High-Speed Moving Horizon Estimation based on Automatic Code Generation

Hans Joachim Ferreau, Tom Kraus, Milan Vukov, Wouter Saeys, Moritz Diehl

Abstract— **Recent theoretical and algorithmic advances have led to efficient algorithms that allow for real-time optimisation of processes with fast nonlinear dynamics. This paper addresses the efficient implementation of algorithms for moving horizon estimation (MHE) for obtaining real-time estimates of process states or parameters that are not measured directly. To this end, we propose to combine the previously proposed concepts of real-time iteration schemes and automatic code generation to obtain highly efficient source code of MHE algorithms. This has led to major extensions of the ACADO Code Generation tool that automatically generates customised plain C code for both model predictive control (MPC) and MHE applications. As a proof of concept, we present numerical results of controlling a nonlinear ODE model by means of combined exported MHE and MPC algorithms in a closed-loop manner. These exported algorithms turn out to be significantly faster than their generically implemented counterparts.**

I. INTRODUCTION

Feedback control strategies such as model predictive control (MPC) are typically designed under the assumption that all current process states and parameters are known. However, in most real applications it is either not possible to measure all these quantities directly or at least not desired for economic reasons. The task of recovering the full knowledge of all current process states and possibly some parameters is referred to as state and parameter estimation, respectively. Sometimes the attribute "online" is added to these terms in order to stress that estimation has to be performed in real-time, i.e. during the runtime of the process.

For linear systems, the classical Kalman filter [17], [13] turned out to be a powerful state estimation algorithm. In case the estimation problem is unconstrained and the error distribution of the estimated states is Gaussian, the Kalman filter is an exact maximum likelihood estimator. In order to avoid accumulation of numerical errors that would destroy this desirable property, so-called square-root Kalman filter variants have been proposed [24].

There exists a great variety of state estimation methods for nonlinear systems. Some of them are extensions of linear state estimators, such as the different variants of the extended Kalman filter (EKF) that linearise the nonlinear system at the current estimate and then apply the usual linear Kalman filter recursion [13], [3]. Unscented Kalman filters replace this simple linearisation by a more accurate approximation of the multidimensional probability density of the state vector, which is also assumed to be Gaussian [16]. In contrast, particle filter methods allow to approximate arbitrary probability densities that are integrated by means of Monte Carlo sampling [10]. Particle filters need to cover the relevant region of the state space and thus suffer from the "curse of dimensionality". Excellent surveys on nonlinear state estimation can be found in [8], [26], for example.

Most the above approaches at the exception of particle filters make explicit assumptions on the error distribution of the estimated states. Thus, they may loose their theoretical justification if either the actual distribution is different or if estimated quantities are subject to (physical) constraints. This is in contrast to estimating the states (or parameters) by simply fitting them to the measurements in a least-squares fashion. This least-squares estimation goes back to Gauss and is a "deterministic" approach in the sense that it does not rely on any specific error distribution (though this insight is very valuable when tuning the weighting matrices of the least-squares objective function). Moreover, incorporation of constraints is straight-forward.

In order to use all available information when estimating the process states, it would be desirable to use all past measurements within the estimation algorithm. However, the computational burden of such full-information estimators might quickly become intractable as the number of measurements grows. Thus, most estimators are based on recursive schemes that only use the most recent measurement explicitly, while information gathered through all earlier measurements is implicitly incorporated within updating rules. An important exception is the optimisation-based least-squares estimator. It allows one to deal with the growing amount of measurement data by only considering a "window" comprising a fixed number of past measurements, while incorporating information of previous measurements into some of the least-squares weighting matrices. This variant of least-squares estimation is usually referred to as moving horizon estimation (MHE), see for example [25].

This paper will focus on highly efficient implementations

H.J. Ferreau, M. Vukov, and M. Diehl are with the Electrical Engineering Department, KU Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium. Correspondence to: milan.vukov@esat.kuleuven.be

T. Kraus and W. Saeys are with the Division of Mechatronics, Biostatistics and Sensors (MeBioS), KU Leuven, Kasteelpark Arenberg 30, 3001 Leuven, Belgium.

