Efficient NMPC for nonlinear models with linear subsystems

Efficient NMPC for nonlinear models with linear subsystems

Rien Quirynen, Sébastien Gros and Moritz Diehl

Abstract— **Real-time optimal control algorithms for fast, mechatronic systems need to be run on embedded hardware and they need to respect tight timing constraints. When using nonlinear models, the simulation and generation of sensitivities forms a computationally demanding part of any algorithm. Automatic code generation of Implicit Runge-Kutta (IRK) methods has been shown to reduce its CPU time significantly. However, a typical model also shows a lot of structure that can be exploited in a rather elegant and efficient way. The focus of this paper is on nonlinear models with linear subsystems. With the proposed model formulation, the new auto generated integrators can be considered a powerful generalization of other solvers, e.g. those that support quadrature variables. A speedup of up to $5-10$ is shown in the integration time for two examples from the literature.**

Keywords **: structure exploitation, code generation, IRK methods, NMPC, embedded optimization**

I. INTRODUCTION

Nonlinear Model Predictive Control (NMPC) [1] and Moving Horizon Estimation (MHE) [2] are popular approaches for real-time optimal control and estimation, since they can explicitly handle constraints and nonlinear dynamics. In the case of a system with fast dynamics, the high computational burden forms a major challenge. An Optimal Control Problem (OCP) needs to be solved at each sampling time, respecting the hard timing constraints which are imposed by real-time applications. Recent algorithmic progress [3], [4], [5] allows to consider NMPC and MHE also for fast systems. Among the available online algorithms, the Real-Time Iteration (RTI) scheme [6] has been proposed as a highly competitive approach.

In addition to using an online algorithm, efficient implementations are needed in a specific programming language to allow running it in real-time on embedded control hardware. One way to achieve this is by automatic code generation, i.e. by exporting a customized solver. Significant improvements in the computation time can also be obtained by removing unnecessary computations, optimizing the memory access and cache usage, and by exploiting the problem structure and sparsity patterns. This rather old idea has become quite popular for e.g. convex optimization [7]. It is now possible to assemble a complete algorithm for solving an OCP, starting from different components of which each could be code generated when advantageous. This is the approach pursued by the ACADO code generation tool, which exports highly efficient C-code based on the RTI scheme [8]. It has a MATLAB interface to export and use the same code in simulations [9].

A rather important component in case of NMPC is the one that handles integration and sensitivity generation for the nonlinear model. It forms typically a major or even dominating factor in the total computation time, while the accuracy of the derivatives also strongly defines the success of the optimizer. Because of the need for a more or less deterministic runtime in case of real-time applications, adaptivity is avoided where possible. The assumption in this paper is therefore that the step size and order of the integration method are fixed. It has been shown that code generation for Implicit Runge-Kutta (IRK) methods with a tailored approach for computing the sensitivities can be very efficient [10]. This is important because of the attractive properties of A-stable IRK methods and their natural extension from systems of Ordinary Differential Equations (ODE) to Differential Algebraic Equations (DAE) of index 1. In the specific case of collocation methods, extra output functions can be evaluated on an arbitrary grid and this at a rather low cost [9].

In this paper, the following OCP formulation is addressed

$$\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_t^{t+T} (\|x(\tau)-x_{\text{ref}}(\tau)\|_Q^2 + \|u(\tau)-u_{\text{ref}}(\tau)\|_R^2)\,\mathrm{d}\tau \\
& + \|x(t+T)-x_{\text{ref}}(t+T)\|_P^2 \\
\text{s.t.} \quad & x(t) = \bar{x}_t, \\
& \dot{x}(\tau) = f(\tau, x(\tau), u(\tau)), \\
& \underline{u} \le u(\tau) \le \bar{u}, \\
& \underline{x} \le x(\tau) \le \bar{x}, \qquad \forall \tau \in [t, t+T],
\end{aligned} \tag{1}$$

in which x and u are the states and controls, x_{ref} and u_{ref} are the state and control references, t denotes the current time and T the length of the prediction horizon. Of particular interest is the nonlinear function f which defines the system dynamics. It is known that the models used in applications, either ODE or DAE systems, often have a clear structure resulting in e.g. a sparse Jacobian matrix. A scalable way to

R. Quirynen, S. Gros and M. Diehl are with the Optimization in Engineering Center (OPTEC), KU Leuven, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium. rien.quirynen@esat.kuleuven.be

* 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. Flemish Government: IOF / KP / SCORES4CHEM, FWO: PhD/postdoc grants and projects: G.0320.08 (convex MPC), G.0377.09 (Mechatronics MPC); IWT: PhD Grants, projects: SBO LeCoPro; Belgian Federal Science Policy Office: IUAP P7 (DYSCO, Dynamical systems, control and optimization, 2012-2017); EU: FP7- EMBOCON (ICT-248940), FP7-SADCO (MC ITN-264735), ERC ST HIGHWIND (259 166), Eurostars SMART, ACCM. The first author holds a PhD fellowship of the Research Foundation – Flanders (FWO). The authors also thank M. Vukov for fruitful discussions and joint development of the ACADO code generation tool.

