ \mathbb{0} & \tilde{\beta}\, H & \tilde{\alpha}\, H + J \end{bmatrix}, \tag{18}$$

where the scaled eigenvalues are defined as $\tilde{\gamma} = \frac{1}{T_s}\gamma$, $\tilde{\alpha} = \frac{1}{T_s}\alpha$ and $\tilde{\beta} = \frac{1}{T_s}\beta$. The linear system (17) is therefore split into two subsystems of dimension $n_k = n_x + n_z$ and $2\, n_k$. The latter is further transformed into a n_k-dimensional but complex subsystem as discussed in (Hairer and Wanner, 1991). Note that these transformations are deployed for

computing both the collocation variables K_i and their derivatives K_i^w in Eq. (12).

This approach has been implemented using complex arithmetic in the linear algebra routines. It is labeled *simplified Newton* in the numerical experiments of Section 6.

4.3 Scheme III: Single Newton

We detail next a third and last inexact lifted scheme, which will be using Single Newton-type methods as discussed for example by (Cooper and Vignesvaran, 1993) and (González-Pinto et al., 1995). They are based on the observation that if the inverse of the coefficient matrix A^{-1} has only one real eigenvalue γ, then the matrix (18) reads as:

$$\begin{bmatrix} \tilde{\gamma}\, H + J & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \tilde{\gamma}\, H + J & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & \tilde{\gamma}\, H + J \end{bmatrix}, \tag{19}$$

which makes the linear system (17) equivalent to three separate linear subsystems with the same real $n_k \times n_k$-matrix $\tilde{\gamma}\, H + J$. For most high order IRK methods, however, A^{-1} does not have this property (Hairer and Wanner, 1991).

Therefore, in the following we approximate the coefficient matrix A by $\tilde{A}$, selected such that its inverse has only one real eigenvalue γ. More details on how to construct this approximation can be found, e.g. in (González-Pinto et al., 1995). The matrix $\tilde{A}$ needs to be invertible and has a decomposition of the form $\tilde{A}^{-1} = \gamma\, W(\mathbb{1}_3 - E)W^{-1}$, where E is a strictly lower triangular matrix. Using the new matrix $\tilde{A}$ in (16) and premultiplying the linear system by $(T_s\, \tilde{A})^{-1} \otimes \mathbb{1}_{n_k}$, one obtains the equivalent expression:

$$\begin{aligned} (\mathbb{1}_3 \otimes (\tilde{\gamma}\, H + J))\, \Delta\hat{K}_i = &- (\tilde{\gamma}\, (\mathbb{1}_3 - E)\, W^{-1} \otimes \mathbb{1}_{n_k}) G_i \\ &+ (E \otimes \tilde{\gamma}\, H) \Delta\hat{K}_i, \end{aligned} \tag{20}$$

where $\tilde{\gamma} = \frac{1}{T_s}\gamma$ and $\Delta\hat{K}_i = (W^{-1} \otimes \mathbb{1}_{n_k})\Delta K_i$. The solution of the latter system requires merely the computation of one LU factorization of the $n_k \times n_k$-matrix $\tilde{\gamma}\, H + J$. Since matrix E is strictly lower triangular, the linear system in (20) results in three separable subsystems of dimension n_k which can be solved sequentially.

The specific Single Newton-type schemes which will be used further in this paper for the 3- and 4-stage Gauss method, can be found in (González-Pinto et al., 1995) and (González-Pinto et al., 2001), respectively.

4.4 Computational Complexity

The three proposed inexact Newton schemes allow for considerably reducing the computational cost of the implicit integrator. Table 1 shows a relative comparison of the computational complexity for the presented lifted collocation schemes, including the novel inexact variants proposed in this paper. The comparison assumes that the LU factorization of a $n \times n$ matrix requires $\sim \frac{2}{3}n^3$ flops and the back substitutions accordingly require $\sim 2n^2$ flops. Note that the table is set up for the Gauss collocation method, for which the coefficient matrix A has $\frac{S}{2}$ complex conjugate pairs of eigenvalues when the number of stages S is even or it has one real eigenvalue and $\frac{S-1}{2}$ complex conjugate pairs in case S is odd (Hairer and Wanner, 1991). This is important information for the Simplified Newton scheme.

