To appear in the *International Journal of Control*
Vol. 00, No. 00, Month 20XX, 1–23

From Linear to Nonlinear MPC: Bridging the Gap via Real-Time Iteration.

Sébastien Gros[a]* and Mario Zanon[a] and Rien Quirynen[b] and Alberto Bemporad[c] and Moritz Diehl[d]

[a]*Signals & Systems, Chalmers University of Technology, Göteborg, Sweden;* [b]*Elec. Eng. Dept. (ESAT-SCD) and the Optimization in Engineering Center (OPTEC), K.U. Leuven, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee;* [c]*IMT Institute for Advanced Studies Lucca, Piazza San Francesco 19, 55100 Lucca, Italy;* [d] *IMTEK, University of Freiburg, Georges-Koehler-Allee 102, 79110 Freiburg, Germany*

(*November 2015*)

Linear Model Predictive Control (MPC) can be currently deployed at outstanding speeds, thanks to recent progress in algorithms for solving online the underlying structured Quadratic Programs. In contrast, Nonlinear MPC (NMPC) requires the deployment of more elaborate algorithms, which require longer computation times than linear MPC. Nonetheless, computational speeds for NMPC comparable to those of MPC are now regularly reported, provided that the adequate algorithms are used. In this paper we aim at clarifying the similarities and differences between linear MPC and NMPC. In particular, we focus our analysis on NMPC based on the Real-Time Iteration (RTI) scheme, as this technique has been successfully tested and, in some applications, requires computational times that are only marginally larger than linear MPC. The goal of the paper is to promote the understanding of RTI-based NMPC within the linear MPC community.

Keywords: linear MPC, real-time NMPC

1. Introduction

Linear Model Predictive Control is an advanced control technique able to deal with multiple input multiple output constrained linear systems [36, 43]. Recent algorithmic developments have significantly sped up computational times, allowing for the deployment of MPC at outstanding speeds [17, 19, 20, 21, 39].

Nonlinear Model Predictive Control (NMPC) is an effective way of tackling problems with nonlinear constraints and dynamics. Although not as widely used as linear MPC, NMPC has a long history of deployment in the process industry [3], where the relatively slow systems at hand leave room for computation-intensive control algorithms. However, thanks to progress in algorithms for optimal control and embedded control platforms, NMPC is getting more and more considered also for fast applications [2, 23, 24, 25, 26, 27, 33, 44, 45, 48, 51].

One of the most successful and largely used approaches to fast NMPC is arguably based on the Real-Time Iteration (RTI) [13] and its predecessors [35, 38]. The RTI approach exploits the fact that NMPC requires to successively solve optimal control problems (OCPs) that are closely related, in the sense that at every time instant the solution of the OCP at hand is very similar to the solution obtained at the previous time instant. RTI achieves the convergence of the NMPC solution "on-the-fly", i.e. conjointly to the evolution of the system dynamics. The reliability of this strategy hinges on the fast contraction rate of Newton type optimisation techniques. It has been formally studied in [16], and has been verified in many deployments of the RTI approach.

In this paper, we aim at bridging the gap between linear and RTI-based nonlinear MPC by highlighting

*Corresponding author. Email: grosse@chalmers.se

the similarities and differences of the two approaches. Because the NMPC problem is solved approximately by solving only one properly formulated QP per sampling instant, RTI can be seen as a special case of linear time-varying MPC with two important features:

(1) the linearisation of the system dynamics occurs *online* and is done at the current state and control prediction rather than on the reference trajectory;
(2) the system dynamics are simulated using a numerical integration scheme.

A theoretical justification for this approach has been provided in [13]. An intuitive justification can instead be provided by the so-called *real-time dilemma*, which we describe in Section 3.3.

We remark that the connection between linear and nonlinear MPC has already been highlighted from an algorithmic point of view in [5, 6]. Highlighting the similarity between linear and nonlinear MPC was however not the main focus of these contributions, where the emphasis was rather on approximate RTI implementations and efficient computations. In this paper we aim instead at clarifying the connection between the two techniques from the point of view of control engineers, based on a more tutorial and less algorithm-focused approach.

The paper is organised as follows. Sections 2 and 3 formulate, respectively, the linear and nonlinear MPC problems. Section 4 describes in detail the RTI-based NMPC approach, establishing its connection to linear MPC in Section 4.3. Section 5 describes numerical methods to obtain the discrete-time nonlinear prediction model from a continuous-time description of the system. A discussion on how to reliably implement RTI-based NMPC is given in Section 6. Conclusions are drawn in Section 7.

2. Linear Model Predictive Control (MPC)

At every discrete-time instant i, for a given state estimate $\hat{x}_i$ of the system, the control policy u_i is defined by solving the following optimal control problem

$$\begin{aligned}
&\mathrm{QP}_{\mathrm{MPC}}\left(\hat{x}_i,\, \boldsymbol{x}_i^{\mathrm{ref}},\, \boldsymbol{u}_i^{\mathrm{ref}}\right) = \\
\arg\min_{\Delta\boldsymbol{x}_i,\Delta\boldsymbol{u}_i} \; &\sum_{k=0}^{N-1} \frac{1}{2}\begin{bmatrix}\Delta x_{i,k}\\ \Delta u_{i,k}\end{bmatrix}^{\top} W_{i,k}\begin{bmatrix}\Delta x_{i,k}\\ \Delta u_{i,k}\end{bmatrix} && \text{(1a)}\\
\text{s.t.} \quad &\Delta x_{i,0} = \hat{x}_i - x_{i,0}^{\mathrm{ref}} && \text{(1b)}\\
&\Delta x_{i,k+1} = A_{i,k}\Delta x_{i,k} + B_{i,k}\Delta u_{i,k} + r_{i,k}, \qquad k = 0,\dots,N-1, && \text{(1c)}\\
&C_{i,k}\Delta x_{i,k} + D_{i,k}\Delta u_{i,k} + h_{i,k} \le 0, \qquad k = 0,\dots,N-1, && \text{(1d)}
\end{aligned}$$

where $\boldsymbol{x}_i^{\mathrm{ref}}$, $\boldsymbol{u}_i^{\mathrm{ref}}$ are the reference trajectories provided at time i, constraint (1b) enforces that the prediction starts at the current state, constraint (1c) enforces the system dynamics and constraint (1d) enforces path constraints which include e.g. actuator limitations, obstacle avoidance, etc. We use here and in the following the notation $x_{i,k}^{\mathrm{ref}}$ with $k = 0,...,N$ to denote the k^{th} element of the reference trajectory $\boldsymbol{x}_i^{\mathrm{ref}}$ provided at time i. The same applies to $\boldsymbol{u}_i^{\mathrm{ref}}$ with $k = 0,...,N-1$. The trajectories $\Delta\boldsymbol{x}_i = (\Delta x_{i,0},\dots,\Delta x_{i,N})$, $\Delta\boldsymbol{u}_i = (\Delta u_{i,0},\dots,\Delta u_{i,N-1})$ are the deviation of the system trajectories $\boldsymbol{x}_i, \boldsymbol{u}_i$ predicted at time i from the reference trajectories $\boldsymbol{x}_i^{\mathrm{ref}}, \boldsymbol{u}_i^{\mathrm{ref}}$ provided at time i, i.e.:

$$\Delta x_{i,k} = x_{i,k} - x_{i,k}^{\mathrm{ref}}, \quad k = 0,...,N, \qquad \Delta u_{i,k} = u_{i,k} - u_{i,k}^{\mathrm{ref}}, \quad k = 0,...,N-1. \tag{2}$$

At every discrete-time instant i, the input applied to the system is given by:

$$u_i^{\mathrm{MPC}} = u_{i,0}^{\mathrm{ref}} + \Delta u_{i,0}, \quad (\Delta\boldsymbol{x}_i,\, \Delta\boldsymbol{u}_i) = \mathrm{QP}_{\mathrm{MPC}}\left(\hat{x}_i,\ \boldsymbol{x}_i^{\mathrm{ref}},\ \boldsymbol{u}_i^{\mathrm{ref}}\right). \tag{3}$$

Matrices $W_{i,k}$ are symmetric positive semidefinite. For the sake of brevity, we omit the terminal cost in this section. Formulation (1) is a structured Quadratic Program (QP). In case matrices $A_{i,k}$, $B_{i,k}$ are constant, we refer to linear MPC based on a Linear Time-Invariant (LTI) model, otherwise to Linear Time-Varying (LTV) MPC. If the reference trajectories $\boldsymbol{x}_i^{\text{ref}}$, $\boldsymbol{u}_i^{\text{ref}}$ satisfy the dynamics of the prediction model, the affine terms $r_{i,k}$ are zero in the dynamic constraints (1c). Otherwise, the affine term $r_{i,k}$ is non-zero and given by the offset terms $r_{i,k} = A_{i,k}x_{i,k}^{\text{ref}} + B_{i,k}u_{i,k}^{\text{ref}} - x_{i,k+1}^{\text{ref}}$.