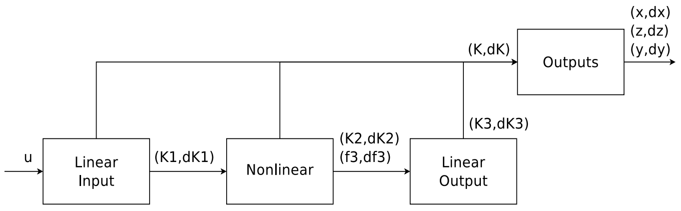

Fig. 1. Schematic illustrating the three stages, together with the workflow in the structure exploiting integrators.

handle this is by using sparse solutions for the linear algebra operators when that is advantageous. It will however create little to no gain for small to medium scale systems which appear in the problems that are targeted by this paper. The goal is therefore to have a closer look at the structure specific to these systems and to propose an alternative approach for structure exploitation, that will be shown to perform very well. The addressed model class contains as a subclass the Wiener-Hammerstein models which are since a long time used in system identification [11].

Contribution: This paper presents algorithms to exploit a frequently occurring model structure, resulting in very efficient auto generated integrators.

The paper is organized as follows. In Section II, a three stage model formulation will be proposed. Section III summarizes the used framework of collocation methods with sensitivity generation. Then, the issue of how to exploit the three stage structure in these IRK methods, is addressed by Section IV. Finally, some real-world applications are used in Section V to illustrate the structure exploiting integrators and their performance.

II. The three stage structure

In this section, a typical structure consisting of three different stages in the system of differential equations is discussed.

A. Model formulation

When modeling for example a mechatronic system, the result is typically a set of nonlinear differential equations with possibly some algebraic states. In the case of an explicit ODE such as $\dot{x} = f(x,u)$ from the OCP in (1), one would often recognize one or more of the following three subsystems in this specific order

$$\begin{aligned} \dot{x}^{[1]} &= A_1 x^{[1]} + B_1 u, && \text{(linear input system)} \\ \dot{x}^{[2]} &= f_2(x^{[1]}, x^{[2]}, u), && \text{(nonlinear system)} \\ \dot{x}^{[3]} &= A_3 x^{[3]} + f_3(x^{[1]}, x^{[2]}, u), && \text{(linear output system)} \end{aligned} \tag{2}$$

with the matrices A_1, B_1 and A_3 and the nonlinear functions f_2 and f_3 defining the subsystems. Figure 1 illustrates this chain of three subsystems. In what follows, $N_x^{[1]}$, $N_x^{[2]}$ and $N_x^{[3]}$ denote the number of differential states in each of the three subsystems. As mentioned earlier, this class comprises Wiener-Hammerstein models which consist of a linear dynamic system followed by a static nonlinearity and another linear dynamic system. In our notation, they are of form (2) with $N_x^{[2]} = 0$. Let us generalize this structure to an implicit DAE system of index 1 as follows:

$$C_1 \dot{x}^{[1]} = A_1 x^{[1]} + B_1 u, \tag{3a}$$

$$0 = f_2(x^{[1]}, x^{[2]}, \dot{x}^{[1]}, \dot{x}^{[2]}, z, u), \tag{3b}$$

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

with invertible matrices C_1 and C_3 and function f_2 satisfying $\frac{\partial f_2}{\partial (z, \dot{x}^{[2]})}$ invertible.

B. Motivation

The introduced structure is not something that comes up only in rare occasions. These stages almost always arise naturally when modeling for control. A linear input system (3a) can result from

- some partially linear dynamics which are independent of the states in the nonlinear equations. A classical example would be the double integrator

$$\begin{aligned} \dot{x}_1 &= u \\ \dot{x}_2 &= x_1 \end{aligned} \tag{4}$$

  but any linear variant of this is possible.
- any linear high order differential equation, which would be transformed into a set of first order equations

$$\begin{aligned} \frac{d^n x}{dt^n} &= h(t, u(t), x(t), \dots, \frac{d^{n-1}x}{dt^{n-1}}) \\ &\Downarrow \quad x_1 = x, \dots, x_n = \frac{d^{n-1}x}{dt^{n-1}} \\ \dot{x}_1 &= x_2 \\ &\dots \\ \dot{x}_n &= h(t, u(t), x_1(t), \dots, x_n(t)), \end{aligned} \tag{5}$$

  with h a linear function.
- implementing a filter on the controls or input states. In optimal control, it e.g. makes sense to extra penalize the high frequency content. Therefore, a High-Pass RC filter such as

$$\frac{dx(t)}{dt} = -\omega_C x(t) + \frac{du(t)}{dt} \tag{6}$$

  can be used, with $\omega_C = \frac{1}{RC} = 2\pi f_C$ in which f_C is called the cutoff frequency.