Table 1. Computational cost of the lifting schemes for a Gauss collocation based implicit integrator ($n_k = n_x + n_z$ and $n_w = n_x + n_u$).

	computations (#flops)
Standard integrator	$(1 + N_s)\, \frac{2}{3}(S\, n_k)^3 + N_s\, 2(S\, n_k)^2(n_w + L)$
Lifted integrator	$N_s\, \frac{2}{3}(S\, n_k)^3 + N_s\, 2(S\, n_k)^2(n_w + 1)$
Inexact Lifted scheme - Newton with Reuse	$\frac{2}{3}(S\, n_k)^3 + N_s\, 2(S\, n_k)^2(n_w + 1)$
Inexact Lifted scheme - Simplified Newton	$\frac{4\,S}{3} n_k^3 + N_s\, (4\, S)\, n_k^2(n_w+1)$ [S even] $\frac{(4\,S-2)}{3} n_k^3 + N_s (4\, S-2)\, n_k^2(n_w+1)$ [S odd]
Inexact Lifted scheme - Single Newton	$\frac{2}{3} n_k^3 + N_s\, (2\, S)\, n_k^2(n_w + 1)$

Similarly to the discussion in (Quirynen et al., 2015), Table 1 shows the advantages of a lifted implicit integrator over the standard implementation from Algorithm 1. The lifted scheme performs exactly one Newton iteration, i.e. $L = 1$ and it eliminates the need for an extra LU factorization to compute the sensitivity information (Quirynen et al., 2014). The first Inexact Lifted scheme can further reduce this computational cost by reusing the factorization of the Jacobian matrix over all integration steps N_s within one shooting interval. As mentioned earlier, the Simplified and Single Newton-type schemes can be used to considerably reduce the cost of this one LU factorization and its use in back substitutions by transforming each linear system into multiple smaller subsystems.

As can be seen from Table 1, the Single Newton scheme is the cheapest to implement but it additionally approximates the coefficient matrix which affects the convergence as studied in more details by (Bickart, 1977), (Butcher, 1976) and (Cooper and Vignesvaran, 1993). A detailed convergence analysis for these Inexact Lifted schemes within a multiple shooting based SQP method is part of ongoing research. Section 6 will however illustrate the performance of the proposed schemes using the results of numerical experiments for a real-world application of Nonlinear MPC. For a discussion on the additional memory requirements which result from lifting an implicit integrator, we refer to (Quirynen et al., 2015).

5. SOFTWARE IMPLEMENTATION

All schemes proposed in this paper can be deployed via an open-source software implementation, which allows for easily auto generating lifted implicit integrators within a multiple shooting method for real-time optimal control applications. The algorithms are part of the open-source ACADO Toolkit software (Houska et al., 2011a) which can be downloaded from `www.acadotoolkit.org`. It is however its code generation tool which offers an easy way to export highly efficient C-code for fast optimal control, as presented in (Houska et al., 2011b; Quirynen et al., 2014).

The auto generated solvers are based on the Real-Time Iteration (RTI) scheme, which was proposed as an efficient online SQP-type algorithm for Nonlinear MPC (Diehl

et al., 2002). In the numerical results of this paper, the QP solutions will be obtained using the active-set solver qpOASES (Ferreau, 2006) in combination with a condensing technique to numerically eliminate the state variables as proposed by (Bock and Plitt, 1984).

6. CASE STUDY: NMPC FOR A DELTA ROBOT

This section illustrates the numerical performance of the proposed variants of the lifted implicit integrator within the RTI algorithm for Nonlinear MPC. All simulations are carried out using the ACADO code generation tool on a standard computer, equipped with Intel i7-3720QM processor, and using a 64-bit version of Ubuntu 14.04 and the g++ compiler version 4.8.2.