Linear MPC is often deployed to control nonlinear dynamical systems. Consider a time-invariant discrete nonlinear system:

$$x^+ = f\left(x,\, u\right), \tag{4}$$

with the inequality constraints

$$h\left(x, u\right) \leq 0. \tag{5}$$

In order to deploy linear MPC on (4)-(5), one needs to *prepare* the QP problem (1) (offline, whenever possible), where the matrices $A_{i,k}$, $B_{i,k}$, $C_{i,k}$, $D_{i,k}$ stem from the linearisation of the dynamics and of the inequality constraints at the reference trajectory $\boldsymbol{x}_i^{\text{ref}}$, $\boldsymbol{u}_i^{\text{ref}}$, i.e.:

$$A_{i,k} = \left.\frac{\partial f(x,u)}{\partial x}\right|_{x_{i,k}^{\text{ref}}, u_{i,k}^{\text{ref}}}, \quad B_{i,k} = \left.\frac{\partial f(x,u)}{\partial u}\right|_{x_{i,k}^{\text{ref}}, u_{i,k}^{\text{ref}}}, \tag{6a}$$

$$C_{i,k} = \left.\frac{\partial h(x,u)}{\partial x}\right|_{x_{i,k}^{\text{ref}}, u_{i,k}^{\text{ref}}}, \quad D_{i,k} = \left.\frac{\partial h(x,u)}{\partial u}\right|_{x_{i,k}^{\text{ref}}, u_{i,k}^{\text{ref}}}, \tag{6b}$$

$$r_{i,k} = f\left(x_{i,k}^{\text{ref}}, u_{i,k}^{\text{ref}}\right) - x_{i,k+1}^{\text{ref}}, \quad h_{i,k} = h\left(x_{i,k}^{\text{ref}}, u_{i,k}^{\text{ref}}\right). \tag{6c}$$

The dynamic constraints (1c) and inequality constraints (1d) then approximate the nonlinear dynamics (4) and inequality constraints (5) at the reference trajectories, i.e.:

$$\Delta x_{i,k+1} = A_{i,k}\Delta x_{i,k} + B_{i,k}\Delta u_{i,k} + \underbrace{f\left(x_{i,k}^{\text{ref}}, u_{i,k}^{\text{ref}}\right) - x_{i,k+1}^{\text{ref}}}_{=r_{i,k}}, \tag{7}$$

$$C_{i,k}\Delta x_{i,k} + D_{i,k}\Delta u_{i,k} + \underbrace{h\left(x_{i,k}^{\text{ref}}, u_{i,k}^{\text{ref}}\right)}_{=h_{i,k}} \leq 0. \tag{8}$$

If the reference trajectory $\boldsymbol{x}_i^{\text{ref}}$, $\boldsymbol{u}_i^{\text{ref}}$ is feasible for the system at hand, i.e. it satisfies (4) and (5), then $\boldsymbol{r}_i = 0$ such that the task of the MPC consists solely in rejecting disturbances and the error yielded by the linear model (7). Feasible reference trajectories are typically designed off-line via e.g. open-loop optimal control. For a given reference trajectory computed offline, the corresponding linearisation (7) ensues, and can also be computed offline. If an infeasible reference trajectory is used, then the term $r_{i,k}$ does not vanish, and triggers a first-order correction in the QP (1) for the infeasible trajectory.

When a set-point regulation problem is considered, a fixed reference trajectory is used. In this case $\boldsymbol{x}_i^{\text{ref}}$, $\boldsymbol{u}_i^{\text{ref}}$ are constant, and satisfy the stationarity condition $x_{i,k+1}^{\text{ref}} = x_{i,k}^{\text{ref}} = f(x_{i,k}^{\text{ref}}, u_{i,k}^{\text{ref}})$, resulting in $r_{i,k} = 0$. At every discrete time instant k, the input applied to the system is given by (3).

3. Nonlinear MPC (NMPC)

NMPC is sometimes preferred over linear MPC because it can treat the nonlinear dynamics and constraints directly and explicitly, as opposed to using linear approximations. We consider here a generalization of the linear MPC formulation (1) to nonlinear systems of the form (4):

$$\begin{aligned}
&\mathrm{NLP}\left(\hat{x}_i,\, \boldsymbol{x}_i^{\mathrm{ref}},\, \boldsymbol{u}_i^{\mathrm{ref}}\right) = \\
&\qquad \arg\min_{\boldsymbol{x},\boldsymbol{u}} \quad \sum_{k=0}^{N-1} \frac{1}{2} \begin{bmatrix} x_{i,k} - x_{i,k}^{\mathrm{ref}} \\ u_{i,k} - u_{i,k}^{\mathrm{ref}} \end{bmatrix}^{\top} W_{i,k} \begin{bmatrix} x_{i,k} - x_{i,k}^{\mathrm{ref}} \\ u_{i,k} - u_{i,k}^{\mathrm{ref}} \end{bmatrix} && \text{(9a)} \\
&\qquad \text{s.t.} \quad x_{i,0} = \hat{x}_i, && \text{(9b)} \\
&\qquad\qquad x_{i,k+1} = f\left(x_{i,k}, u_{i,k}\right), \qquad k = 0, \ldots, N-1, && \text{(9c)} \\
&\qquad\qquad h\left(x_{i,k}, u_{i,k}\right) \leq 0, \qquad k = 0, \ldots, N-1. && \text{(9d)}
\end{aligned}$$

While we restrict here to a quadratic positive (semi-)definite stage cost, in the context of nonlinear MPC more generic costs can be preferable. However, the use of a non positive-definite stage cost makes it hard to ensure closed-loop stability [42] and additional care might be needed at the algorithmic level [40]. Note however that, as proven in [49, 50], the feedback control law of any locally stabilising non positive-definite NMPC formulation can be approximated up to first-order by an (N)MPC formulation with positive-definite stage cost.

Similarly to Section 2, for the sake of brevity, we omit the terminal cost in this Section. At every time instant i, Problem (9) provides the NMPC control solutions for system (4), given by:

$$u_i^{\mathrm{NMPC}} = u_{i,0}, \quad (\boldsymbol{x}_i,\, \boldsymbol{u}_i) = \mathrm{NLP}\left(\hat{x}_i,\, \boldsymbol{x}_i^{\mathrm{ref}},\, \boldsymbol{u}_i^{\mathrm{ref}}\right). \tag{10}$$

Problem (9) is a structured Nonlinear Program (NLP), which can be solved efficiently using various solution approaches. We briefly recall next the Sequential Quadratic Programming (SQP) approach, which iteratively solves quadratic approximations of the NLP until convergence is achieved.

3.1 *Sequential Quadratic Programming (SQP) for NMPC*

In the SQP approach, Problem (9) is sequentially approximated by QPs delivering Newton directions for performing steps towards the solution starting from the available guess. The iteration is repeated taking (not necessarily full) Newton steps until convergence.