The linear output system in (3c) is somewhat similar to the input system in (3a), but with a nonlinear relation with respect to the previous states and the control inputs. An output system can therefore result from some partially linear dynamics which also depend on the previous states or it can implement a filter for these states. In the latter cases, the matrix A_3 will generally be nonzero. When $A_3 = 0$ and C_3 is an identity matrix, Equation (3c) reduces to

$$\dot{x}^{[3]} = f_3(x^{[1]}, x^{[2]}, \dot{x}^{[1]}, \dot{x}^{[2]}, z, u) \tag{7}$$

which are known in a different context as quadrature states [12]. They are typically used to formulate objective and constraint functions in an OCP.

III. Auto generated IRK methods

This section compactly describes the general implementation of auto generated IRK methods, presented earlier in [10] and [9]. Their formulation is handled in Subsection III-A, for ODE as well as DAE systems of index 1. Some comments on the implementation are repeated in Subsection III-B and the used methods for efficient generation of continuous output and sensitivities are respectively described by Subsection III-C and III-D.

A. Formulation

The general IRK methods from [9] solve the following Initial Value Problem (IVP) over a certain time:

$$\begin{aligned} 0 &= f(t, \dot{x}(t), x(t), z(t), u(t)), \\ x(0) &= x_0, \end{aligned} \tag{8}$$

with $x(t)$ a vector of N_x differential states, $\dot{x}(t)$ the corresponding time derivatives, $z(t)$ a vector of N_z algebraic states and $u(t)$ a vector of control inputs so that f defines a system of $N_x + N_z$ equations. Note that $N_x = N_x^{[1]} + N_x^{[2]} + N_x^{[3]}$ in case of the structure in (3). These methods handle models ranging from explicit ODE until fully implicit DAE systems. The Jacobian matrix $\frac{\partial f}{\partial(z,\dot{x})}$ needs to be invertible, so that the algebraic states z and the differential state derivatives $\dot{x}$ are uniquely defined, i.e. the DAE system should be of index 1 [14].

Because of their high order of accuracy and good stability properties, the focus of this paper will be on A-stable IRK methods such as the Gauss methods [15]. Applied to the system in (8) for the integration from t_n until $t_{n+1} = t_n + h$, an s-stage IRK method can be formulated as follows:

$$\begin{aligned} 0 &= f\left(t_n + c_i h, k_i, x_n + h \sum_{j=1}^{s} a_{ij} k_j, Z_i\right), \ i = 1, \ldots, s, \\ x_{n+1} &= x_n + h \sum_{i=1}^{s} b_i k_i, \end{aligned} \tag{9}$$

with Z_i the stage values of the algebraic states, k_i the stage values of the differential state derivatives and the coefficients c_i, b_i and a_{ij} are defined by the Butcher table. The method results in a nonlinear system of $(N_x + N_z) \times s$ equations that can be solved using a Newton method. Using this scheme, consistency of the state values is only assured at the s collocation nodes. For stiffly accurate methods such as the Radau IIA methods, the endpoint is included in these nodes.

B. Implementation

With real-time applications in mind, the ACADO code generation tool exports a customized integrator with a deterministic runtime. The step size, the order of the method and the number of Newton iterations are therefore fixed. Let us write the nonlinear system from (9) as $G(x_n, K) = 0$, resulting in the following Newton-type iterations:

$$\begin{aligned} M &= \frac{\partial G}{\partial K}(x_n, K^{[0]}), \\ K^{[i]} &= K^{[i-1]} - M^{-1} G(x_n, K^{[i-1]}), \ i = 1, \ldots, L, \end{aligned} \tag{10}$$

with M the evaluation of the Jacobian $\partial G / \partial K$ in the initialization point and the variables $K = (k_1, \ldots, k_s, Z_1, \ldots, Z_s)$. Note that one Jacobian evaluation and a small amount of iterations L is sufficient if this initialization $K^{[0]}$ is good enough. The evaluation of the Jacobian can be done using Algorithmic Differentiation (AD) [16]. A custom linear solver can even be exported, e.g. based on the LU decomposition of the matrix M.