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. Financial support of the Research Foundation – Flanders (FWO) is acknowledged.

of such nonlinear MHE algorithms that are also suited to run on embedded hardware. The question whether or not MHE is the most suited estimation algorithm for a specific application is beyond the scope of this paper.

A. Real-Time Iterations and Code Generation

Moving horizon estimation for processes described by nonlinear dynamic equations requires the online solution of a (possibly constrained) nonlinear least-squares optimisation problem at each estimation instant. Such problems can be efficiently solved by a generalised Gauss-Newton method [4], which is a tailored variant of the classical Newton method. When using this method online, one faces a dilemma: it would be ideal to solve to this problem exactly in no time to have an optimal estimate available instantaneously. But on the other hand the Gauss-Newton method might need several iterations to find a (locally) optimal solution and each iteration increases the time delay. In [9] the so-called real-time iteration (RTI) scheme has been proposed to tackle this issue. It only performs a single Gauss-Newton iteration per sampling instant in order to quickly deliver an approximate solution to the optimisation problem. A clever initialisation of each optimisation improves convergence of the Gauss-Newton method. The RTI scheme has been initially developed for solving nonlinear MPC problems using a shooting method for discretization [5]. It has been adapted recently to yield a real-time MHE algorithm [18], [19].

The RTI scheme also divides the computational load of each Gauss-Newton iteration into a short estimation phase and a typically much longer preparation phase. Once a new measurement arrives, only the estimation phase needs to be performed to obtain the current state or parameter estimates. This further reduces the time delay between obtaining a new measurement and calculating an approximately optimal solution to the nonlinear least-squares problem.

A similar approximative approach using simultaneous collocation has been presented in [30]. It proposes to iterate each MHE problem till convergence during the preparation step and corrects for disturbances during the feedback step. In [2] a moving horizon estimation approach is presented without focusing on the numerical solution algorithm.

Another recent approach for speeding-up online computations in (nonlinear) MPC is the use of automatic code generation [23], [21], [15]. The `ACADO Code Generation` tool [15] allows the user to export a tailored real-time MPC algorithm that contains only the absolutely essential algorithmic components. Based on a symbolic representation of the MPC problem to be solved online, problem-specific structure such as dimensions, sparsity patterns etc. is exploited during the code generation process to avoid irrelevant computations.

B. Outline of the Paper

The paper is organised as follows: Section II introduces the MHE problem formulation to be used throughout the paper and summarises the adapted RTI scheme to be used within the exported MHE algorithm. Section III presents the `ACADO Code Generation` tool and explains in detail all algorithmic extensions that have been developed to automatically generate customised real-time MHE algorithms. Such an exported MHE algorithm is then used in Section IV for state and parameter estimation of a nonlinear tractor model. Closed-loop simulation results of controlling the tractor model by means of combined exported MHE/MPC algorithms are presented. Section V concludes the paper and gives directions of future research.

II. REAL-TIME MOVING HORIZON ESTIMATION

A. Problem Formulation

Throughout this paper we assume that the process whose states or unknown parameters are to be estimated is described by a dynamic model of the following form:

$$\begin{aligned} x(T_0) &= x_0\,, && \text{(1a)}\\ \dot{x}(t) &= f(x(t),u(t),p) \quad \forall\, t\in\mathcal{T}:=[T_0,\infty)\,, && \text{(1b)}\\ y(t) &= h(x(t),u(t),p) \quad \forall\, t\in\mathcal{T}\,, && \text{(1c)} \end{aligned}$$

with an unknown initial value $x_0\in\mathbb{R}^{n_x}$, differential states $x:\,\mathcal{T}\to\mathbb{R}^{n_x}$, unknown inputs $u:\,\mathcal{T}\to\mathbb{R}^{n_u}$, parameters $p\in\mathbb{R}^{n_p}$, and process outputs $y:\,\mathcal{T}\to\mathbb{R}^{n_y}$. We assume that measurements are made at discrete time instants