At a guess $(\boldsymbol{x}_i^{\mathrm{guess}},\, \boldsymbol{u}_i^{\mathrm{guess}})$, Problem (9) is approximated by the QP:

$$\begin{aligned}
&\mathrm{QP}_{\mathrm{NMPC}}\left(\hat{x}_i,\, \boldsymbol{x}_i^{\mathrm{guess}},\, \boldsymbol{u}_i^{\mathrm{guess}},\, \boldsymbol{x}_i^{\mathrm{ref}},\, \boldsymbol{u}_i^{\mathrm{ref}}\right) = \\
&\qquad \arg\min_{\Delta\boldsymbol{x}_i, \Delta\boldsymbol{u}_i} \quad \sum_{k=0}^{N-1} \frac{1}{2} \begin{bmatrix} \Delta x_{i,k} \\ \Delta u_{i,k} \end{bmatrix} H_{i,k} \begin{bmatrix} \Delta x_{i,k} \\ \Delta u_{i,k} \end{bmatrix} + J_{i,k}^{T} \begin{bmatrix} \Delta x_{i,k} \\ \Delta u_{i,k} \end{bmatrix} && \text{(11a)} \\
&\qquad \text{s.t.} \quad \Delta x_{i,0} = \hat{x}_i - x_{i,0}^{\mathrm{guess}}, && \text{(11b)} \\
&\qquad\qquad \Delta x_{i,k+1} = A_{i,k}\Delta x_{i,k} + B_{i,k}\Delta u_{i,k} + r_{i,k}, && \text{(11c)} \\
&\qquad\qquad C_{i,k}\Delta x_{i,k} + D_{i,k}\Delta u_{i,k} + h_{i,k} \leq 0, && \text{(11d)}
\end{aligned}$$

where

$$A_{i,k} = \left.\frac{\partial f(x,u)}{\partial x}\right|_{x_{i,k}^{\text{guess}}, u_{i,k}^{\text{guess}}}, \qquad B_{i,k} = \left.\frac{\partial f(x,u)}{\partial u}\right|_{x_{i,k}^{\text{guess}}, u_{i,k}^{\text{guess}}}, \tag{12a}$$

$$C_{i,k} = \left.\frac{\partial h(x,u)}{\partial x}\right|_{x_{i,k}^{\text{guess}}, u_{i,k}^{\text{guess}}}, \qquad D_{i,k} = \left.\frac{\partial h(x,u)}{\partial u}\right|_{x_{i,k}^{\text{guess}}, u_{i,k}^{\text{guess}}}, \tag{12b}$$

$$r_{i,k} = f\left(x_{i,k}^{\text{guess}}, u_{i,k}^{\text{guess}}\right) - x_{i,k+1}^{\text{guess}}, \quad h_{i,k} = h\left(x_{i,k}^{\text{guess}}, u_{i,k}^{\text{guess}}\right), \quad J_{i,k} = W_{i,k}\begin{bmatrix} x_{i,k}^{\text{guess}} - x_{i,k}^{\text{ref}} \\ u_{i,k}^{\text{guess}} - u_{i,k}^{\text{ref}} \end{bmatrix}, \tag{12c}$$

and $H_{i,k}$ is some approximation for the Hessian of the Lagrangian in Problem (9). The popular Gauss-Newton Hessian approximation [4] is given directly by $H_{i,k} = W_{i,k}$ in this case. The SQP algorithm at time instant i is detailed in Algorithm 1.

Algorithm 1: SQP for NMPC at discrete time i

Input: current state estimate $\hat{x}_i$, reference trajectory $(\boldsymbol{x}_i^{\text{ref}}, \boldsymbol{u}_i^{\text{ref}})$ and initial guess $(\boldsymbol{x}_i^{\text{guess}}, \boldsymbol{u}_i^{\text{guess}})$

while *Not converged* **do**

1. Evaluate $r_{i,k}$, $h_{i,k}$ and the sensitivities $A_{i,k}$, $B_{i,k}$, $C_{i,k}$, $D_{i,k}$, $H_{i,k}$, $J_{i,k}$ using (12)
2. Construct and solve $\text{QP}_{\text{NMPC}}\left(\hat{x}_i, \boldsymbol{x}_i^{\text{guess}}, \boldsymbol{u}_i^{\text{guess}}, \boldsymbol{x}_i^{\text{ref}}, \boldsymbol{u}_i^{\text{ref}}\right)$ as in (11) to get the Newton direction $(\Delta\boldsymbol{x}_i, \Delta\boldsymbol{u}_i)$
3. Compute step-size $\alpha \in]0, 1]$ to guarantee descent [37]
4. Update $(\boldsymbol{x}_i^{\text{guess}}, \boldsymbol{u}_i^{\text{guess}})$ with the Newton step:

$$(\boldsymbol{x}_i^{\text{guess}}, \boldsymbol{u}_i^{\text{guess}}) \leftarrow (\boldsymbol{x}_i^{\text{guess}}, \boldsymbol{u}_i^{\text{guess}}) + \alpha(\Delta\boldsymbol{x}_i, \Delta\boldsymbol{u}_i) \tag{13}$$

end

return NMPC solution $(\boldsymbol{x}_i, \boldsymbol{u}_i) = (\boldsymbol{x}_i^{\text{guess}}, \boldsymbol{u}_i^{\text{guess}})$

The NMPC solution is then obtained from the SQP Algorithm 1 as follows:

$$u_i^{\text{NMPC}} = u_{i,0}, \qquad (\boldsymbol{x}_i, \boldsymbol{u}_i) = \text{SQP}\left(\hat{x}_i, \boldsymbol{x}_i^{\text{guess}}, \boldsymbol{u}_i^{\text{guess}}, \boldsymbol{x}_i^{\text{ref}}, \boldsymbol{u}_i^{\text{ref}}\right), \tag{14}$$

where by $\text{SQP}\left(\hat{x}_i, \boldsymbol{x}_i^{\text{guess}}, \boldsymbol{u}_i^{\text{guess}}, \boldsymbol{x}_i^{\text{ref}}, \boldsymbol{u}_i^{\text{ref}}\right)$ we denote the solution of $\text{NLP}\left(\hat{x}_i, \boldsymbol{x}_i^{\text{ref}}, \boldsymbol{u}_i^{\text{ref}}\right)$ obtained by applying the SQP Algorithm 1 starting from the initial guess $(\boldsymbol{x}_i^{\text{guess}}, \boldsymbol{u}_i^{\text{guess}})$.

We aim here at stressing the similitudes and differences between computing the control solution using the linear MPC approach (3) and using the NMPC approach (14). Both (1) with (6) and (11)–(12) form a QP approximation of the NMPC problem (9). This statement is true by construction for QP_{NMPC}. By inspection, it is also easy to verify that QP_{MPC} in (1) is an approximation of (9) if the reference trajectory $(\boldsymbol{x}_i^{\text{ref}}, \boldsymbol{u}_i^{\text{ref}})$ is used as the linearisation point and a Gauss-Newton Hessian approximation is used. This similarity can be formally construed via the following Lemma:

Lemma 1: *If a Gauss-Newton Hessian approximation is used, i.e. $H_{i,k} = W_{i,k}$, then the following equivalence holds:*

$$\text{QP}_{\text{NMPC}}\left(\hat{x}_i, \boldsymbol{x}_i^{\text{ref}}, \boldsymbol{u}_i^{\text{ref}}, \boldsymbol{x}_i^{\text{ref}}, \boldsymbol{u}_i^{\text{ref}}\right) = \text{QP}_{\text{MPC}}\left(\hat{x}_i, \boldsymbol{x}_i^{\text{ref}}, \boldsymbol{u}_i^{\text{ref}}\right). \tag{15}$$

Proof. by inspection, the linearisation (12) taken at $x_{i,k}^{\text{guess}} = x_{i,k}^{\text{ref}}$, $u_{i,k}^{\text{guess}} = u_{i,k}^{\text{ref}}$ is identical to the linearisation (6) used in QP_{MPC} in (1). Additionally, $J_{i,k}$ evaluated at $x_{i,k}^{\text{guess}} = x_{i,k}^{\text{ref}}$, $u_{i,k}^{\text{guess}} = u_{i,k}^{\text{ref}}$ is zero. Con-

