



FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs

A. Zanelli, A. Domahidi, J. Jerez & M. Morari

To cite this article: A. Zanelli, A. Domahidi, J. Jerez & M. Morari (2017): FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs, International Journal of Control, DOI: [10.1080/00207179.2017.1316017](https://doi.org/10.1080/00207179.2017.1316017)

To link to this article: <http://dx.doi.org/10.1080/00207179.2017.1316017>



Accepted author version posted online: 10 Apr 2017.



Submit your article to this journal [↗](#)



Article views: 2



View related articles [↗](#)



View Crossmark data [↗](#)

To appear in the *International Journal of Control*
Vol. 00, No. 00, Month 20XX, 1–25

Publisher: Taylor & Francis

Journal: *International Journal of Control*

DOI: <https://doi.org/10.1080/00207179.2017.1316017>

FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs

A. Zanelli^a, A. Domahidi^{c,d}, J. Jerez^{b,c} and M. Morari^{*b}

^a*Systemtheorie, Regelungstechnik und Optimierung, Albert-Ludwigs-Universität, Freiburg, Germany*

^b*Automatic Control Laboratory, ETH Zurich, Zurich, Switzerland*

^c*embotech GmbH, Zurich, Switzerland*

^d*inspire AG, Zurich, Switzerland*

(June 30, 2016)

Real-time implementation of optimization-based control and trajectory planning can be very challenging for nonlinear systems. As a result, if an implementation based on a fixed linearization is not suitable, the nonlinear problems are typically locally approximated online, in order to leverage the speed and robustness of embedded convex quadratic programming (QP) solver technology developed during the last decade. The purpose of this paper is to demonstrate that, using simple standard building blocks from nonlinear programming (NLP), combined with a structure-exploiting linear system solver, it is possible to achieve computation times in the range typical of solvers for QPs, while retaining nonlinearities and solving the NLP to local optimality. The implemented algorithm is an interior-point method with approximate Hessians and adaptive barrier rules, and is provided as an extension to the C code generator FORCES. Three detailed examples are provided that illustrate a significant improvement in control performance when solving NLPs, with computation times that are comparable with those achieved by fast approximate schemes and up to an order of magnitude faster than the state-of-the-art interior-point solver IPOPT.

Keywords: Predictive control, nonlinear systems, numerical optimization, interior-point methods, embedded systems

1. Introduction

Model predictive control (MPC) has drawn a lot of attention in both academia and industry since the late 1970s (Garcia, Prett, and Morari, 1988). The possibility of formulating an optimization problem that directly encodes control objectives and constraints, together with its inherent multi-variable nature, has made MPC a promising technology for the systematic design of control systems in several fields (Qin and Badgwell, 2003). This has become possible thanks to both the availability of reliable and efficient numerical methods and the increasing computational capabilities of embedded computing units. Despite the increased complexity of the algorithms in comparison with classical control techniques, considerable progress has been made in the last decade to make MPC

*Corresponding author. Email: morari@control.ee.ethz.ch

a viable technology in many challenging applications. For linear systems, fast implementations of convex solvers running at millisecond and microsecond timescale (Domahidi, Zraggen, Zeilinger, Morari, and Jones, 2012; Ferreau, Kirches, Potschka, Bock, and Diehl, 2014; Frison, Sorensen, Dammann, and Jørgensen, 2014; Jerez, Goulart, Richter, Constantinides, Kerrigan, and Morari, 2014; Richter, Jones, and Morari, 2012) have been developed and, for some methods, complexity certifications have been derived (Giselsson, 2012; Richter et al., 2012), providing bounds on the maximum number of iterations to guarantee convergence to a global optimum.

As a result, methods for the solution of convex quadratic programs (QPs) have become a reliable technology. When handling nonlinear nonconvex problems, it is common practice to approximate nonlinearities, potentially present in the system dynamics, constraints and objectives, in order to rely on existing methods and implementations for linear-quadratic MPC. A common approach to handle nonlinear models is to linearize the dynamics around the current approximation of the optimal trajectories. This procedure provides a linear time-varying model that locally approximates the dynamics and can be readily used to formulate a linear time-varying MPC problem as a convex QP. This strategy is employed, for example, in (Falcone, Borrelli, Asgari, Tseng, and Hrovat, 2007), which addresses the problem of actively controlling the front steering system of an autonomous vehicle. Alternatively, in (Kothare, Balakrishnan, and Morari, 1996) the effect of the nonlinear dynamics is taken into account by formulating a robust optimization problem. The original model is approximated by a polytopic uncertain linear time-varying system. Prior information on the Jacobian of the dynamics is used to guarantee that the trajectories of the system are confined inside polytopes. In this way a *min-max* problem is formulated that can be solved with semidefinite programming methods. However, the resulting scheme is potentially highly conservative and still too computationally intensive for most practical purposes. Authors of (Bemporad and Morari, 1999) propose instead to approximate the dynamics of a nonlinear system with a piecewise affine function, requiring the introduction of integer variables into the optimization problem, which greatly increases the computational complexity of the solution algorithms. Alternative approaches based on feedback linearization are proposed, for example, in (Deng, Baccara, and Stobart, 2009; Simon, Lofberg, and Glad, 2013). Feedback linearization results in problems with nonlinear and, in general, nonconvex inequality constraints that require specific handling. Despite the satisfactory results reported in some specific cases, these approaches, in general, result in formulations that are computationally infeasible for real-time applications or degrade the control performance significantly.

Including nonlinearity directly into the optimal control problem allows one to circumvent these difficulties and provides a systematic approach for handling systems with nonlinear dynamics, constraints and objectives. Following this path relies on the ability to efficiently and reliably solve in real-time the nonlinear and nonconvex optimization problems arising from an optimal control formulation. While there exist approaches in the literature that rely on dynamic programming techniques or indirect methods (Diehl, Ferreau, and Haverbeke, 2009), most existing strategies are nowadays based on the so-called *direct methods* (Diehl et al., 2009), which formulate a finite-dimensional discrete time nonlinear program (NLP). State-of-the-art numerical solvers for nonlinear programming can be used to solve the resulting discretized formulations.

Compared to the convex QPs arising in linear-quadratic MPC, optimization problems arising from nonlinear MPC formulations present several additional challenges that can have a major impact on both the reliability and efficiency of solvers. This is especially important for embedded systems that have limited computational power and must run autonomously without user interaction. Firstly, the discretization of the continuous-time problem requires the solution of nonlinear algebraic differential equations and the computation of derivatives for the current predicted trajectories with respect to the optimization variables, two operations that can be computationally demanding. Secondly, the discretized problem is in general nonconvex and a number of additional factors must be taken into account in order to guarantee convergence of the solver to a local minimum. This leads to a significant increase in complexity of the methods.

Due to the high computational burden associated with the solution of nonlinear optimization problems, several approximate schemes have been proposed to trade control performance for speed (Diehl et al., 2009). Among these schemes, the *Real-Time Iteration* (RTI) proposed in (Diehl, Bock, Schlöder, Findeisen, Nagy, and Allgöwer, 2002) is based on the idea of refining the solution while the problem changes (Diehl et al., 2009). A single convex QP that locally approximates the original optimization problem is solved per feedback step. If the iterates are initialized sufficiently close to a local minimum, a solution is gradually approached, while controlling the system using the current approximation of the solution. Local stability properties of such a scheme have been investigated in (Diehl, Findeisen, and Allgöwer, 2007). A region of attraction can be defined such that, if the combined system-optimizer state lies in it, the system can be stabilized using the mentioned approximate feedback law. When using this method, stability problems can arise if, after a large disturbance, the system leaves the region of attraction (Houska, Ferreau, and Diehl, 2011). A second approximate scheme in the literature is the so called *Continuation-GMRES* (C-GMRES) method in (Ohtsuka, 2004). As for the RTI, closed-loop stability of C-GMRES is in principle covered by the considerations in (Diehl et al., 2007) and analogous issues can be encountered in the presence of large disturbances or reference changes.

In order to avoid the eventual stability and suboptimality issues related to approximate schemes, it is possible to solve the optimization problems to a local minimum. In this way, classical results on nominal and robust stability (Nicolao, Magni, and Scattolini, 1998) can be applied. However, due to long computation times, feedback delays can degrade control performance in practice. Practical and theoretical issues have been discussed in the literature (Chen, Balance, and O'Reilly, 2000; Findeisen and Allgöwer, 2004; Santos, Afonso, Castro, Oliveira, and Biegler, 2001). The authors of (Zavala and Biegler, 2009) propose the so called *advanced-step NMPC* controller to overcome these difficulties. The main idea is to predict the state of the system in order to solve the future optimal control problem in advance, hence compensating for the computational delay. Before applying the computed control input to the system, a linearization of the solution manifold is used to perform a computationally cheap correction step based on the actual state estimate or measurement. This approach has been proven to perform well if a good prediction of the state of the system is available, while a considerable performance degradation is experienced in case of plant-model mismatch or large disturbances (Zavala and Biegler, 2009).

Main Contribution and Outline

Ideally, one would not need to trade off control performance for computational delay. This paper presents a software package that implements an efficient interior-point method for nonlinear and nonconvex optimization problems arising from optimal control formulations. The main contributions are the following:

- (1) The main features of the C code generator FORCES NLP (Domahidi and Jerez, 2016) are presented. The multistage structure of the NLPs is exploited in order to efficiently build Hessian approximations and solve the Karush-Kuhn-Tucker (KKT) systems to compute search directions. In particular, a block-wise Cholesky factorization is used based on the approach in (Domahidi et al., 2012). The class of problems solvable by the structure-exploiting QP solver in (Domahidi et al., 2012) is extended to the more general class of smooth nonlinear nonconvex programs.
- (2) Most available methods and software packages capable of achieving fast feedback times rely on approximate solution schemes. Despite the good control performance attainable in many cases, the system is often operated suboptimally. The proposed approach solves each optimization problem to a local minimum exploiting an efficient interior-point method in order to reduce the computational burden. The numerical results in Section 4 show how the resulting control performance can be considerably improved, especially for strongly nonlinear

problems, with moderately higher computation times.

