A fast nonlinear MPC solver for real-time control of linear motors

Riera Segui, Antoni

Award date:
2018

Technische Universiteit
Eindhoven
University of Technology

Department of Electrical Engineering
Control Systems group
Postbus 513, 5600 MB Eindhoven
The Netherlands
www.tue.nl

Master
Systems and Control

Student IDNR
1033755

Date
October 26, 2018

A Fast Nonlinear MPC Solver for Real-Time Control of Linear Motors

Master Thesis

Antoni Riera Segui

Supervision:	dr.	M. (Mircea) Lazar
	dr.ir.	T.T. (Tuan) Nguyen
Committee chairman:	prof.dr.ir.	P.M.J. (Paul) Van den Hof
Committee members:	prof.dr.ir.	H. (Hans) Butler
	dr.	A. (Alessandro) Saccon

Technische Universiteit
Eindhoven
University of Technology

Declaration concerning the TU/e Code of Scientific Conduct for the Master's thesis

I have read the TU/e Code of Scientific Conduct[i].

I hereby declare that my Master's thesis has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct

Date

14 - 11 - 2018

Name

Riera Segui, A

ID-number

1033755

Signature

Submit the signed declaration to the student administration of your department.

[i] See: http://www.tue.nl/en/university/about-the-university/integrity/scientific-integrity/
The Netherlands Code of Conduct for Academic Practice of the VSNU can be found here also.
More information about scientific integrity is published on the websites of TU/e and VSNU

A Fast Nonlinear MPC Solver for Real-Time Control of Linear Motors

Antoni Riera
IDNR: 1033755 - Master: Systems and Control
Supervision: Dr. M. Lazar, Dr. T. Nguyen
Control Systems Group - Electrical Engineering Department

***Abstract*—Model Predictive Control (MPC) is a control technique which has traditionally been applied to systems with slow dynamics, due to its high computational requirements. The increase in computational power has recently opened the door to applying MPC to areas requiring a high sampling rate, such as mechatronic systems. One of such applications is the control of Ironless Linear Motors (ILMs). When accounting for non-ideal geometry, an ILM is modelled as a nonlinear system. Therefore, applying Model Predictive Control requires solving a nonlinear optimization problem at every sample. In this work, a tailor-made algorithm for the fast solution of a class of Nonlinear MPC problems for systems with input nonlinearities is developed, a class which includes ILMs. The algorithm implements an Interior Point scheme while exploiting the particular problem structure in order to accelerate the required linear algebra operations. Simulation results show that nonlinear MPC may be used to control ILMs at sampling times in the microsecond range. The derived algorithm and software outperforms other freely available solvers for real-time nonlinear MPC.**

I. Introduction

Model Predictive Control (MPC) is a control technique which uses a system model to optimize a sequence of inputs over future time instants (horizon). Its main advantages include the possibility to consider constraints in the current and future time instants, anticipate to future reference or system changes, easily deal with multivariate systems, and in general deal with a wide range of (nonlinear) systems which describe processes in many disciplines, among others. Furthermore, the time response of the controller can be adjusted intuitively by tuning its parameters.

A MPC controller determines each input to the system from the solution of an optimization problem, which needs to be calculated in real time. The technique was originally developed for the control of large-scale processes in the chemical and petroleum industry, where the slow dynamics allowed for sampling times which could be measured in minutes or hours. Hence, the computation time required for solving the corresponding optimization problems was not a problem. Recent advances in computational power and numerical optimization have opened the door to the application of MPC to systems with fast dynamics [19], [20].

In general, the solution of a MPC controller can only be pre-computed in the case of unconstrained problems which use linear, time-invariant systems as prediction models. For constrained problems, the solution can also be computed explicitly for each combination of active constraints, this is referred to as Explicit MPC [1]. The control input is then described by a piecewise-affine function of the current state. The number of regions grows exponentially with the total number of constraints, making this approach competitive only for problems with a modest control horizon and number of variables and inequalities. For many applications, solving the optimization problems in real time is the only option, or the most efficient one.

Depending on the problem structure, the fastest approach to solve MPC optimization problems will vary. Important decisions to take into account are the algorithm type, the treatment of the inequalities (for Newton methods, typically Interior Point or SQP with Active Set techniques), the approach taken to solve the required linear algebra operations, the use of exact or approximated second order information, the real-time implementation of the algorithm, and so on. The optimization problem can be also modified beforehand, for example, in other to preserve convexity or make it computationally easier to solve. The computational (software) implementation of the algorithm is equally important in achieving low computational times. Recently, automatic code generation for optimization solvers has become popular. The generation of tailor-made code for each particular optimization problem provides significant gains since it can exploit the fact that the problem size and structure are known beforehand. Examples of well-known code generation tools are CVXGEN [13] and ACADO Toolkit [10].

In this work, a custom solver for the solution of Nonlinear Model Predictive Control (NMPC) problems for Ironless Linear Motors (ILMs) is developed. We consider a nonlinear, position dependent model which accounts for Lorentz and Reluctance forces on the driving and non-driving directions [15]. The model can account for non-perfect geometry, and can be obtained via system identification techniques [14, Chapter 3]. The use of NMPC allows the linear motor to be controlled with optimal control sequences which take into account constraints on the currents.

The developed solver implements a modified Interior Point scheme which allows to be warm-started, with a custom structure-exploiting solution of the required linear algebra operations via block factorization. A code generation routine accelerates the linear algebra operations by fixing the problem size and performing off-line pre-computations to the best extent possible. The developed algorithm can also be used for a class of NMPC problems which use a prediction model

consisting of a linear system with a (time-varying) input nonlinearity. This includes Hammerstein systems as a general case. The developed algorithm is particularly advantageous for prediction models with more inputs than states, for prediction models in which the input is determined by solving a system of (linked) nonlinear equations, and for problems with a large prediction horizon.

The remainder of this thesis is structured as follows. Section II formulates a model predictive control problem for a class of dynamical systems with input nonlinearities. Section III introduces a fast NMPC algorithm and software for the control of dynamical systems with the mentioned structure. In Section IV, a NMPC controller for linear motors is designed and validated in simulation, which makes use of the designed algorithm. In section V, the conclusions are summarized.

A. Basic definitions

Let $\mathbb{N}$ and $\mathbb{R}$ denote the set of natural and real numbers, respectively. $\mathbb{R}^n$ denotes the set of real column vectors of dimension n, and $\mathbb{R}^{m\times n}$ denotes the set of real matrices of size $m \times n$. A zero matrix of size $m \times n$ is denoted as $0_{m\times n}$, and a $n \times n$ identity matrix is denoted as I_n. Let $\mathbb{S}_n^+$ denote the set of symmetric, strictly positive definite matrices of size $n \times n$. Let *blkdiag*$(X_1 \dots X_n)$ denote a block diagonal matrix, with submatrices $X_1 \dots X_n$ as the diagonal elements. Given a vector $x \in \mathbb{R}^n$, let *diag*(x) denote a $n \times n$ diagonal matrix with the elements of x on the diagonal, and let *length*(x) denote the length of the vector x, i.e. n. Given $X \in \mathbb{S}_n^+$, let $chol(X)$ denote a Cholesky factorization of X, i.e., $chol(X) = LL^T$ where L is a real, lower triangular matrix of size $n \times n$. Given a matrix A, the inverse of its transpose is denoted as A^{-T}. Given a vector x, its i-th element is referenced as $x^{[i]}$. A floating point operation (flop) is defined as one addition, substraction, multiplication or division [3, Appendix C].

II. Problem formulation

The main objective of this work is the development of a fast, tailor-made algorithm for the solution of Nonlinear Model Predictive Control (NMPC) problems for a class of dynamical systems with input nonlinearities. The considered class of nonlinear systems was selected based on the modeling of Ironless Linear Motors for controller design purposes. Such systems consist of a nonlinear quasi-static part, due to electromagnetics, and a linear dynamical system, due to moving mechanical parts.. More details about the modeling of ILMs will be given in Section IV of this thesis, and an application example will be provided.

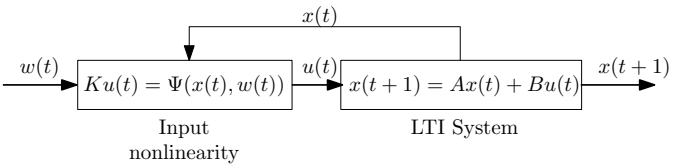

Fig. 1: Linear dynamical system with state-dependent input nonlinearities.

We consider discrete-time dynamical systems with the structure shown in Figure 1. A state-space representation is given by

$$x(t+1) = Ax(t) + Bu(t), \qquad t = 0, 1, 2, \dots \quad (1)$$

$$Ku(t) = \begin{bmatrix} \Psi_1(w(t), x(t)) \\ \vdots \\ \Psi_{n_f}(w(t), x(t)) \end{bmatrix} \coloneqq \Psi(w(t), x(t)), \quad (2)$$

where $t \in \mathbb{N}$ denotes the discrete time index, $x(t) \in \mathbb{R}^{n_x}$ the state vector, $w(t) \in \mathbb{R}^{n_w}$ the input vector to the overall nonlinear system, and $u(t) \in \mathbb{R}^{n_u}$ the input vector to the linear dynamical subsystem. The matrices $A \in \mathbb{R}^{n_x\times n_x}$ and $B \in \mathbb{R}^{n_x\times n_u}$ represent the linear, time-invariant system described by (1). We assume that rank$(A) = n_x$ The functions $\Psi_i : \mathbb{R}^{n_w\times n_x} \to \mathbb{R}^{n_f}$ describe a static nonlinear relationship between the input terms $w(t)$, the state $x(t)$ and $Ku(t)$, with $K \in \mathbb{R}^{n_f\times n_u}$ and $rank(K) = n_u$. Note that a proper discretization for the position-dependent terms is required. Also, observe that the number of nonlinear equalities is n_f, which may be bigger than n_u. Ψ is assumed to be twice differentiable. The mapping $w(t) \to u(t)$ needs to be surjective (each $w(t)$ maps to some $u(t)$), but it might be non-bijective and/or non-convex. In the latter case, the possible implications will be outlined throughout the document. We consider, without loss of generality, that the output of the linear system is $x(t)$, i.e., all the states can be measured. Throughout this document, we refer to $w(t)$ and $u(t)$ as the *nonlinear input* and the *linear input*, respectively, because that is how they relate to the output signal $x(t)$. The linear dynamical system is assumed to have a number of inputs equal to or higher than the number of states.

This problem formulation includes a subset of MPC problems for quasi-LPV systems that received an increased attention recently, see [4] and the references therein. Note that, when disregarding the position dependency of the terms $\Psi(x(t), w(t))$, equations (1)-(2) describe a Hammerstein system (linear dynamical system with a static nonlinear input term).

A nonlinear model predictive control scheme with the dynamical system (1) – (2) used as prediction model is considered in what follows. The associated optimization problem determines the predicted states and inputs at time t, which are given by

$$U_k = \left(u_{0|k}, \dots, u_{N-1|k}\right) \in \mathbb{R}^{n_u\times N}, \quad (3a)$$

$$W_k = \left(w_{0|k}, \dots, w_{N-1|k}\right) \in \mathbb{R}^{n_w\times N}, \quad (3b)$$

$$X_k = \left(x_{1|k}, \dots, x_{N|k}\right) \in \mathbb{R}^{n_x\times N}. \quad (3c)$$

For brevity, we will omit the reference to current discrete time index in the above sequences unless specified otherwise, i.e., $p_{j|k}$ is referred to as p_j for any $p \in \{u, w, x\}$. The optimal predicted inputs and states are noted by U_k^*, W_k^* and X_k^*, and are given by the solution of the following problem:

Optimization Problem 1 (State-dependent Input Nonlinearity NMPC Controller)**.**

$$\min_{U_k, X_k, W_k} \sum_{j=0}^{N-1} L_j(x_j, u_j, w_j) + L_N(x_N), \tag{4}$$

subject to

$$\begin{aligned}
x_0 &= x(t), && && (5a)\\
x_{j+1} &= Ax_j + Bu_j, && \forall j = 0, \ldots, N-1, && (5b)\\
Ku_j &= \Psi(w_j, x_j), && \forall j = 0, \ldots, N-1, && (5c)\\
G_0^u u_0 &\leq f_0^u, && && (5d)\\
G_j^{xu} \left(x_j^T, u_j^T\right)^T &\leq f_j^{xu}, && \forall j = 1, \ldots, N-1, && (5e)\\
G_j^w w_j &\leq f_j^w, && \forall j = 0, \ldots, N-1, && (5f)\\
G_N^x x_N &\leq f_N^x, && && (5g)
\end{aligned}$$

with prediction horizon length N, and stage and terminal cost functions $L_j : \mathbb{R}^{n_x+n_u+n_w} \rightarrow \mathbb{R}_+$ and $L_n : \mathbb{R}^{n_x} \rightarrow \mathbb{R}_+$. Equations (5b) – (5c) represent the discretized system dynamics as defined in this section, and equations (10e) – (10g) represent linear inequality constraints on the predicted states and inputs, with right hand sides $f_j^x \in \mathbb{R}^{l_x}$, $f_j^u \in \mathbb{R}^{l_u}$, $f_j^w \in \mathbb{R}^{l_w}$, and matrices G_j^x, G_j^u, G_j^w of appropriate size. In total, the problem has $m_i = N(n_x + n_u + n_w)$ inequality constraints and $m = N(n_x + n_f)$ equality constraints. x_j, w_j and u_j refer to the predicted states and inputs at time t as defined in (3a) – (3c), with the second subscript omitted for clarity.