C. Continuous output

Promising possibilities of auto generated IRK methods with continuous output have already been mentioned and illustrated in [9]. In the case of collocation methods, a specific family of IRK methods, this continuous extension comes naturally. Using the collocation variables $K = (k_1, \ldots, k_s, Z_1, \ldots, Z_s)$, a polynomial is defined for $x(t)$, $\dot{x}(t)$ and $z(t)$:

$$\begin{aligned} x(t_n + ch) &\approx x_n + h \sum_{i=1}^{s} k_i \int_0^c l_i(\tau)\, \mathrm{d}\tau, \\ \dot{x}(t_n + ch) &\approx \sum_{i=1}^{s} l_i(c) k_i, \\ z(t_n + ch) &\approx \sum_{i=1}^{s} l_i(c) Z_i, \end{aligned} \tag{11}$$

where $l_i(t) = \prod_{j \neq i} \frac{t - c_j}{c_i - c_j}$ are the Lagrange interpolating polynomials. In the case of an s-stage IRK method of order p, the order of this approximation is $p^* = \min(p, s+1)$ for $x(t)$ and $p^* - 1$ for $\dot{x}(t)$ and $z(t)$ [17]. Output functions as general as

$$y = \psi(t, \dot{x}(t), x(t), z(t)), \tag{12}$$

can efficiently be evaluated on an arbitrarily fine grid.

D. Sensitivity generation

In the context of dynamic optimization, sensitivities with respect to all the independent variables $w_n = (x, u)_n$ are needed in addition to the simulated values of states and outputs. In the case of extra outputs, forward techniques to compute the sensitivities $\partial(x_{n+1}, z_{n+1}) / \partial w_n$ are recommended. A thorough discussion on such techniques of sensitivity generation for IRK methods can be found in [13]. The conclusion there is that the most efficient method is to apply the Implicit Function Theorem (IFT) to $G(w_n, K) = 0$, resulting in

$$\frac{dK}{dw_n} = -\frac{\partial G}{\partial K}^{-1} \frac{\partial G}{\partial w_n}, \tag{13}$$

which can then be used to compute the sensitivities of the states and outputs. This direct approach can provide very accurate sensitivities, which is important for optimization. The Jacobian evaluation in (13) and its factorization can be reused in (10) for the next integration step [10]. Algorithm 1 compactly describes the resulting implementation.

Algorithm 1 The implementation of one step

```
Input: $w_n$, initial $K^{[0]}$, LU factorization of $M$
Output: $(x, \partial x/\partial w_n)_{n+1}$ and $(z, \partial z/\partial w_n)_n$
 1: for $i = 1 \to L$ do
 2:     $K^{[i]} \leftarrow K^{[i-1]} - M^{-1}G(w_n, K^{[i-1]})$
 3: end for
 4: $x_{n+1} \leftarrow x_n + h\sum_{i=1}^{s} b_i k_i$
 5: $z_n \leftarrow \sum_{i=1}^{s} l_i(0) Z_i$   with  $l_i(t) = \prod_{j\neq i} \frac{t-c_j}{c_i-c_j}$
 6: $M \leftarrow \frac{\partial G}{\partial K}(w_n, K^{[L]})$
 7: compute $\frac{dK}{dw_n}$ using $M$ in (13)
 8: $\partial x_{n+1}/\partial w_n \leftarrow \partial x_n/\partial w_n + h\sum_{i=1}^{s} b_i \frac{dk_i}{dw_n}$
 9: $\partial z_n/\partial w_n \leftarrow \sum_{i=1}^{s} l_i(0) \frac{dZ_i}{dw_n}$
10: compute outputs and sensitivities using $(K, \frac{dK}{dw_n})$
```

IV. Structure exploiting IRK methods

This section starts by adapting the IRK methods to the specific three stage structure in Subsection IV-A, followed by some remarks in Subsection IV-B. The availability of these methods in the ACADO Toolkit is briefly illustrated in Subsection IV-C.

A. Implementation details

The methods from Section III could be applied to the system in (3), ignoring the structure that is present. This paper, however, proposes an implementation which is completely equivalent with the latter, while exploiting the structure as much as possible. As illustrated in Figure 1, the scheme can be represented by four blocks in total including the one that handles the extra outputs and their sensitivities. The idea is to compute all the collocation variables K and their derivatives $\frac{dK}{dw_n}$, so that the outputs and the sensitivities can still be generated in a similar way as before.

a) Linear input system: Let us define the collocation variables $K_1 = (k_1^{[1]}, \dots, k_s^{[1]})$ for the linear input system. The resulting equations in (9) will also be linear, namely

$$M_1 = \frac{\partial G_1}{\partial K_1} = \begin{pmatrix} C_1 - ha_{11}A_1 & \cdots & -ha_{1s}A_1 \\ \vdots & \ddots & \vdots \\ -ha_{s1}A_1 & \cdots & C_1 - ha_{ss}A_1 \end{pmatrix} \tag{14}$$

is a constant $sN_x^{[1]} \times sN_x^{[1]}$ matrix in which G_1 defines the collocation equations. Instead of performing L Newton iterations like in (10), the variables can now be computed by $K_1 = -M_1^{-1}G_1(x_n)$. The inverse matrix M_1^{-1} can be precomputed offline, which leaves us with only a matrix-vector multiplication. Even the derivatives $\frac{dK_1}{dw_n}$ are constant and will therefore be precomputed.