- (3) Thanks to the structure-exploiting linear algebra, a considerable speedup can be achieved with respect to state-of-the-art solvers for nonlinear programming. A comparison with the state-of-the-art code IPOPT shows the large computational benefits of the proposed implementation for multistage problems.

The remainder of this paper is organized as follows: in Section 2 a brief introduction to nonlinear MPC is given and state-of-the-art numerical methods for the solution of the arising optimization problems are outlined. Section 3 discusses the implementation details of the software package presented and, in Section 4, numerical results are discussed that compare the solver with two different schemes: the fast and approximate RTI scheme and the exact, but generally computationally demanding, interior-point solver IPOPT (Wächter and Biegler, 2006).

2. Preliminaries

2.1 Nonlinear Model Predictive Control

2.1.1 Problem Formulation

In this work the following continuous-time, infinite-dimensional optimization problem will be considered:

$$\begin{aligned}
 \min_{x(\cdot), u(\cdot)} \quad & \int_{t_0}^{t_f} l_c(t, x(t), u(t)) dt + e(x(t_f)) \\
 \text{s.t.} \quad & x(t_0) = x_0 \\
 & \dot{x}(t) = f(t, x(t), u(t)) \\
 & g(t, x(t), u(t)) \leq 0,
 \end{aligned} \tag{1}$$

where $x \in \mathbb{R}^{n_x}$ are the states of the system, $u \in \mathbb{R}^{n_u}$ are its inputs and l_c , e , f and g are potentially nonlinear twice continuously differentiable functions. In order to recast the problem into a finite-dimensional program that can be solved in practical computation times for large system dimensions, the following reformulations are applied:

- (1) As closed form solutions for nonlinear differential equations are generally not available, the state trajectories need to be approximated numerically using an integration scheme;
- (2) A piece-wise constant parametrization of the control trajectories is used that results in a problem with a finite number of variables.

This so-called direct approach results in a formulation that can be solved with state-of-the-art methods for nonlinear optimization that can handle large-scale problems and can deal with constraints directly, avoiding the main difficulties related to indirect methods and approaches based on dynamic programming. Notice that these problems are in general nonconvex, hence computing the global minimum with direct methods can be a computationally intensive task. Most global optimization algorithms require the solution of several local NLPs (Nocedal and Wright, 2006), which is often not practical for embedded applications.

Among several approaches available to discretize problem (1), *direct multiple shooting* (Bock and Plitt, 1984) will be considered in this work, due to the special structure present in the obtained discretized problems. State trajectories at time $t = t_0, \dots, t_N$ are approximated using a numerical

integration scheme, obtaining the following discrete-time finite-dimensional problem

$$\begin{aligned} \min_{z_0, z_1, \dots, z_N} \quad & \sum_{i=0}^N l_i(z_i) \\ \text{s.t.} \quad & c_i(z_{i+1}, z_i) = 0, \quad i = 0, \dots, N-1 \\ & h_i(z_i) \leq 0, \quad i = 0, \dots, N, \end{aligned} \quad (2)$$

with stage variables $z_i \in \mathbb{R}^{p_i}$, stage cost functions $l_i : \mathbb{R}^{p_i} \rightarrow \mathbb{R}$, and equality constraints $c_i : \mathbb{R}^{p_i+p_{i+1}} \rightarrow \mathbb{R}^{r_i}$. General twice differentiable functions l_i and c_i will be considered and only simple bounds

$$h_i := \begin{bmatrix} \bar{z}_i - z_i \\ z_i - \underline{z}_i \end{bmatrix} \quad (3)$$

on the stage variables are present in the discretized formulation.

Notice that problems with general nonlinear inequalities can be recast into form (2) by introducing additional variables and equality constraints. Hence, although problems with general inequalities can be directly specified from the interface of FORCES NLP, in the following, formulation (2) will be considered, without loss of generality. The structure of (2) is the same as in convex QPs arising from linear-quadratic MPC formulations and it can be exploited in order to efficiently compute the Newton search directions in an interior-point method, as outlined in Section 3.

The continuous time formulation (1) can be discretized using other schemes as well. The so-called *direct single shooting* scheme provides a more compact formulation obtained by simulating the dynamics in order to eliminate the state variables. The resulting problem has fewer variables than the one obtained by applying multiple shooting. On the other hand, the favorable multistage sparsity structure is lost (Diehl et al., 2009). A different approach consists in formulating a problem that embeds the nonlinear equations used to discretize the dynamics, for instance, the equations of an implicit Runge-Kutta integration scheme including all intermediate variables. Such a scheme, referred to as *direct collocation*, results in a larger optimization problem with an alternative structure that can be exploited using different approaches to the ones used in this paper (Kang, Cao, Word, and Laird, 2014), (Quirynen, Gros, Houska, and Diehl, 2016). Further details on integration schemes used to obtain formulation (2) from the continuous-time optimal control problem (1) are given in Section 3.5.

2.2 Newton-type Approaches to Nonlinear Programming

Newton-type approaches to nonlinear programming can be interpreted as variations of the Newton method applied to the KKT system associated with (2):

$$r_S := \nabla_z \mathcal{L} = 0 \quad (4a)$$

$$r_C := c(z) = 0 \quad (4b)$$

$$r_D := h(z) + s = 0 \quad (4c)$$

$$r_N := Y_D s = 0 \quad (4d)$$

$$y_d \geq 0, \quad s \geq 0, \quad (4e)$$

where

$$\mathcal{L} := \sum_{i=0}^N l_i(z_i) + y_c^T c(z) + y_d^T h(z) \quad (5)$$

is the Lagrangian of the problem, y_c and y_d are the multipliers associated with equality and inequality constraints respectively, s are slack variables and Y_D is the diagonal matrix having the elements of y_d on its diagonal.

2.2.1 Sequential Quadratic Programming

A first approach that can be used in order to numerically solve problem (2) is sequential quadratic programming (SQP). The main idea consists in iteratively approximating the nonlinear problem with QPs. Given a candidate solution z^k the local quadratic approximation takes the form

$$\begin{aligned} \min_{\Delta z_0, \Delta z_1, \dots, \Delta z_N} & \sum_{i=0}^N \left(\frac{1}{2} \Delta z_i^T H_i \Delta z_i + \nabla l_i(z_i^k)^T \Delta z_i \right) \\ \text{s.t.} & c_i(z_{i+1}^k, z_i^k) + \nabla c_i(z_{i+1}^k, z_i^k)^T [\Delta z_i^T \Delta z_{i+1}^T]^T = 0, \quad i = 0, \dots, N-1 \\ & h_i(z_i^k) + \nabla h_i(z_i^k)^T \Delta z_i \leq 0, \quad i = 0, \dots, N, \end{aligned} \quad (6)$$

where H_i is an approximation of the block of the Hessian of the Lagrangian associated with z_i . After solving (6), the solution is updated

$$z^{k+1} = z^k + \Delta z \quad (7)$$

and it can be shown that, under mild assumptions, the iterates converge to a local solution of the discretized optimization problem (Nocedal and Wright, 2006).

Due to the potentially high computational burden associated with the solution of several QPs and the computation of several linearizations, there exist approaches in the literature that rely on a limited number iterations. The RTI scheme is based on this idea. It only performs a single step of an SQP-based method at each sampling instant and the resulting approximate solution is used to control the system. Although performance degradation can be experienced due to the suboptimality of the approximate solution, this approach has been proven to work well in several practical cases (Ferreau, Houska, Geebelen, and Diehl, 2011), (Quirynen, Gros, and Diehl, 2013a).

2.2.2 Interior Point Methods

An alternative approach consists in applying the Newton method directly to the set of nonlinear equations (4). In order to do so, the nonsmooth condition (4d) is relaxed, such that a set of smooth nonlinear equations is obtained to which the Newton method can be applied directly. Once a solution to the relaxed KKT system is obtained, the relaxation parameter τ is decreased and, as $\tau \rightarrow 0$, a local optimum is approached (Nocedal and Wright, 2006). The main computational cost is associated with evaluating functions and their derivatives, and computing search directions. At every iteration of the interior-point algorithm the KKT system is linearized at the current iterate and a linear system is solved to obtain a search direction. For stiff differential equations and differential algebraic equations, a high computational burden can be associated with function evaluations and derivative generation. Recent results on the efficient implementation of numerical methods for the integration of such problems (Quirynen, Vukov, and Diehl, 2012), (Quirynen et al., 2013a), (Quirynen, Gros, and Diehl, 2013b), can be exploited in order to considerably reduce

computation times required for linearization. In the following section the problem of efficiently computing the search directions is addressed.

3. Implementation

3.1 Structure-Exploiting Linear System Solver

An efficient interior-point method is used to solve (4) based on the implementation proposed in (Domahidi et al., 2012). While the KKT solver in (Domahidi et al., 2012) targets convex QPs and convex quadratically constrained quadratic problems (QCQPs), in this work an extension that can handle general nonlinear and nonconvex multistage problems is presented.

The linearized KKT version of (4) for problem (2) is

$$\begin{bmatrix} \mathcal{H}(z, y_c, y_d) & J_{\text{eq}}(z)^T & J_{\text{ineq}}(z)^T \\ J_{\text{eq}}(z) & & \\ J_{\text{ineq}}(z) & & \\ & S & Y_D \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta y_c \\ \Delta y_d \\ \Delta s \end{bmatrix} = - \begin{bmatrix} r_S \\ r_C \\ r_D \\ r_N \end{bmatrix}, \quad (8)$$

where $\mathcal{H}(z, y_c, y_d)$ is block-diagonal with blocks given by

$$\mathcal{H}_i := \nabla_{z_i}^2 l_i(z_i) + \sum_{k=1}^{r_{i-1}} y_{c,i-1[k]} \nabla_{z_i}^2 c_{i-1[k]}(z_i, z_{i-1}) + \sum_{k=1}^{r_i} y_{c,i[k]} \nabla_{z_i}^2 c_{i[k]}(z_{i+1}, z_i) + \sum_{k=1}^{q_i} y_{d,i[k]} \nabla_{z_i}^2 h_{i[k]}(z_i),$$

where r_i and q_i are the number of equality and inequality constraints in stage i , respectively, and $[k]$ denoting the k -th entry or constraint. J_{ineq} is also block diagonal with blocks consisting of $\nabla h_i(z_i)$, while J_{eq} is block banded with rows consisting of $\nabla c_i(z_{i+1}, z_i)$ in the appropriate position. Moreover, $Y_D \in \mathbb{R}^{q \times q}$ and $S \in \mathbb{R}^{q \times q}$ have been introduced to denote the diagonal matrices having y_d and s on their diagonals, respectively.

