Auto-generated Algorithms for Nonlinear Model Predictive Control on Long and on Short Horizons

Milan Vukov  Alexander Domahidi  Hans Joachim Ferreau  Manfred Morari  Moritz Diehl

Abstract—We present a code generation strategy for handling long prediction horizons in the context of real-time nonlinear model predictive control (NMPC). Existing implementations of fast NMPC algorithms use the real-time iteration (RTI) scheme and a condensing technique to reduce the number of optimization variables. Condensing results in a much smaller, but dense quadratic program (QP) to be solved at every time step. While this approach is well suited for short horizons, it leads to unnecessarily long execution times for problem formulations with long horizons. This paper presents a new implementation of auto-generated NMPC code based on a structure exploiting auto-generated QP solver. Utilizing such a QP solver, the condensing step can be avoided and execution times scale linearly with the horizon length instead of cubically. Our simulation results show that this approach significantly decreases the execution times of NMPC with long horizons. For a nonlinear test problem that comprises 9 states and 3 controls on a horizon with 50 time steps, an improvement by a factor of 2 was observed, reducing the execution time for one RTI to below 4 milliseconds on a 3 GHz CPU.

I. INTRODUCTION

Model predictive control (MPC) has been originally designed and used for control of large-scale processes, typically in the chemical and petroleum industry. Slow dynamics of those systems allowed large control intervals measured in tens of seconds or even hours, leaving enough time to compute a solution to the underlying optimization problem. However, progress in the area of optimization algorithms and computational hardware in the last two decades have extended the applicability of numerical optimization on embedded platforms. These developments made MPC suitable for control of fast dynamical systems with time constants in micro- and millisecond range.

Milan Vukov and Moritz Diehl are with the Department of Electrical Engineering, KU Leuven, Kasteelpark Arenberg 10, 3001 Leuven, Belgium. Alexander Domahidi and Manfred Morari are with the Automatic Control Laboratory, Department of Information Technology and Electrical Engineering, ETH Zurich, 8092 Zurich, Switzerland. Hans Joachim Ferreau is with the ABB Corporate Research, Baden-Dättwil, Switzerland.

Contact: milan.vukov@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 (DYSO, Dynamical systems, control and optimization, 2012-2017); EU: FP7-EMBOCON (ICT-240890), FP7-SADCO (MC ITN-264735), ERC ST HIGHLWind (259 166), Eurostars SMART, ACCM.

The authors would like to thank Janick Frasch and Timon Achtnich for helpful discussions during the preparation of this paper.

Nonlinear model predictive control (NMPC) allows one to apply MPC to nonlinear dynamical systems [3], [4]. The continuation/GMRES method [19], the advanced step NMPC controller [28] and the real-time iteration (RTI) scheme [7], [8] represent efficient algorithms for fast NMPC. The reader is referred to [9] for a more detailed overview.

Automatic code generation for convex optimization solvers became very popular recently. The basic idea of code generation is to exploit the fact that all problem dimensions and structure is known a priori. This allows one to generate highly customized and fast solver code, which then solves instances of a specific problem. The code generation framework CVXGEN [17] was the first implementation that allowed one to export tailored interior-point (IP) solvers that can solve quadratic programs (QPs) very fast on embedded platforms. Following the observation that linear MPC problems fall into the class of multistage problems with particular structure [11], FORCES has been designed to overcome the limitations of CVXGEN in terms of problem size. The FORCES code generation framework [10] exports primal-dual interior point codes.

In a similar fashion, nonlinear MPC (NMPC) relevant dimensions are also known before deployment of a controller and highly efficient code can be generated to speed up computations. To our knowledge, the existing implementations of NMPC code generators are AutoGenU [23] and the ACADO Code Generation tool [15]. The ACADO Code Generation tool exports efficient self-contained C-code that implements the RTI scheme. The algorithm implements an sequential quadratic programming (SQP) framework and consists of simulation and linearization of the nonlinear system, and solution of a convex subproblem – a QP. The RTI scheme [8] performs a single SQP iteration per sampling instant in order to quickly deliver an approximate solution to the optimization problem. An experimental verification of the exported NMPC code is presented in [25], showing execution times below 1.1 milliseconds per RTI step for a nontrivial dynamic model.