$$T_0<T_1<T_2<\dots$$

Consequently, we are only interested in the solution of the ordinary differential equation (ODE) (1) at these discrete-time instants:

$$\begin{aligned} x(T_0) &= x_0\,, && \text{(2a)}\\ x(T_{k+1}) &= F(x(T_k),u(T_k),p) \quad \forall\, k\geq 0\,, && \text{(2b)}\\ y(T_k) &= h(x(T_k),u(T_k),p) \quad \forall\, k\geq 0\,, && \text{(2c)} \end{aligned}$$

where F is either determined analytically or implicitly by means of numerical integration[1]. Moreover, we assume that the unknown inputs are piecewise-constant over the time intervals $[T_k,T_{k+1})$. Finally, we define $x_k:=x(T_k)$, $u_k:=u(T_k)$, $y_k:=y(T_k)$ etc. for ease of notation.

As mentioned in Section I, a moving horizon estimator does not consider all past measurements explicitly, but only minimises the misfit of model prediction and measurements on a fixed horizon of length $M>0$. This can be formulated in form of a constrained, nonlinear least-squares optimisation problem. The MHE problem at measurement instant T_K looks as follows:

$$\begin{aligned} \min_{x_k,u_k,p}\quad & \left\| \begin{array}{c} x_L-x_L^{\text{est}} \\ p-p_L^{\text{est}} \end{array} \right\|_{P_L}^2 \\ & +\sum_{k=L}^{K}\left\|\tilde{y}_k-h(x_k,u_k,p)\right\|_{V_K}^2+\sum_{k=L}^{K-1}\left\|\tilde{u}_k-u_k\right\|_{W_K}^2 && \text{(3a)}\\ \text{s.t.}\quad & x_{k+1}=F\big(x_k,u_k,p\big) \quad \forall\, L\leq k\leq K-1, && \text{(3b)}\\ & x_k^{\text{lower}}\leq x_k\leq x_k^{\text{upper}} \quad \forall\, L\leq k\leq K, && \text{(3c)}\\ & u_k^{\text{lower}}\leq u_k\leq u_k^{\text{upper}} \quad \forall\, L\leq k\leq K, && \text{(3d)}\\ & p^{\text{lower}}\leq p\leq p^{\text{upper}}, && \text{(3e)} \end{aligned}$$

[1]The `ACADO Code Generation` tool discretises the ODE by means of a multiple (or single) shooting method, see Section III.

where $\tilde{y}_k$ and $\tilde{u}_k$ denote the actually measured process outputs and inputs at time T_k, respectively, and $L := K - M + 1$ is defined for notational convenience. All estimated quantities might be subject to lower or upper limits reflecting process insight or possible physical limitations. Deviations of the oldest states considered within the moving horizon window and the parameters from some a priori estimates x_L^{est}, p_L^{est} are penalised by a symmetric positive definite matrix P_L. Additionally, deviations of the predicted process outputs (2c) and the actually measured outputs are penalised by a symmetric positive semi-definite weighting matrix V_K. Also deviations of the unknown inputs from $\tilde{u}_k$ are penalised by a symmetric positive definite weighting matrix W_K. Note that the dependence of F from state noise terms is omitted here but could be handled within the proposed framework.

B. Choice of Weighting Matrices and Arrival Cost Updates

Within formulation (3), it mostly depends on the choice of the weighting matrices P_L, V_k and W_k which estimates are considered optimal. In order to choose them properly, it is common practice to rely on the assumption that the measured quantities exhibit Gaussian distributions. Let us assume that the initial value x_L and the parameter p are normally distributed random variables with covariance matrices Q_L^x and Q_L^p and mean values x_L^{est} and p_L^{est}, respectively. Let also the measured outputs $\tilde{y}_k$ and unknown inputs $\tilde{u}_k$ be Gaussian with mean value y_k and u_k as well as covariance matrices R_k^y and R_k^u, respectively. Then, it is well-known that (3) delivers maximum-likelihood estimates for the true trajectories of the current window if (i) P_L is chosen as block-diagonal matrix with the two blocks $(Q_L^x)^{-\frac{1}{2}}$ and $(Q_L^p)^{-\frac{1}{2}}$ on the diagonal, (ii) V_k and W_k are chosen as $(R_k^y)^{-\frac{1}{2}}$ and $(R_k^u)^{-\frac{1}{2}}$, respectively, and (iii) no constraints are present. This result has been extended in [27] to the case where constraints on the estimated quantities are present.