When dealing with nonlinear problems instead of convex QPs, \mathcal{H} is a function of the primal variables z , but also of the equality and inequality constraint multipliers y_c and y_d , due to the potential nonlinearity in the dynamics and constraints. The Jacobians of the equality and inequality constraints, for the same reason, are functions of z rather than being constant as in QPs and QCQPs. Moreover, it is important to mention that it is in general difficult to ensure that the KKT matrix in (8) has the desired inertia, i.e. the right number of positive, negative and zero eigenvalues, which is required to guarantee certain descent properties (Wächter and Biegler, 2006). With the Hessian of the Lagrangian potentially being indefinite, additional care has to be taken when solving system (8) as outlined in the following subsection.

The search direction (8) is computed with a direct structure-exploiting factorization method. In particular, the linear system is first reduced by eliminating Δs :

$$\Delta s = Y_D^{-1} (-r_N - S \Delta y_d) \quad (9)$$

and successively Δy_d :

$$\Delta y_d = S^{-1} Y_D (r_D + J_{\text{ineq}}(z) \Delta z) - S^{-1} r_N, \quad (10)$$

obtaining the following symmetric indefinite system:

$$\begin{bmatrix} \Phi & J_{\text{eq}}(z)^T \\ J_{\text{eq}}(z) & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta y_c \end{bmatrix} = - \begin{bmatrix} r_d \\ r_C \end{bmatrix}, \quad (11)$$

where

$$\begin{aligned}\Phi &:= \mathcal{H}(z, y_c, y_d) + J_{\text{ineq}}(z)^T S^{-1} Y_D J_{\text{ineq}}(z) \\ r_d &:= r_S + J_{\text{ineq}}(z)^T S^{-1} Y_D r_D - J_{\text{ineq}}(z)^T S^{-1} r_N.\end{aligned}\quad (12)$$

This is the form of the KKT system used for example in (Wächter and Biegler, 2006), which will be referred to as the *symmetric indefinite* form. As done in (Domahidi et al., 2012), a more compact and symmetric system can be obtained by forming the Schur complement of the iteration matrix (11):

$$Y \Delta y_c = \beta, \quad (13)$$

where

$$\begin{aligned}Y &:= J_{\text{eq}}(z) \Phi^{-1} J_{\text{eq}}(z)^T \\ \beta &:= r_C - J_{\text{eq}}(z) \Phi^{-1} r_d \\ \Delta z &= \Phi^{-1} (-r_d - J_{\text{eq}}(z)^T \Delta y_c).\end{aligned}\quad (14)$$

This form of the KKT system is also referred to as the *normal equations* form.

Another additional complication with respect to QPs arises because the Hessian and Jacobian matrices forming the linearized KKT matrix (8) are functions of the current iterate. As a result, matrices Φ and $J_{\text{eq}}(z)$ can become rank deficient. In general, it is necessary to add regularization terms to the Hessian and equality constraint Jacobian to be able to compute approximate search directions reliably.

FORCES NLP provides different linear solvers to solve the symmetric indefinite form (11) and the normal equations form (13). In both cases a tailored structure-exploiting, block-wise Cholesky decomposition (Wright, 1993) with complexity $\mathcal{O}(N)$ can be used under the assumption that Φ and Y are positive-definite. While in the quadratic convex case this assumption always holds for well-posed problems, the same does not hold for general nonlinear nonconvex problems. In the latter case, the Hessian of the Lagrangian is potentially indefinite and Y cannot be factorized with a Cholesky decomposition. Inertia correction techniques are used, for example in (Wächter and Biegler, 2006), to guarantee positive-definiteness of the Hessian projected onto the null space of the constraint Jacobian. However, this approach requires several successive factorizations and regularizations of the KKT matrix (8), which lead to considerably increased computational demands. A common workaround used in nonconvex optimization consists in approximating $\mathcal{H}(z, y_c, y_d)$ in order to guarantee its positive-definiteness. In the following subsection several customized approaches for multistage programs implemented in the FORCES NLP package are presented and discussed.

3.2 Hessian Approximation

3.2.1 Gauss-Newton

Consider a constrained nonlinear least-squares problem with cost function terms

$$l_i(z_i) := \frac{1}{2} \|\eta_i(z_i)\|_2^2. \quad (15)$$

This is a rather common formulation which, in the MPC context, can encode, for example, regulation and tracking control objectives. The blocks of the Hessian of the Lagrangian become

$$\begin{aligned} \mathcal{H}_i(z_{i+1}, z_i, y_{c,i}, y_{d,i}) := & \sum_{k=1}^{r_{i-1}} y_{c,i-1[k]} \nabla_{z_i}^2 c_{i-1[k]}(z_i, z_{i-1}) + \sum_{k=1}^{r_i} y_{c,i[k]} \nabla_{z_i}^2 c_{i[k]}(z_{i+1}, z_i) + \\ & \sum_{k=1}^{q_i} y_{d,i[k]} \nabla_{z_i}^2 h_{i[k]}(z_i) + \nabla_{z_i} \eta_i(z_i) \nabla_{z_i} \eta_i(z_i)^T + \sum_{k=1}^{v_i} (\eta_{i[k]}(z_i) \nabla_{z_i}^2 \eta_{i[k]}(z_i)), \end{aligned} \quad (16)$$

where v_i is the dimension of the residual η_i . Under the assumption of sufficiently small curvature of the functions η_i , h_i and c_i and small residuals $\eta_i(z_i)$, it is possible to neglect all the terms in (16) containing second order derivatives, obtaining the following positive-semidefinite approximation:

$$\mathcal{H}_i(z, y_c, y_d) \approx \nabla \eta_i(z_i) \nabla \eta_i(z_i)^T \succeq 0 \quad (17)$$

and, assuming that the Jacobian of the residuals $\nabla \eta_i^T$ has full rank, a positive-definite approximation $\nabla \eta_i(z_i) \nabla \eta_i(z_i)^T \succ 0$ is obtained. This approach has been proven to work very well in practice in many cases (Diehl et al., 2009). The fact that the approximation is always positive-semidefinite allows one to exploit the efficient Cholesky decomposition proposed in (Domahidi et al., 2012), while avoiding the evaluation of second order derivatives for the Lagrangian, an operation that is potentially computationally demanding.

3.2.2 Block-wise Broyden-Fletcher-Goldfarb-Shanno Hessian updates

Whenever the problem does not take the form of a constrained nonlinear least-squares problem, or in case of highly nonlinear problems where the second derivatives cannot be neglected, it is necessary to use a different approximation.

A different approach is to use the high-rank block updates proposed in (Bock and Plitt, 1984). This second method consists in applying the low-rank Broyden-Fletcher-Goldfarb-Shanno (BFGS) update for each stage $i = 0, \dots, N$ separately. Given the updated primal and dual variables z_i^+ , $y_{c,i}^+$ and $y_{d,i}^+$, and the previous primal iterate z_i , the Hessian approximation B_i of the i -th block (3.1) is updated as follows:

$$B_i \leftarrow B_i - \frac{B_i \delta \delta^T B_i}{\delta^T B_i \delta} + \frac{\gamma \gamma^T}{\delta^T \gamma}, \quad (18)$$

where

$$\begin{aligned} \delta &= z_i^+ - z_i \\ \gamma &= \nabla \mathcal{L}(z_i^+, y_{c,i}^+, y_{d,i}^+) - \nabla \mathcal{L}(z_i, y_{c,i}^+, y_{d,i}^+). \end{aligned} \quad (19)$$

The update (18) is guaranteed to preserve positive-definiteness of B_i if the condition $\delta^T \gamma > 0$ holds. In order to guarantee a positive-definite Hessian, it is possible to proceed in different ways. A first workaround is to skip the Hessian update whenever $\delta^T \gamma \leq 0$ and re-initialize the estimate whenever the update is skipped several times in succession. Such an approach is for example used in the IPOPT software package (Wächter and Biegler, 2006) and is also an option in FORCES NLP.

An alternative method is based on the modified update proposed by Powell in (Powell, 1978)

$$B_i \leftarrow B_i - \frac{B_i \delta \delta^T B_i}{\delta^T B_i \delta} + \frac{\omega \omega^T}{\delta^T \omega}, \quad (20)$$

with

$$\begin{aligned} \omega &:= \theta \gamma + (1 - \theta) B \delta \\ \theta &:= \begin{cases} 1 & \text{if } \delta^T \gamma \geq 0.2 \delta B \delta \\ \frac{0.8 \delta^T B \delta}{\delta^T B \delta - \delta^T \gamma} & \text{if } \delta^T \gamma < 0.2 \delta B \delta. \end{cases} \end{aligned} \quad (21)$$

It can be proved that the above update gives rise to superlinear convergence (Powell, 1978). The initialization of Hessian estimates B_i significantly affects the performance of the algorithm. For example, the common initialization $B_i = \rho I$ with some a-priori chosen scalar $\rho > 0$ can be used. Alternatively, a user-defined initialization can be provided.

3.3 Barrier Strategy

In order to guarantee quick progress towards a local minimum, the KKT system (8) needs to be relaxed according to a given policy, commonly referred to as the *barrier strategy* which adjusts the barrier parameter τ in

$$\begin{aligned} r_S &= 0 \\ r_C &= 0 \\ r_D &= 0 \\ r_N &= \tau \mathbf{1}. \end{aligned} \quad (22)$$

While a Mehrotra predictor-corrector approach is used in (Domahidi et al., 2012), due to its good empirical performance for convex quadratic problems, for nonlinear problems this heuristic can perform poorly in certain circumstances (Nocedal, Wächter, and Waltz, 2006).