6.1 Problem formulation

As an illustrative example, we consider NMPC to perform point-to-point motions with a delta robot. The modeling follows similar lines as in (Codourey, 1998). In the Lagrange formalism, the Lagrangian $\mathcal{L} = \kappa - \nu$ of the robot has the kinetic and potential energy functions:

$$\kappa = m\dot{p}^\top \dot{p} + z^\top c + \frac{1}{2} J \dot{\theta}^\top \dot{\theta}, \qquad \nu = mgp_3, \tag{21}$$

where $p \in \mathbb{R}^3$ denotes the position of the robot nacelle, and $\theta \in \mathbb{R}^3$ represents the angle of the three arms. The constraints $c \in \mathbb{R}^3$, given by

$$c_k = \left(\|p - R_k a_k\|^2 - l^2 \right), \quad k = 1, 2, 3$$

enforce the geometry of the robot, with:

$$R_k = \begin{bmatrix} \cos\alpha_k & -\sin\alpha_k & 0 \\ \sin\alpha_k & \cos\alpha_k & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad a_k = l_{\text{arm}} \begin{bmatrix} \cos\theta_k \\ 0 \\ \sin\theta_k \end{bmatrix} \tag{22}$$

for $\alpha = \begin{bmatrix} 0 & \frac{2\pi}{3} & \frac{4\pi}{3} \end{bmatrix}$. Constant $l = 0.6$m stands for the length of the robot parallelograms, $l_{\text{arm}} = 0.2$m for the robot arms, $m = 5{\cdot}10^{-2}$kg for the nacelle mass, and $J = 1{\cdot}10^{-1}\text{kg}\cdot\text{m}^2$ for the motor-arms inertia. The inertia of the parallelograms is neglected. Defining $q^\top = \begin{bmatrix} \theta^\top & p^\top \end{bmatrix}$, the dynamics then take the simple DAE index-1 form:

$$\begin{bmatrix} M & \nabla c \\ \nabla c^\top & 0 \end{bmatrix} \begin{bmatrix} \ddot{q} \\ z \end{bmatrix} = \begin{bmatrix} F_{\text{g}} \\ \frac{\partial}{\partial q} \left(\frac{\partial c}{\partial q} \dot{q} \right) \dot{q} \end{bmatrix} \tag{23}$$

with $M = \text{diag}\left(\begin{bmatrix} J & J & J & m & m & m \end{bmatrix}\right)$ and

$$F_{\text{g}} = \begin{bmatrix} T \\ \mathbf{0}_{3\times 1} \end{bmatrix} + \nabla_q \nu. \tag{24}$$

Here $T \in \mathbb{R}^3$ stands for the motor torques. The control inputs u are then chosen as $\dot{T} = u$ in order to impose a penalty on the motor torque time-derivatives, and the integral states $I \in \mathbb{R}^3$ are introduced to reject permanent positioning error, i.e. $\dot{I} = p - p_{\text{ref}}$. The robot model then has the states:

$$x^\top = \begin{bmatrix} q^\top & \dot{q}^\top & T^\top & I^\top \end{bmatrix} \in \mathbb{R}^{18}, \qquad z \in \mathbb{R}^3. \tag{25}$$

The OCP objective is of the form (1a), and consists of a least squares cost minimizing the deviation of the position for the end effector from its desired reference values, and the effort of the actuators. In addition to the index-1 DAE model which describes the system dynamics, simple bound constraints on the torques were included.

Table 2. Average NMPC timing results in [ms] for the delta robot example using a lifted implicit integrator, based on Gauss collocation.

	Standard	Lifted	Inexact Lifted		
$S = 4, N_s = 1$	$L = 5$		Reuse	Simplified	Single
integration	12.84	5.92	-	3.05	**1.71**
total RTI step	13.41	6.52	-	3.66	2.31
$S = 3, N_s = 2$					
integration	11.30	5.96	5.07	3.14	**2.54**
total RTI step	11.90	6.55	5.67	3.77	3.15

6.2 Numerical Results

Similar to the computational complexity comparison in Table 1, let us implement each of the collocation based integrator schemes using ACADO code generation. Table 2 presents the average computation time per RTI step for the delta robot, including the average time spent in the integrator. Note that $N_s = 2$ integration steps of the Gauss method of order 6 ($S = 3$ stages) and only one step of the order 8 method ($S = 4$) was used (Hairer and Wanner, 1991), in order to obtain a similar simulation accuracy (Quirynen et al., 2014). The standard implementation used a fixed number of $L = 5$ Newton iterations.