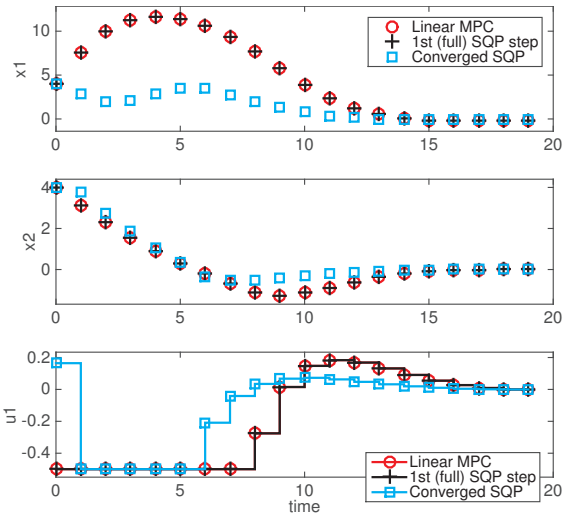


Figure 1. Comparison between the open-loop predicted trajectories obtained via linear MPC (circles), via performing a single iteration of Algorithm 1 using a full step ($\alpha = 1$) and $\boldsymbol{x}_i^{\text{guess}} = \boldsymbol{x}_i^{\text{ref}}$, $\boldsymbol{u}_i^{\text{guess}} = \boldsymbol{u}_i^{\text{ref}}$ (crosses), and via running Algorithm 1 to full convergence (squares).

sequently, the solution of QP_{MPC} given by (1) is identical to the solution of QP_{NMPC} given by (11). □

If the SQP Algorithm 1 is fed with the reference trajectory as an initial guess, i.e. is run as

$$\boldsymbol{x}_i,\, \boldsymbol{u}_i = \text{SQP}\left(\hat{x}_i,\, \boldsymbol{x}_i^{\text{ref}},\, \boldsymbol{u}_i^{\text{ref}}, \boldsymbol{x}_i^{\text{ref}},\, \boldsymbol{u}_i^{\text{ref}}\right). \tag{16}$$

Then the first Newton direction $(\Delta \boldsymbol{x}_i,\, \Delta \boldsymbol{u}_i)$ computed in Algorithm 1, line 2 is identical to the solution of the linear MPC problem computed via (3). This observation is illustrated in Figure 1 for the simple problem:

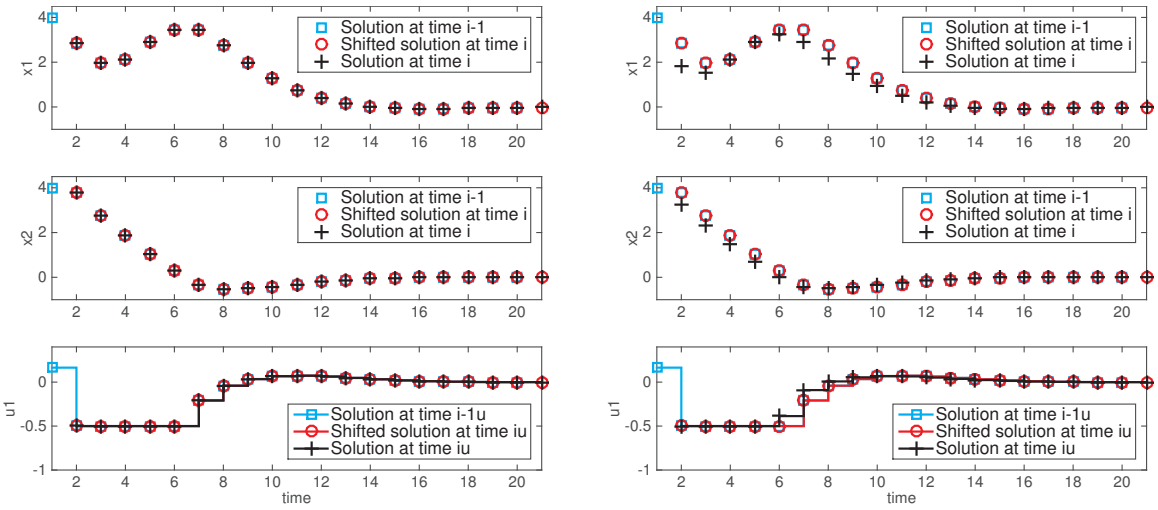
$$\begin{aligned}
\min_{\boldsymbol{x}_i, \boldsymbol{u}_i} \quad & \sum_{k=0}^{N} \|\boldsymbol{x}_{i,k}\|_2^2 + 20 \sum_{k=0}^{N-1} \|\boldsymbol{u}_{i,k}\|_2^2 && \text{(17a)}\\
\text{s.t.} \quad & x_{i,0} = \hat{x}_i, && \text{(17b)}\\
& x_{i,k+1} = 0.9 x_{i,k} + \begin{bmatrix} \sin\left(\begin{bmatrix} 0 & 1 \end{bmatrix} x_{i,k}\right) \\ u_{i,k} + u_{i,k}^3 \end{bmatrix}, && \text{(17c)}\\
& |u_{i,k}| \leq 0.5, \quad k = 0, \ldots, N-1, && \text{(17d)}
\end{aligned}$$

with state $x_{i,k} \in \mathbb{R}^2$, input $u_{i,k} \in \mathbb{R}$, and $N = 19$. The optimal trajectories obtained via linear MPC (1) are reported using circles (○). The trajectories obtained from performing a single iteration of Algorithm 1, with a full step ($\alpha = 1$) and $\boldsymbol{x}_i^{\text{guess}} = \boldsymbol{x}_i^{\text{ref}}$, $\boldsymbol{u}_i^{\text{guess}} = \boldsymbol{u}_i^{\text{ref}}$ as the initial guess are displayed using plus signs (+). The trajectories resulting from running Algorithm 1 to full convergence are displayed using squares (□).

3.2 *Warm-started SQP for NMPC*

Algorithm 1 requires the initial guess $\boldsymbol{x}_i^{\text{guess}}$, $\boldsymbol{u}_i^{\text{guess}}$ as an input. Selecting an adequate initial guess is crucial for obtaining a fast and reliable convergence of the SQP iterations. First, a good initial guess allows for avoiding the SQP algorithm to exit with an infeasible solution; secondly, it allows for taking full Newton steps ($\alpha = 1$) in the SQP algorithm, hence allowing for a fast convergence rate.

In the previous section, we showed that when the reference trajectory is provided as an initial guess, the first SQP step, if full, provides the same solution as linear MPC. However, the reference trajectory is



(a) In the absence of disturbance and model error, the guess obtained via shifting the previous solution is typically an excellent approximation of the current solution. In this graph, the guess for time i and the corresponding solution are indistinguishable.

(b) In the presence of disturbances, $\hat{x}_i$ diverges significantly from the predicted one $x_{i-1,1}$, and therefore the guess x_i^{guess} obtained via shifting needs a correction.

Figure 2. Illustration of the shifting procedure.

sometimes a rather poor initial guess for the SQP strategy, e.g. when the actual system trajectory is not in the neighbourhood of the reference.

In the specific context of SQP for NMPC, a very good initial guess for the discrete time instant i can be constructed, provided that a good solution has been obtained at the previous time instant $i-1$. In such a case, the following *shifting procedure* can be used. Shifting constructs an initial guess $\boldsymbol{x}_i^{\text{guess}}$, $\boldsymbol{u}_i^{\text{guess}}$ for time i using the solution:

$$(\boldsymbol{x}_{i-1},\, \boldsymbol{u}_{i-1}) = \text{SQP}\left(\hat{x}_{i-1},\, \boldsymbol{x}_{i-1}^{\text{guess}},\, \boldsymbol{u}_{i-1}^{\text{guess}},\, \boldsymbol{x}_{i-1}^{\text{ref}},\, \boldsymbol{u}_{i-1}^{\text{ref}}\right), \tag{18}$$

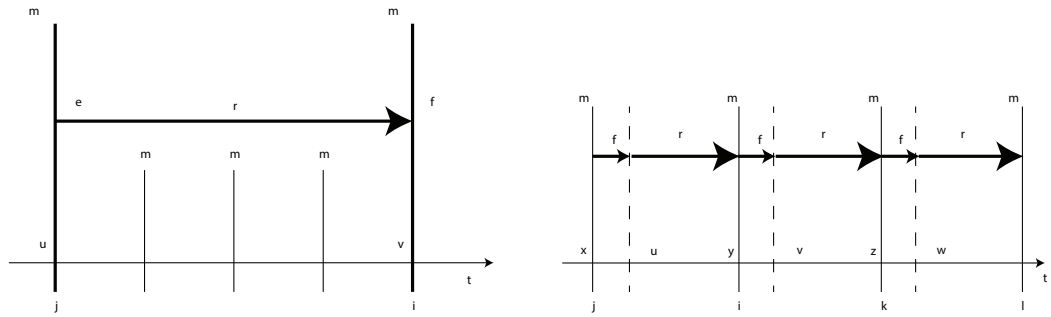
obtained for time $i-1$. Shifting assumes that the system evolution follows closely the predicted trajectory, i.e. it assumes $\hat{x}_i \approx x_{i-1,1}$. The shifting procedure then reads as follows

$$x_{i,k}^{\text{guess}} = x_{i-1,k+1}, \quad k = 0, ..., N-1, \tag{19a}$$
$$u_{i,k}^{\text{guess}} = u_{i-1,k+1}, \quad k = 0, ..., N-2, \tag{19b}$$
$$x_{i,N}^{\text{guess}} = f\left(x_{i,N-1}^{\text{guess}}, u_{i,N-1}^{\text{guess}}\right). \tag{19c}$$