A more robust approach is to monotonically decrease τ (Nocedal and Wright, 2006). Starting from a fixed initial value τ_0 , the relaxation is tightened whenever a solution that satisfies (22) to a given accuracy is found. In particular, given the current values of primal and dual variables and the barrier parameter, the optimality error is defined taking into account the maximum residuals in (8):

$$E_\tau := \max \{ \|r_S\|_\infty, \|r_C\|_\infty, \|r_D\|_\infty, \|r_N - \tau \mathbf{1}\|_\infty \} \quad (23)$$

and the barrier parameter is decreased whenever

$$E_\tau \leq \kappa_\epsilon \tau \quad (24)$$

for a given $\kappa_\epsilon > 0$. When condition (24) holds, the relaxation is updated according to

$$\tau \leftarrow \max \left\{ \frac{\epsilon_{\text{tol}}}{5}, \kappa_\tau \tau \right\}, \quad (25)$$

with constants $\kappa_\tau \in (0, 1)$ and ϵ_{tol} . When using such a barrier strategy, global convergence can be ensured, but slow convergence might be experienced (Nocedal et al., 2006). For this reason several accelerating adaptive strategies have been proposed in (Nocedal et al., 2006), while global

convergence is ensured by switching back to the monotone strategy when sufficient progress towards a solution cannot be made.

Different strategies have been analyzed and compared in (Nocedal et al., 2006), based on variants of the Mehrotra predictor corrector and other heuristics. In the following, an efficient approach (Vanderbei and Shanno, 1999) is reported that requires neglectable additional computational effort. This rule, used in the software package LOQO, determines the so called *centering parameter* σ as

$$\sigma = 0.1 \min \left\{ 0.05 \frac{1 - \xi}{\xi}, 2 \right\}^3 \quad \text{with} \quad \xi = \min_i \left\{ \frac{s_i y_{d,i}}{y_d^T s / q} \right\} \quad (26)$$

and the KKT relaxation is updated as follows:

$$\tau = \sigma \frac{s^T y_d}{q}. \quad (27)$$

The interested reader is referred to (Nocedal et al., 2006) for more details on adaptive barrier strategies and their comparison. The FORCES NLP package implements several monotone and adaptive barrier strategies, including the ones described above.

3.4 Line-Search Filter Method

In order to ensure convergence of the iterations, a backtracking line-search has been used based on the successful algorithm described in (Wächter and Biegler, 2006). Once the search direction has been computed, the maximum step lengths that maintain positivity in the slacks and multipliers are computed according to

$$\alpha_{\max} := \max \{ \alpha \in (0, 1] : s + \alpha \Delta s \geq \tau s \}, \quad (28)$$

$$\alpha_l := \max \{ \alpha \in (0, 1] : y_d + \alpha \Delta y_d \geq \tau y_d \}. \quad (29)$$

Then, a backtracking procedure is invoked, which iteratively reduces the primal step length from α_{\max} until a point is found that either improves feasibility or improves the barrier objective function:

$$\tilde{l}(\alpha_p) := \sum_{i=0}^N l_i(z_i + \alpha_p \Delta z_i) - \tau \sum_{i=0}^N \sum_{k=1}^{r_i} \log(s_{i[k]} + \alpha_p \Delta s_{i[k]}).$$

For more details, see (Wächter and Biegler, 2006).

3.5 Integration and sensitivity generation

Building the linear system (8) associated with the computation of the Newton steps requires integration of differential equations and computation of the sensitivities associated with the obtained trajectories. In order to do so, explicit or implicit integration methods can be chosen depending on several parameters such as stiffness of the differential equation, computational burden associated with function evaluations and number of states.

Integration and sensitivity generation routines can be computationally expensive and their careful implementation is of crucial importance in order to reduce computation times. While explicit schemes might be appropriate for non-stiff systems and when function evaluations can be carried out cheaply, in the presence of stiff dynamics or expensive function evaluations it is generally

necessary to use implicit schemes. The latter require the solution of implicit nonlinear equations that can be rather expensive.

FORCES NLP implements both explicit integrators and the state-of-the-art algorithms for implicit Runge-Kutta schemes introduced in (Quirynen, Vukov, Zanon, and Diehl, 2014), that have been proven to be orders of magnitude faster than commercial packages for problems of small to medium size. Such schemes exploit Newton-type iterations with a fixed Jacobian of the Runge-Kutta equations. In this way, a single factorization is carried out per integration step for both integration and sensitivity generation resulting in a highly efficient algorithm. A detailed description and benchmarking of the scheme can be found in (Quirynen et al., 2014).

3.6 Initialization and Warm-start

In order to initialize the algorithm, the user-provided initial point z^0 is modified such that it is feasible, i.e. ensuring that bounds on the primal variables are satisfied and that positivity constraints on slacks and bounds multipliers are satisfied. The multipliers $y_{c,i}$ associated with bounds are initialized to one componentwise and a least-squares estimate for the equality multipliers $y_{d,i}$ is computed using the possibly modified initial point (Wächter and Biegler, 2006).

Notice that, in the context of nonlinear MPC, where a series of neighbouring problems have to be solved as the initial conditions change, SQP methods can exploit warm-starts available from the solution of previous instances. This feature can lead to a significant reduction in the number of iterations required. In particular, for the RTI scheme theoretical results are present in the literature that guarantee local convergence if an initial guess that is “sufficiently” close to the optimal solution is available Diehl et al. (2007).

Warm-start of interior-point methods is less straightforward. Issues and potential solutions associated with the warm-start of interior-point methods are discussed in the literature for linear (Yildirim and Wright, 2002), quadratic (Gondzio and Grothey, 2006) and nonlinear nonconvex programs (Benson and Shanno, 2008), (Forsgren, 2006). Moreover, algorithms tailored to MPC applications have been proposed in (Shahzad and Goulart, 2011), (Shahzad, Kerrigan, and Constantinides, 2010) and (Zanelli, Quirynen, Jerez, and Diehl, 2017). The current implementation of FORCES NLP does not exploit warm-starts, but the warm-starting techniques available in the literature can be in principle applied to the algorithm implemented.

4. Numerical Results

The algorithm described has been implemented in the software package for embedded optimization FORCES NLP as an extension of the previously existing primal-dual interior-point solver for convex QCQPs (Domahidi et al., 2012). The tool consists of a code generation engine capable of providing efficient, statically-allocated, standalone C code with a small memory footprint.

In this section, the performance achieved with FORCES NLP (Domahidi and Jerez, 2016) is compared with IPOPT (Wächter and Biegler, 2006) and the solvers generated with the ACADO Toolkit (Houska et al., 2011). ACADO is a software environment for automatic control and dynamic optimization that provides code generation functionalities to export C code that implements the RTI algorithm (Diehl, Bock, and Schlöder, 2005). While IPOPT can, in general, more reliably solve NLPs to local optimality, it often leads to long computation times, the RTI-based solver in ACADO has been designed to provide approximate solutions within shorter computation times. By using FORCES NLP it is possible, for the examples presented in the following, to combine the same closed-loop performance as the one obtained by IPOPT, with computation times in the range of the ones achievable with the RTI scheme, hence offering a better trade-off between control performance and computational delay.

In order to compare the three solvers, tracking nonlinear optimal control problems of the form

$$\begin{aligned}
& \min_{\substack{x_0, \dots, x_N \\ u_0, \dots, u_{N-1}}} \sum_{i=0}^{N-1} l_i(x_i, u_i) + l_N(x_N) \\
& \text{s.t.} \quad x_{i+1} = c(x_i, u_i) \quad \forall i = 0, \dots, N-1 \\
& \quad \quad x_{min} \leq x_i \leq u_{max} \quad \forall i = 0, \dots, N \\
& \quad \quad u_{min} \leq u_i \leq u_{max} \quad \forall i = 0, \dots, N-1,
\end{aligned}$$

will be formulated and solved for three different nonlinear systems. The function c describes the discretized dynamics obtained using an explicit Runge-Kutta integrator of order four (RK4). The simulations in the following have been done using Gauss-Newton RTI solvers code-generated with ACADO using two different QP solvers: the active-set solver qpOASES (Ferreau et al., 2014) and the interior-point solver FORCES PRO (Domahidi et al., 2012). FORCES NLP has been configured to use block-wise BFGS Hessian approximations and a LOQO adaptive barrier strategy (Nocedal et al., 2006). Routines that evaluate functions and their derivatives required for integration, sensitivity generation and the BFGS updates have been code generated using CasADi (Andersson, Akesson, and Diehl, 2012). For IPOPT, the CasADi interface has been used to set up the optimal control formulations. As compiling the code generated by CasADi for functions, Jacobians and Hessians evaluations resulted in *out-of-memory* errors for some of the examples, the routines have been compiled and timed separately in order to have a fair estimate of the total execution time for IPOPT. Results with both exact and BFGS approximate Hessians are reported and the sparse direct linear solver MA57 (Duff, 2004) is used. For FORCES NLP and IPOPT the default termination conditions have been used. All the benchmarks have been run on a laptop XPS13 9350 with an Intel i7-6500U CPU (maximum boost frequency of 3.10 GHz) and Ubuntu 15.10 as operating system. All the examples described in the next sections are available online at <https://github.com/embotech/forcesnlp/examples>.

4.1 Robotic manipulator

Consider the robotic manipulator with two links and two revolute joints (Siciliano, Sciavicco, Villani, and Oriolo, 2009) illustrated in Figure 1. Two electric drives apply torques to the joints in order to steer the end-effector to the desired position.