The first term in the objective function (3a) is commonly referred to as arrival cost and summarises all information gathered through measurements before time T_L. The reference estimates x_L^{est} and p_L^{est} are typically taken from the solution of the MHE problem at the previous estimation instant. The arrival cost matrix P_L can be chosen in different ways: A constant zero matrix has been proposed in [22].Alternatively, [27] proposed a so-called smoothed EKF-update based on sensitivity information obtained while solving the previous MHE problem, which is also what our MHE algorithm uses. It has been shown that the classical EKF is equivalent to MHE using smoothed EKF-updates and a horizon of length one [27].

C. Real-Time Iterations for MHE

The nonlinear RTI scheme, as originally proposed in [9] for use in nonlinear MPC, has been recently adapted for solving nonlinear MHE problems [18], [19]. It is a direct method and thus first discretises the MHE problem to arrive at formulation (3) by means of a multiple shooting method [5]. This finite-dimensional least-squares nonlinear programming (NLP) problem is then solved using a generalised Gauss-Newton method, which solves one quadratic programming (QP) problem at each iteration [4]. This allows for multiple active-set changes and thus ensures that the nonlinear MHE algorithm cannot perform worse than a MHE estimator based on a linear ODE model.

Real-Time Iterations for Nonlinear MHE:

Initialisation: Choose initial values for x and u at all shooting nodes as well as for p and initialise EKF-updates.

Repeat Online:

1a) Update arrival cost data x_L^{est}, p_L^{est}, and P_L based on a smoothed EKF-update.

1b) Evaluate objective function and ODE equations and their derivatives at current NLP iterate.

1c) Condense resulting large-scale QP into a smaller-scale dense QP (obtain all QP data except its gradient vector).

1d) Wait for the measurement $\tilde{y}_K$.

2a) Use measurement $\tilde{y}_K$ and calculate gradient vector of dense QP problem.

2b) Solve dense QP problem.

2c) Expand solution of dense QP to obtain current estimates.

2d) Update current iterates and possibly shift them in time.

Alg. 1. Summary of the nonlinear RTI scheme based on a multiple shooting discretisation (similar to [19]).

The QP problem resulting from linearising problem (3) has a special block-sparse structure that can be exploited to speed-up its solution. This sparsity might either be used directly within a sparse QP solver. Alternatively, all but the first state variables can be easily eliminated from the QP formulation in order to arrive at a smaller-scale but dense "condensed" QP problem.

The main steps of the nonlinear RTI scheme for MHE are summarised in Alg. 1. The RTI scheme only performs one Gauss-Newton iteration per estimation instant. When doing so, one observes that most computations of this iteration can be prepared before the most current measurement $\tilde{y}_K$ is available. Once the new measurement arrives, only a small-scale QP problem needs to be solved in a so-called estimation phase in order to obtain the current state or parameter estimates.

III. ACADO CODE GENERATION TOOL

A. Code Generation for MPC

The idea to speed-up online computations by means of automatic code generation has been introduced about ten years ago in the context of MPC. The software tool `AutoGenU` by Ohtsuka and Kodama provides `Mathematica` scripts to generate customised C-code implementing the so-called Continuation/GMRES method [23]. At each sampling instant their algorithm requires the online solution of just one system of linear equations [28]. Recently, code generation has attracted great attention due to the software package `CVXGEN` [21], which allows the user to generate customised

interior-point solvers for small-scale LP and convex QP problems. By doing so, linear MPC problems can be solved based on highly efficient, auto-generated code.

The recently introduced ACADO Code Generation tool extends these ideas and allows the user to export a generalised Gauss-Newton algorithm for solving nonlinear MPC problems [15]. The exported C-code implements the real-time iteration scheme for nonlinear MPC as described in [9]: at each sampling instant it performs a system linearisation and solves the resulting convex QP online.