b) Nonlinear system: For the implicit nonlinear system, the methods are still applied as described in Section III for the variables $K_2 = (k_1^{[2]}, \dots, k_s^{[2]}, Z_1, \dots, Z_s)$ of dimension $s(N_x^{[2]} + N_z)$. The only difference is that the derivatives $\frac{dK_1}{dw_n}$ are now needed to evaluate $\frac{\partial G_2}{\partial w_n}$ for the sensitivity generation with G_2 defining the nonlinear collocation equations. In Figure 1, evaluations of the function f_3 in (3c) and of its Jacobian are also considered to be part of this block so that it handles all nonlinear parts.

c) Linear output system: Completely similar to the input system, the variables $K_3 = (k_1^{[3]}, \dots, k_s^{[3]})$ can be computed by $K_3 = -M_3^{-1}G_3(x_n)$ in which $M_3 = \frac{\partial G_3}{\partial K_3}$ is a $sN_x^{[3]} \times sN_x^{[3]}$ matrix and G_3 defines the collocation equations for the linear output system. The inverse matrix M_3^{-1} and the derivatives of K_3 with respect to $x_n^{[3]}$ are constant and can be precomputed. The rest of the derivatives are computed by the matrix-vector multiplication $\frac{dK_3}{dw_n} = -M_3^{-1}\frac{\partial G_3}{\partial w_n}$, using the derivatives $\frac{dK_1}{dw_n}$ and $\frac{dK_2}{dw_n}$ in the evaluation of $\frac{\partial G_3}{\partial w_n}$.

B. Some small remarks

First of all, the Jacobian matrix $\frac{dK}{dw_n}$ has a clear sparsity structure

$$\frac{dK}{dw_n} = \begin{pmatrix} \frac{dK_1}{dx^{[1]}} & 0 & 0 & \frac{dK_1}{du} \\ \frac{dK_2}{dx^{[1]}} & \frac{dK_2}{dx^{[2]}} & 0 & \frac{dK_2}{du} \\ \frac{dK_3}{dx^{[1]}} & \frac{dK_3}{dx^{[2]}} & \frac{dK_3}{dx^{[3]}} & \frac{dK_3}{du} \end{pmatrix}, \tag{15}$$

which is exploited in the implementation of the integrators where possible. In addition, it is interesting to note that the in- and output system actually have an analytical solution since they consist of a set of linear differential equations with constant coefficients. Using this would however have little impact on the computational complexity and the overall accuracy of the integration method for the complete system.

C. The ACADO software

The presented integrators are part of the open source ACADO code generation tool [8]. It is implemented in C++ and efficient C-code is exported, tailored to a specific problem. But a brief glance at this tool is made possible by its MATLAB interface. Consider a simple artificial system

$$C_1\dot{x}_1 = A_1x_1 + B_1u, \tag{16a}$$
$$\dot{x}_2 = ux_1, \tag{16b}$$
$$C_3\dot{x}_3 = A_3x_3 + x_2^2, \tag{16c}$$

where the A, B and C matrices are constant, x_1, x_2 and x_3 are vectors of each 2 differential states, u is the control input and (16a)-(16c) corresponds to the three stages from Section II. Let us now export a standalone integrator that exploits the structure in this model using ACADO from MATLAB:

```
DifferentialState x1(2) x2(2) x3(2);
Control u;

A1 = [ ]; A3 = [ ];       % 2x2 matrix
C1 = [ ]; C3 = [ ];       % 2x2 matrix
B1 = [ ];                 % 2x1 vector
sim = acado.SIMexport( 0.1 );

sim.setLinearInput(C1, A1, B1);
sim.setModel(dot(x2) == [u*x1]);
sim.setLinearOutput(C3, A3, [x2.^2]);

sim.set( 'INTEGRATOR_TYPE', 'INT_IRK_GL4' );
sim.set( 'NUM_INTEGRATOR_STEPS', 2 );
sim.exportCode('export')
```

Note that it is also possible to export these integrators directly as part of an NMPC algorithm, which is done in the next section.

V. Applications in Optimal Control

The goal of this section is both to illustrate the reasoning behind the proposed three stage structure in II-A and to present speedups that could be expected from the structure exploiting integrators. To strengthen this message, two real-world problems will be tackled. An application of NMPC on an overhead crane is discussed in Subsection V-A, followed by a quadcopter example in Subsection V-B. The experiments presented in this section are performed using the ACADO code generation tool on an ordinary computer (Intel i7-3720QM 6MB cache, 2.60 GHz, 64-bit Ubuntu 12.04 and Clang 3.0).