The original implementation of the code generator for nonlinear MPC within the ACADO toolkit utilizes the embedded version of the QP solver qpOASES [1]. The QP solver is based on an (online) active set strategy [12] and employs dense linear algebra routines. To obtain feedback times in the milliseconds range, a technique called condensing [6] is applied, which significantly reduces the number of optimization variables in the QP. However, both the condensing procedure and an active-set QP solver have cubic complexity in number of prediction intervals, leading to prohibitively
long execution times when long prediction horizons are needed.

In this paper we present an extension of the ACADO Code Generation tool for NMPC that can efficiently handle long prediction horizons. In particular, we present a variant of an auto-generated NMPC that is utilizing the structure exploiting QP solver FORCES [11], [10]. Employing FORCES bypasses usage of the condensing procedure, thus saves a significant amount of computational time for long prediction horizons. The new implementation of the NMPC is tested against the original, condensing based, NMPC on a scalable benchmark.

The paper is organized as follows. In Section II, we introduce the problem formulation of the nonlinear MPC purpose of this paper is to explore at which horizon length an auto-generated NMPC that is utilizing the structure exploited QP solver FORCES [11], [10].

Furthermore, in order to ensure continuity of the optimal state trajectory on the whole horizon matching constraints are added to the NLP:

\[ s_{k+1} = F_k(s_k, u_k), \quad k = 0, \ldots, N - 1, \tag{3} \]

where \( F_k \) represents the solution of the initial value problem (IVP) \((2)\) at time \( t = t_{k+1}. \) The objective and path constraints are discretized at the same grid as the states and controls.

The NLP that comes as a result of the multiple shooting discretization reads:

\[
\begin{align*}
\min_{u, s} & \; \frac{1}{2} \sum_{k=0}^{N-1} \left( ||s_k - x_k^\text{ref}||_Q^2 + ||u_k - u_k^\text{ref}||_R^2 \right) + ||s_N - x_N^\text{ref}||_P^2 \\
\text{s.t.} & \quad s_0 = \hat{x}_0, \\
& \quad s_{k+1} = F(t_{k+1}; t_k, s_k, u_k), \quad k = 0, \ldots, N - 1, \\
& \quad u \leq u_k \leq \overline{u}, \quad k = 0, \ldots, N - 1, \\
& \quad \underline{x} \leq s_k \leq \overline{x}, \quad k = 1, \ldots, N, \tag{4a}
\end{align*}
\]

\[
\begin{align*}
\min_{\Delta u, \Delta s} & \; \frac{1}{2} \sum_{k=0}^{N-1} \left[ \Delta s_k \right]^T \begin{bmatrix} Q & R \\ R^T & \Delta_u \end{bmatrix} \begin{bmatrix} \Delta s_k \\ \Delta u_k \end{bmatrix} + \begin{bmatrix} g_s^T \\ g_u^T \end{bmatrix} \Delta u_k \tag{6a}
\end{align*}
\]

\[
\begin{align*}
\text{s.t.} & \quad \Delta s_0 = \hat{x}_0 - s_0, \\
& \quad \Delta s_{k+1} = a_k + A_k \Delta s_k + B_k \Delta u_k, \quad k = 0, \ldots, N - 1, \\
& \quad u - u_k \leq \Delta u_k \leq \overline{u} - u_k, \quad k = 0, \ldots, N - 1, \\
& \quad \underline{x} - s_k \leq \Delta s_k \leq \overline{x} - s_k, \quad k = 1, \ldots, N, \tag{6b}
\end{align*}
\]
with
\[ a_k = F_k(s_k, u_k) - s_{k+1}, \quad A_k = \frac{\partial F_k}{\partial s_k}(s_k, u_k), \quad B_k = \frac{\partial F_k}{\partial u_k}(s_k, u_k), \]
(7a)
\[ g_k^e = Q(s_k - s_k^{\text{ref}}), \quad g_k^u = R(u_k - u_k^{\text{ref}}). \]  
(7b)

The RTI scheme [8] uses the control and state variables of the previous optimization run, possibly after a shift, as new linearization point, and performs only one Newton-type iteration per sampling instant. Due to the fact that the initial value enters the problem linearly, it can be shown to deliver a generalized tangential predictor to perturbations, and nominal convergence of the resulting NMPC loop can be proven. We refer to [9] and the references therein for a detailed survey on the RTI and other algorithms for nonlinear MPC.