In a basic Model Predictive Control implementation, the system is controlled with the policy $w(t) = w_{0|t}^*$.

We consider the following stage and terminal cost formulation:

$$\begin{aligned}
L_i(x_i, u_i, w_i) = &\begin{bmatrix} x_i - x_i^r \\ u_i - u_i^r \end{bmatrix}^T \begin{bmatrix} Q & S \\ S^T & R_u \end{bmatrix} \begin{bmatrix} x_i - x_i^r \\ u_i - u_i^r \end{bmatrix} \\
&+ \left(w_i - w_i^r\right)^T R_w \left(w_i - w_i^r\right), \qquad (6a)
\end{aligned}$$

$$L_N(x_N) = \left(x_N - x_N^r\right)^T P \left(x_N - x_N^r\right) \tag{6b}$$

with quadratic weight matrices Q, R, S, N, P, and reference vectors x_i^r, u_i^r, w_i^r of appropriate dimensions. Furthermore, we assume the stage and terminal cost functions to be convex, this is fulfilled when $\left(\begin{smallmatrix} Q & S \\ S^T & R_u \end{smallmatrix}\right) \in \mathbb{S}_+$, $R_w \in \mathbb{S}_+$, and $P \in \mathbb{S}_+$.

The stage cost formulation (6) comprises a large subset of Model Predictive Control problems. The cross-term S is typically used when penalizing an output channel with direct feed-through. We do not consider any cross-terms for the nonlinear input w.

A. Removal of the position dependency

In Optimization Problem 1, the constraint (5c) depends on the current state. A technique used in model predictive control formulations of position dependent systems is the pre-evaluation of those terms at guess $\hat{X}_k$ of X_k [4]. This technique will be applied to the Linear Motor in Section IV-B1 in order to reduce the computational complexity of the problem and in order to avoid non-convexity. This subsection gives the structure of the modified NMPC optimization problem.

We consider the following change to the structure of Optimization Problem 1: Equation (5c) is replaced by

$$Ku_j = \Psi(w_j, \hat{x}_j) \coloneqq \Psi^j(w_j) \tag{7}$$

where

$$\hat{X}_k = \Gamma(X_{k-1}^*) = (\hat{x}_1, \ldots, \hat{x}_N) \tag{8}$$

and where $\Gamma : \mathbb{R}^{N+1} \rightarrow \mathbb{R}^N$ is a function used to predict the future states. This function is used to remove the dependency on the current state from $\Psi(w_j, x_j)$ in (5c), which is now converted into a time-dependency across the predicted horizon. The modified optimization problem is given by

Optimization Problem 2 (Time-varying Input Nonlinearity NMPC Controller)**.**

$$\min_{U_k, X_k, W_k} \sum_{j=0}^{N-1} L_j(x_j, u_j, w_j) + L_N(x_N), \tag{9}$$

subject to

$$\begin{aligned}
x_0 &= x(t), && && (10a)\\
x_{j+1} &= Ax_j + Bu_j, && \forall j = 0, \ldots, N-1, && (10b)\\
Ku_j &= \Psi^j(w_j), && \forall j = 0, \ldots, N-1, && (10c)\\
G_0^u u_0 &\leq f_0^u, && && (10d)\\
G_j^{xu} \left(x_j^T, u_j^T\right)^T &\leq f_j^{xu}, && \forall j = 1, \ldots, N-1, && (10e)\\
G_j^w w_j &\leq f_j^w, && \forall j = 0, \ldots, N-1, && (10f)\\
G_N^x x_N &\leq f_N^x, && && (10g)
\end{aligned}$$

where the nonlinear dynamics are now time-dependent ($\Psi^j(w_j)$ in (10c)), and the rest of the terms are as in Optimization Problem 1. This modification avoids the potential non-convexity issues caused by the position dependency of (5c). Note that the problem remains non-convex, but now only with respect to w_j.

Note that an appropriate discretization of a continuous-time system is difficult to express in the structure of Figure 1. This is because the nonlinear terms $\Psi(w(t), x(t))$ may vary between samples as $x(t)$ varies continuously, which in turn affects $u(t)$, providing a feedback which is difficult to model. This effect may be compensated for with a proper choice of Γ.

III. Fast nonlinear model predictive control for systems with time-varying input nonlinearity

In this chapter, a fast structure-exploiting algorithm is derived for the solution of Optimization Problem 2, which describes nonlinear model predictive control problems involving a time-dependent input nonlinearity in the prediction model.

An Interior Point scheme with fixed barrier term is implemented in order to solve Optimization Problem 2. In order to accelerate the required linear algebra operations, a block-factorization scheme which exploits the sparsity pattern of the problem will be used. The developed option is especially favourable for problems with relatively large prediction horizons, and problems with a number of inputs which is similar or higher than the number of states.

A. A modified interior point scheme for real time control

A primal-dual interior point scheme is implemented to solve the optimization problem (9) – (10). In this section, a standard version of the algorithm will be outlined, then a variation favourable for real-time model predictive control will be introduced. For simplicity, standard notation is used in this section. In the next sections, the structure of the optimization problem (9) – (10) will be taken into account. In order to introduce the basics of the interior point method, consider the following optimization problem in standard form

$$\min_{z} f(z) \tag{11}$$

$$\text{subject to } g_i(z) = 0, \quad i = 1 \dots m, \tag{12}$$

$$h_i(z) > 0, \quad i = 1 \dots m_i, \tag{13}$$

with optimization variables $z \in \mathbb{R}^n$, cost $f : \mathbb{R}^n \to \mathbb{R}$, m functions $g_i : \mathbb{R}^n \to \mathbb{R}$ and m_i functions $h_i : \mathbb{R}^n \to \mathbb{R}$ such that the set defined by (13) is convex.

A barrier approach is taken to derive the interior point method [17, Section 19.1]. The problem (11) – (13) is rewritten as

$$\min_{z,s} \; f(z) - \mu \sum_{i=1}^{m_i} \log s_i \tag{14}$$

$$\text{subject to } g_i(z) = 0, \quad i = 1 \dots m, \tag{15}$$

$$h_i(z) - s_i = 0, \quad i = 1 \dots m_i, \tag{16}$$

where $\mu > 0$ is a parameter and the terms $s_i \in \mathbb{R}$ are slack variables. The term $-\mu \sum_{i=1}^{m_i} \log s_i$ is a barrier term replacing the inequality constraints. Note that this also introduces the implicit constraint $s_i > 0$. Problems (14) – (16) and (11) have the same solution when $\mu = 0$ and the constraint $s_i > 0$ is enforced.

The first order (KKT) optimality conditions for (14) can be expressed as (when $s_i > 0 \; \forall i$)

$$\nabla f(z) - C^T(z)\nu - J^T(z)\lambda = 0, \tag{17a}$$

$$-\mu e + S\lambda = 0, \tag{17b}$$

$$g(z) = 0, \tag{17c}$$

$$h(z) - s = 0, \tag{17d}$$

where $g(z) = (g_1(z), \dots, g_m(z))^T$ are all equality constraints stacked, $h(z) = (h_1(z), \dots, h_{m_i}(z))^T$ are all inequality constraints stacked, $s = \text{diag}(s_1, \dots, s_{m_i})$ are slack variables, $\lambda \in \mathbb{R}^{m_i}$, $\nu \in \mathbb{R}^m$ are multipliers associated with the inequality and equality constraints (such that $\forall i$, $\lambda_i > 0$), $e = (1, \dots, 1)^T$ and

$$C(z) = \left[\nabla g_0^T(z), \dots, \nabla g_{m_i}^T(z)\right]^T, \tag{18}$$

$$J(z) = \left[\nabla h_0^T(z), \dots, \nabla h_{m_i}^T(z)\right]^T. \tag{19}$$

The solution of the original problem (11) may be obtained by solving a sequence of modified problems (14) for a decreasing value of μ. For low values of μ, the solution of the modified problem lies closer to the one of the original problem. Although an optimal solution of the modified problem will not be an optimal solution of (11), all solutions are primal feasible for the original problem, i.e. $g_i(z) = 0$ and $h_i(z) > 0$ are respected. When solving the NMPC problem (9) – (10), $g_i(z) = 0$ means that the model dynamics are respected across the prediction horizon.

Problem (14) only has equality constraints, but with the implicit constraint that $s_i > 0$ in the cost function. Applying Newton's method to (14) involves solving the system

$$\begin{bmatrix} \nabla^2_{xx}\mathcal{L} & 0 & -C^T & -J^T \\ 0 & Z & 0 & S \\ C & 0 & 0 & 0 \\ J & -I & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta s \\ \Delta \nu \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} \nabla f - C^T\nu - J^T\lambda \\ -\mu e + S\lambda \\ g(z) \\ h(z) - s \end{bmatrix}, \tag{20}$$

with $Z = diag(z)$, Δz, Δs, $\Delta \nu$ and $\Delta \lambda$ represent Newton steps, and $\nabla^2_{xx}\mathcal{L}$ denotes the Hessian of the Lagrangian of problem (14), which is given by

$$\mathcal{L} = f(z) - \nu^T g(z) - \lambda^T (h(z) - s). \tag{21}$$

The terms Δs and $\Delta \lambda$ are removed from (20) by eliminating the corresponding rows, resulting in the system (with the dependency of each term on z, ν or μ)

$$\begin{bmatrix} \Phi(z,\nu,\mu) & C^T(z) \\ C(z) & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta \nu \end{bmatrix} = \begin{bmatrix} -r_d(z,\nu,\mu) \\ -g(z) \end{bmatrix}, \tag{22}$$

with new terms

$$\Phi(z,\nu,\mu) = \nabla^2_{xx}\mathcal{L}(z,\nu) + \mu J^T(z)\Sigma(z)J(z), \tag{23}$$

$$r_d(z,\nu,\mu) = \nabla f(z) - C^T(z)\nu - \mu J^T(z)d(z), \tag{24}$$

and in which

$$d(z) = \left(\frac{1}{h_1(z)}, \dots, \frac{1}{h_{m_i}(z)}\right), \tag{25}$$

$$\Sigma(z) = \text{diag}(d(z))^2. \tag{26}$$

After the transformation, the first order optimality conditions can be expressed as $g(z) = 0$ and $r_d(z,\nu,\mu) = 0$. The conversion from (20) to (22) reduces the size of the problem and yields a favourable structure for solving the linear system. The reduced system (22) may become ill conditioned because, as $\mu \to 0$, some terms in $d(z)$ diverge to ∞ while others remain bounded. For MPC problems, the term Φ will be block diagonal with relatively small blocks, which reduces the effects of this ill-conditioning. Furthermore, high quality solution approximations may be obtained with modest values of μ, as will be shown in Section IV.

1) Standard Interior Point implementations: In standard Interior Point implementations, the original problem (11) is solved via the successive solution of approximate subproblems (14) for decreasing values of μ. The method is started at a guess z_0 and ν_0 which satisfies the inequality constraints. Then, each subproblem is solved by performing multiple iterations until the norm of the residual (r_d, g) is small enough. The solution of each subproblem is used to start the next. The barrier parameter μ is typically reduced geometrically, i.e. $\mu \leftarrow k\mu$ with $k \in (0, 1)$. Each iteration is performed by computing the search directions Δz and $\Delta \nu$ with (22), then determining a step size $s \in (0, s^{max})$ by performing a line search on the norm of a merit function or residual, and then updating the current guess with $z_0^+ \leftarrow z_0 + s\Delta z$

and $\nu_0^+ \leftarrow \nu_0 + s\Delta\nu$ ($s^{max} \in (0,1)$ is an upper bound which ensures that the updated point satisfies the inequality constraints). See, for example, [17, Section 19.4] for a formal definition.