The dynamical model of such a system can be derived with the Lagrangian approach:

$$\begin{aligned}
\ddot{\theta}_1 &= \gamma_1 \\
\ddot{\theta}_2 &= \gamma_2 \\
\dot{\tau}_1 &= \tau_{1r} \\
\dot{\tau}_2 &= \tau_{2r},
\end{aligned} \tag{30}$$

with

$$\begin{aligned}
\gamma_1 &= \frac{1}{(\alpha_1 - \alpha_2 \frac{\beta_1}{\beta_2}) \beta_2} (\beta_4 + \beta_3 \dot{\theta}_1^2 - \tau_2) - \alpha_3 \dot{\theta}_1 \dot{\theta}_2 + -\alpha_4 \dot{\theta}_2 - \alpha_5 + \tau_1 \\
\gamma_2 &= \frac{1}{\beta_2} (\tau_2 - \beta_1 \gamma - \beta_3 \dot{\theta}_1^2 - \beta_4),
\end{aligned}$$

and

$$\begin{aligned}
\alpha_1 &= I_{l_1} + m_{l_1} l_1^2 + k_{r_1}^2 I_{m_1} + I_{l_2} + m_{l_2} (a_1^2 + l_2^2 + 2a_1 l_2 \cos(\theta_2)) + I_{m_2} + m_{M_2} a_1^2 \\
\alpha_2 &= I_{l_2} + m_{l_2} (l_2^2 + a_1 l_2 \cos(\theta_2)) + k_{r_2} I_{m_2} \\
\alpha_3 &= -2.0 m_{l_2} a_1 l_2 \sin(\theta_2) \\
\alpha_4 &= -m_{l_2} a_1 l_2 \sin(\theta_2) \\
\alpha_5 &= (m_{l_1} l_1 + m_{M_2} a_1 + m_{l_2} a_1) g \cos(\theta_1) + m_{l_2} l_2 g \cos(\theta_1 + \theta_2) \\
\beta_1 &= I_{l_2} + m_{l_2} (l_2^2 + a_1 l_2 \cos(\theta_2)) + k_{r_2} I_{m_2} \\
\beta_2 &= I_{l_2} + m_{l_2}^2 + k_{r_2} k_{r_2} I_{m_2} \\
\beta_3 &= m_{l_2} a_1 l_2 \sin(\theta_2) \\
\beta_4 &= m_{l_2} l_2 g \cos(\theta_1 + \theta_2),
\end{aligned}$$

where θ_1, θ_2 are the angles describing the configurations of the manipulator and τ_1, τ_2 are the torques applied to the joints. The numerical values of the fixed geometrical and mechanical parameters used for the simulations are reported in Table 1.

In order to formulate the optimal control problem, states $x := [\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2, \tau_1, \tau_2]^T$ and controls $u := [\tau_{1r}, \tau_{2r}]^T$ are defined and the following constraints are imposed:

$$\begin{aligned}
-100 \text{ Nm} &\leq \tau_1 \leq 70 \text{ Nm} \\
-100 \text{ Nm} &\leq \tau_2 \leq 70 \text{ Nm} \\
-200 \text{ Nm/s} &\leq \tau_{1r} \leq 200 \text{ Nm/s} \\
-200 \text{ Nm/s} &\leq \tau_{2r} \leq 200 \text{ Nm/s}
\end{aligned}$$

A quadratic cost function is used that penalizes the deviation of states and inputs from a reference:

$$\begin{aligned}
l_i &:= (x - x_r)^T Q (x - x_r) + u^T R u \\
l_N &:= (x_N - x_r)^T Q_N (x_N - x_r),
\end{aligned}$$

with

$$\begin{aligned}
Q = Q_N &:= \text{diag}(10^3, 10^{-1}, 10^3, 10^{-1}, 10^{-2}, 10^{-2}) \\
R &:= \text{diag}(10^{-2}, 10^{-2})
\end{aligned}$$

and reference defined as

$$x_r = \begin{cases} t < 10s & [1.2, 0, -1.2, 0, 0, 0] \\ t \geq 10s & -[1.2, 0, -1.2, 0, 0, 0]. \end{cases} \quad (31)$$

A prediction horizon of $T_f = 2s$ over $N = 20$ shooting nodes is used with two intermediate RK4 integration steps per shooting interval.

Figure 2 shows the closed-loop trajectories obtained. In this case the difference between performance achieved by ACADO and FORCES NLP is minor and the relative closed-loop suboptimality is 0.85%. The same local minima are obtained with both FORCES NLP and IPOPT at each sampling

instant. The same holds for the two QP solvers used, FORCES PRO and qpOASES. For this reason, only the trajectories obtained with qpOASES are shown in the plots.

Figure 3 compares the number of iterations and computation times for the three solvers. ACADO and FORCES NLP achieve the fastest computation times, with FORCES NLP being only about two to three times slower than ACADO with FORCES PRO as QP solver. In FORCES NLP 0.0286 ms per iteration are spent in the integrators on average, in order to compute state trajectories and sensitivities.

It was not possible to compile the generated C code for the evaluations of functions, Jacobians and Hessians for CasADi-IPOPT, due to *out-of-memory* errors. For this reason, these routines have been code-generated, compiled and timed separately and the total execution time has been estimated.

4.2 Quadcopter

In this section, a tracking formulation is considered with the goal of stabilizing a quadcopter around its hover state. The control of such a system with nonlinear MPC has been previously addressed in (Quirynen et al., 2013a), where the RTI algorithm is used to obtain approximate solutions. In the following, it will be shown how the performance can be significantly improved when solving the problems to a local minimum at every sampling instant with moderate additional computational burden using the solver introduced. Consider the following model of a quadcopter adapted from (Quirynen et al., 2013a):

$$\begin{aligned}
 \dot{p} &= v \\
 \dot{q} &= \frac{1}{2}E^T\Omega \\
 \dot{v} &= \frac{1}{m}RF - g\mathbf{1}_z \\
 \dot{\Omega} &= J^{-1}(T + \Omega \times J\Omega) \\
 \dot{\omega} &= \omega_r,
 \end{aligned} \tag{32}$$

where p and v represent position and velocity of the quadcopter respectively and q and Ω are its orientation expressed in quaternion representation and its angular velocity. It is assumed that angular accelerations of the propellers ω_r can be tracked instantaneously, hence they are considered as inputs to the system. The constants J and m are the inertia matrix of the vehicle and mass, respectively, while the forces and torques applied to the system are described by

$$\begin{aligned}
 F &= \sum_{k=1}^4 \frac{1}{2}\rho AC_L \omega_k^2 \mathbf{1}_z \\
 T &= [T_1 \ T_2 \ T_3]^T,
 \end{aligned} \tag{33}$$

with

$$T_1 := \frac{AC_L L \rho (\omega_2^2 - \omega_4^2)}{2}, \quad T_2 := \frac{AC_L L \rho (\omega_1^2 - \omega_3^2)}{2}, \quad T_3 := \frac{AC_d L \rho (\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)}{2},$$

where ρ is the air density, C_D and C_L are the drag and lift coefficients and A is the area of the propellers. The values of the parameters appearing in the model are listed in Table 2.

States and controls are defined as $x := [p^T, q^T, v^T, \Omega^T, \omega^T]^T$ and $u := \omega_r$ and the following con-

straints are imposed:

$$\begin{aligned}
20 \text{ rad/s} &\leq \omega_k \leq 50 \text{ rad/s}, & \forall k = 1, \dots, 4 \\
-40 \text{ rad/s}^2 &\leq \omega_{rk} \leq 40 \text{ rad/s}^2, & \forall k = 1, \dots, 4 \\
-40 \text{ rad/s} &\leq \Omega_k \leq 40 \text{ rad/s}, & \forall k = 1, \dots, 3 \\
-10 \text{ m} &\leq p_k \leq 10 \text{ m}, & \forall k = 1, \dots, 3 \\
-100 \text{ m/s} &\leq v_k \leq 100 \text{ m/s}, & \forall k = 1, \dots, 3.
\end{aligned}$$

A prediction horizon of $T_f = 1s$ over $N = 30$ shooting nodes is used with a single RK4 integration step per shooting interval. A strictly convex quadratic cost function and terminal cost have been used to penalize deviation of states and inputs from a given reference

$$\begin{aligned}
l_i &:= (x_i - x_r)^T Q (x_i - x_r) + (u_i - u_r)^T R (u_i - u_r) \\
l_N &:= (x_N - x_{rN})^T Q_T (x_N - x_{rN}),
\end{aligned} \tag{34}$$

$$\begin{aligned}
Q_N = Q &= \text{diag}(\mathbf{1}_3^T \cdot 50, \mathbf{1}_4^T \cdot 20, \mathbf{1}_3^T \cdot 5, \mathbf{1}_3^T \cdot 5, \mathbf{1}_4^T \cdot 0.01) \\
R &= \text{diag}(\mathbf{1}_3 \cdot 5)
\end{aligned}$$

and constant references $x_r := [\mathbf{0}_3^T, 1, 0, 0, 0, \mathbf{0}_3^T, \mathbf{0}_3^T, \bar{\omega} \mathbf{1}_4^T]$ and $u_r := [0, 0, 0, 0]$, with $\bar{\omega} = 39.939 \text{ rad/s}$.

Figure 4 shows the resulting closed loop trajectories when using $N = 30$ shooting nodes. Both controllers manage to stabilize the system, steering the states to the specified reference. However, by computing a local minimum at every feedback step, the nonlinear interior-point solver achieves better control performance. The closed-loop trajectories obtained with IPOPT differ from the one obtained with FORCES NLP, due to the fact that the solver computes different local minima. However, the trajectories computed by IPOPT are not reported for clarity of the plots and are not taken into account in the suboptimality comparison in Table 3 as both solvers find locally optimal trajectories. The closed-loop cost achieved with FORCES NLP is taken as a reference to compute the suboptimality of the RTI scheme.

In order to improve the closed-loop performance obtained with the RTI, it is possible to perform multiple SQP steps per OCP instance. Every iteration would require an additional linearization and QP solution. Table 3 shows the obtained computation times and suboptimality for the OCP with an increasing number of shooting nodes. When applying several iterations to the same instance, shifts are applied only when updating the initial condition.

ACADO is always faster than FORCES NLP, but, especially for problems with longer horizons, the worst-case execution time is only moderately increased. A possible reason for this is that, for this example, the time spent in solving the ODE and computing sensitivities is a rather small fraction of the time required to carry out an interior-point iteration.