C. Condensing procedure

The goal of the condensing procedure [6] is to reduce the number of optimization variables in the QP (6). The system of linearized dynamic equations (6c) can be regarded as an affine time variant dynamic system with steps \( \Delta s_0 \). Assuming \( \Delta s_0 \) and the variables \( \Delta w_1 = (\Delta u_0, \ldots, \Delta u_{N-1}) \) are known, the variables \( \Delta w_2 = (\Delta s_1, \ldots, \Delta s_N) \) can be obtained by a forward simulation. Thus, the resulting map is also affine:
\[ \Delta w_2 = d + C \Delta s_0 + E \Delta w_1, \]  
(8)
Here, matrices \( C \in \mathbb{R}^{(N-n_t) \times n_t} \) and \( E \in \mathbb{R}^{(N-n_u) \times (N-n_u)} \) have a special block structure:
\[ C = \begin{bmatrix} C_0 \\ C_1 \\ \vdots \\ C_{N-1} \end{bmatrix}, \quad E = \begin{bmatrix} E_{0,0} & E_{0,1} & \cdots & E_{0,N-1} \\ E_{1,0} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ E_{N-1,0} & \cdots & E_{N-1,N-1} \end{bmatrix}, \]
(9)
where the corresponding blocks are computed as:
\[ C_k = A_k \cdots A_0, \quad k = 0, \ldots, N-1; \quad E_{k,m} = A_k E_{k-1,m}, \quad k = 1, \ldots, N-1, \; m = 0, \ldots, k-1; \quad E_{k,k} = B_k, \quad k = 0, \ldots, N-1. \]
(10)

Vector \( d \) can be computed by a forward simulation of the affine dynamic system (6c) with \( \Delta s_0 = 0 \) and \( \Delta w_1 = 0 \). This yields a recursion:
\[ d_0 = a_0, \quad d_k = a_k + A_k d_{k-1}, \quad k = 1, \ldots, N-1 \]  
(11)

Applying the condensing procedure to the original sparse QP (6), an equivalent, condensed, QP is obtained as:
\[ \min_{\Delta w_1} \frac{1}{2} \Delta w_1^T H_c \Delta w_1 + \Delta w_1^T g_c \]  
(12a)
\[ \text{s.t.} \quad \bar{w}_1 - w_1 \leq \Delta w_1 \leq \bar{w}_1 - w_1 \]  
(12b)
\[ w_2 - \bar{w}_2 \leq E \Delta w_1 \leq w_2 - \bar{w}_2 \]  
(12c)

with \( \bar{w}_2 = w_2 + d + C \Delta s_0 \). The Hessian and the linear term of the condensed QP are obtained as:
\[ H_c = E^T \bar{Q} E + \bar{R}, \]  
(13a)
\[ g_c = E^T \bar{Q} d + E^T g^s + g^u + E^T \bar{Q} C \Delta s_0, \]  
(13b)

Condensing based real-time iterations for NMPC

**Initialization:** Initialize \( s_k, k = 0, \ldots, N \) and \( u_k, k = 0, \ldots, N-1 \).

**Repeat online:**

1. **Preparation step**
   1.1 Solve IVPs (2) and generate sensitivities \( A_k \) and \( B_k \).
   1.2 Evaluate objective.
   1.3 Condense large QP (6) into a smaller dense QP (12).
   1.4 Wait for a new state feedback \( \bar{x}_0 \).

2. **Feedback step**
   2.1 Use the state feedback to compute the linear term of the dense QP and bounds on affine constraints.
   2.2 Solve the dense QP (12).
   2.3 Send the first control, \( u_0^+ \), to the process.