The use of a decreasing sequence of values for μ is intended to aid in convergence speed. For a value of μ too big, the solution of the barrier problem differs too much from the one of the original problem. On the other hand, when μ is too small, the contribution of the barrier function to the overall Hessian and optimality residuals is small until later iterations (this contribution is given by the rightmost terms in (23) and (24)). Starting with a small μ causes the search directions to aim towards the solution of the unconstrained problem, until the border of the inequality constraints is reached. Once in the border, convergence is slowed down.

2) Modified interior point scheme: In order to apply MPC in real time, the optimization problem (9) – (10) needs to be solved successively and in a fixed time period. Each pair of successive optimization problems (9) – (10) is similar, and the predicted inputs and states U_k, W_k and X_k can be used to compute an initial guess for the solution of the next optimization problem (warm start). In other words, before solving the problem for z and ν, we can obtain a z^{ws} and ν^{ws} which are sufficiently close. However, Interior Point methods do not work well with warm start techniques, since the solution of the previous problem was obtained with a different barrier parameter μ.

A variation which uses a constant barrier parameter μ is considered in Algorithm 1 given below. This allows the problem to be warm-started. A value of μ relatively small needs to be selected: this will provide a solution which is close enough to the one of the original problem (11), but the contribution of the barrier parameters to the Hessian and optimality residuals will be significant since the method starts at the points z^{ws} and ν^{ws}. The value of μ will be selected via simulations. A similar scheme with a fixed barrier term has been applied to linear MPC problems in [19]. In the Real Time Iteration scheme for Nonlinear MPC proposed in [5] (implemented in the software ACADO Toolkit [11]), also a limited number of SQP iterations are performed to solve the MPC problem, achieving convergence in time thanks to the warm start.

In Algorithm 1, z^{ws} and ν^{ws} represent an initial guess for the values of z and ν. The parameters $\alpha \in (0, 0.5)$ and $\beta \in (0,1)$ are used for the line search procedure, and $\gamma \in (0,1)$ is used to numerically determine s^{max}. The following merit function is used for the line search procedure:

$$\phi(z,\tau) = f(z) - \mu \sum_{i=1}^{m_i} \log h_i(z) + \tau ||g(z)||_2, \qquad (27)$$

where $\tau \in \mathbb{R}^+$ is a weighting parameter. Since, in real time, the amount of computational time available is limited, the number of iterations is truncated at K^{max}. The use of a fixed iteration number is also made possible by the use of a fixed barrier parameter μ because, in case of early termination of the algorithm (before convergence), the current z will be close to the optimum.

Algorithm 1 Fixed Barrier Interior Point Method

given z^{ws}, ν^{ws}, μ, α, β, δ, τ
1: $z \leftarrow z^{ws}$
2: $\nu \leftarrow \nu^{ws}$
3: $k \leftarrow 0$
4: **while** $k < K^{max}$ **do**
5: *Compute search directions* Δx, $\Delta\nu$ *using* (22)
6: Calculate s^{max} satisfying $h(z) > 0$
7: $s^{max} \leftarrow 1$
8: **while** $h_i(z + s^{max}\Delta z) \leq 0$, $i \in (1, \ldots, m_i)$ **do**
9: $s^{max} = \delta s^{max}$
10: **end**
11: Perform line search on merit function
12: $s \leftarrow s^{max}$
13: **while** $\phi(z+s\Delta z, \tau) > \phi(z,\tau) + \alpha s \nabla_z \phi(z,\tau)^T \Delta z$
14: $s = \beta s$
15: **end**
16: Update solution
17: $z \leftarrow z + s\Delta z$
18: $\nu \leftarrow \nu + s\Delta\nu$
19: $k \leftarrow k + 1$
20: **end**

The amount of time required to execute Algorithm 1 is largely determined by the solution time of step 5, which involves the solution of (22). The complexity of this step is iteration independent (it depends on the problem structure and size, but not on any parameters of the algorithm). In the following subsections, an efficient algorithm for the solution of this step will be derived.

B. Problem structure

In this subsection, the structure the NMPC optimiztion problem (9) – (10) is detailed in terms of the interior-point problem (14) – (16). This structure will be used in the next subsection to derive an efficient factorization method for the KKT matrix (22).

The optimization variables U_k, W_k, X_k are sorted and stacked in the optimization vector

$$z = (u_0, w_0, x_1, \ldots, u_{N-1}, w_{N-1}, x_N) \in \mathbb{R}^n, \qquad (28)$$

where the dependency on k (second subscript) has been omitted for ease of notation, and $n = N(n_x + n_u + n_w)$ is the number of optimization variables. We assign a dual variable (Lagrange multiplier) to each equality constraint, such that $\nu_i^l \in \mathbb{R}^{n_x}$ are the Lagrange multipliers assigned to (10b) for $j = i$, and similarly $\nu_i^{nl} \in \mathbb{R}^{n_f}$ are the ones assigned to (10c). The dual variables are stacked in the vector

$$\nu = \left(\nu_0^l, \nu_0^{nl}, \ldots, \nu_{N-1}^l, \nu_{N-1}^{nl}\right) \in \mathbb{R}^m, \qquad (29)$$

where $m = N(n_x + n_f)$ is the number of equality constraints. This proposed arrangement of z and ν will reduce the computational complexity of the subsequent linear algebra operations.

With the arrangement (28) of the optimization variables, the Hessian of the Lagrangian has the structure

$$\Phi(z,\nu,\mu) \coloneqq \begin{bmatrix} \Phi_0(z,\nu,\mu) & & 0 \\ & \ddots & \\ 0 & & \Phi_N(z,\mu) \end{bmatrix} \in \mathbb{R}^{n\times n}, \quad (30)$$

where the blocks $\Phi_0 = \Phi_0(z,\nu,\mu)$, $\Phi_i = \Phi_i(z,\nu,\mu)$ and $\Phi_N(z,\mu) = \Phi_N$ have the structure

$$\Phi_0 = \begin{bmatrix} \Phi_0^u & 0 \\ 0 & \Phi_0^w \end{bmatrix} \in \mathbb{R}^{(n_u+n_w)^2}, \quad (31a)$$

$$\Phi_i = \begin{bmatrix} \Phi_i^{xu} & 0 \\ 0 & \Phi_i^w \end{bmatrix} \in \mathbb{R}^{(n_x+n_u+n_w)^2}, \quad i{=}1\ldots N{-}1, \quad (31b)$$

$$\Phi_N = \Phi_N^x \in \mathbb{R}^{(n_x)^2}, \quad (31c)$$

with sub-blocks

$$\Phi_0^u = R + \mu G_0^{uT}\Sigma_0^u G_0^u \qquad \in \mathbb{R}^{{n_u}^2}, \quad (32a)$$

$$\Phi_N^x = P + \mu G_N^{x\ T}\Sigma_N^x G_N^x \qquad \in \mathbb{R}^{{n_x}^2}, \quad (32b)$$

$$\Phi_i^w = \Omega + \mu G_i^{wT}\Sigma_i^w G_i^w + \nabla_{ww}^2 F(w)\nu_i^{nl} \quad \in \mathbb{R}^{{n_w}^2}, \quad (32c)$$

$$\Phi_i^{xu} = \begin{bmatrix} Q & S \\ S^T & R \end{bmatrix} + \mu G_i^{xuT}\Sigma_i^{xu} G_i^{xu} \in \mathbb{R}^{(n_x+n_u)^2}, \quad (32d)$$

and in which the terms Σ_i^j are given by

$$\Sigma_i^j = \text{blkdiag}\left(\frac{1}{\sigma_i^{j\,[1]}}, \ldots, \frac{1}{\sigma_i^{j\,[k]}}\right)^2, \qquad i = 0,\ldots,N,$$
$$j \in \{u,x,w,xu\}, \quad k = \text{length}(\sigma_i^j), \quad (33)$$

with

$$\sigma_i^{xu} = f_i^{xu} - G_i^{xu}(x_i,u_i)^T, \quad \sigma_i^w = f_i^w - G_i^w w_i, \quad (34a)$$

$$\sigma_N^x = f_N^x - G_N^x x_N, \qquad \sigma_0^u = f_0^u - G_0^u u_0. \quad (34b)$$

In equations (32a) – (32d), the terms with Σ_i^j represent the contributions of the barrier function to the overall Hessian.

Remark 1 (Positive definiteness of the Hessian)**.** *In order to satisfy the second order sufficient conditions (SOSCs) of the intermediate Newton steps [17, Chapter 3], the Hessian Φ needs to be positive definite on the null space of the linearized constraints C. Note that the subterms Φ_i^w are not guaranteed to be positive definite, because they depend on ν, which is unbounded. In case this assumption does not hold, a number of techniques can be applied, such that using an approximation $\tilde{\Phi} = \Phi + \gamma I$, which sacrifices part of the second-order information (See e.g. [17, Section 19.3]). With this problem structure, and adequate γ can be possibly obtained analytically from* (32b)*. We assume Φ to be positive definite from this point onwards.*

With the given arrangement of z and ν, the constraints gradient matrix has the structure

$$C(z) \coloneqq \begin{bmatrix} E_0 & F_1 & 0 & \cdots & 0 & 0 \\ 0 & E_1 & F_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & E_{N-2} & F_{N-1} & 0 \\ 0 & 0 & \cdots & 0 & E_{N-1} & F_N \end{bmatrix} \in \mathbb{R}^{m\times n} \quad (35)$$

with components

$$E_0 \coloneqq \left[\begin{array}{cc|c} B & & 0_{n_x\times n_w} \\ K & & \psi_0(w_0) \end{array}\right] = \left[\begin{array}{c|c} E_0^L & E_0^R \end{array}\right], \quad (36a)$$

$$E_i \coloneqq \left[\begin{array}{cc|c} A & B & 0_{n_x\times n_w} \\ 0 & K & \psi_i(w_i) \end{array}\right] = \left[\begin{array}{c|c} E_i^L & E_i^R \end{array}\right], i{=}1\ldots N{-}1 \quad (36b)$$

$$F_i \coloneqq \left[\begin{array}{cc|c} -I_{n_x} & 0_{n_x\times n_u} & 0_{n_x\times n_w} \\ 0_{n_f\times n_x} & 0_{n_f\times n_u} & 0_{n_f\times n_w} \end{array}\right] = \left[\begin{array}{c|c} F_i^L & 0 \end{array}\right], i{=}1\ldots N \quad (36c)$$

$$F_N \coloneqq \begin{bmatrix} -I_{n_x} \\ 0_{n_f\times n_x} \end{bmatrix}, \quad (36d)$$

where the matrices E_0, E_i and F_i have been partitioned as shown, and $\psi_i(w_i)$ represents a linearization of the nonlinear equality constraints at $w = w_i$,

$$\psi_i(w_i) \coloneqq \frac{\partial \Psi^i(w)}{\partial w} = \begin{bmatrix} \frac{\partial \Psi_1^i}{\partial w^{[1]}} & \cdots & \frac{\partial \Psi_1^i}{\partial w^{[n_w]}} \\ \vdots & & \vdots \\ \frac{\partial \Psi_{n_f}^i}{\partial w^{[1]}} & \cdots & \frac{\partial \Psi_{n_f}^i}{\partial w^{[n_w]}} \end{bmatrix}_{w=w_i} \in \mathbb{R}^{n_f\times n_w}. \quad (37)$$

C. Structured solution of the Newton step

In this subsection, a structured algorithm for the solution of Newton steps is derived. The greatest computational cost in optimization algorithms lies in the calculation of the primal and dual search directions $\Delta z \in \mathbb{R}^n$ and $\Delta v \in \mathbb{R}^m$, which are given by the solution of the KKT system

$$\begin{bmatrix} \Phi(z_0,\nu_0,\mu_0) & C^T(z_0) \\ C(z_0) & 0 \end{bmatrix} \begin{bmatrix} \Delta z \\ \Delta\nu \end{bmatrix} = \begin{bmatrix} -r_d(z_0,\nu_0,\mu_0) \\ -r_p(z_0) \end{bmatrix}, \quad (38)$$

where $\Phi(z_0,\nu_0,\mu_0) \in \mathbb{S}_n^+$ is either the actual Hessian of the Lagrangian of problem (14), or a positive-definite approximation, and $C(z_0) \in \mathbb{R}^{m\times n}$ is the constraints gradient. Both terms are evaluated at the current guess (z_0,ν_0) and for the current barrier parameter $\mu = \mu_0$.