A. NMPC of an overhead crane

A dynamic model very similar to the one in [18] is used for the same overhead crane. Let us immediately write down the equations in the presented three-stage format. The linear input system consists of

$$\begin{aligned} \dot{x}_T &= v_T & \dot{v}_T &= a_T & \dot{u}_T &= u_{TR} \\ \dot{x}_L &= v_L & \dot{v}_L &= a_L & \dot{u}_L &= u_{LR}, \end{aligned} \tag{17}$$

followed by two nonlinear equations:

$$\begin{aligned} \dot{\theta} &= \omega \\ \dot{\omega} &= -\frac{1}{x_L}(g\sin(\theta) + a_T\cos(\theta) + 2v_L\omega), \end{aligned} \tag{18}$$

with:

$$a_T = -\frac{1}{\tau_1}v_T + \frac{A_1}{\tau_1}u_T \qquad a_L = -\frac{1}{\tau_2}v_L + \frac{A_2}{\tau_2}u_L. \tag{19}$$

Note that there is no linear output system and the different parameters are defined in [18]. The NMPC algorithm can now use an OCP formulation similar to (1) with this model for e.g. a simple point-to-point motion. Table I illustrates the speedup for one iteration of the RTI scheme implemented in ACADO. Using 10 control intervals over a horizon of 1s and 4 integration steps of the 4th order Gauss method per interval, it presents the average computation times for the different components with and without structure exploitation in the auto generated integrator. For this relatively small model, a speedup factor of 3 can already be observed for the total computation time spent in integration and sensitivity generation.

It has been illustrated that a linear input system can appear naturally. Consider now the situation where one would like to extra penalize the high frequency content in the states. One way to do this is by implementing a High-Pass RC filter such as in (6) for each of the states. This results in the following 6 equations being added to the input system:

$$\begin{aligned} \dot{x}_T^{HP} &= \dot{x}_T - \omega_C x_T^{HP} & \dot{x}_L^{HP} &= \dot{x}_L - \omega_C x_L^{HP} \\ \dot{v}_T^{HP} &= \dot{v}_T - \omega_C v_T^{HP} & \dot{v}_L^{HP} &= \dot{v}_L - \omega_C v_L^{HP} \\ \dot{u}_T^{HP} &= \dot{u}_T - \omega_C u_T^{HP} & \dot{u}_L^{HP} &= \dot{u}_L - \omega_C u_L^{HP}, \end{aligned} \tag{20}$$

	unstructured	structured
integration method	**220 µs**	**67 µs**
condensing	6 µs	6 µs
QP solution (qpOASES)	16 µs	16 µs
remaining operations	3 µs	3 µs
one real-time iteration	245 µs	92 µs

TABLE I
Average computation times for NMPC of an overhead crane.

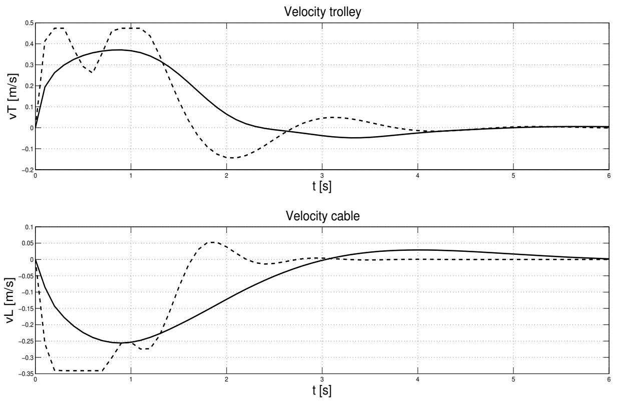

Fig. 2. Closed-loop trajectories for the velocity of both trolley and cable, before (dashed) and after extra penalization of high frequencies (solid).

but also in a linear output system consisting of

$$\dot{\theta}^{HP} = \dot{\theta} - \omega_C\theta^{HP} \qquad \dot{\omega}^{HP} = \dot{\omega} - \omega_C\omega^{HP}. \tag{21}$$

The new states in these equations are modeling the high frequency content in the corresponding states from the original system, leading to a complete system of 16 differential states. Weighting these high frequency states in the NMPC formulation causes a certain smoothing of the closed-loop trajectories, as depicted in Figure 2. Similar to before, Table II now presents a speedup factor of 6 for the integration time due to the structure exploitation on this extended model.