Fig. 1: A simplified description of the classical condensing based RTI scheme for NMPC.

where \( \bar{Q} \in \mathbb{R}^{(N-n_t) \times (N-n_t)} \) and \( \bar{R} \in \mathbb{R}^{(N-n_u) \times (N-n_u)} \) are block diagonal matrices with matrices \( Q \) and \( R \) on the corresponding diagonals.

Analyzing (12) and (13), it can be easily concluded that all quantities except the linear term \( g_c \) and two terms in the lower and upper affine constraint bounds can be precomputed – in the RTI this is called as the preparation phase. As soon as the current state \( \bar{x}_0 \) becomes available, the linear term and lower and upper bounds of the affine constraints are computed and the QP (12) is solved. This is called the feedback phase. Once the QP is solved, the control vector on the first shooting interval is updated, \( u_0^+ = u_0 + \Delta u_0^*, \) and immediately sent to the process. Afterwards, we can apply the so-called expansion step to obtain all variables from the original QP (6). The control variables are updated with \( u_k^+ = u_k + \Delta u_k^*, \quad k = 1, \ldots, N-1 \), while variables \( s_k \) are updated using (8):
\[ w_2^+ = w_2 + d + C \Delta s_0 + E \Delta w_1^*. \]
(14)

The classical condensing based RTI scheme is shown on Fig. 1.

The main computational bottleneck of the condensing technique is the computation of the condensed Hessian \( H_c \). In particular, \( O(N^3) \) complexity results from the multiplication of \( E' \) with \( \bar{Q} E \). A naive approach would lead to exactly \( N^3 \) block multiplications. However, significant savings can be obtained by taking into account that matrix \( E \) is lower block triangular and \( \bar{Q} \) is block diagonal, along with the fact that we need to calculate only upper (or lower) triangular blocks of the reduced Hessian \( H_c \). This approach results in \( N(N+1)(N+2)/6 \) block multiplications, each of runtime complexity of \( n_t^2 n_t \) floating point operations (flops\(^1\)). In other words, more than 75% block multiplications can be saved for \( N > 7 \) compared to the naive approach.

On the other hand, condensing technique greatly reduces the number of optimization variables, from \( (N+1)n_t + Nn_u \) to \( Nn_u \). This allows us to solve\((12)\) with an efficient

\(^1\)flops - addition, multiplication or fused multiply-accumulate operations of two floating point numbers
dense linear algebra QP solver, such as qpOASES [1], and significantly reduce the feedback delays. The condensing technique is known to perform well for relatively short horizons.

D. Solution of the sparse problem

The second way of solving (6) is to keep and optimize all variables of the original problems: $\Delta s$ and $\Delta u$, i.e. avoid the aforementioned condensing procedure. This leaves far more variables in the problem. However, compared to the the condensed, dense QP (12), the problem described in (6) is highly structured. This structure can be exploited to yield a computational complexity of $O(N(n^2_3+n^3_3))$, significantly reducing the execution time of one real-time iteration for long prediction horizons compared to the condensing based approach. The exploration of this banded structure in the solution of nonlinear control problems was first proposed in [24] and for linear MPC in [22].

There are currently three freely available fast embedded QP solvers that can exploit the structure of (6): an efficient prototype implementation of a primal-barrier interior point method named fastMPC [26] and two auto-generated solvers that implement a primal-dual interior point method: CVXGEN [18], [16] and FORCES [11], [10]. CVXGEN is optimized for small problems and provides C-code that can be executed in orders of milliseconds or even microseconds, while FORCES has about the same speed but supports problems of arbitrary size.