The ACADO Code Generation tool is part of the open-source software package ACADO Toolkit for automatic control and dynamic optimisation [14]. Based on a symbolic representation of the MPC problem to be solved online, problem-specific structure such as dimensions, sparsity patterns etc. is exploited during the code generation process to avoid irrelevant or redundant computations. Auto-generated nonlinear MPC algorithms have been found to outperform their counterparts implemented in a generic way by up to a factor of 20 on small-scale examples [12]. A first experimental validation of an auto-generated real-time MPC algorithm is described in [29]. More details on the ACADO Code Generation tool can be found in [1].

B. An Auto-Generated Real-Time MHE Algorithm

The newly developed extension to the ACADO Code Generation tool allows the user to export highly efficient nonlinear real-time MHE algorithms. The exported code is customised based on a symbolic representation of the MHE problem formulation (3), but with a continuous-time ODE formulation (1). The corresponding interface has been written in C++ and we refer to [14], [15] for more details.

Once a specific MHE problem has been set up in the ACADO syntax, a dedicated MHEexport class generates customised C-code comprising five major optimised functions:

1) The typically nonlinear right-hand side ODE function f in Eq. (1) is exported as a plain C-function together with its symbolically derived associated variational differential equation (VDE). Integrating this VDE delivers exact first-order derivatives of f with respect to the states, controls and parameters [7]. The VDE is symbolically simplified employing automatic differentiation tools and exploiting zero-entries in the Jacobian of f.
2) The output function h in Eq. (1) and its Jacobian with respect to all arguments are exported as plain C-functions.
3) A multiple shooting discretisation algorithm is exported. For integrating the dynamic model equations along with the associated VDEs, the shooting algorithm makes use of an auto-generated explicit Runge-Kutta integrator. Currently, only fixed-step methods are provided, which means that the user needs to manually tune the constant number of integrator steps in offline simulations. On the other hand, this fact guarantees a deterministic run-time of the integration routine.
4) The source code of a customised condensing routine for eliminating all but the initial state x_L from the linearised MHE problem is generated. This routine differs from its MPC counterpart in two important respects: (i) the initial state is kept within the condensed QP, (ii) the least-squares weighting matrices typically vary from one MHE problem to the next and thus can be adapted online.
5) The nonlinear RTI Gauss-Newton method for MHE as described in Section II-C is exported and makes use of all the above-mentioned algorithmic parts. Moreover, it employs a tailored algorithm for solving the underlying small-scale, dense QP problems. Interfaces to an embedded variant of the online QP solver qpOASES [11] or to QP solvers generated by CVXGEN [21] can be exported.

The ACADO Code Generation tool generates self-contained C-code, which facilitates its use on embedded hardware. Branching in the exported code is minimized and loops are unrolled whenever reasonable in order to increase computational efficiency.

The exported code provides two main functions to be called by the user: preparationStep() and estimationStep(real_t* y) that collect all online computations within the preparation or estimation phase, corresponding to step 1) and 2) in Alg. 1, respectively. The estimation phase function can be called once the most recent measurement is available.

C. New Features and Software Design

In addition to the MHE extensions, the software design of the ACADO Code Generation tool as first presented in [15] has been recently improved and the following algorithmic features have been added:

- handling of parameters,
- handling of free initial values,
- online adaptation of weighting matrices,
- Levenberg-Marquardt regularisation.

Moreover, the ACADO Code Generation tool now also offers to auto-generate an interface to Simulink and to adapt the code to different operating systems.

Finally, the software design now encapsulates the export of all kinds of matrix-vector operations. Thus, their memory management and strategies such as loop unrolling can be tuned centrally. Moreover, the setup of new C-functions or variables to be exported has been made more flexible.