B. NMPC of a quadcopter

In this second example, NMPC will be applied to a Quadcopter (see e.g. [19]). This paper uses a simple Quad-

	unstructured	structured
integration method	**1380 µs**	**238 µs**
condensing	25 µs	25 µs
QP solution (qpOASES)	12 µs	12 µs
remaining operations	13 µs	13 µs
one real-time iteration	1430 µs	288 µs

TABLE II
Average computation times for NMPC of an overhead crane, with extra penalization of higher frequencies.

copter model, where it is assumed that low-level speed controllers ensure the tracking of the reference velocities of the propellers, which are computed by the NMPC scheme. The model equations are briefly summarized by

$$\ddot{\omega}_k^{\text{ref}} = U_k, \qquad \dot{\omega}_k = \tau^{-1}\left(\omega_k^{\text{ref}} - \omega_k\right), \tag{22}$$

$$\dot{q} = \frac{1}{2}E^T\Omega, \qquad \dot{\Omega} = J^{-1}\left(T + \Omega \times J\Omega\right), \tag{23}$$

$$\dot{v} = m^{-1}RF - g\mathbf{1}_z, \tag{24}$$

$$F = \sum_{k=1}^{4}\frac{1}{2}\rho A C_{\text{L}}\omega_k^2\mathbf{1}_z, \quad T = \sum_{k=1}^{4}\frac{(-1)^k}{2}\rho A C_{\text{D}}\omega_k^2\mathbf{1}_z, \tag{25}$$

$$\dot{p} = v, \qquad \dot{I}_p = p - p_{\text{ref}}, \tag{26}$$

where U_k for $k = 1,\dots,4$ are the control inputs, commanding the 2nd time derivative of the propeller reference velocities ω_k^{ref}, ω_k are the actual propeller velocities, $q \in \mathbb{R}^4$ is the quaternion vector used to represent the orientation, Ω is the main body angular velocity in the body frame, v is the linear velocity in the inertial frame, p is the position and $I_{\text{p}} \in \mathbb{R}^3$ the integral of the position error. Matrix $R = EG^T$ is the rotation matrix between the body frame and the inertial frame, with

$$G = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix}, \; E = \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & q_3 & -q_2 \\ -q_3 & q_0 & q_1 \\ q_2 & -q_1 & q_0 \end{bmatrix}.$$

Parameter τ is the time constant of the speed control loops, $J \in \mathbb{R}^{3\times 3}$ is the inertia matrix of the Quadcopter, and m its total mass. Coefficients C_{L} and C_{D} are respectively the lift and drag coefficients, A is the individual area of the propellers, ρ is the air density, and $\mathbf{1}_z = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$.

It can be observed that (22) forms a 12 states linear input system, (23)-(24) forms a 10 states nonlinear dynamic system and (26) is a 6 states linear output system. The input dynamics $\ddot{\omega}_k^{\text{ref}} = U_k$ are used to implement a penalty on the high-frequency components in the control input, i.e. the Lagrange term

$$\Pi_{\text{input}} = \sum_{k=1}^{4} W_1 U_k^2 + W_2\left(\dot{\omega}_k^{\text{ref}}\right)^2$$

is added to the objective, with positive weights W_1 and W_2. Moreover, the integral of the position error $\dot{I}_p = p - p_{\text{ref}}$ is introduced and penalized in order to eliminate steady-state errors in the Quadcopter position, which typically result from constant disturbances and model errors.

The positive definite weighting matrices Q, R and P from the OCP formulation in (1) are chosen to achieve quick point-to-point motions of the quadcopter while taking into account its limitations. A possible closed-loop trajectory for the position and orientation of the quadcopter is presented in Figure 3, in the event of a motion from the point $(1,1,1)$ to the origin. It shows the position as well as the orientation of the quadcopter at multiple time instants, which are 0.4s apart from each other.

Table III shows a speedup factor of 12 for the total integration time. It even seems to cause condensing to become a more dominating computational cost in one real-time iteration. For these experiments, a horizon of length 5s with 25 control intervals has been used and 2 integration steps of the 6th order Gauss method are performed per interval.

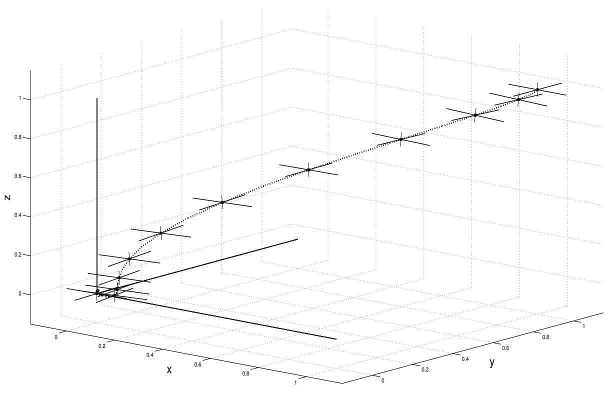

Fig. 3. Illustration of a closed-loop trajectory of the position and orientation of the quadcopter.

	unstructured	structured
integration method	**35.8 ms**	**3.1 ms**
condensing	2.6 ms	2.6 ms
QP solution (qpOASES)	0.1 ms	0.1 ms
remaining operations	0.2 ms	0.2 ms
one real-time iteration	38.7 ms	6.0 ms

TABLE III
AVERAGE COMPUTATION TIMES FOR NMPC OF A QUADCOPTER.