It shall be observed that if the solution $(\boldsymbol{x}_{i-1},\, \boldsymbol{u}_{i-1})$ is feasible, then the *shifted* solution is feasible with respect to the dynamic constraints. Additionally, if the guess for the time instant i obtained via the shifting procedure is sufficiently close to the exact solution of the NMPC problem (9), then the following statements hold [14]:

- full Newton steps are selected in the SQP iterations, i.e. $\alpha = 1$ at line 3 of Alg. 1;
- the first iteration of Alg. 1 provides already an excellent approximation of the exact solution to the NMPC problem (9).

The guess for the last control input $u_{i,N-1}^{\text{guess}}$ can be selected via different approaches. Whenever available, a control law $\kappa(x)$ which locally stabilises the system while enforcing the path constraints can be used to compute $u_{i,N-1}^{\text{guess}} = \kappa(x_{i,N-1}^{\text{guess}})$. In the absence of unmodelled perturbations, this choice guarantees recursive feasibility of the MPC scheme, i.e. feasibility not only with respect to the system dynamics, but also with



(a) SQP: a model-based prediction of the state estimate $\hat{x}_i$ can be obtained at time $i-1$ in order to start the next SQP algorithm after the latest control policy u_{i-1}^{NMPC} is delivered. The prediction can be used to account for the physical time elapsing while the SQP algorithm is running. However, subsequent measurements of the system evolution occurring as the SQP algorithm is running are not incorporated in the SQP iteration, and are therefore ignored.

(b) RTI: every time a state estimate $\hat{x}_i$ is obtained from the measurements, the feedback phase of the RTI step is triggered. Once the feedback phase is completed, a new preparation phase is started. RTI takes successive full Newton steps, always using the latest information available on the system.

Figure 3. Illustration of the SQP and RTI timelines. Note that the sampling time for the two approaches is different, so that $t_i^{\text{SQP}} > t_i^{\text{RTI}}$.

respect to the path constraints. Under some mild assumptions, it moreover ensures a decrease in the MPC cost, therefore enforcing stability of the closed-loop system [43]. In practice, simpler approaches are often used. The simplest one is to create a copy of the control input at stage $N-2$, i.e.

$$u_{i,N-1}^{\text{guess}} = u_{i,N-2}^{\text{guess}} = u_{i-1,N-1}. \tag{20}$$

The shifting procedure is illustrated in Figures 2(a) and 2(b). It can be seen how, for the unperturbed case, shifting provides a guess which is extremely close to the solution. In the perturbed case, a correction of the guess is necessary but shifting is nevertheless close to the solution. These observations provide an important intuitive justification of the Real-Time Iteration approach, described in Section 4. Before presenting it in detail, we first review the *real-time dilemma*.

3.3 *The real-time dilemma*

Upon obtaining a new state estimate $\hat{x}_i$, the SQP procedure can be started. The real-time dilemma stems from the fact that while the SQP iterations are performed, the physical system evolves, and the information used to compute the state estimate $\hat{x}_i$ becomes outdated.

Clearly, this problem can be partially addressed by using a *prediction* of what the state of the system will be at the time the SQP algorithm will be completed, as opposed to directly using the current state estimate. However, even when using such a prediction approach, since updating the control policy requires the completion of the SQP algorithm, the SQP procedure introduces a potentially large delay between gathering measurements from the physical system and delivering the corresponding required control action. It follows that even if the SQP computational time is accounted for via a state prediction, the SQP algorithm is nonetheless based on outdated system information. We illustrate this key issue in Fig. 3(a).

The key idea of the RTI procedure detailed in Section 4, and first presented in [14] is to consistently incorporate the latest information on the system evolution in the iterations computing the NMPC solutions. The real-time dilemma then consists in choosing between applying an exact solution computed using outdated information versus applying an approximate solution computed using the most up-to-date information.

Summary of the section

- When SQP is deployed on NMPC, and the reference trajectory is used as an initial guess, the first step of a full step Gauss-Newton SQP delivers the same control solution as linear MPC (Lemma 1).

- In the context of NMPC, the SQP iteration can be efficiently warm-started by shifting the solution obtained at the previous time instant (Sec. 3.2). In the presence of reasonably small disturbances, the SQP algorithm then needs only a couple of full Newton steps to reach full convergence.
- When running SQP to full convergence, only the first iteration is using an up-to-date estimate of the system state $\hat{x}_i$. Subsequent iterations are still performed based on $\hat{x}_i$, while the system state evolves, making $\hat{x}_i$ outdated (Sec. 3.3).

4. The Real-Time Iteration (RTI)

In this section we recall the RTI approach first introduced in [14]. It is important to remark that several approaches for real-time NMPC have been proposed in the literature. The *Newton-Type Controller* [35] shares many similarities with the RTI approach, in particular the fact that it performs only one full Newton step per sampling time. The main difference is that it does not use the generalised tangential predictor of the initial value embedding [15] and it is based on a sequential discretisation of the system dynamics. The *Continuation/GMRES Method* [38] is also based on taking a single full Newton step per sampling instant. However, rather than on SQP it is based on an interior-point like approach where the barrier parameter is fixed at a prescribed value and the barrier function is not self concordant. Convergence is improved by making use of a tangential predictor. The *advanced step controller* [52] is based on an interior point approach and consists in solving the NLP to convergence at each iterate. However, (a) the computational delay is accommodated for by using a prediction of the future initial state and (b) once the actual state is known, the solution is corrected using a tangential predictor, similarly to the Continuation/GMRES method. In this paper, we decided to further restrict our attention to the RTI approach because of the stronger similarities to linear MPC, since both are based on the solution of a QP at each sampling instant and can therefore account for active set changes [15].

RTI approach consists in performing the Newton steps always using the latest information on the system evolution. For the sake of clarity, we first introduce a simplified version of the RTI algorithm (Section 4.1), then the complete RTI algorithm (Section 4.2). Finally, we establish a comparison between RTI-based NMPC and linear MPC that will reveal a strong connection between the two approaches (Section 4.3).

4.1 *Single full Newton step in SQP*

To introduce the RTI algorithm, it is useful to first consider the following Algorithm 2, which is a simplified version of Algorithm 1 with the addition of a shifting procedure for constructing the initial guess. Here, at every discrete time instant i, the NMPC solution is updated using a single full Newton step, instead of performing the SQP algorithm to full convergence. The Newton step is taken on the NMPC solution obtained at the previous time $i-1$, after the shifting procedure (19) is applied.