Even though lifting the implicit integrator as described in Algorithm 2 requires little coding effort, Table 2 shows a numerical speedup of about factor 2 over the standard approach. Using our novel inexact Newton based lifted schemes, this computational cost can however significantly be reduced further up to a speedup of about factor $6-8$. From Table 2, it can be observed that the use of inexact lifted methods is particularly advantageous in combination with higher order collocation schemes. This can also be seen from the effect of the number of stages S on the cost comparison in Table 1.

Figure 1 presents convergence results for the SQP method in the multiple shooting framework (4), using different integrators based on the Gauss method of order 6 ($S = 3$). It shows for each iteration, the distance $\|W - W^*\|_\infty$ of the current optimization values from the local minimum $W^* = (X^*, U^*)$ of the NLP in Eq. (2). As discussed in more details by (Quirynen et al., 2015), the standard integrator based algorithm does not fully converge to the solution W^* which is consistent with the collocation equations from (5). This could be resolved for a specific tolerance by sufficiently increasing the number of iterations, which was chosen $L = 5$ in this case. Note that the convergence for variants II and III of the inexact lifted scheme, is very close to that of the exact lifting method in Figure 1.

In practice, the RTI scheme involves only one SQP iteration per time step for real-time feasible control. Figure 2 presents closed-loop NMPC simulations, showing the position and motor torques for the delta robot. It allows us to compare the RTI algorithm using either a standard integrator or the Single Newton based inexact lifted scheme, with a fully converged SQP based implementation. There is however little to no noticeable difference in the closed-loop behaviour, as illustrated in Figure 2.

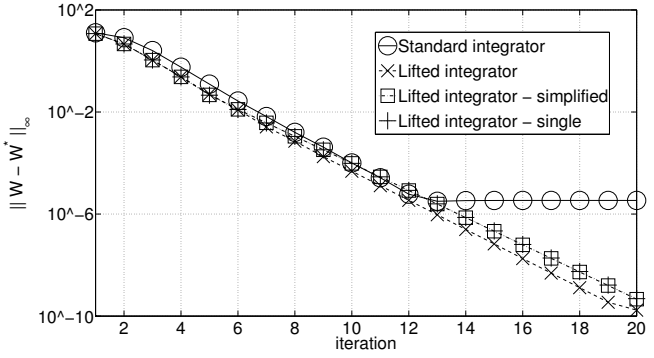

Fig. 1. SQP convergence results for the delta robot OCP, using integrators based on a Gauss method ($S = 3$).

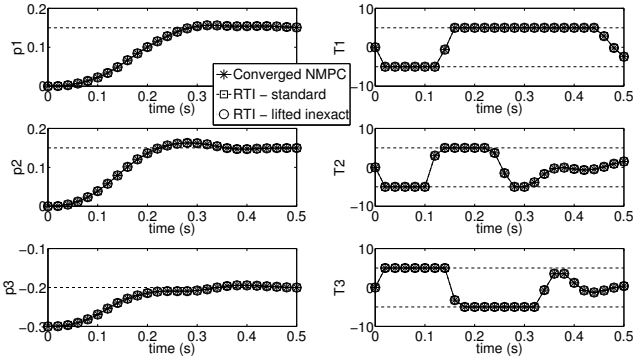

Fig. 2. Closed-loop NMPC behaviour for the delta robot, comparing RTI using the standard or the inexact lifting scheme with a fully converged SQP method.