The KKT matrix for this particular optimization problem shows a specific structure which can be exploited. In particular, $\Phi(z_0,\nu_0)$ is positive-definite and block diagonal, the rows and columns of $C(z_0)$ have been arranged such that only consecutive terms are connected (in terms of horizon index), and some terms in these matrices may be independent of current guess (z_0,ν_0). We will use block elimination on the KKT matrix in order to expose its sparsity pattern. Block elimination has been applied to the solution of Linear MPC optimization problems in [19] and [7, Chapter 4] (where it is referred as the Schur Complement Method). In [6], the same method is outlined with a more general notation and some modifications. Here we derive a new formulation for

the special case of Nonlinear MPC problems with the input nonlinearity structure defined in this chapter.

From here onwards, the notation is simplified such that $\Phi = \Phi(z_0, \nu_0, \mu_0)$ and $C = C(z_0)$. By taking the Schur complement of the KKT matrix on the block Φ, the linear system becomes

$$Y\Delta\nu = \beta, \tag{39}$$
$$\Phi\Delta z = -r_d - C^T\Delta\nu, \tag{40}$$

with

$$Y = C\Phi^{-1}C^T \in \mathbb{R}^{m\times m}, \tag{41}$$
$$\beta = r_p - C\Phi^{-1}r_d \in \mathbb{R}^m, \tag{42}$$

in which both matrices involved in a linear system (Y and ψ) are now positive definite. This provides a computational advantage in solving the corresponding linear systems, since Cholesky factorizations can be used.

Remark 2. *$Y = C\Phi^{-1}C^T$ is positive definite when $null(C^T) = \{0\}$ and because of the assumption $\Phi \in \mathbb{S}^n_+$. It can be verified that $null(C^T) = \{0\}$ holds when $rank(\begin{bmatrix}K & \psi^i(w_i)\end{bmatrix}) = n_f \quad \forall i$, i.e. when the linearized equality constraints at the current iteration are linearly independent. It should be verified for every particular application whether the rank of C can decrease for all possible values of ψ. Note that, in some applications, dropping all but one of the linearly dependent linearized constraints may suffice to calculate a search direction, however, this could be a symptom that the problem is converging towards a stationary point which does not fulfill all equality constraints.*

The structured solution of (39)-(42) comprises the following steps:

i) Factorize Φ.
ii) Calculate Y and β.
iii) Factorize Y.
iv) Solve (39) for $\Delta\nu$.
v) Solve (40) for Δz.

Each step will be outlined, commenting the possible structure exploitations and the computational cost in terms of floating point operations (flops). We use the definition of flop as one addition, substraction, multiplication or division [3, Appendix C]. Remarks about this definition are provided in Section III-D. The considered costs for some basic linear algebra operations are summarized in Table I.

Linear algebra operation costs (flops) [3]		
Operation	Input size	Cost
Matrix multiplication $C = AB$	$A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times p}$	2mnp
Cholesky decomposition $Y = LL^T$	$Y \in \mathbb{S}^n_+$	$n^3/3$
Banded Cholesky $Y = LL^T$	$Y \in \mathbb{S}^n_+$, $bw(Y) = k \ll n$	nk^2
Forward substitution $XL = B$	$L \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$	mn^2

TABLE I: Basic linear algebra operation costs.

i) Factorization of Φ: A Cholesky factorization of Φ is performed, such that $\Phi = L^\Phi L^{\Phi^T}$. Given the block diagonal structure of Φ, each of its blocks can be factored independently, that is

$$\Phi_i^j = L_i^j L_i^{j^T}, \qquad i = 0\dots N, \quad j \in \{xu, w\}. \tag{43}$$

Assuming dense blocks, the total cost is $N\left((n_x+n_u)^3 + n_w^3/3\right)$ flops. In case the problem is state-input separable (in both the stage cost an inequality constraints), the cost is lowered to $1/3N\left(n_x^3+n_u^3+n_w^3\right)$ flops. In case of diagonal blocks (fulfilled when P, Q are diagonal, $S = 0$, and box inequality constraints), there are no costs associated with this step.

In special cases, it may be possible to pre-compute some factorizations. In the general case, it is not possible because $\Phi = \Phi(z_0, \nu_0, \mu_0)$. The dependence on z comes from the contribution to the Hessian of the barrier function (terms with Σ in Equations (32a) – (32d)). When some x_i and u_i is not affected by any inequality constraints, the factorization affecting its block ($\Phi_i^{xu} = L_i^{\Phi^{xu}} L_i^{\Phi^{xu}T}$) can be pre-computed. This is, in general, not the case for Φ_i^w since it also depends on the dual variables.

ii) Calculation of Y and β: With the arrangement (29) of the Lagrange multipliers, $Y = C\Phi^{-1}C^T$ will have the block tridiagonal structure

$$Y \coloneqq \begin{bmatrix} Y_{11} & Y_{21}^T & 0 & \cdots & 0 & 0 \\ Y_{21} & Y_{22} & Y_{32}^T & \cdots & 0 & 0 \\ 0 & Y_{32} & Y_{33} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & Y_{N-1,N-1} & Y_{N,N-1}^T \\ 0 & 0 & 0 & \cdots & Y_{N,N-1} & Y_{N,N} \end{bmatrix} \in \mathbb{R}^{m\times m}, \tag{44}$$

where

$$Y_{ii} = E_{i-1}\Phi_{i-1}^{-1}E_{i-1}^T + F_i\Phi_{i-1}^{-1}F_i^T, \tag{45a}$$
$$Y_{i+1,i} = E_i\Phi_i^{-1}F_i^T, \tag{45b}$$

with $Y_{ii}, Y_{i+1,i} \in \mathbb{R}^{(n_x+n_f)^2}$. Each of the three terms in (45a) – (45b) is solved with the block factorization of Φ.

The term $E_i\Phi_i^{-1}E_i^T$ can be obtained by performing two independent operations. By dividing the blocks $E_i = \begin{bmatrix}E_i^L | E_i^R\end{bmatrix}$ as in (36a) – (36b), and considering the block diagonal structure of Φ_i (see (31b)),

$$E_i\Phi_i^{-1}E_i^T = E_i^L\Phi_i^{xu-1}E_i^{L^T} + E_i^R\Phi_i^{w-1}E_i^{R^T}, \tag{46}$$

where, by using the factorizations of each Φ_i block,

$$\begin{aligned} &E_i\Phi_i^{-1}E_i^T \\ &= E_i^L\left(L_i^{xu}L_i^{xu^T}\right)^{-1}E_i^{L^T} + E_i^R\left(L_i^wL_i^{w^T}\right)^{-1}E_i^{R^T}, \\ &= \underbrace{\left(E_i^LL_i^{xu^{-T}}\right)}_{\pi_i}\left(E_i^LL_i^{xu^{-T}}\right)^T + \underbrace{\left(E_i^RL_i^{w^{-T}}\right)}_{\theta_i}\left(E_i^RL_i^{w^{-T}}\right)^T, \\ &= \pi_i\pi_i^T + \theta_i\theta_i^T. \end{aligned} \tag{47}$$

Similarly, the term $F_i\Phi_{i-1}^{-1}F_i^T$ takes the form

$$\begin{aligned} F_i^L\Phi_{i-1}^{-1}F_i^T &= \underbrace{\left(F_i^LL_i^{xu^{-T}}\right)}_{\phi_i}\left(F_i^LL_i^{xu^{-T}}\right)^T, \\ &= \phi_i\phi_i^T. \end{aligned} \tag{48}$$

The term $E_i\Phi_i^{-1}F_i^T$ can be then expressed as a combination of two of the previous terms,

$$E_i\Phi_{i-1}^{-1}F_i^T = \underbrace{\left(E_iL_i^{xu^-{}^T}\right)}_{\pi_i}\underbrace{\left(F_iL_i^{xu^-{}^T}\right)^T}_{\phi_i^T}, \\ = \pi_i\phi_i^T. \tag{49}$$

Note that, in (48) and (49), the rightmost n_w zeros from (36c) have removed any influence from Φ_i^w and E_i^R. The matrices π_i, θ_i, ϕ_i are obtained by solving

$$E_i^L = \pi_i L_i^{xu^T} \tag{50a}$$
$$E_i^R = \theta_i L_i^{w^T} \tag{50b}$$
$$F_i^L = \phi_i L_i^{xu^T} \tag{50c}$$

which involves three backward substitution operations, since the matrices $L_i^{j^T}$ are lower triangular. Then, the components of Y can be determined by

$$Y_{ii} = \pi_{i-1}\pi_{i-1}^T + \theta_{i-1}\theta_{i-1}^T + \phi_i\phi_i^T, \tag{51a}$$
$$Y_{i+1,i} = \pi_i\phi_i^T. \tag{51b}$$

The same operations are performed for $i = 0 \dots N$, note, however, that the matrix sizes are different for $i = 0$ and $i = N$, but for clarity we have used the same notation for all values of i. The total cost for the $3N$ required forward substitution operations is $N\left(n_x + n_f\right)\left(2(n_x + n_u)^2 + n_w^2\right)$ flops, and the matrix self-multiplications in (51a) – (51b) have a total cost of $2N(n_x + n_f)(n_x + n_u)^2 + N\left(n_x + n_u)((n_x + n_f)^2 + n_w^2\right)$ flops. The leading terms, when considering both operations, are $4N(n_x + n_f)(n_x + n_u)^2 + Nn_w^2(2n_x + n_u + n_f)$. On an implementation side, note that the matrices E_i^L and F_i^L do not depend on the current iterate.

β is obtained by solving (42). The term $\Phi^{-1}r_d$ is calculated with the block factorization of Φ, and the structure of C is exploited for the matrix multiplication. The cost of this operation is negligible and grows linearly with N.

iii) Factorization of Y: Y is a block tridiagonal matrix with the structure shown in (44). A Cholesky factorization is performed, such that

$$Y = L^Y L^{Y^T}. \tag{52}$$

Given the block tridiagonal structure of Y, its factorization takes the form

$$L^Y = \begin{bmatrix} L_{11}^Y & 0 & 0 & \cdots & 0 & 0 \\ L_{21}^Y & L_{22}^Y & 0 & \cdots & 0 & 0 \\ 0 & L_{32}^Y & L_{33}^Y & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & L_{N-1,N-1}^Y & 0 \\ 0 & 0 & 0 & \cdots & L_{N,N-1}^Y & L_{NN}^Y \end{bmatrix} \tag{53}$$

where $L_{ii}^Y \in \mathbb{R}^{(n_x+n_f)^2}$ are lower triangular and $L_{i+1,i}^Y \in \mathbb{R}^{(n_x+n_f)^2}$ are, in general, dense. A Cholesky factorization of a block tridiagonal matrix is performed by successively solving

$$L_{11}^Y L_{11}^{Y^T} = Y_{11}, \tag{54a}$$
$$L_{ii}^Y L_{i+1,i}^{Y^T} = Y_{i+1,i}, \qquad i = 1, \dots, N-1, \tag{54b}$$
$$L_{ii}^Y L_{i,i}^{Y^T} = Y_{ii} - L_{i,i-1}^Y L_{i,i-1}^{Y^T}, \qquad i = 2, \dots, N, \tag{54c}$$

in which steps (54a) and (54c) involve obtaining L_{ii}^Y via a Cholesky factorization of the right hand side, and step (54b) involves a forward substitution in order to obtain $L_{i+1,i}^Y$. The total costs are $1/3N(n_x+n_f)^3$ for (54a) and (54c), and $(N-1)(n_x+n_f)^3$ for (54b).