Algorithm 2: Newton iteration for NMPC at discrete time i

Input: state estimate $\hat{x}_i$, reference trajectory $(\boldsymbol{x}_i^{\text{ref}}, \boldsymbol{u}_i^{\text{ref}})$ and previous NMPC solution $(\boldsymbol{x}_{i-1}, \boldsymbol{u}_{i-1})$

1 Shift $(\boldsymbol{x}_{i-1}, \boldsymbol{u}_{i-1})$ according to (19) to construct $(\boldsymbol{x}_i^{\text{guess}}, \boldsymbol{u}_i^{\text{guess}})$

2 Evaluate $r_{i,k}$, $h_{i,k}$ and the sensitivities $A_{i,k}$, $B_{i,k}$, $C_{i,k}$, $D_{i,k}$, $H_{i,k}$, $J_{i,k}$ at $(\boldsymbol{x}_i^{\text{guess}}, \boldsymbol{u}_i^{\text{guess}})$ using (12)

3 Construct and solve $\text{QP}_{\text{NMPC}}\left(\hat{x}_i, \boldsymbol{x}_i^{\text{guess}}, \boldsymbol{u}_i^{\text{guess}}, \boldsymbol{x}_i^{\text{ref}}, \boldsymbol{u}_i^{\text{ref}}\right)$ as in (11) to get $(\Delta\boldsymbol{x}_i, \Delta\boldsymbol{u}_i)$

4 Apply the full Newton step

$$(\boldsymbol{x}_i, \boldsymbol{u}_i) \leftarrow (\boldsymbol{x}_i^{\text{guess}}, \boldsymbol{u}_i^{\text{guess}}) + (\Delta\boldsymbol{x}_i, \Delta\boldsymbol{u}_i)$$

return NMPC solution $(\boldsymbol{x}_i, \boldsymbol{u}_i)$

The efficiency of Algorithm 2 at providing a good approximation of the fully converged NMPC solution

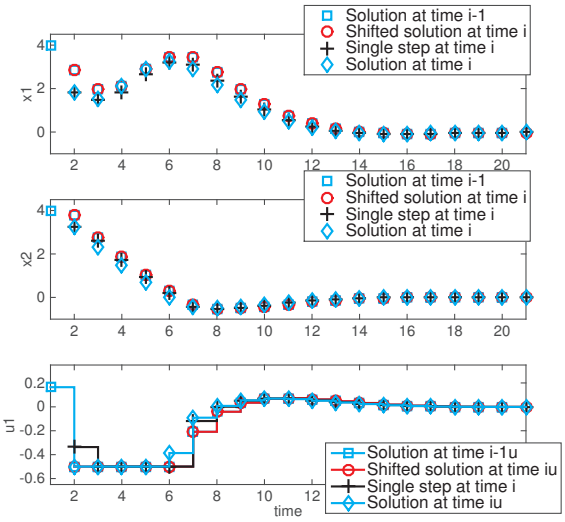


Figure 4. Illustration of the single full Newton step approach, including state noise in the closed-loop simulation.

hinges on the assumption that the shifted NMPC solution obtained at time $i-1$ is a good initial guess for the NMPC solution at time i. Under this assumption, a full Newton step can be taken ($\alpha = 1$), and provides an excellent approximation of the fully converged NMPC solution. An illustration of this fact is given in Figure 4, where a single full Newton step strategy is compared to a fully converged SQP method. Algorithm 2 computes the RTI feedback control policy. However, as we will present in the next section, the genuine RTI algorithm divides the computations in two phases so as to achieve shorter feedback latencies. It is also interesting to note that Algorithm 2 constructs the QP using the shifted solution guess directly, even though the initial state might be different from the state estimate $\hat{x}_i$ in this trajectory. This concept is typically referred to as initial value embedding and it allows for a generalized tangential predictor from one time step to the next, as discussed in more detail in [15].

4.2 *The RTI algorithm: preparation-feedback split*

The RTI algorithm is an improved version of Algorithm 2, where its *feedback time* is reduced. The improvement is using the fact that steps 1 and 2 of Algorithm 2 do not require the knowledge of the state estimate $\hat{x}_i$, and can therefore be performed *before* the state estimate $\hat{x}_i$ becomes available.

The RTI scheme (see Algorithm 3) thus proposes to split the operation between:

- a *preparation phase*, performing the computations involved in the steps 1 and 2 of Algorithm 2 *prior* to obtaining the new state estimate $\hat{x}_i$.
- a *feedback phase*, performing the computations involved in steps 3 and 4 upon obtaining the latest state estimate $\hat{x}_i$.

Note that usually the Gauss-Newton Hessian approximation [4], i.e. $H_{i,k} = W_{i,k}$, is used because (a) it does not require the computation of second order derivatives and (b) it always delivers a positive (semi)definite Hessian approximation. For a detailed overview on the RTI scheme, including a proof of nominal stability, we refer to [11, 12, 16].

It is important to remark that:

- The delay introduced by the feedback time can be accommodated as in linear MPC, by including a corresponding prediction in the state estimate.
- The overall sampling time $t_i - t_{i-1}$ that can be achieved by the RTI scheme is limited by the total time spent in solving both the feedback phase and the preparation phase.
- The time required to perform the feedback phase is practically the same as the time required to solve

Algorithm 3: RTI for NMPC at discrete time i

Preparation phase performed over the time interval $[t_{i-1},\, t_i[$

Input: previous NMPC solution $(\boldsymbol{x}_{i-1},\, \boldsymbol{u}_{i-1})$, reference $(\boldsymbol{x}_{i-1}^{\text{ref}},\, \boldsymbol{u}_{i-1}^{\text{ref}})$

1 Shift $(\boldsymbol{x}_{i-1},\, \boldsymbol{u}_{i-1})$ according to (19) to construct $(\boldsymbol{x}_i^{\text{guess}},\, \boldsymbol{u}_i^{\text{guess}})$

2 Evaluate $r_{i,k},\, h_{i,k}$ and the sensitivities $A_{i,k},\, B_{i,k},\, C_{i,k},\, D_{i,k},\, H_{i,k},\, J_{i,k}$ at $(\boldsymbol{x}_i^{\text{guess}},\, \boldsymbol{u}_i^{\text{guess}})$ using (12)

3 Form QP (11) omitting $\hat{x}_i$, prepare all possible computations (e.g. condensing, matrix factorisations)

return QP (11)

Feedback phase performed at time t_i upon availability of $\hat{x}_i$

Input: $\hat{x}_i$, prepared QP (11)

4 Compute $(\Delta \boldsymbol{x}_i,\, \Delta \boldsymbol{u}_i)$ by introducing $\hat{x}_i$ in QP (11) and solving it

5 Apply the full Newton step

$$(\boldsymbol{x}_i,\, \boldsymbol{u}_i) \leftarrow (\boldsymbol{x}_i^{\text{guess}},\, \boldsymbol{u}_i^{\text{guess}}) + (\Delta \boldsymbol{x}_i,\, \Delta \boldsymbol{u}_i)$$

return NMPC solution $(\boldsymbol{x}_i,\, \boldsymbol{u}_i)$

the linear MPC problem.

- Part of the computations related to the QP solution can often be moved to the preparation phase, e.g. using a technique called condensing [4, 44].
- The sampling time that can be achieved via RTI-based NMPC increases from standard linear MPC by the time required for the preparation phase.
- It is typically desirable that the feedback time is only a fraction of the overall sampling time. Because the preparation phase can often fit in the time after the feedback phase and before the next state estimate is available, RTI NMPC can in many cases be applied at the same sampling frequency of linear MPC based on a model pre-linearised offline.

An illustration of the RTI timeline is displayed in Figure 3(b).

4.3 *Extending linear MPC to NMPC via the RTI*

We want to first establish a clear connection between linear MPC and the RTI approach described in Algorithm 3, and then clarify how linear MPC can be extended to NMPC. First it is useful to observe the following:

Lemma 2: *When a Gauss-Newton Hessian approximation is used, i.e.* $H_{i,k} = W_{i,k}$ *and* $\boldsymbol{x}_{i-1} = \boldsymbol{x}_i^{\text{ref}},\, \boldsymbol{u}_{i-1} = \boldsymbol{u}_i^{\text{ref}}$ *are fed as inputs to the preparation and feedback phases of Algorithm 3, then RTI delivers the same solutions as the linear MPC scheme* (1)–(3).

Proof. Follows from Lemma 1. □

As a consequence of Lemma 2, linear MPC can be regarded as an RTI scheme where the preparation phase is performed only once, usually offline, based on the reference trajectory. Linear MPC then runs only the feedback phase of Algorithm 3. This observation entails that a linear MPC scheme can be easily extended to an approximate NMPC scheme via the RTI approach. The extension requires that one performs the preparation phase online, i.e. that a new linearisation of the NMPC problem is performed at every discrete time instant, based on a shifting of the previous NMPC solution. Note that algorithms which could be classified as intermediate between linear MPC and RTI based NMPC have been proposed in [5, 6].