VI. CONCLUSIONS & FURTHER DEVELOPMENTS

This paper proposed a new format for defining nonlinear dynamic models and showed how the three-stage structure therein can be strongly exploited by IRK methods. The general idea of code generation for IRK methods with efficient generation of sensitivities and continuous output was first briefly repeated. The mentioned structure in the differential equations has then been introduced and motivated, followed by a discussion on the consequences for the implementation when exploiting this. Two problems, namely the real-time control of an overhead crane and of a quadcopter, illustrated the relevance of the proposed three-stage structure. Numerical experiments have shown that important speedups can be achieved with these auto generated, structure exploiting integrators which are available in the ACADO toolkit [20].

It is useful to note that the proposed three-stage structure could be detected and exploited automatically in any nonlinear model. Future work could include implementing such an automatic structure detection.

References

[1] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: stability and optimality," *Automatica*, vol. 26, no. 6, pp. 789–814, 2000.

[2] C. Rao and J. Rawlings, "Nonlinear Moving Horizon State Estimation," in *Nonlinear Predictive Control* (F. Allgöwer and A. Zheng, eds.), (Basel Boston Berlin), pp. 45–69, Birkhäuser, 2000.

[3] C. Jones and M. Morari, "Polytopic approximation of explicit model predictive controllers," *IEEE Transactions on Automatic Control*, vol. 55, no. 11, pp. 2542–2553, 2010.

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

[5] M. Diehl, H. J. Ferreau, and N. Haverbeke, *Nonlinear model predictive control*, vol. 384 of *Lecture Notes in Control and Information Sciences*, ch. Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation, pp. 391–417. Springer, 2009.

[6] M. Diehl, H. Bock, and J. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM Journal on Control and Optimization*, vol. 43, no. 5, pp. 1714–1736, 2005.

[7] J. Mattingley and S. Boyd, *Convex Optimization in Signal Processing and Communications*, ch. Automatic Code Generation for Real-Time Convex Optimization. Cambridge University Press, 2009.

[8] H. Ferreau, *Model Predictive Control Algorithms for Applications with Millisecond Timescales*. PhD thesis, K.U. Leuven, 2011.

[9] R. Quirynen, S. Gros, and M. Diehl, "Fast auto generated ACADO integrators and application to MHE with multi-rate measurements," in *Proceedings of the European Control Conference*, 2013.

[10] R. Quirynen, M. Vukov, and M. Diehl, "Auto Generation of Implicit Integrators for Embedded NMPC with Microsecond Sampling Times," in *Proceedings of the 4th IFAC Nonlinear Model Predictive Control Conference, Noordwijkerhout, The Netherlands* (M. Lazar and F. Allgower, eds.), vol. 4, 2012.

[11] M. Boutayeb and M. Darouach, "Recursive identification method for MISO Wiener-Hammerstein model," *Automatic Control, IEEE Transactions on*, vol. 40, no. 2, pp. 287–291, Feb 1995.

[12] R. Serban and A. Hindmarsh, "CVODES: the sensitivity-enabled ODE solver in SUNDIALS," in *Proceedings of IDETC/CIE 2005*, 2005.

[13] R. Quirynen, "Automatic code generation of implicit runge-kutta integrators with continuous output for fast embedded optimization," Master's thesis, KU Leuven, 2012.

[14] R. Findeisen and F. Allgöwer, "Nonlinear model predictive control for index–one DAE systems," in *Nonlinear Predictive Control* (F. Allgöwer and A. Zheng, eds.), (Basel Boston Berlin), pp. 145–162, Birkhäuser, 2000.

[15] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems*. Berlin Heidelberg: Springer, 2nd ed., 1991.

[16] A. Griewank and A. Walther, *Evaluating Derivatives*. SIAM, 2 ed., 2008.

[17] N. H. Cong and L. N. Xuan, "*Parallel-Iterated RK-type PC Methods with Continuous Output Formulas*," *International Journal of Computer Mathematics*, vol. 80:8, pp. 1025–1035, 2003.

[18] M. Vukov, W. V. Loock, B. Houska, H. Ferreau, J. Swevers, and M. Diehl, "Experimental Validation of Nonlinear MPC on an Overhead Crane using Automatic Code Generation," in *The 2012 American Control Conference, Montreal, Canada.*, 2012.

[19] G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin, "Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007.

[20] Sourceforge, "Acado toolkit." URL: `http://sourceforge.net/p/acado/wiki/Home/`, last checked on June 18, 2013.