In this paper, we focus on employing FORCES, which supports more general convex multi-stage problems of the form

$$\min \ z \ \frac{1}{2} \sum_{k=0}^{N-1} z^T_k H_{F,k} z_k + g^T_{F,k} z_k$$

s.t. $C_{F,k} z_k + D_{F,k+1} z_{k+1} = c_{F,k}$, $k = 0, \ldots, N-1$, (15b)

$h_{F,k}(z_k) \leq 0$, $k = 0, \ldots, N$. (15c)

with $N+1$ stage variables $z_k$, coupling equality constraints (15b) and convex inequality constraints $h_{F,k}$. The problem (15) can be solved efficiently by exploiting the inherent problem structure. In particular, the Hessian in (15) is block-diagonal, and together with (15b), this introduces a block-banded structure in the reduced coefficient matrix of the linearized Karush-Kuhn-Tucker (KKT) system. Solving this system to obtain a search direction is the bottleneck of interior point methods, and can be done as suggested in [26], [11], [22], [24] quickly by exploiting the block-banded structure. Moreover, the FORCES solver is able to exploit certain special cases of MPC, for example how to efficiently deal with diagonal Hessian block $H_{F,k}$ and take advantage of the fact that (6) has only box constraints. More general constraints such as quadratic constraints are supported as well, but not considered in this paper.

III. ACADO CODE GENERATION TOOL

The ACADO Code Generation tool is part of the open-source software package ACADO Toolkit for automatic control and dynamic optimization [14]. The tool implements the algorithmic ideas based on the real-time iteration scheme. The user interface allows one to specify nonlinear dynamic model equations as well as objective and constraint functions. The original implementation of code-generator allowed for the export of generalized Gauss-Newton method for nonlinear MPC [15], and was followed up with extensions that made it possible to specify and export a solver for moving horizon estimation (MHE) [13]. The tool exploits problem structure and dimensions together with sparsity patterns to remove all unnecessary computations and remove the need for dynamic memory allocation. More details on the ACADO Code Generation tool can be found in [2].

More recent extensions of the ACADO Code Generation tool include support for implicit integrators for ODEs and differential algebraic equations (DAEs) [21], [20]. Those extensions allow handling of more challenging stiff dynamic models where usage of explicit integrators would require far more integration steps per prediction interval. This would directly lead to prohibitively long execution times.

Both the original implementation of auto-generated MPC and the later MHE extension implement a condensing procedure in order to solve a smaller-scale QP by means of a dense linear algebra QP solver. Cubic complexities of the condensing procedure and active-set based QP solver reduce the applicability of the considered approach for OCPs with long horizons.

The latest extension of the tool that is presented in this paper makes it possible to solve long horizon OCPs with interior point based solver FORCES inside ACADO. This way the expensive $O(N^3)$ condensing procedure is avoided, and in fact exchanged for $O(N)$ complexity coming from structure-exploiting IP based QP solver. From user point of view, only two options have to be changed in order to generate an MPC solver that is internally using FORCES to solve the QP (6).

The ACADO Code Generation tool generates self-contained ANSI-C compliant code, which can be deployed on any platform supporting the standard C library. The same holds for the auto-generated FORCES based QP solver. The qpOASES solver is provided in form of self-contained ANSI C++ code, thus creates slightly more dependencies. Branching in the exported code is minimized leading to improved code locality, thus faster execution times. Furthermore, heuristics have been implemented to unroll for-loops whenever possible to increase efficiency of the compiled code.

IV. NUMERICAL SIMULATIONS

In order to test both NMPC implementations we have chosen a chain of masses connected with springs [27], illustrated in Fig. 2. The dynamics of this system is nonlinear due to nonlinear spring forces. Furthermore, this model is easily scalable, meaning that the problem complexity can be increased by adding additional masses to the chain. The goal of each controller is to move back the chain from its...
disturbed to its equilibrium state, respecting upper and lower bounds on both states and input variables.

A. Simulation model

We consider a chain with $M$ balls of equal mass $m$ that are connected with springs. All springs have equal rest length $L$ and spring constant $D$. At both ends additional identical springs are attached. One end of the chain is attached to a fixed point, while the velocity of the other end $u = (u_x, u_y, u_z)$ can be controlled. The center of each ball is represented by a 3-dimensional coordinate $p_i = (p_{x,i}, p_{y,i}, p_{z,i})$. For the chain comprising $M$ balls, an ODE that describes model dynamics has $n_x = (2M + 1) \cdot 3$ states and $n_u = 3$ controls. In the rest of the paper it is assumed that all states can be measured. For all details specific to the model we refer to [27].