In case the stage cost and inequality constraints are state-input separable, and because of the particular input nonlinearity structure, the cost of (54b) can be further reduced. The left hand terms L_{ii}^Y and $L_{i+1,i}^{Y^T}$ are partitioned in the blocks

$$L_{ii}^Y = \left[\begin{array}{c|c} L_A & 0 \\ \hline V_C & L_B \end{array}\right], \qquad L_{i+1,i}^{Y^T} = \left[\begin{array}{c|c} W_A & W_C \\ \hline W_D & W_B \end{array}\right], \tag{55}$$

so that the upper left blocks are of size $n_x \times n_x$ and the lower right ones are $n_f \times n_f$, and the right hand term of (54b) is partitioned as (see Appendix A)

$$Y_{i+1,i} = \left[\begin{array}{c|c} Y_{i+1,i}^A & 0 \\ \hline 0 & 0 \end{array}\right], \tag{56}$$

with blocks of the same size. Then, $L_{i+1,i}^{Y^T}$ can be obtained by solving

$$L_A W_A = Y_{i+1,i}^A \tag{57a}$$
$$L_B W_D = -V_C W_A \tag{57b}$$
$$W_C = 0 \tag{57c}$$
$$W_B = 0 \tag{57d}$$

which involves a forward substitution to obtain W_A in (57a) and a multiplication plus another forward substitution in (57b), with a total cost of $(N-1)(n_x^3 + n_x^2 n_f + n_x n_f^2)$, which is notably inferior to the cost of a dense solution of (54b), especially when $n_f > n_x$.

This comes directly from the multiplication of the terms in (55), and the observation that $null(L_A) = \{0\}$ and $null(L_B) = \{0\}$. This step can also be performed (in theory less efficiently) with a banded solver, see Appendix A.

iv-v) Calculation of $\Delta\nu$ and Δz: Equations (39) and (40) are solved with the block factorizations of Y and Φ. The cost of this operation is negligible and grows linearly with N.

D. Complexity analysis

In this section, the computational complexity of the block factorization routine is assessed. The leading (asymptotic) cost of the proposed structure-exploiting solution of the Newton step, in terms of floating point operations and when considering the general case, is

$$N\Bigg(\frac{5}{3}(n_x+n_f)^3 + \frac{1}{3}\left((n_u+n_x)^3 + n_w^3\right) \\ + \left(2(n_u+n_x)^2 + n_w^2\right)(n_f+n_x) \\ + 2(n_f+n_x)(n_u+n_x)^2 \\ + (n_u+n_x)\left((n_f+n_x)^2 + n_w^2\right)\Bigg). \tag{58}$$

The full cost (not just the leading terms) is shown in Figure 2 for different combinations of n_x, n_u, n_f and n_w, with $N = 10$ (note that the cost grows linearly with N). This cost largely dominates the cost of a full interior point step, as defined

in Algorithm 1. In the figure, three cases are considered: the general case, the case in which the stage cost and inequality constraints are input-state separable, and the case with diagonal stage cost and box constraints. The proposed approach is especially effective when considering over-actuated systems (n_w similar or bigger than n_u or n_x).

The proposed block factorization approach is compared with the solution of (22) via a banded LDL^T factorization (after rearranging its rows and columns). This is one of the main approaches used in fast (linear) MPC [7, Chapter 2.2.2]. With this approach, the computational complexity also grows linearly with N. In this particular problem, a favourable rearrangement of the KKT matrix can be obtained with a bandwidth (maximum distance from the diagonal of its non-zero elements) of $2n_x+n_u+n_f+n_w$. The (again theoretical) cost of performing an Interior Point operation with this approach is added to Figure 2. This approach offers no gains for the special cases considered. In terms of (raw) flops, the block factorization approach is shown to be between 50% and 85% faster for the parameter combinations considered, due to the better exploitation of the sparsity pattern. Another approach to factorize a KKT matrix consists on removing the equality constraints and some variables in order to form smaller, dense quadratic programs, with a cost of order $\mathcal{O}(N^3 n_w^3)$ [12]. For nonlinear problems, the condensing needs to be performed online at each iteration. This approach has not been analysed in this section because our problem formulation focuses on systems with more inputs than states, for which the removal of variables does not outweight breaking the sparsity pattern. (A solver which utilizes condensing will be benchmarked in Section V).

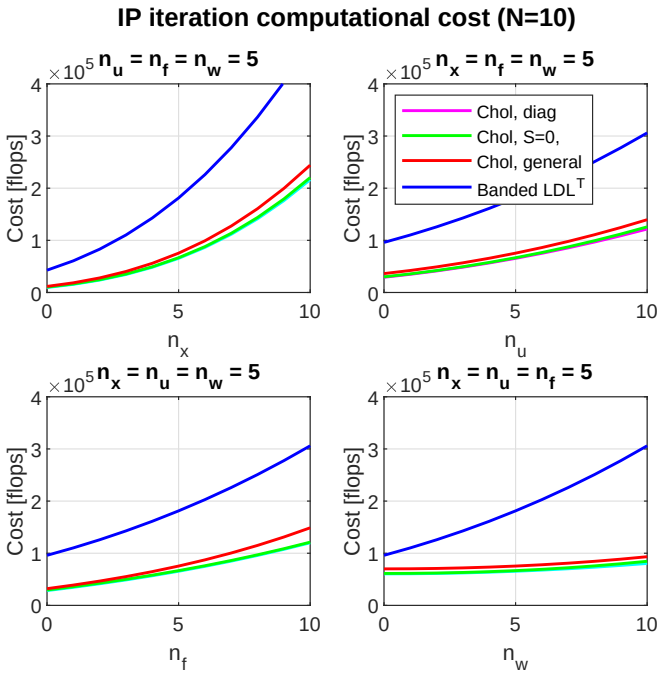

Fig. 2: Computational cost of an Interior Point iteration, in number of floating point operations, for different values of n_x, n_u, n_f and n_w, with $N = 10$. In each of the subplots, all values are fixed, except the one labeled in the horizontal axis.

While the number of required floating point operations is a good indicator of the algorithm complexity, it is not an exact predictor of the required computational time. A modern processor can usually perform multiple operations in one instruction if the required data is properly arranged in memory, on the other hand, retrieving scattered data in the memory may require a handful of instructions. In high speed optimal control, the arrangement of the data in memory is the biggest bottleneck, outweighing the algorithmic cost. See e.g. [8]. For a better implementation, focus is punt on algorithms which allow sequential access to the required data. Most steps of the block factorization algorithm of this chapter can be executed in a sequential and parallelized way, as will be explained in Appendix B. In contrast, the LDL^T factorization of a medium-large matrix requires row pivoting operations. This involves a non-sequential access to the elements in the matrices when they are stored in memory and, at some point, the memory access operations bottleneck the floating point operations. This makes the LDL^T approach less favourable, especially for problems with a large size.

E. Algorithm implementation

A software implementation of the algorithm described in this section has been developed and is described in Appendix B. It comprises a code generation routine which, given a problem structure, generates efficient, tailor-made C code for every particular problem, and makes use of fast routines for the required linear algebra operations. The appendix also describes the approaches taken by other implementations of fast MPC solvers and compares them with the developed solver.

The derived software tool generates solvers which are tailor-made for the structure (9) – (10) with dynamics (1)-(2), being especially advantageous for cases in which $n_w > n_x$ or $n_f > n_x$, and can also generate solvers for linear MPC problems, being the block-factorization procedure especially useful for LMPC problems with high prediction horizons.

IV. Application: Fast Nonlinear Model Predictive Control for Linear Motors

In this section, a Nonlinear Model Predictive Control scheme for Linear Motors is implemented with the fast algorithm derived in Section III, validated by simulation. A collection of techniques are applied in order to achieve a balance between solution speed and accuracy for the NMPC optimization problem.

This work focuses on a fast application of Model Predictive Control. Topics regarding the quality of the control (stability, reference tracking, etc.) are not the main goal of this thesis.

A. System description and modelling

In this subsection, the dynamics of a Ironless Linear Motor (ILM) are introduced, together with the model and assumptions that we consider. Figure 3 shows a cross-section view of a ILM. The stator contains two arrays of permanent magnets mounted on alternating directions. The translator moves

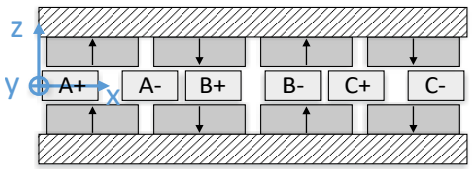

Fig. 3: Cross-section representation of a coreless linear motor with one set of coils [15].

linearly between the permanent magnets and contains one or more sets of three-phase coils (one set is represented in the figure).

A position-dependent model of the linear motor that takes into account the Lorentz and reluctance forces is used [15], [16]. The model can be identified from a physical system [14], and allows using the Lorentz and reluctance forces to control the non-driving directions.

A continuous-time model of the system is given by

$$\dot{x}(t) = Ax(t) + Bu_x(t), \tag{59a}$$
$$u_x(t) = \mathcal{F}_x\left(x(t), i(t)\right), \tag{59b}$$
$$u_z(t) = \mathcal{F}_z\left(x(t), i(t)\right), \tag{59c}$$
$$u_t(t) = \mathcal{F}_t\left(x(t), i(t)\right), \tag{59d}$$

where the input to the system is given by the current vector $i \in \mathbb{R}^{n_c}$, with n_c the number of coils, the intermediate state $u_x(t) \in \mathbb{R}$, $u_z(t) \in \mathbb{R}$ represent the forces applied to the translator in the driving and normal directions, respectively, $u_t(t) \in \mathbb{R}$ the torque applied to the translator. $A \in \mathbb{R}^{2\times 2}$ and $B \in \mathbb{R}^{2\times 1}$ represent a state space realization of the translator dynamics in the driving direction, such that $x(t) = (x^v(t), x^p(t))$ represent the velocity and position of the translator in the driving direction. The model is decoupled into two independent parts: the translator itself is modelled as a linear mass system, described in (59a), and the electromagnetic part is modelled as the static nonlinearity defined by (59b) – (59d), which relates currents and forces. The coordinates are given as defined in Figure 3: x and z represent the position and normal offset of the translator, and r_y represents the rotation over y.

1) Electromagnetic part modelling: The relationships between currents and forces are modelled via a Fourier approximation [14, Chapter 2]. This model evaluates the Lorentz and reluctance forces at a given position. It can be determined from the geometry of the motor via first principles modelling methods, or via system identification methods. The problem is analyzed in the x-z plane. Using the model and notation of [16], the net forces and torque to the translator can be expressed as

$$\mathcal{F}_x(x,i) = K_x(x)i, \tag{60a}$$
$$\mathcal{F}_z(x,i) = K_z(x)i + i^T G_z i, \tag{60b}$$
$$\mathcal{F}_t(x,i) = K_t(x)i + i^T G_t i, \tag{60c}$$

where $\mathcal{F}_x(x,i)$ is the function describing the propulsion force, $\mathcal{F}_z(x,i)$ is the normal force, and $\mathcal{F}_t(x,i)$ is the torque to the translator. The linear terms $K_x(x) \in \mathbb{R}^{n_c}$, $K_z(x) \in \mathbb{R}^{n_c}$ and $K_t(x) \in \mathbb{R}^{n_c}$ represent the Lorentz force functions and the weights $G_z \in \mathbb{R}^{N_c\times N_c}$, $G_t \in \mathbb{R}^{N_c\times N_c}$ represent a quadratic relationship between the currents and the reluctance force and torque. The vector $i \in \mathbb{R}^{n_c}$ represents the currents through the coils,

$$i(t) = \left(i_{A_1}(t), i_{B_1}(t), i_{C_1}(t), \ldots, i_{A_{N_c}}(t), i_{B_{N_c}}(t), i_{C_{N_c}}(t)\right)^T \tag{61}$$

where N_c is the number of coil sets, and n_c the total number of currents (so $n_c = 3N_c$). It is enforced that the sum of the three currents through every coil set is zero, i.e.

$$i_{A_j} + i_{B_j} + i_{C_j} = 0, \qquad j = 1 \ldots N_c. \tag{62}$$

This is required because of the star connection of the three-phase coils.

At a given position, the propulsion force depends linearly on the currents, while the normal force and torque are quadratic functions of i because they are caused, in part, by the reluctance forces. The model considers no displacements in the z and r_y directions. In this case, the linear terms depend only on x and the quadratic terms G_z, G_t are constant.