The difference between linear MPC, RTI NMPC and fully converged NMPC is illustrated in Figures 5, 6(a) and 6(b). Figure 5 shows the open loop predictions at time instant $i = 2$ in the presence of

state noise. It can be seen that, while the RTI prediction is close to the fully converged SQP solution, the linear MPC scheme delivers a different solution. Figure 6(a) displays the closed-loop trajectory in the absence of state noise. Again, RTI and fully converged SQP are indistinguishable, while linear MPC differs significantly. Figure 6(b) proposes the same simulation with the addition of state disturbances. The same behaviour is observed also in this case.

4.4 *Global vs. local optimality*

Linear MPC can be preferred because the convexity of the underlying optimisation problem guarantees one to compute its global solution at every time instant. In contrast, the nonconvexity of the optimisation problem underlying NMPC problems prevents such guarantees to be provided. We will argue here, however, that under some assumptions, the solution provided by the RTI scheme will follow the global solution of the NMPC problem. The required assumptions are the following ones:

(1) the RTI scheme is warm-started at the global optimum,
(2) the sampling frequency is sufficiently high,
(3) there are no jumps in the reference and the state,
(4) the OCP underlying the NMPC problem fulfills Second Order Sufficient Conditions (SOSC) [37] for every feasible initial condition,
(5) the global optimum depends continuously on the initial state and reference.

A formal proof of this statement is rather involved and can be found in [11] for local optimality. Assumption 1 additionally ensures that the local optimum followed by RTI is also the global one. In order to give an intuitive understanding, we remark that Assumption 4 guarantees that the solution manifold is smooth and has no bifurcation. This entails that the RTI will keep track of the global solution manifold as long as it starts on that manifold and the initial conditions $\hat{x}_i$ are sufficiently close to the predicted ones. The latter is guaranteed by Assumptions 1, 2, and 3. Faster sampling results in a larger set of disturbances for which RTI tracks the global solution manifold. Assumption 5 ensures that the solution manifold is continuous in time. In practice, the warm starting can be performed by setting the system at a reference steady state and initialising the RTI algorithm accordingly. Consequently, RTI is initialised at the global optimum.

In Section 4.3 we established the connection between RTI and linear MPC. We remark that, when controlling a nonlinear system, linear MPC can be seen as a specific case of RTI initialised at the reference rather than at the current state and control prediction. In this framework, it becomes clear that the global optimum achieved by linear MPC is actually a local approximation of the NMPC solution in a neighbourhood of the reference. Therefore, the linear MPC solution is not the global optimum for the nonlinear problem.

Summary of the section:

- NMPC based on RTI performs a single full Newton step at every discrete time instant, relying on the fast convergence of Newton-type optimisation. This approach allows for performing the Newton steps using the latest information on the system evolution.
- RTI can be divided in a preparation phase and a feedback phase, minimising the delay between obtaining a new state estimate and updating the control policy.
- Linear MPC can be regarded as an RTI scheme where the preparation phase is performed only once, offline. In this context, the extension of linear MPC to RTI-based NMPC then simply requires that the preparation phase is repeatedly performed online.

5. MPC and NMPC for continuous-time systems

The preparation phase of Algorithm 3 requires to compute online the evaluation of the discretised system dynamics (4) with the associated sensitivities, namely $\nabla f\left(x,u\right)$. In the case the system is readily described as a discrete dynamic system, computing $f\left(x,u\right)$ and $\nabla f\left(x,u\right)$ is straightforward. However, in many

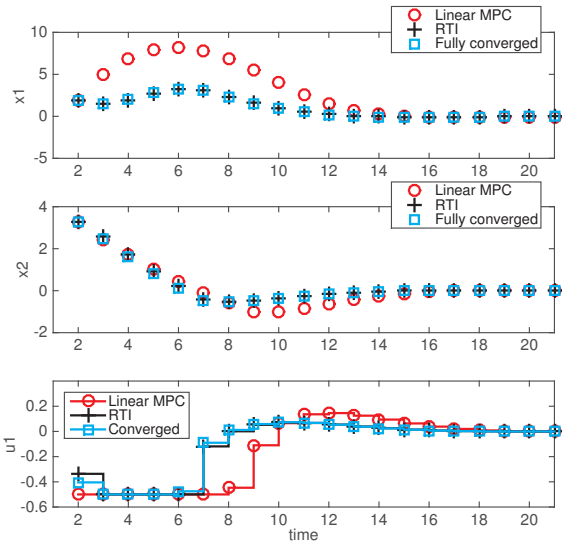
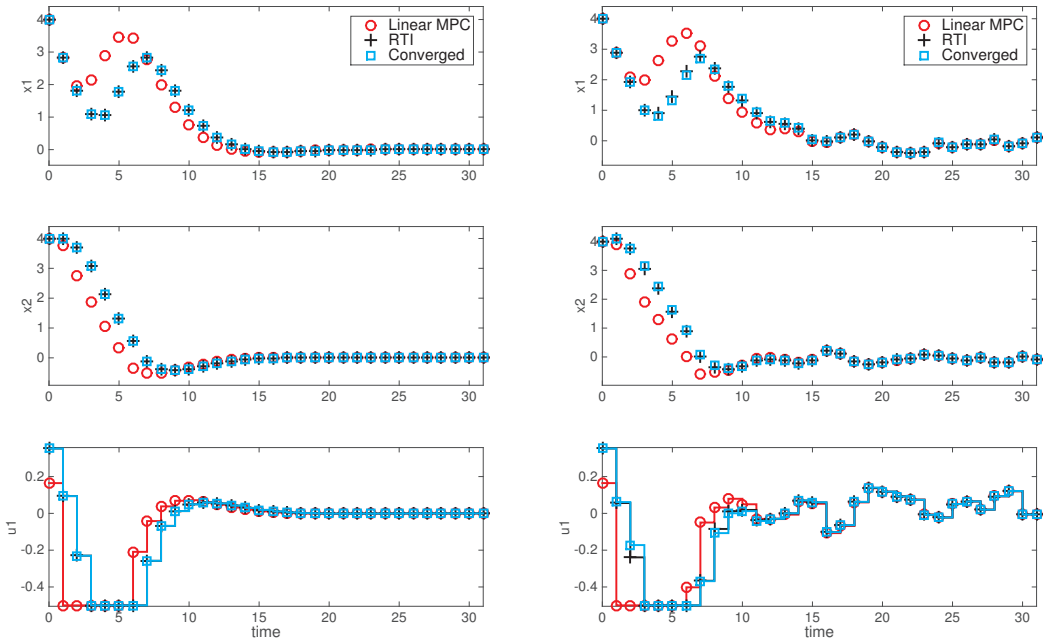


Figure 5. Illustration of the RTI solution vs. the linear MPC solution at the discrete time instant $i = 2$, with state disturbances.



(a) No state disturbances.

(b) State disturbances of covariance 0.1.

Figure 6. Illustration of the RTI solution vs. the linear MPC solution in closed-loop simulations, with and without state disturbances.

applications, the system dynamics are available in a continuous form, typically as an Ordinary Differential Equation (ODE) of the form:

$$\dot{s}(t) = F\left(s(t), v(t)\right), \tag{21}$$

where we use $s(t)$ and $v(t)$ to denote the continuous-time states and controls and, thus, distinguish them from their discrete-time counterparts.

In this section, we will present a family of numerical methods for simulation and sensitivity generation. It is important to stress that the well-known matrix exponential can also be considered as such a method for numerical simulation. However, depending on the system considered, other methods might be more

accurate and less computationally intensive. We also want to stress the fact that several integration steps can be taken inside each control interval in order to increase the accuracy of the simulation. We will also sketch how the sensitivities can be propagated in case multiple integration steps are taken.