As it was not possible to compile the code generated by the interface to IPOPT in CasADi for function evaluations, the computation times reported in Table 3 are estimates computed based on separate timings for function evaluations. Routines for the evaluation of functions, Jacobians and Hessians have been code-generated in CasADi, compiled and timed separately. When using FORCES NLP, a considerable speedup can be achieved with respect to timings estimated for IPOPT.

4.3 Vehicle path planning

Real-time trajectory generation is an ubiquitous task for next generation robotics and autonomous driving applications. An optimization-based approach provides a systematic way for computing optimal trajectories with respect to the performance goals of the application. However, the resulting optimization problems are often challenging due to nonlinear models and nonconvex constraints, for instance, when avoiding obstacles.

Consider the following simple nonlinear vehicle model

$$\begin{aligned}\dot{y} &= v \cos(\theta) \\ \dot{z} &= v \sin(\theta) \\ \dot{v} &= F/m \\ \dot{\theta} &= s/I.\end{aligned}\tag{35}$$

where y and z are the Cartesian coordinates of the vehicle, v is the linear velocity and θ is the heading angle. There are two control inputs to the model: the acceleration force F and the steering torque s . The vehicle mass is $m = 1$ kg and its moment of inertia is $I = 1$ kgm².

For this example we formulate a trajectory generation problem where the objective is to maximize progress on the z direction, while minimizing the applied force and torque, i.e.

$$l_i := -100z + 0.1F^2 + 0.001s^2$$

The dynamics (35) are discretized with the explicit RK4 integrator using $N = 50$ shooting nodes over a prediction horizon of 5s. The maneuver is subject to a set of constraints, involving both the simple bounds

$$\begin{aligned}-5 \text{ N} &\leq F \leq 5 \text{ N}, \\ -1 \text{ Nm} &\leq s \leq 1 \text{ Nm}, \\ -3 \text{ m} &\leq y \leq 0 \text{ m}, \\ 0 \text{ m} &\leq z \leq 3 \text{ m}, \\ 0 \text{ m/s} &\leq v \leq 2 \text{ m/s}, \\ 0 \text{ rad} &\leq \theta \leq \pi \text{ rad},\end{aligned}\tag{36}$$

as well the nonlinear nonconvex constraints representing obstacles

$$\begin{aligned}1 \text{ m}^2 &\leq y^2 + z^2 \leq 9 \text{ m}^2 \\ 0.9025 \text{ m}^2 &\leq (y + 2)^2 + (z - 2.5)^2.\end{aligned}\tag{37}$$

In addition, the vehicle should be at standstill and with a heading angle of 0 degrees at the end of the maneuver. The simulation using IPOPT exploits code-generated C routines for the evaluations of functions and their first and second order derivatives. The generated code has been compiled with optimization flag `-O3`. Figure 5 shows the computed optimal trajectories, which are the same for FORCES NLP and IPOPT. However, there is a large difference in the time required to compute the solution with the two solvers. Table 4 shows that FORCES NLP is more than one order of magnitude faster than IPOPT for this trajectory generation problem.

5. Conclusions

The software package FORCES NLP that provides an efficient tailored implementation of an NLP solver for nonlinear nonconvex multistage problems has been presented. The package implements a structure-exploiting nonlinear interior-point method with approximate Hessians, and generates standalone efficient C code that can be readily deployed on any embedded platform. The performance of the solver has been assessed on three optimal control and trajectory generation examples. It has been shown that, for examples for which it makes a difference to solve NLPs to local optimality, computation times in the same range as with fast approximate schemes can be achieved, while considerable improvements in the control performance can be attained. Moreover, the presented software has been compared to the state-of-the-art code for nonlinear optimization IPOPT, achieving speedups of up to an order of magnitude. The numerical examples comparing FORCES NLP, ACADO and IPOPT used in this paper are all available on <https://github.com/embotech/forcesnlp/examples>.

References

- Andersson, J., Akesson, J., and Diehl, M. (2012). CasADi – A symbolic package for automatic differentiation and optimal control. In *Recent Advances in Algorithmic Differentiation*, volume 87 of *Lecture Notes in Computational Science and Engineering*, 297–307. Springer.
- Bemporad, A. and Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3), 407–427.
- Benson, H. and Shanno, D. (2008). Interior-point methods for nonconvex nonlinear programming: regularization and warmstarts. *Computational Optimization and Applications*, 40(143), 143–189.
- Bock, H. and Plitt, K. (1984). A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings of the 9th IFAC World Congress*, 242–247. Pergamon Press, Budapest, Hungary.
- Chen, W., Balance, D., and O’Reilly, J. (2000). Model predictive control of nonlinear systems: computational burden and stability. *IEEE Proc. Control Theory Appl.*, 147(4), 387–394.
- Deng, J., Baccara, V.M., and Stobart, R. (2009). Input constrained handling in an MPC/feedback linearization scheme. *Int. J. Appl. Math. Comput. Sci.*, 19(2), 219–232.
- Diehl, M., Bock, H.G., and Schlöder, J.P. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5), 1714–1736. .
- Diehl, M., Bock, H.G., Schlöder, J.P., Findeisen, R., Nagy, Z., and Allgöwer, F. (2002). Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4), 577–585.
- Diehl, M., Ferreau, H., and Haverbeke, N. (2009). Nonlinear model predictive control, efficient numerical methods for nonlinear MPC and moving horizon estimation. *Lecture notes in control and information sciences*, 384, 391–417.
- Diehl, M., Findeisen, R., and Allgöwer, F. (2007). A stabilizing real-time implementation of nonlinear model predictive control. *Real-Time and Online PDE-Constrained Optimization*, 23–52.
- Domahidi, A. and Jerez, J. (2016). *FORCES Nonlinear Programming Solver*. embotech GmbH. URL <https://www.embotech.com/FORCES-Pro/User-Manual/High-level-Interface>.
- Domahidi, A., Zraggen, A., Zeilinger, M., Morari, M., and Jones, C. (2012). Efficient interior point methods for multistage problems arising in receding horizon control. In *IEEE Conference on Decision and Control*, 668–674. Maui, HI, USA.
- Duff, I. (2004). Ma57—a code for the solution of sparse symmetric definite and indefinite systems. *ACM Transactions on Mathematical Software*, 30(2), 118–144.
- Falcone, P., Borrelli, F., Asgari, J., Tseng, H.E., and Hrovat, D. (2007). Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on Control Systems Technology*, 15(3), 566 – 580.
- Ferreau, H., Houska, B., Geibelen, K., and Diehl, M. (2011). Real-time control of a kite-model using an auto-generated nonlinear MPC algorithm. *18th IFAC World Congress*, 44(1), 2488–2493.
- Ferreau, H., Kirches, C., Potschka, A., Bock, H., and Diehl, M. (2014). qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4), 327–363.

- Findeisen, R. and Allgöwer, F. (2004). Computational delay in nonlinear model predictive control. In *Proc. Int. Symp. Adv. Control of Chemical Processes*, 427–432. Hong Kong.
- Forsgren, A. (2006). *On Warm Starts for Interior Methods*, 51–66. Springer US, Boston MA.
- Frison, G., Sorensen, H.B., Dammann, B., and Jørgensen, J.B. (2014). High-performance small-scale solvers for linear model predictive control. In *ECC*, 128–133.
- Garcia, C., Prett, D., and Morari, M. (1988). Model predictive control: theory and practice – A survey. *Automatica*, 25(3), 335–348.
- Giselsson, P. (2012). Execution time certification for gradient-based optimization in model predictive control. In *51st IEEE Conference on Decision and Control*, 3165–3170. Maui, Hawaii, USA.
- Gondzio, J. and Grothey, A. (2006). A new unblocking technique to warmstart interior point methods based on sensitivity analysis. *Siam J. Control Optim.*, 19(3), 1184–1210.
- Houska, B., Ferreau, H., and Diehl, M. (2011). An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica*, 47(10), 2279–2285.
- Jerez, J., Goulart, P., Richter, S., Constantinides, G., Kerrigan, E., and Morari, M. (2014). Embedded online optimization for model predictive control at megahertz rates. *IEEE Transactions on Automatic Control*, 59(12), 3238–3251.
- Kang, J., Cao, Y., Word, D., and Laird, C. (2014). An interior-point method for efficient solution of block-structured NLP problems using an implicit Schur-complement decomposition. *Computers & Chemical Engineering*, 71, 563–573.
- Kothare, M.V., Balakrishnan, V., and Morari, M. (1996). Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32(10), 1361–1379.
- Nicolao, G.D., Magni, L., and Scattolini, R. (1998). Nonlinear model predictive control, Stability and robustness of nonlinear receding horizon control. *Progress in Systems and Control Theory*, 3–22.
- Nocedal, J., Wächter, A., and Waltz, R. (2006). Adaptive barrier update strategies for nonlinear interior methods. *SIAM Journal on Optimization*, 19(4), 1674–1693.
- Nocedal, J. and Wright, S.J. (2006). *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2 edition.
- Ohtsuka, T. (2004). A continuation/GMRES method for fast computation of nonlinear receding horizon control. *Automatica*, 4(40), 563–574.
- Powell, M. (1978). A fast algorithm for nonlinearly constrained optimization calculations. *Watson, G.A. (ed.) Numerical Analysis*, 144–157.
- Qin, S.J. and Badgwell, T.A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7), 733–764.
- Quirynen, R., Gros, S., and Diehl, M. (2013a). Efficient NMPC for nonlinear models with linear subsystems. In *IEEE Conference on Decision and Control*, 5101–5106. Florence, Italy.
- Quirynen, R., Gros, S., and Diehl, M. (2013b). Fast auto generated ACADO integrators and application to MHE with multi-rate measurements. In *European Control Conference*, 3077–3082. Zurich, Switzerland.
- Quirynen, R., Gros, S., Houska, B., and Diehl, M. (2016). Lifted collocation integrators for direct optimal control in ACADO toolkit. *Mathematical Programming Computation (under review, preprint available at Optimization Online)*.
- Quirynen, R., Vukov, M., and Diehl, M. (2012). Auto generation of implicit integrators for embedded NMPC with microsecond sampling times. In *IFAC Nonlinear Model Predictive Control Conference*, 175–180. Noordwijkerhout, The Netherlands.
- Quirynen, R., Vukov, M., Zanon, M., and Diehl, M. (2014). Autogenerating microsecond solvers for nonlinear MPC: a tutorial using ACADO integrators. *Optimal Control Applications and Methods*, 36, 685–704.
- Richter, S., Jones, C., and Morari, M. (2012). Computational complexity certification for real-time MPC with input constraints based on the fast gradient method. *IEEE Transactions on Automatic Control*, 57(6), 1391–1403.
- Santos, L.O., Afonso, P., Castro, J., Oliveira, N., and Biegler, L.T. (2001). On-line implementation of nonlinear MPC: An experimental case study. *Control Engineering Practice*, 9(8), 847–857.
- Shahzad, A. and Goulart, P. (2011). A new hot-start interior-point method for model predictive control. In *Proceedings of the IFAC World Congress*.
- Shahzad, A., Kerrigan, E., and Constantinides, G. (2010). A warm-start interior-point method for predictive control. Technical report, Imperial College London.
- Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2009). *Robotics: Modelling, Planning and Control*.