The position dependency of each of the terms $K_x(x)$, $K_z(x)$ and $K_t(x)$ is described by a Fourier model [14]. Each term is given by

$$K_j(x) = \begin{bmatrix} \sum_{i=0}^{l_j} \left(a_1^{ij} \cos\left(ix^p \frac{\pi}{\tau_p}\right) + b_1^{ij} \sin\left(ix^p \frac{\pi}{\tau_p}\right)\right) \\ \vdots \\ \sum_{i=0}^{l_j} \left(a_{n_c}^{ij} \cos\left(ix^p \frac{\pi}{\tau_p}\right) + b_{n_c}^{ij} \sin\left(ix^p \frac{\pi}{\tau_p}\right)\right) \end{bmatrix}^T, \tag{63}$$

for $j = x, z, t$, where τ_p represents the magnet pole pitch, l_j defines the number of harmonics used, and the terms a_k^{ij} and b_k^{ij} are identified parameters, for $i = 1 \ldots l_j$ and $k = 1 \ldots n_c$. Therefore, a total of $2 \times 3 \times n_c \times l_j$ parameters are identified.

2) Mechanical part modelling: The translator is modelled as a linear time-invariant dynamical system described by the state space representation $\dot{x}(t) = Ax(t) + Bu_x(t)$ with

$$A = \begin{bmatrix} -d/m & 0 \\ 1 & 0 \end{bmatrix}, \qquad B = \begin{bmatrix} 1/m \\ 0 \end{bmatrix}, \tag{64}$$

where m is the mass of the translator and d is a viscous friction parameter. Note that the dynamical model only considers the driving direction. The displacement in the directions z and r_y may be modelled as a mass-spring-damper system with high stiffness, but we consider the displacements in z and r_y to be unobservable, and we desire them to be zero. No dynamical model is used for those directions, instead, the constraints $\mathcal{F}_z = 0$ and $\mathcal{F}_t = 0$ will be enforced.

B. Problem formulation

A Nonlinear Model Predictive Controller is designed in order to control a linear motor with current constraints, by posing an optimization problem with the structure of (9) – (10). The predicted states of previous executions are used to discretize the problem and remove the position dependency.

The Linear Motor model defined in (59a) – (59d) is used as a prediction model. We consider $N_c = 2$ sets of coils. The optimization problem at time t is posed as

Optimization Problem 3 (Linear Motor NMPC Problem)**.**

$$\min_{U_k, X_k, W_k} \sum_{j=0}^{N-1} \Big((x_i - x_j^r)^T Q_x (x_j - r_j) + (u_j - u_j^r)^T R_u (u_j - u_j^r) + (w_j - w_j^r)^T R_w (w_j - w_j^r) \Big) + (x_N - x_N^r)^T P (x_N - x_N^r), \quad (65)$$

subject to

$$x_0 = x(t), \quad (66a)$$
$$x_{j+1} = A_d x_j + B_d u_j, \quad \forall j = 0, \dots, N-1, \quad (66b)$$
$$u_j = \mathcal{F}_x(\hat{x}_j, w_j), \quad \forall j = 0, \dots, N-1, \quad (66c)$$
$$0 = \mathcal{F}_y(\hat{x}_j, w_j), \quad \forall j = 0, \dots, N-1, \quad (66d)$$
$$0 = \mathcal{F}_t(\hat{x}_j, w_j), \quad \forall j = 0, \dots, N-1, \quad (66e)$$
$$-w_{max} \le w_j \le w_{max}, \quad \forall j = 0, \dots, N-1, \quad (66f)$$
$$-u_{max} \le u_j \le u_{max}, \quad \forall j = 0, \dots, N-1, \quad (66g)$$
$$-x_{max} \le x_j \le x_{max}, \quad \forall j = 1, \dots, N, \quad (66h)$$
$$0 = w_j^{[k+1]} + w_j^{[k+2]} + w_j^{[k+3]}, \quad \forall k = 0, 3, \quad \forall j = 0, \dots, N-1, \quad (66i)$$

with prediction horizon length N, state cost $Q_x \in \mathbb{R}^{2\times 2}$, current cost $R_w \in \mathbb{R}^{6\times 6}$, force cost $R_u \in \mathbb{R}$, position reference r_k ($k = 1 \dots N$). The input nonlinearity terms (66c) – (66e) represent the electromagnetic model defined in (60a) – (60c). The position dependency of these terms is evaluated at the guess $\hat{x}$ before solving the optimization problem, as defined below. The constraints (66d) – (66e) impose that the normal force and torque are zero. The constraint (66i) imposes that the sum of currents through every coil set is zero. The terms in (66b) represent the linear dynamics. $w_{max} \in \mathbb{R}^6$, $u_{max} \in \mathbb{R}$ and $x_{max} \in \mathbb{R}^2$ represent box constraints on the predicted currents, forces, and states, respectively.

1) Discretization and treatment of the position dependency: The linear dynamical system (64) is discretized with a zero-order hold on the force with discretization time T_d, which is described by the equations $x(t+1) = A_d x(t) + B_d u(t)$.

The position-dependent terms $\mathcal{F}_x(x_j, w_j)$, $\mathcal{F}_z(x_j, w_j)$, and $\mathcal{F}_t(x_j, w_j)$ are pre-evaluated at $x = \hat{x}$ before solving the optimization problem. $\hat{x}$ is a guess that is derived from the previous predicted state vector X_{k-1}^*. This solves a double purpose: the position-dependent terms of the model are converted into time-varying terms, which reduce the complexity of the problem, and serve as a discretization of the position-dependent functions. A zero-order hold is implemented on the currents $w(t)$ between samples, therefore, the zero-order hold assumed in the predicted forces u_k is an approximation, which is only exact when $x(t)$ is constant between x_k and x_{k+1} due to the position-dependency of the input terms. We propose to evaluate $\mathcal{F}_x(x_j, w_j)$, $\mathcal{F}_z(x_j, w_j)$, and $\mathcal{F}_t(x_j, w_j)$ functions at $x_j = \hat{x}_j$, which is given by

$$\hat{x}_j = \frac{x^*_{j+1|k-1} + x^*_{j|k-1}}{2}, \qquad j = 1, \dots, N-1, \quad (67)$$

i.e. the average between the predicted x_j and x_{j+1} at time $k-1$. Therefore, the predicted currents w_j are modelled (and implemented) with a zero-order hold, and the predicted forces u_i are assumed to be also (approximately) constant between sampling periods. This assumption is valid when $K_j(x(t)) - K_j(x(t+T_d))$ is relatively small. This is verified in simulation results. In practice, the variation of $u(t)$ between samples is very small, and the discretization error is also small when considering the simpler approximation $\hat{x}_j = x_{j|k-1}$, see [14, Chapter 5].

The technique of using a pre-computed sequence of predicted states to remove the state dependency at the next time instant has been successfully implemented in a number of applications, particularly quasi-LPV systems (see e.g. [4]). Although the recursive convergence of this technique is difficult to prove, it provides good results in practice.

Remark 3 (Including the position dependency in the optimization problem)**.** *Although the identified functions $K_x(x)$, $K_z(x)$ and $K_t(x)$ are smooth and easily differentiable, they are evaluated before they are introduced in the nonlinear optimization problem. Specifically, the full model of* (63) *is not introduced in the problem because the computational cost of doing so does not compensate the gains, and especially because the introduction of such functions makes the optimization problem highly non-convex.*

C. Optimization problem solution

In this subsection, a solution for the optimization problem (65) – (66) is described, which makes use of the algorithm derived in Section III and allows to control the motor in real time.

1) Variable elimination: The three-phase coil constraints (66i) are removed with the transformation

$$\underbrace{\begin{bmatrix} i_{A_1} \\ i_{B_1} \\ i_{C_1} \\ i_{A_2} \\ i_{B_2} \\ i_{C_2} \end{bmatrix}}_{w_k} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & -1 \end{bmatrix}}_{T} \underbrace{\begin{bmatrix} i_{A_1} \\ i_{B_1} \\ i_{A_2} \\ i_{B_2} \end{bmatrix}}_{\overline{w}_k} \quad (68)$$

which is applied to condense the problem by removing the inputs i_{C_1} and i_{C_2}, by replacing the following variables with their overlined correspondent (e.g. $R_w \to \overline{R}_w$)

$$\overline{R}_w = T R_w T^T, \quad (69a)$$
$$\overline{G}_z = T G_z T^T, \quad (69b)$$
$$\overline{G}_t = T G_t T^T, \quad (69c)$$
$$\overline{w} = T^T w, \quad (69d)$$
$$\overline{w}_{max} = T^T w_{max}. \quad (69e)$$

This operation preserves the sparsity of the problem, except for the terms Φ_i^w which become dense (but still positive definite, if originally), also in the case of diagonal R_w and box constraints, but $2N$ optimization variables and $2N$ equality constraints are removed from the problem. No further variables nor constraints are removed from the problem in order to preserve and exploit its sparsity pattern.

2) Problem size and notation: The optimization problem is expressed in the notation of (2) in Section II. The problem size is given by $n_x = 2$, $n_u = 1$, $n_f = 3$, $n_w = 4$, and the input nonlinearity can be described in the syntax of (2) by considering

$$K = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T, \tag{70a}$$

$$\Psi^j(w_j) = \begin{bmatrix} \mathcal{F}_x(\hat{x}_j, T\bar{w}_j) \\ \mathcal{F}_z(\hat{x}_j, T\bar{w}_j) \\ \mathcal{F}_t(\hat{x}_j, T\bar{w}_j) \end{bmatrix}. \tag{70b}$$

3) Real-time strategy: The system is controlled, at time t_0, by reading/estimating the current state $x(t_0)$, solving the optimization problem (65), and applying the first predicted input $w_{0|t_0}$ with a zero-order hold, until the next sampling instant $t_0 + T_s$. The state $x(t_0)$ is assumed to be known.

The system is sampled at a fixed period T_s, and that is the time limit to solve the optimization problem (65), which is solved with the interior point scheme formulated in Algorithm 1. The problem is solved for a fixed number of iterations K^{max}, in this case, the solution is only guaranteed to satisfy the inequality constraints.

The full solution (predicted states and inputs) of the optimization problem at time t_0 is used as an initial guess for the optimization problem at time $t_0 + T_s$ (*warm-start*). The dual variables (ν in Algorithm 1) are also used in the warm-start. This is made possible by the constant value of μ. Note that we consider different discretization (prediction) and sampling times, T_d and T_s. In the case $T_d = T_s$, the predicted sequences X^*_{k-1}, U^*_{k-1}, W^*_{k-1} can be shifted one sample ahead for a more accurate warm start. No shifting is required when $T_d \gg T_s$.

D. Closed-loop simulation example

The scheme described in this chapter is implemented in simulation, and the consequences of the design decisions previously taken are commented.

1) Example problem: The electromagnetic model from [15] is used (terms $K_x(x)$, $K_z(x)$, $K_t(x)$, G_z, G_t). The first three terms (given by Eq. 63) are shown in Figure 4. A mass $m = 20kg$ and a damping of $s = 100kg/s$ are considered.

The MPC controller is designed with prediction horizon length $T = 10$, weighting matrices $Q_x = 10^8 I_2$, $R_u = 1$, $\overline{R}_w = I_6 T$, and current inequality constraints $w_{max} = (1, \ldots, 1)^T A$. The system tracks a step reference from $x = 0.1m$ to $x = 0.3m$, which activates the input constraints. The reference is not known in advance (before $t = 0$). The reference currents and forces are set to zero. The system is discretized at $T_d = 10ms$, therefore, the prediction horizon comprises the next $NT_d = 100ms$.

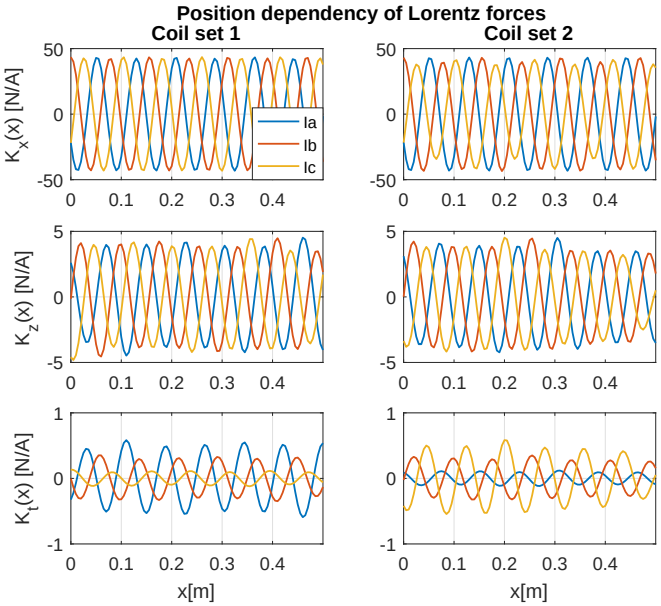

Fig. 4: Identified terms $K_x(x)$, $K_z(x)$ and $K_t(x)$, showing the position dependency of the Lorentz forces, each term divided in two plots for both coil sets.