7. CONCLUSIONS AND FUTURE WORK

This paper presents a novel class of inexact Newton based lifted implicit integrators, to be used in direct multiple shooting for optimal control. Just like the exact lifting scheme, the proposed methods perform only one Newton type step per SQP iteration. Even though these inexact lifting techniques require little extra coding effort, they typically result in a significant reduction of the overall computational cost. An open-source implementation of the presented algorithms is provided as part of the ACADO code generation tool, and their performance was illustrated using the real-world example of controlling a delta robot. Based on these numerical experiments, computational speedups were reported of up to factor $6 - 8$ over the standard collocation-based integrator.

Future work will include a detailed convergence analysis for the proposed lifted implicit integrator schemes.

REFERENCES

Albersmeyer, J. and Diehl, M. (2010). The Lifted Newton Method and its Application in Optimization. *SIAM Journal on Optimization*, 20(3), 1655–1684.

Bickart, T.A. (1977). An efficient solution process for implicit runge-kutta methods. *SIAM Journal on Numerical Analysis*, 14(6), pp. 1022–1027.

Biegler, L.T. (2010). *Nonlinear Programming*. MOS-SIAM Series on Optimization. SIAM.

Bock, H. and Plitt, K. (1984). A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings 9th IFAC World Congress Budapest*, 242–247. Pergamon Press.

Butcher, J. (1976). On the implementation of implicit runge-kutta methods. *BIT Numerical Mathematics*, 16(3), 237–240.

Codourey, A. (1998). Dynamic Modeling of Parallel Robots for Computed-Torque Control Implementation. *International Journal of Robotics*, 17(12), 1325–1336.

Cooper, G. and Vignesvaran, R. (1993). Some schemes for the implementation of implicit runge-kutta methods. *Journal of Computational and Applied Mathematics*, 45(1 - 2), 213 – 225.

Dennis, J.E., J. (1968). On newton-like methods. *Numerische Mathematik*, 11(4), 324–330.

Diehl, M., Bock, H., Schlöder, J., Findeisen, R., Nagy, Z., and Allgöwer, F. (2002). Real-time optimization and Nonlinear Model Predictive Control of Processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4), 577–585.

Diehl, M., Ferreau, H.J., and Haverbeke, N. (2009). *Nonlinear model predictive control*, volume 384 of *Lecture Notes in Control and Information Sciences*, chapter Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation, 391–417. Springer.

Ferreau, H. (2006). *An Online Active Set Strategy for Fast Solution of Parametric Quadratic Programs with Applications to Predictive Engine Control*. Master's thesis, University of Heidelberg.

González-Pinto, S., Montijano, J.I., and Rández, L. (1995). Iterative schemes for three-stage implicit runge-kutta methods. *Appl. Numer. Math.*, 17(4), 363–382.

González-Pinto, S., Pérez-Rodríguez, S., and Montijano, J.I. (2001). Implementation of high-order implicit runge-kutta methods. *Computers & Mathematics with Applications*, 41(7-8), 1009 – 1024.

Hairer, E. and Wanner, G. (1991). *Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems*. Springer, Berlin Heidelberg, 2nd edition.

Houska, B., Ferreau, H., and Diehl, M. (2011a). ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3), 298–312.

Houska, B., Ferreau, H., and Diehl, M. (2011b). An Auto-Generated Real-Time Iteration Algorithm for Nonlinear MPC in the Microsecond Range. *Automatica*, 47(10), 2279–2285.

Kirches, C., Wirsching, L., Sager, S., and Bock, H. (2010). Efficient numerics for nonlinear model predictive control. In M. Diehl, F. F. Glineur, and E.J.W. Michiels (eds.), *Recent Advances in Optimization and its Applications in Engineering*, 339–357. Springer.

Quirynen, R., Gros, S., and Diehl, M. (2015). Lifted implicit integrators for NMPC based on Multiple Shooting. In *Conference on Decision and Control*. Submitted.

Quirynen, R., Vukov, M., Zanon, M., and Diehl, M. (2014). Autogenerating Microsecond Solvers for Nonlinear MPC: a Tutorial Using ACADO Integrators. *Optimal Control Applications and Methods*.