B. Simulation scenario

The continuous ODE model is parametrized by the multiple shooting technique using intervals of $T_s = 200$ ms. In order to reliably integrate the model we chose implicit Gauss-Legendre integrator of order four, with one integration step per shooting interval. The constraints that the controller must respect are defined as follows. There are simple input bounds

$$-1 \text{ m/s} \leq u_j \leq 1 \text{ m/s}, \quad j \in \{x, y, z\},$$

and there is a set of tight state bounds imposed by a wall next to the equilibrium plane (see Fig. 2):

$$-0.01 \text{ m} \leq p_{y,i}, \quad i = 0, \ldots, M - 1.$$

In other words, the chain must not hit the wall during the simulation. Compared to the simulation settings in [27], the wall is placed at a closer distance to the equilibrium plane. This makes the problem even harder to solve, and this was a design decision made in order to increase the number of active constraints during the simulations. Note that in case of condensing based NMPC state bounds are transformed to affine inequalities, in form of (12c). Relevant controller QP dimensions are summarized in Tables I.

All simulations are performed on a 3GHz Intel Q9650 based desktop computer, running the 64-bit version of Ubuntu Linux 12.04. All codes are compiled with the Intel 13.0.1 compiler, using the flags: -03 -xHost. Execution times are measured with the Linux function clock_gettime(), which yields resolution in the nanosecond range.

C. Results

Simulations are performed for three different variants of chain mass problem, namely for $M = 1, 2, 3$. For each dynamic system ten controllers are tested, for different horizon lengths $N$ in the range from 5 to 50. Simulation results are presented in Fig. 3. All plots show the maximum run-times for both types of controllers. Simulation results show that for a test problem with 9 states, 3 control inputs and 50 prediction intervals, the execution time can be reduced by a factor greater than 2 when using the new extension of the auto-generated NMPC.

Starting at horizon $N = 5$ and $M = 1$ we can observe that condensing based approach is very cheap. The feedback time is very short because a very small QP is solved, comprising only 15 variables. On the other hand, the runtime of the sparse QP solver is more expensive due to its internal factorizations which are of $O(n_x^3 + n_u^3)$ complexity; in this case number of optimization variables is more than four times larger that in the condensing based NMPC. Increasing the number of prediction intervals $N$, runtime of the condensing based NMPC scales with $N^3$, while the full-space NMPC scales linearly in $N$, as expected.

For a fixed number number of prediction intervals $N$, in the case of condensing based NMPC number of optimization variables is $N n_u$, i.e. independent of the number of states $n_x$. However, an increase of the number of states $n_x$ leads to an increased number of affine constraints $(12c) - MN$. This directly makes an NMPC problem harder to solve, i.e. requires more working set recalculations when using an active set based QP solver, resulting in longer runtimes. Increase of number of intervals $N$ directly leads to a cubic increase in the runtime of one RTI, as can be observed on Fig. 3.

For a fixed number of controls $n_u$ and fixed horizon length $N$, run-time of the interior point QP solver is proportional to the $n_x^3$. For shorter horizons, this makes IP based solver less favorable. In particular, as shown in Fig. 3, the condensing based solver is competitive until horizon lengths of 26, 40, and 50, for $M = 1, 2$, and 3, respectively. However, linear complexity in $N$ makes the IP based solver superior for
V. CONCLUSIONS AND OUTLOOK

This paper presents the most recent extension of the ACADO toolkit Code Generation tool to efficiently handle small to medium scale NMPC problems with long horizons in the millisecond range using the structure exploiting interior point based solver FORCES. The exported code is ANSI C/C++ compatible and can be deployed on arbitrary embedded platforms. The obtained results reflect the benefits of using condensing for short horizons and FORCES for long horizons, making the ACADO code generation tool able to solve a wide range of optimal control problems efficiently.

Future work will include extensions to support more general OCP formulations, as well as support for DAEs.

REFERENCES