Interior Point parameters			
μ	10^2	α	0.01
β	0.95	γ	0.95
τ	10^8	K^{max}	8

TABLE II: Parameters for Algorithm 1.

In order to solve (65), Algorithm 1 is used with the parameters given in Table II.

We consider a sampling period of $T_s = 500\mu s$ (2000Hz). In general, each interior point iteration requires one evaluation of the model and KKT matrix and multiple merit function evaluations for the line search procedure. We allocate $420\mu s$ to the solution of the optimization problem. This limit is translated to an iteration limit for the different steps of the Interior Point algorithm, $K^{max} = 8$ in Algorithm 1, The overall computation times are shown in Table III. The numbers on the first column serve as a demonstration. They need to be adjusted for every particular embedded platform, and have been orientatively picked as twice as much the values obtained with the computer from Appendix B.

Interior point solution time (upper bound)			
	Cost/iter	Max	Sum
Solve KKT system	50 μs	8	$400\mu s$
Line search step	$0.75\mu s$	24	$18\mu s$
Evaluate model	$1\mu s$	1	$1\mu s$
Total			419 μs

TABLE III: Interior point solution time with iteration limit.

The system is simulated with an outer loop running at $20T_s$, which simulates the dynamics, and an inner loop running at T_s, which simulates the controller. The inter-sample behavior

is not shown in the figures, but is taken into account in the simulations.

2) Results: Figure 6 shows the position of the translator, the currents through the coils (inputs), and the produced forces and torque. It can be seen that the system respects the inequality constraints, and is also able to anticipate to future constraint violations: the system starts braking about $100ms$ before reaching the reference value. With a lower prediction horizon, this would have resulted in a larger overshoot. The described trajectory is similar to a second order motion profile, but more efficient since the highest possible force is developed at every sample, taking into account the position dependency of the model and the current constraints.

Although it cannot be appreciated in the figure, the currents saturate at 0.998A instead of 1A. This is due to the use of a the fixed barrier parameter for the inequality constraints. This difference is considered acceptable.

Figure 7 shows the discrepancy between the predicted positions $x_{k|t}$ for $k = 1, 4, 7$ across all t, and the actual values $x(t + kT_d)$, i.e. $k\frac{T_d}{T_s}$ samples later. This is used to evaluate the accuracy of the prediction model. Since the system has been simulated without any noise, this discrepancy represents the accuracy losses due to the pre-evaluation of the terms (66c) – (66e) with (67) and its heuristic discretization. The prediction error peaks between 0% and 2%, depending on k. This is considered acceptable. The horizon is primarily used for predicting future constraints violations on the currents (in this example, the optimization of X_k and U_k (dynamics) is independent from W_k when the constraints are not active, since $Q_x \gg R_w$).

E. Solver performance

For the Linear Motor problem (with problem size $n_x = 2$, $n_u = 1$, $n_f = 3$, $n_w = 4$), the achieved solution times for the solution of the KKT system (38) with the block factorization routine from Section III-C are shown in Figure 5 for different values of N. These numbers represent a large part of the cost of performing an interior point iteration. The numbers correspond to the implementation defined in Appendix B. The achieved computational times are up to 3 orders of magnitude lower than what was achieved in previous work [14, Chapter 5], where computational times in the order of 100ms were achieved for the same problem (with $N = 8$).

The achieved computational times are compared with the time required to solve Optimization Problem 3 with a solver generated with ACADO Toolkit [10], making use of the dense QP (Quadratic Program) solver qpOASES [9]. The tools are used to generate source code used to solve NMPC problems in real time. We have identified those as the best tools which support NMPC problems with the structure of (9) – (10) out of the box. ACADO and qpOASES implement a SQP (Sequential Quadratic Programming) scheme in which the inequality constraints are treated with an Active Set technique. The intermediate QPs are condensed by removing the equality constraints, which makes this approach more favourable for problems with more states than inputs, and makes the computational times grow cubically with N. Note

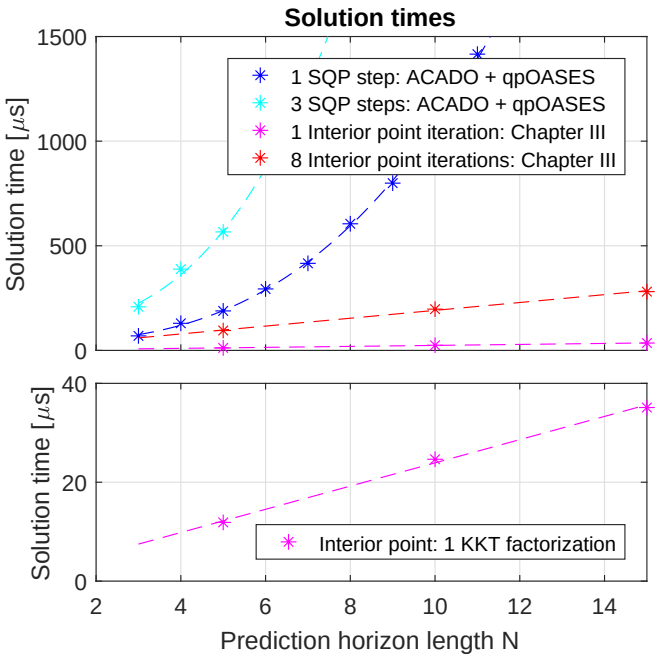

Fig. 5: Achieved times for the solution of the KKT system (38), and comparison with the costs of SQP iterations with ACADO + qpOASES.

that ACADO is meant for generic NMPC problems with nonlinear dynamics, for which a discretization and integration needs to be done online at every sample. This makes the tool not fully appropriate for the problem structure analyzed in this chapter, in which the dynamics are linear.

F. Solution accuracy

In order to achieve a fast solution of the optimization problems, multiple compromises have been made, which could result in suboptimal or inaccurate solutions: the use of a fixed barrier parameter does not guarantee that the achieved solutions are optimal for the original problem (Optimization Problem 3), the use of a fixed iteration limit can potentially cause the algorithm to prematurely stop at infeasible solutions, and the collapse of (20) into (22) is a potential source of numerical conditioning problems.

In the simulation solutions are feasible with respect to all constraints. Figure **??** shows the worst (highest) equality constraint residual of the optimization problem. The equality constraints (66b) – (66e) represent the system dynamics. The residual is negligible in comparison with the numerical values of the parameters in (66b) – (66e), meaning that the system dynamics are respected in all optimization problems solved.

A second closed-loop simulation has been performed by solving Optimization Problem 3 with the general purpose nonlinear solver IPOPT [18], which is considered to be numerically robust and accurate, but is not appropriate for a real-time implementation. We consider the solution obtained with IPOPT to be the true solution of the original optimization problem. In Figure 8, the IPOPT results are compared with those obtained with the solver in Section III, showing a

difference of up to a 0.2%. This difference is caused by the use of a fixed barrier parameter. If necessary, the error may be reduced by picking a smaller μ, which needs to be compensated with a higher iteration limit K^{max}, hence reducing the attainable sampling rate.

The loss in accuracy in the solution of the optimization problems is considered to be small, and is outweighed by the high sampling rates which can be achieved.

V. Conclusions

In this work, a fast Nonlinear Model Predictive Control algorithm has been developed which considers a class of dynamical systems with input nonlinearities as prediction model, a class which includes Hammerstein systems. The algorithm implements an Interior Point scheme in which the required linear algebra operations are performed via block Cholesky factorizations on a reduced KKT matrix. This approach exploits the sparsity pattern of the problem and is especially favourable for problems with relatively large prediction horizons, and problems with a number of inputs which is similar or higher than the number of states. The interior point scheme is implemented in real time via a fixed-barrier approach.

This work is motivated by the Nonlinear Model Predictive Control for Ironless Linear Motors problem. A model accounting for reluctance forces and multiple degrees of freedom has been considered. The problem has been solved in simulation with the developed algorithm, showing that it is possible to control ILMs with sampling times in the microsecond range while satisfying multiple constraints. The proposed approach shows an insignificant loss in numerical accuracy of the solution, while achieving solution times 3 orders of magnitude lower than in previous approaches taken to solve the same problem.

The developed algorithm and software filled a small gap in the available solver implementations for fast nonlinear model predictive control. For the particular problem structure of the ILM NMPC problem (over-actuated system with an input nonlinearity), no other freely available solvers have been considered adequate, and a new implementation of the algorithm described in this paper has been created with the intention of running it in an embedded platform. The developed solver can be used for solving NMPC problems with the described structure, and it can also be used for solving linear MPC problems, being especially competitive when dealing with problems with large prediction horizons and systems with more inputs than states.

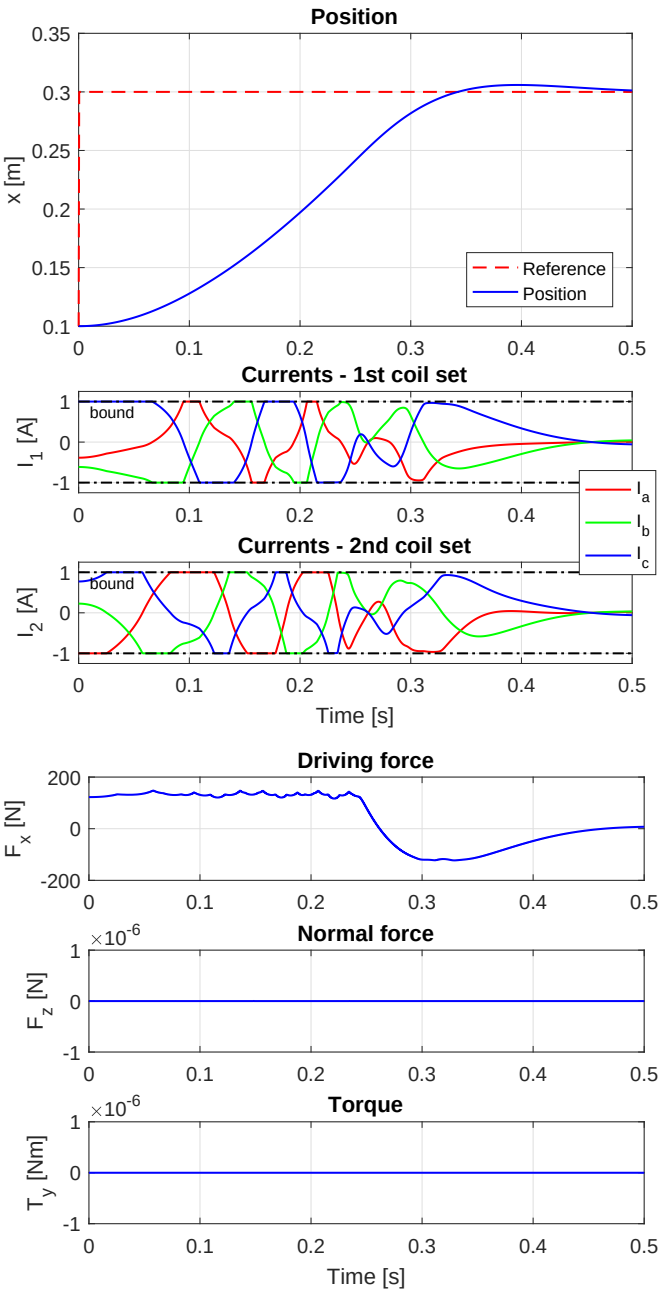

Fig. 6: Simulation results: position, currents, forces and torques at sampling times.

Acknowledgement

The author would like to thank Dr. Mircea Lazar and Dr. Tuan Nguyen for their guidance and patience during the elaboration of this work.

References

[1] Bemporad A. Alessio A. A Survey on Explicit Model Predictive Control. *Magni L., Raimondo D.M., Allgwer F. (eds) Nonlinear Model Predictive Control. Lecture Notes in Control and Information Sciences, vol 384. Springer, Berlin, Heidelberg*, 2009.

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[4] Pablo S. G. Cisneros, Sophia Voss, and Herbert Werner. Efficient Nonlinear Model Predictive Control via quasi-LPV representation. *2016 IEEE 55th Conference on Decision and Control (CDC)*, (Cdc):3216–3221, 2016.