IV. SIMULATION STUDY

To illustrate the performance of the described MHE algorithm, a closed-loop simulation of an autonomous, MPC-controlled tractor was set up. The objective is to follow a user-defined trajectory with high precision at a certain desired ground speed on straight strips while facing difficult and varying soil conditions. This control problem occurs e.g. in industrial agriculture when a desired amount of pesticide is to be sprayed over a given area per time unit while slope

or moisture make operation difficult and off-track navigation results in plant damage.

A. Vehicle Model

The kinematic ODE model is a slight adaptation of the well-known bicycle model. It is known to yield good results when working with slow speeds [6], [20]. The equations are the following:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\beta} \\ \dot{\alpha} \end{bmatrix} = \begin{bmatrix} p_t u_1 \cos\beta \\ p_t u_1 \sin\beta \\ p_t u_1 \frac{\tan\alpha}{l} \\ u_2 \end{bmatrix} \tag{4}$$

Here, x and y represent the vehicle position coordinates, β is the global orientation and α the steering angle. The differential states x, y and α are assumed to be measured. The control variables are the wheel speed u_1 and the steering rate u_2 and since we cannot assume to have perfect actuation, they are also measured.
Additionally, one slowly-changing model parameter being relevant for control cannot be measured directly: the traction factor p_t, to relate wheel speed with ground speed. To avoid bias, this parameter should be estimated along with the full system state in each iteration. The parameter $l = 1.4\,\mathrm{m}$ represents the base of the wheel and is purely geometry-dependent.

B. Simulation Scenario

In this scenario, the vehicle is supposed to ride on a predefined trajectory given as a time-based reference, while facing varying soil conditions. These are reflected by jumps of the model parameter p_t in the "true" model and have to be identified by the MHE algorithms along with the full system state at a sampling time of 200 ms. The measured states are x and y position coordinates and steering angle. These measurements are perturbed by significant Gaussian noise with standard deviations of $\sigma_x = (0.05\,\mathrm{m}, 0.05\,\mathrm{m}, 0.1745\,\mathrm{rad})$. The controls wheel speed and steering angle change are perturbed by a Gaussian noise with $\sigma_u = (0.1\,\mathrm{m/s}, 0.1\,\mathrm{rad/s})$. It is assumed that the error distributions are known to the estimator. The horizon length M is set to 20 for both the MHE and the MPC algorithm.
Even though this model has been used for online estimation with more traditional methods like Kalman filtering before, the presence of the model parameter which has to be identified alongside increases the nonlinearity of the system and makes it a interesting estimation problem despite its relatively small size.

C. Simulation Results

Despite the varying soil conditions and significant measurement errors, the trajectory is followed nicely by the MHE-MPC scheme (see Fig. 2). The filtering of further measurement inputs by the MHE is shown in Fig. 3 and the performance of the MHE state and parameter estimation results are shown in Fig. 4 for the given estimation scenario. In the latter, the graphs compare estimated and true values of

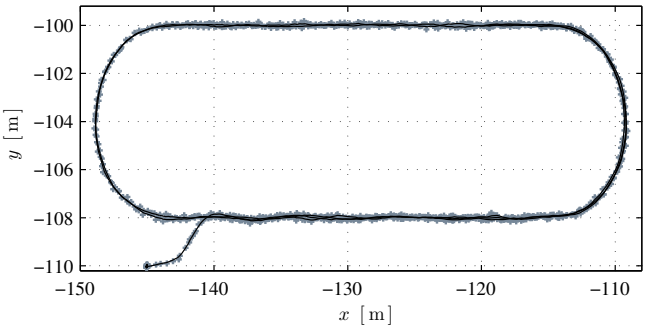

Fig. 2. Positions of the tractor x and y; black line: tracked trajectory, gray markers: measurements.