- Springer.
- Simon, D., Lofberg, J., and Glad, T. (2013). Nonlinear model predictive control using feedback linearization and local inner convex constraint approximations. In *European Control Conference*, 2056–2061. Zurich, Switzerland.
- Vanderbei, R. and Shanno, D. (1999). An interior point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13, 231–252.
- Wächter, A. and Biegler, L. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(25), 25–57.
- Wright, S.J. (1993). Interior point methods for optimal control of discrete time systems. *Journal of Optimization Theory and Applications*, 77(1), 161–187.
- Yildirim, E. and Wright, S.J. (2002). Warm-start strategies in interior-point methods for linear programming. *SIAM Journal on Optimization*, 12(3), 782–810.
- Zanelli, A., Quirynen, R., Jerez, J., and Diehl, M. (2017). A homotopy-based nonlinear interior-point method for NMPC applications. In *Proceedings of 20th IFAC World Congress*. Toulouse, France.
- Zavala, V.M. and Biegler, L.T. (2009). The advanced step NMPC controller: Optimality, stability and robustness. *Automatica*, 45(1), 86–93.

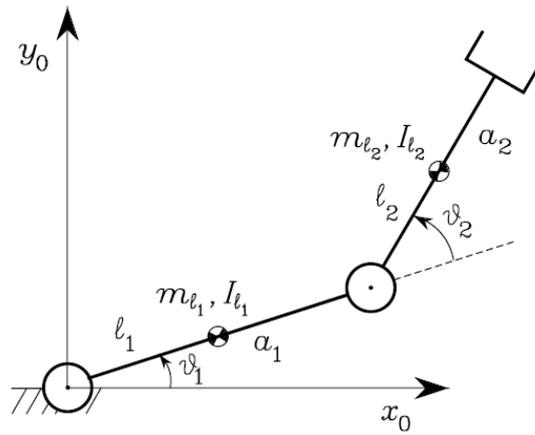


Figure 1.: Robotic manipulator with two links and two joints (Siciliano et al., 2009).

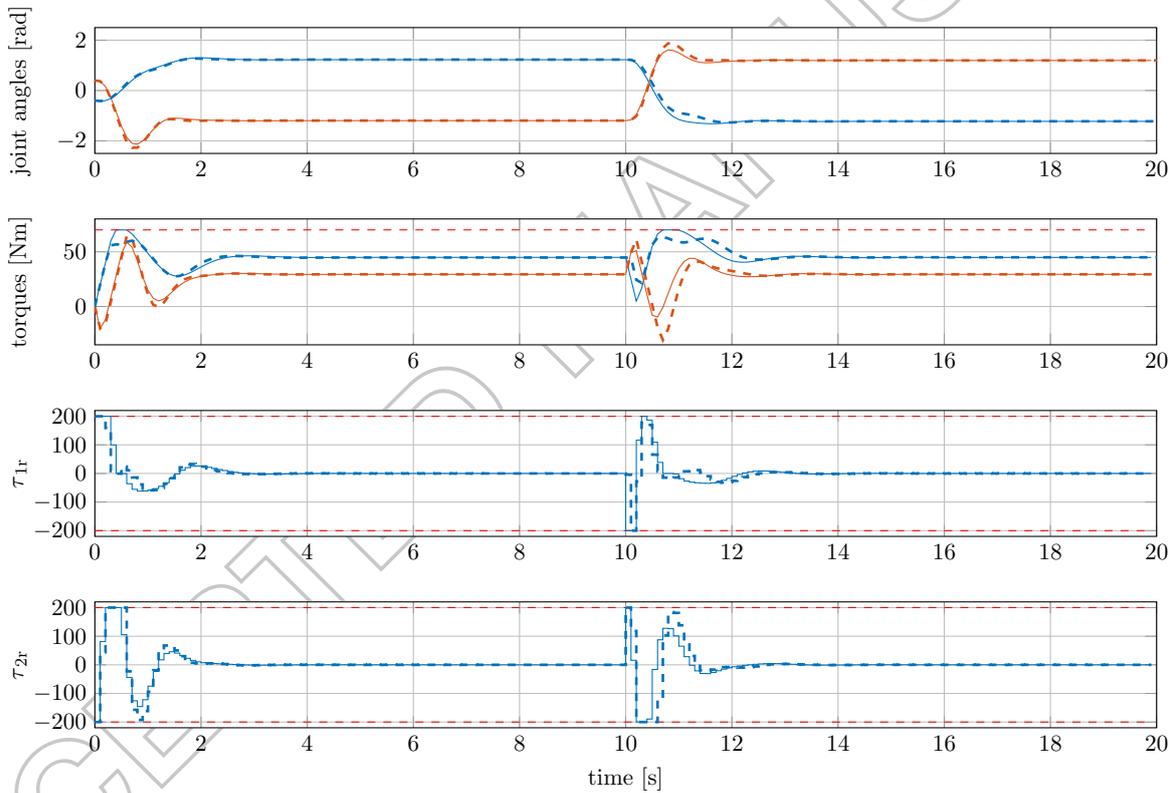


Figure 2.: Robotic manipulator tracking a time-varying reference. The figures compare the closed-loop trajectories obtained with the RTI scheme with shifts (`'shifting.strategy=1'`) (dashed) and FORCES NLP (solid). ACADO uses qpOASES (with warm-starting) as QP solver. In this case, there is a minor deviation in the closed-loop trajectories and the relative suboptimality is 0.85%. The trajectories obtained with IPOPT and ACADO with FORCES PRO as QP solver are not shown in the plot as they are identical to the ones obtained with FORCES NLP and ACADO with qpOASES respectively.

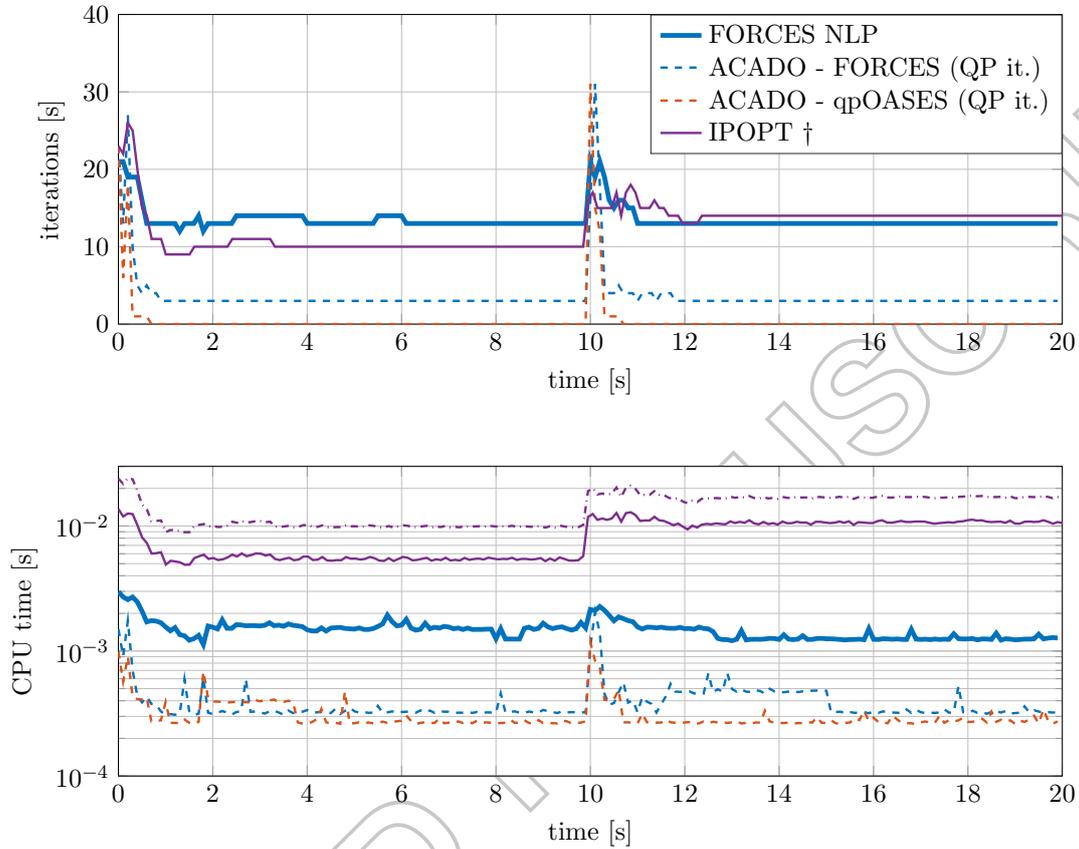


Figure 3.: Number of iterations and computation times for all solvers. Interior-point iterations are reported for IPOPT and FORCES NLP, while the number of QP iterations to solve every QP are reported for ACADO. ACADO exploits shifting (`'shifting.strategy=1'`) and qpOASES is warm-started. ACADO is always faster than FORCES NLP and it achieves a worst-case execution time that is less than two to three times shorter. While both nonlinear interior-point solvers solve the OCPs to a local minimum, FORCES NLP has a worst-case execution time that is about 8 times shorter than the one obtained with IPOPT. Integration and sensitivity generation in FORCES NLP requires, for this example, 0.0286 ms per iteration on average. † *It was not possible to compile the code generated by CasADi-IPOPT for functions, Jacobians and Hessians evaluations, due to out-of-memory errors. These routines have been code-generated, compiled and timed separately in order to estimate CPU time spent in function evaluations (total estimated execution time in dash-dot style).*