[5] Moritz Diehl, Hans Georg Bock, and Johannes P Schl Oder. A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control. *SIAM Journal on Control and Optimization*, 43(5):1714–1736, 2005.

[6] A Domahidi, A U Zgraggen, M N Zeilinger, M Morari, and C N Jones. Efficient interior point methods for multistage problems arising

in receding horizon control. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 668–674, dec 2012.

[7] Gianluca Frison. Numerical Methods for Model Predictive Control. *MSc Thesis, University of Padova*, 2012.

[8] Gianluca Frison. High-performance linear algebra for embedded optimization and its use in the HPMPC solver. IMTEK, University of Freiburg. Technical report, 2016.

[9] M. Diehl H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

[10] B. Houska, H.J. Ferreau, M. Vukov, and R. Quirynen. ACADO Toolkit User's Manual. http://www.acadotoolkit.org, 2009–2013.

[11] Boris Houska, Hans Joachim Ferreau, and Moritz Diehl. An auto-generated real-time iteration algorithm for nonlinear MPC in the microsecond range. *Automatica*, 47(10):2279–2285, 2011.

[12] Juan L. Jerez, Eric C. Kerrigan, and George A. Constantinides. A condensed and sparse QP formulation for predictive control. *Proceedings of the IEEE Conference on Decision and Control*, pages 5217–5222, 2011.

[13] Jacob Mattingley and Stephen Boyd. CVXGEN: a code generator for embedded convex optimization. *Springer Optimization and Engineering*, 13:1–27, 2012.

[14] Tuan T. Nguyen. *Identification and compensation of parasitic effects in coreless linear motors*. PhD thesis, Technische Universiteit Eindhoven, 2018.

[15] Tuan T Nguyen, Mircea Lazar, and Hans Butler. Cancellation of normal and parasitic forces in coreless linear motors. 2015 19th International Conference on System Theory, Control and Computing (ICSTCC), Cheile Gradistei, Romania, 14-16 Oct. 2015.

[16] Tuan T. Nguyen, Mircea Lazar, and Hans Butler. A computationally efficient commutation algorithm for parasitic forces and Torques compensation in ironless linear motors. 7th IFAC Symposium on Mechatronic Systems MECHATRONICS 2016: Loughborough University, Leicestershire, UK, 58 September 2016. *IFAC-PapersOnLine*, 49(21):267–273, 2016.

[17] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.

[18] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, Mar 2006.

[19] Yang Wang and Stephen Boyd. Fast Model Predictive Control Using Online Optimization. *Control Systems Technology, IEEE Transactions on*, 18(2):267–278, 2010.

[20] A. Wills, D. Bates, A. Fleming, B. Ninness, and R. Moheimani. Application of MPC to an active structure using sampling rates up to 25kHz. *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, CDC-ECC '05*, 2005:3176–3181, 2005.

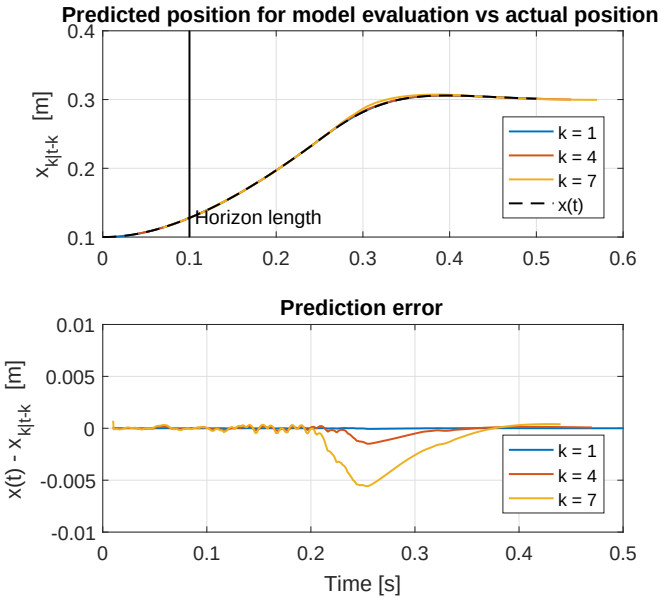

Fig. 7: Simulation results: Assumed positions at model evaluation time compared with the actual values $k\frac{T_d}{T_s}$ samples later.

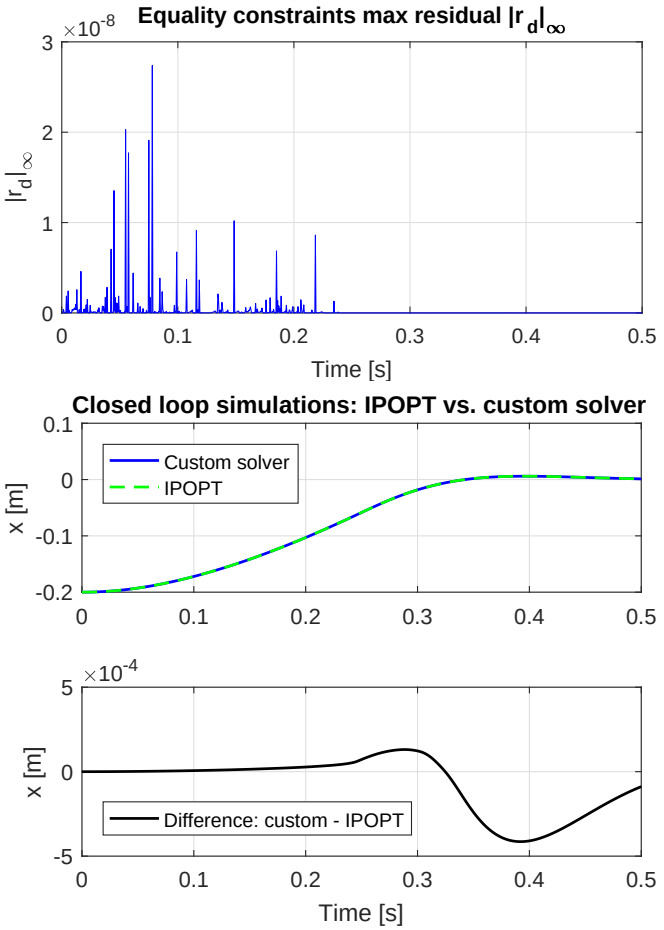

Fig. 8: Closed-loop simulation results with IPOPT, compared with the custom solver from Section III.

APPENDIX A
STRUCTURE OF Y

The structure of $Y = C\Phi^{-1}C^T$ will be analyzed. First, let the structure of Φ^{-1} to be explicitly defined as

$$\Phi^{-1} = \begin{bmatrix} \tilde{R}_0 & & & & & & \\ & \tilde{\Omega}_0 & & & & & \\ & & \tilde{Q}_1 & \tilde{S}_1 & & & \\ & & \tilde{S}_1^T & \tilde{R}_1 & & & \\ & & & & \tilde{\Omega}_1 & & \\ & & & & & \ddots & \\ & & & & & & \tilde{Q}_N \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{71}$$

where the terms $\tilde{S}$ only appear in the case where the stage cost or the inequality constraints are not state-input separable. Then, $Y = C\Phi^{-1}C^T$ takes the form

$$Y \coloneqq \begin{bmatrix} Y_{11} & Y_{21}^T & 0 & \cdots & 0 & 0 \\ Y_{21} & Y_{22} & Y_{32}^T & \cdots & 0 & 0 \\ 0 & Y_{32} & Y_{33} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & Y_{N-1,N-1} & Y_{N,N-1}^T \\ 0 & 0 & 0 & \cdots & Y_{N,N-1} & Y_{N,N} \end{bmatrix} \in \mathbb{R}^{m\times m}, \tag{72}$$

where each block is

$$Y_{11} = \left[\begin{array}{c|c} \tilde{Q}_1 + B\tilde{R}_0B^T & B\tilde{R}_0K^T \\ \hline K\tilde{R}_0B^T & \psi\tilde{\Omega}_0\psi^T + K\tilde{R}_0K^T \end{array}\right]$$

$$Y_{i+1,i+1} \coloneqq \left[\begin{array}{c|c} Y_{i+1,i+1}^A & Y_{i+1,i+1}^B \\ \hline Y_{i+1,i+1}^C & Y_{i+1,i+1}^D \end{array}\right] = \left[\begin{array}{c|c} \tilde{Q}_{i+1} + A\tilde{Q}_iA^T & B\tilde{R}_iK^T + A\tilde{S}_i^TK^T \\ \quad +B\tilde{R}_iB^T + B\tilde{S}_i^TA^T & \\ \quad +A\tilde{S}_iB^T & \\ \hline K\tilde{R}_iB^T + K\tilde{S}_i^TA & \psi\tilde{\Omega}_i\psi^T + K\tilde{R}_iK^T \end{array}\right] \tag{73}$$

$$Y_{i+1,i} \coloneqq \left[\begin{array}{c|c} Y_{i+1,i}^A & Y_{i+1,i}^B \\ \hline Y_{i+1,i}^C & Y_{i+1,i}^D \end{array}\right] = \left[\begin{array}{c|c} -B\tilde{S}^T - A\tilde{Q}_1 & 0_{n_x\times n_f} \\ \hline -K\tilde{S}^T & 0_{n_f\times n_f} \end{array}\right] \tag{74}$$

and where each of the blocks is square, and each block partitioned in sub-blocks so that the upper left ones are of size $n_x \times n_x$ and the lower right ones are $n_f \times n_f$.

Remark 4 (Treatment of Y as a banded matrix)**.** *An alternative approach to step iii) is treating Y as a banded matrix and using a banded solver. As shown in Appendix A, the matrix Y has a maximum bandwidth of $2(n_x + n_f)$ in the general case, and of $n_x + n_f + lbw(A)$ in the case of diagonal and box constraints (where $lbw(A)$ is the lower bandwidth of A). Depending on the dynamical system, it may be possible to rearrange the states such that A is as close as possible to upper triangular, which reduces the cost of this step.*

The cost of the factorization is then $4N(n_x + n_f)^3$ for the first case, and $N(n_x+n_f)(n_x+n_f+lbw(A))^2$ for the second case. Although the number of required operations is higher than in the block tridiagonal approach, in some situations the use of machine-optimized banded solvers may be more efficient than the use of tailor-made matrix routines.

APPENDIX B
ALGORITHM IMPLEMENTATION

A software implementation of the algorithm described in Section III has been developed. It comprises a code generation routine, which automatically generates C code for the solution of the algorithm. The code generation routine hardcodes all constants and the sizes of all the intermediate arrays before compilation (given by n_x, n_u, n_f, n_w and N), pre-allocates memory for all the required intermediate computations, and generates matrix operation functions for those sizes. Then the generated loops can be further optimized by the compiler. The code can be run on an embedded platform. The operations are performed in an order such that the access to intermediate values is grouped (i.e. instead of solving steps i-v in order, first steps i-iii are solved for $i = 0$, then again for $i = 1$, and so on). This reduces the number of memory operations between the memory and the cache of the microcontroller, which is the main bottleneck when performing linear algebra operations on a computer [8]. This also the operations to be parallelized if the compiler/platform supports it.

The current implementation relies on a few calls to the linear algebra libraries BLAS and LAPACK [2], but we are working to remove that dependency since auto-generated code is faster for small scale operations, and more portable. In particular, the factorization of Y is performed with a factorization routine for banded, positive definite matrices. The computational times reported in previous chapters correspond to a computer with an *Intel Core i7 3630QM* processor, and the code is compiled with the *Intel C Compiler 16.0* with the *-O2* option. The calls to MA57 and LAPACK are done via the *Intel Math Kernel Library 2017.0.31*. Equivalent compilers and libraries are available for embedded platforms.

To the author's best knowledge, there are no comparable code generation routines for the solution of NMPC problems with the structure defined in Section II. Neither is the author aware of the existence of freely available code generation tools for the structure-exploiting solution of KKT systems for MPC problems via block factorizations (for linear or nonllinear systems). A popular code generation tool for linear MPC problems is CVXGEN [13], however, the approach taken is different. CVXGEN generates code that solves convex QPs by performing an LDL^T factorization of the full KKT matrix. An LDL^T factorization is usually performed via Gaussian elimination, which requires row permutation operations. These operations involve a memory bottleneck since the elements of the matrix are being accessed in a non-sequential way. CVXGEN uses a fixed-permutation algorithm in order to overcome this problem. This approach is fast and numerically stable for small problems, and becomes unsuitable for large problems. CVXGEN only supports linear systems.