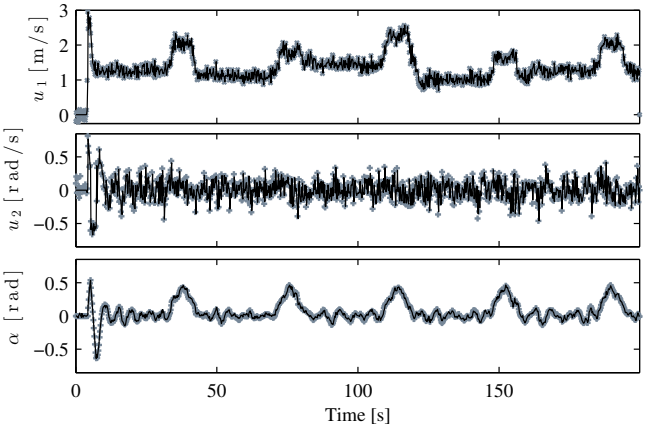

Fig. 3. The MPC outputs u_1 and u_2, and the steering angle α; black lines: estimates, gray markers: measurements.

the unmeasured state β and the parameter p_t. The parameter jumps are detected each time and the estimator arrives at the true values after only a few iterations. Note that the values of the differential states and parameters used to initialise the estimator were all disturbed. Despite the relatively large noise level, the MHE tracks the true states well. This demonstrates its good filtering properties. Therefore, the given estimator performance, enables a good performance of the MPC controller by providing accurate estimates.

In the given simulations one real-time iteration is performed for the MHE as well as for the MPC step, which both consist of a typically much longer preparation and a shorter estimation/feedback phase. Table I summarises computations times obtained on a standard PC (Intel Core 2 Quad Q9650) with 3.0 GHz processors. The average as well as maximum runtimes of both the preparation as well as the estimation/feedback phase are given. It does not come at a surprise that runtimes of both the exported MHE and MHE algorithm are quite similar as both solve optimisation problems of similar size. Note that employing these combined MHE/MPC real-time algorithms always allowed one to give optimised feedback to the tractor less than 1 ms after the measurements have been obtained.

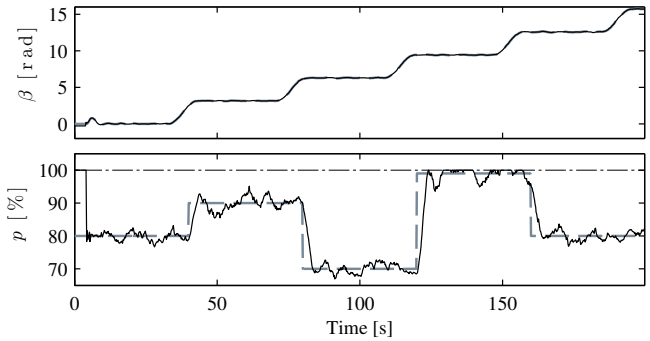

Fig. 4. Estimation of the unmeasured state β and the unmeasured parameter p; black lines: estimates, dashed gray lines: true values, dashed-dotted black line: constraint.

TABLE I

RUNTIMES OF THE AUTO-GENERATED MHE/MPC ALGORITHMS.

		Average	Maximum
MHE	Preparation phase	0.62 ms	0.62 ms
	Estimation phase	0.10 ms	0.42 ms
	Overall execution time	0.72 ms	1.04 ms
MPC	Preparation phase	0.61 ms	0.61 ms
	Feedback phase	0.06 ms	0.42 ms
	Overall execution time	0.67 ms	1.03 ms

V. CONCLUSIONS

We have presented a novel major extension to the open-source `ACADO Code Generation` tool that allows the user to export customised real-time iteration MHE algorithms. This tool automatically generates highly efficient, self-contained C-code based on a user-specified symbolic MHE problem formulation. As a proof of concept, closed-loop simulations for a small-scale nonlinear tractor model were performed. In these simulations auto-generated implementations of both the moving horizon estimator as well as the model predictive controller have been used and exhibited fast computation times.

Future work will focus on further extensions of the `ACADO Code Generation` tool. Namely, the ability to handle processes modelled by differential algebraic equations is currently under development. Moreover, the auto-generated MHE/MPC algorithms for the presented tractor model will be used to control a vehicle using embedded hardware. Preliminary tests on the real vehicle have already been performed.

REFERENCES

[1] ACADO Toolkit Homepage. http://www.acadotoolkit.org, 2009–2011.