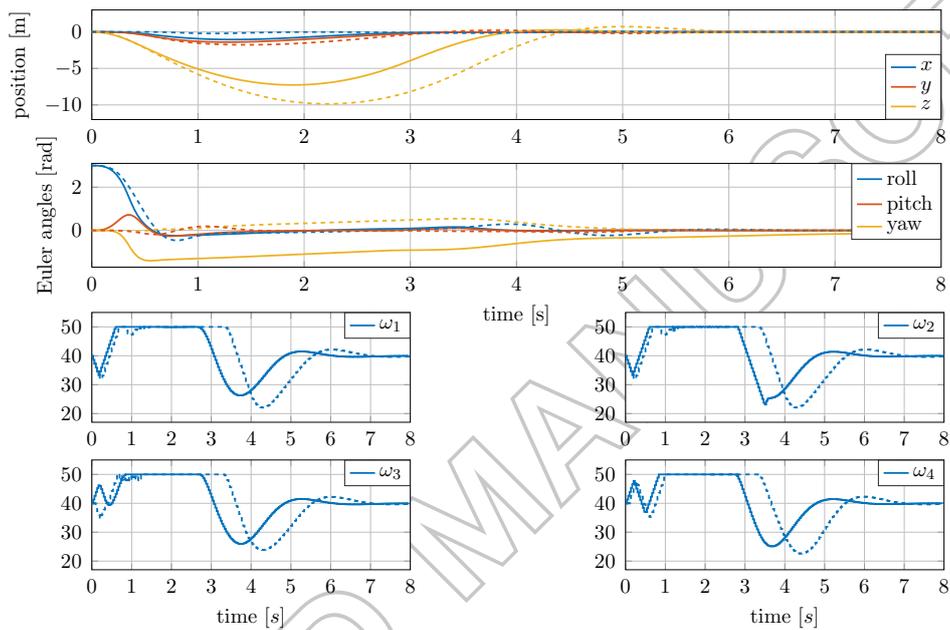


Figure 4.: Stabilization of the quadcopter around hovering state: closed-loop trajectories obtained with FORCES NLP (solid) and ACADO with qpOASES (dashed) for $N = 30$. Although different trajectories are obtained with IPOPT, the latter are not reported for clarity of the plots. ACADO uses warm-starts and shifts to initialize subsequent subproblems. The nonlinear interior-point solver computes a local minimum at every feedback step resulting in an improved performance. Relative closed-loop suboptimality associated with the approximate scheme is 69.07%.

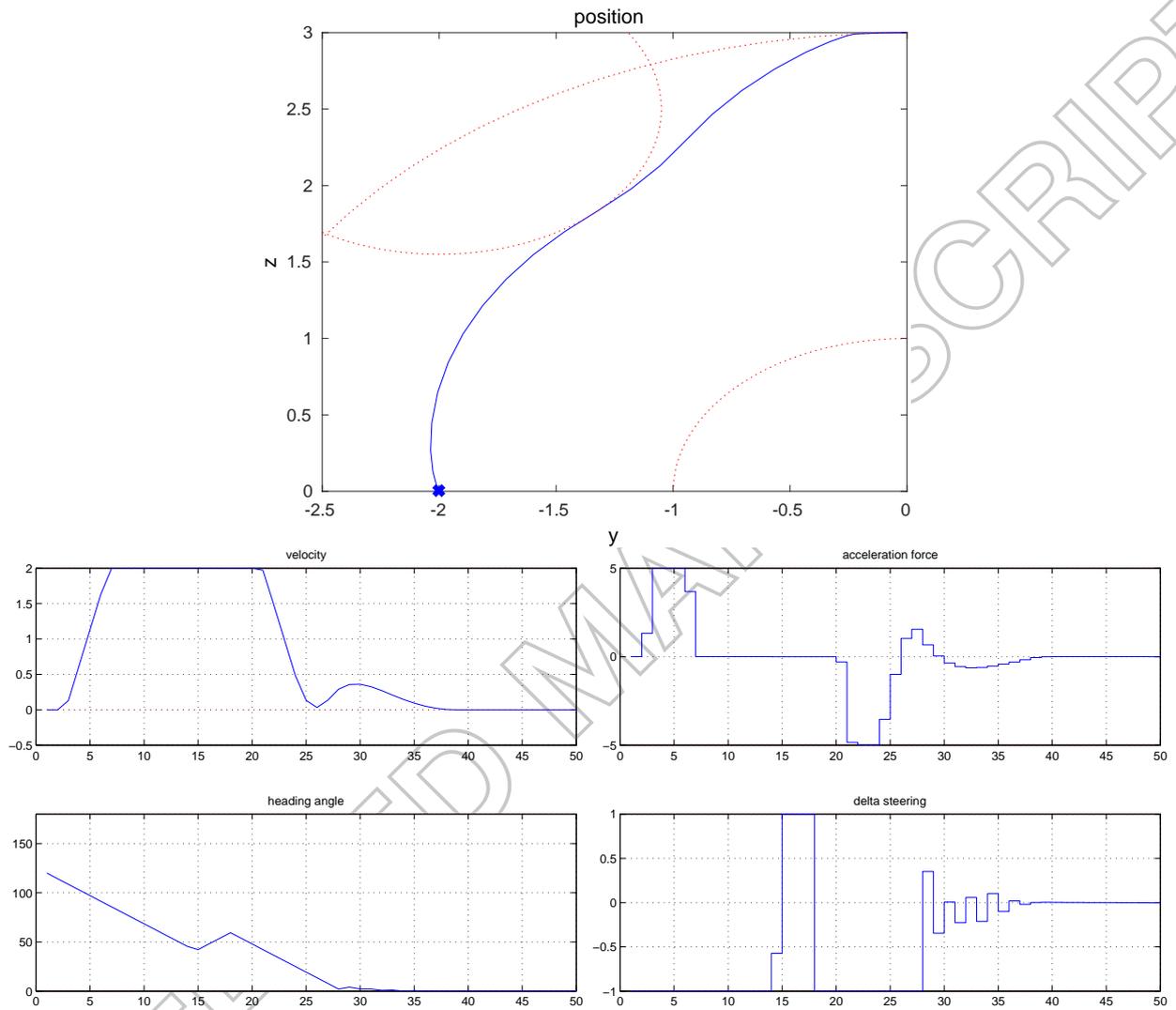


Figure 5.: Trajectory of the vehicle on a 2D space (top), vehicle's velocity and heading angle (left), and actuators (right). Constraints are represented with dotted red lines. The vehicle starts at position $(-2,0)$ and should maximize progress in y direction, while being at standstill and at heading angle of zero degrees at its final position after 50 integration steps.

Table 1.: Robotic manipulator: parameter values

Parameter	Value	Description
$m_{M_1} = m_{M_2}$	0.3 kg	motor mass
$k_{r_1} = k_{r_2}$	1.0	gear reduction ratio
$r_{M_1} = r_{M_2}$	0.05 m	motor radius
$I_{M_1} = I_{M_2}$	$\frac{1}{2}m_{M_1}r_{M_1}^2$	motor moment of inertia
$m_{l_1} = m_{l_2}$	3.0 kg	link mass
$I_{l_1} = I_{l_2}$	$\frac{1}{12}m_{l_1}l_1^2$	link moment of inertia
$a_1 = a_2$	$\frac{1}{2}l_1$	link center of mass relative position
$l_1 = l_2$	1 m	link length
g	$9.81 m/s^2$	gravitational acceleration

Table 2.: Quadcopter: parameter values

Parameter	Value	Description
ρ	$1.225 kg/m^3$	air density
A	$0.1 m^2$	propeller area
C_l	0.25	lift coefficient
C_d	$0.3 \cdot C_l$	drag coefficient
m	10 Kg	quadcopter mass
g	$9.81 m/s^2$	gravitational acceleration
$J_1 = J_2 = J_3$	$0.25 Kg \cdot m/s^2$	link length

	$N = 10$		$N = 20$		$N = 30$	
	time [ms] (it.)	sub. %	time [ms] (it.)	sub. %	time [ms] (it.)	sub. %
RTI - qpOASES	5.09 (3)	11.86	24.43 (2)	41.19	42.72(1)	69.07
RTI - FORCES	15.52(3)*	20.27	35.76(3)*	11.13	44.57(2)*	10.47
FORCES NLP	18.17	-	45.18	-	51.71	-
IPOPT	109.7 [†]	-	283.1 [†]	-	494.6 [†]	-

Table 3.: Quadcopter example: worst-case CPU time and closed-loop relative suboptimality for increasing horizon lengths. For ACADO, the minimum number of SQP iterations required to stabilize the system is taken into account (in brackets next to timings). For small problems, ACADO - qpOASES can achieve moderate suboptimality with considerably shorter computation times. Although FORCES NLP is always slower than ACADO, when increasing the horizon length, rather competitive computation times can be achieved with improved performance with respect to the RTI algorithm. *Notice that, when using FORCES as QP solver, it is necessary to damp the Newton steps in order to avoid ill-conditioned QPs which would make the solver fail. This justifies the different suboptimality incurred with respect to ACADO - qpOASES. [†] Computation times for IPOPT are estimated based on separate timings of code-generated function evaluations.

	Time w/o f. evals	Total solution time	Iterations
FORCES NLP - BFGS	7.0 ms	8.0 ms	74
IPOPT - BFGS	95 ms	140 ms	70
IPOPT - exact Hessian	105 ms	258 ms	117

Table 4.: Solve times and number of iterations for the vehicle path planning problem. Timings with and without function evaluations are reported. The routines evaluating functions and their first and second order derivatives have been code generated using the CasADi interface to IPOPT. Notice that, especially when computing exact Hessians, the time spent in function evaluations can be significant.