[2] A. Alessandri, M. Baglietto, and G. Battistelli. Moving-horizon state estimation for nonlinear discrete-time systems: New stability results and approximation schemes. *Automatica*, 44(7):1753 – 1765, 2008.

[3] V.M. Becerra, P.D. Roberts, and G.W. Griffiths. Applying the extended Kalman Filter to systems described by nonlinear di!erential-algebraic equations. *Control Engineering Practice*, 9:267–281, 2001.

[4] H.G. Bock. Recent advances in parameter identification techniques for ODE. In P. Deuflhard and E. Hairer, editors, *Numerical Treatment of Inverse Problems in Differential and Integral Equations*. Birkhäuser, Boston, 1983.

[5] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings 9th IFAC World Congress Budapest*, pages 243–247. Pergamon Press, 1984.

[6] S. Cameron and P. Probert. *Advanced Guided Vehicles*. World Scientific, Singapore, 1994.

[7] E.A. Coddington and N. Levinson. *Theory of Ordinary Differential Equations*. McGraw-Hill, New York, 1955.

[8] F. Daum. Nonlinear Filters: Beyond the Kalman Filter. *IEEE A & E Systems Magazine*, 20(8):57–69, 2005.

[9] M. Diehl, H.G. Bock, J.P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and Nonlinear Model Predictive Control of Processes governed by differential-algebraic equations. *J. Proc. Contr.*, 12(4):577–585, 2002.

[10] A. Doucet and N. De Freitas ans N.J. Gordon. *Sequential Monte Carlo methods in practice*. Springer, 2001.

[11] H. J. Ferreau, H. G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, 2008.

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

[13] A. Gelb. *Applied Optimal Estimation*. MIT Press, 1974.

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

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

[16] S.J. Julier, J.K. Uhlmann, and H.F. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45:477–482, 2000.

[17] R.E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82:35–45, 1960.

[18] T. Kraus, P. Kühl, L. Wirsching, H. G. Bock, and M. Diehl. A Moving Horizon State Estimation algorithm applied to the Tennessee Eastman Benchmark Process. In *Proc. of IEEE Robotics and Automation Society conference on Multisensor Fusion and Integration for Intelligent Systems*, 2006.

[19] Peter Kühl, Moritz Diehl, Tom Kraus, Johannes P. Schlöder, and Hans Georg Bock. A real-time algorithm for moving horizon state and parameter estimation. *Computers & Chemical Engineering*, 35(1):71–83, January 2011.

[20] S.M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

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

[22] H. Michalska and D. Q. Mayne. Moving horizon observers and observer-based control. *IEEE Transactions on Automatic Control*, 40(6):995–1006, 1995.

[23] T. Ohtsuka and A. Kodama. Automatic code generation system for nonlinear receding horizon control. *Transactions of the Society of Instrument and Control Engineers*, 38(7):617–623, 2002.

[24] P.G. Park and T. Kailath. New Square-Root Algorithms for Kalman Filtering. *IEEE Transactions on Automatic Control*, 40(5):895–899, 1995.

[25] C.V. Rao and J.B. Rawlings. Nonlinear Moving Horizon State Estimation. In F. Allgöwer and A. Zheng, editors, *Nonlinear Predictive Control*, pages 45–69, Basel Boston Berlin, 2000. Birkhäuser.

[26] J.B. Rawlings and B.R. Bakshi. Particle filtering and moving horizon estimation. *Computers and Chemical Engineering*, 30:1529–1541, 2006.

[27] D. Robertson. *Development and Statistical Interpretation of Tools for Nonlinear Estimation*. PhD thesis, Auburn University, 1996.

[28] H. Seguchi and T. Ohtsuka. Nonlinear Receding Horizon Control of an Underactuated Hovercraft. *International Journal of Robust and Nonlinear Control*, 13(3–4):381–398, 2003.

[29] M. Vukov, W. Van Loock, B. Houska, H.J. 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.

[30] V.M. Zavala, C.D. Laird, and L.T. Biegler. A Fast Moving Horizon Estimation Algorithm Based on Nonlinear Programming Sensitivity. *Journal of Process Control*, 18(9):876 – 884, 2008.