For the sake of simplicity we consider here an explicit ODE having time-invariant dynamics, though the following developments can be easily extended to the time-varying case and to implicit ODE or Differential Algebraic Equation (DAE) systems. We consider a piecewise constant parametrization of the control inputs $v(t)$ with parameters $u_0, \ldots, u_{N-1}$, such that $v(t) = u_k, \quad t \in [t_k, t_{k+1}[$. Note that, in principle, the restriction to piecewise constant control parameterizations is not required as long as local basis functions are used to maintain the structure of the resulting OCP. Also note that piecewise constant controls are most commonly used in practice for the ease of implementation using zero-order holders. The following notation will be used further to refer to the respective Jacobian evaluations of the ODE in (21):

$$F_s(x,u) = \left.\frac{\partial F(s,v)}{\partial s}\right|_{s=x,v=u}, \quad F_v(x,u) = \left.\frac{\partial F(s,v)}{\partial v}\right|_{s=x,v=u}. \tag{22}$$

5.1 *First linearise then discretise*

In the context of linear MPC, it is common to compute the linearisation in (6) by directly linearising the continuous time formulation using the reference trajectory. Subsequently, a discrete time description can be obtained via the matrix exponential. This means that for any time instant i in the reference trajectory, matrices A_k, B_k are given by:

$$A_k = e^{F_s\left(x_{i,k}^{\mathrm{ref}},\, u_{i,k}^{\mathrm{ref}}\right)T_{\mathrm{s}}}, \qquad B_k = \int_0^{T_{\mathrm{s}}} e^{F_s\left(x_{i,k}^{\mathrm{ref}},\, u_{i,k}^{\mathrm{ref}}\right)(T_{\mathrm{s}}-\tau)} F_v\left(x_{i,k}^{\mathrm{ref}},\, u_{i,k}^{\mathrm{ref}}\right) \mathrm{d}\tau. \tag{23}$$

Note that T_{s} is the chosen physical sampling time satisfying $T_{\mathrm{s}} = t_i - t_{i-1}$, i.e. $T_{\mathrm{s}} = \frac{T}{N}$ in case of an equidistant grid over the control horizon with length T.

For a fully predefined reference, (23) can be performed off-line, prior to deploying the linear MPC scheme such as performed e.g. in [7, 10]. If the reference is time-invariant and feasible, Eq. (23) provides an exact linearisation of the continuous dynamics as long as the system resides in this steady state. In all other cases, the linearisation provided by (23) becomes inexact. Therefore, it can be preferable to first discretise the system using numerical integration. In the following, we will illustrate some integration schemes which allow for highly accurate discretisations for any chosen sampling time.

5.2 *First discretise then linearise*

The exact discrete dynamics (4) corresponding to the system in (21) are formally given by:

$$f\left(x_k, u_k\right) := s\left(t_{k+1}\right), \qquad \text{with} \qquad \begin{cases} \dot{s}(\tau) \;= F\left(s(\tau), u_k\right), & \tau \in [t_k, t_{k+1}] \\ s\left(t_k\right) = x_k, \end{cases} \tag{24}$$

where $T_{\mathrm{s}} = t_{k+1} - t_k$ and the piecewise constant control parameterization has been used. In case T_{s} is not negligible compared to the time constant of the nonlinear dynamics of the system, it is not advisable to use the matrix exponential (23) to obtain the discrete linearisation (11c). A more generally applicable approach is to numerically approximate the discrete dynamics from (24), in order to obtain a linearisation with a specific desired accuracy. This requires the numerical simulation of a nonlinear system of differential equations, see [28, 29] for a detailed overview.

For the purpose of this article, let us restrict the discussion to the class of one-step methods, an important family which includes Runge-Kutta (RK) methods. Unlike multistep schemes, these one-step methods have the advantage that they do not require any start-up procedure which makes them rather suitable for short

simulation times. In addition, we will consider the integration over one shooting interval using a fixed step size T_{i} resulting in $N_{\mathrm{S}} = \frac{T_{\mathrm{s}}}{T_{\mathrm{i}}}$ integration steps per shooting interval. It is important to note that these choices are made only for the sake of simplicity, while in general any integration method could be used, e.g. a multistep method and/or an adaptive step size implementation.

5.2.1 Explicit Euler

Let us start with the simplest but also typically not the most efficient integration scheme, known as the Euler method and detailed in Algorithm 4. This first order method provides us with the following approximation of the exact discrete dynamics

$$\| f\left(x_k, u_k\right) - f_{\mathrm{Euler}}\left(x_k, u_k\right) \| = O\left(T_{\mathrm{i}}\right), \tag{25}$$

which corresponds to the global or transported error [28].

The resulting discrete time nonlinear system needs to be linearised to obtain (12a), which can be performed efficiently using the principle of Internal Numerical Differentiation [4], which computes the needed sensitivities A_k, B_k by differentiating the integration method itself. The required derivatives can be evaluated using the techniques of Algorithmic Differentiation (AD) [22] and propagated forward through the integration algorithm using the chain rule as detailed in Algorithm 4, where we denote the identity matrix by I. Note that while the numerical integration scheme is approximating the real system dynamics up to the integrator order, the sensitivities computed via the approach described here are the exact derivatives of the integration scheme, up to machine precision. Hence the approach departs from computing the sensitivities using e.g. variational approaches, where both the integration of the system dynamics and the sensitivities are inexact [8]. For more information on the application of AD to explicit integration schemes and an extension to adjoint sensitivity analysis, we refer the reader to [46].

Algorithm 4: N_{S} steps explicit Euler with forward AD

Input: x_k, u_k

$x^+ = x_k, \quad A_k = I, \quad B_k = \mathbb{0}$

for $n = 1 : N_{\mathrm{S}}$ **do**

$\quad x^+ \leftarrow x^+ + T_{\mathrm{i}}\, F\left(x^+, u_k\right)$

$\quad \begin{bmatrix} A_k & B_k \end{bmatrix} \leftarrow \left(I + T_{\mathrm{i}}\, F_s\left(x^+, u_k\right)\right) \begin{bmatrix} A_k & B_k \end{bmatrix} + \begin{bmatrix} \mathbb{0} & T_{\mathrm{i}}\, F_v\left(x^+, u_k\right) \end{bmatrix}$

end

return $x^+,\ A_k,\ B_k$

5.2.2 Alternative explicit schemes

Even though the explicit Euler scheme is very simple, it typically does not yield the most efficient approach to obtain the desired accuracy because it is only of order one. Indeed, many more explicit integration methods have been developed and can be found e.g. in [28]. A popular example is the 4-stage Runge-Kutta method of order 4, detailed in Algorithm 5 together with its forward sensitivity propagation. Unlike explicit Euler, this scheme results in a global error which satisfies

$$\| f\left(x_k, u_k\right) - f_{\mathrm{RK4}}\left(x_k, u_k\right) \| = O\left(T_{\mathrm{i}}^4\right), \tag{26}$$

where T_{i} is again the chosen step size. Similar to our previous discussion on Euler, the computed sensitivities are the exact derivatives $A_k = \frac{\mathrm{d}f_{\mathrm{RK4}}(x_k,u_k)}{\mathrm{d}x_k}$, $B_k = \frac{\mathrm{d}f_{\mathrm{RK4}}(x_k,u_k)}{\mathrm{d}u_k}$ and can be interpreted as applying this RK scheme to the forward system of variational differential equations (VDE) [46].

Note that both Algorithm 4 and 5 have been included mostly for illustration purposes, while efficient implementations typically rely on AD tools (often implemented using either operator overloading or source

code transformation) as described in e.g. [22, 46]. The open-source ACADO Toolkit software [32] provides a high-level framework for deploying a code-generated RTI approach, including auto generated integrators with tailored sensitivity propagation [41].

Algorithm 5: N_S steps Runge-Kutta 4 with forward AD

Input: $x_k,\ u_k$

$x^+ = x_k, \quad A_k = I, \quad B_k = \mathbb{0}$

for $n = 1 : N_\mathrm{S}$ **do**

$$k_1 \leftarrow F(x^+, u_k)$$

$$\begin{bmatrix} \frac{dk_1}{dx_k} & \frac{dk_1}{du_k} \end{bmatrix} \leftarrow F_s\left(x^+, u_k\right) \begin{bmatrix} A_k & B_k \end{bmatrix} + \begin{bmatrix} \mathbb{0} & F_v\left(x^+, u_k\right) \end{bmatrix}$$

$$k_2 \leftarrow F(x^+ + \tfrac{T_\mathrm{i}}{2} k_1, u_k)$$

$$\begin{bmatrix} \frac{dk_2}{dx_k} & \frac{dk_2}{du_k} \end{bmatrix} \leftarrow F_s(x^+ + \tfrac{T_\mathrm{i}}{2} k_1, u_k) \begin{bmatrix} \left(A_k + \frac{T_\mathrm{i}}{2} \frac{dk_1}{dx_k}\right) & \left(B_k + \frac{T_\mathrm{i}}{2} \frac{dk_1}{du_k}\right) \end{bmatrix} + \begin{bmatrix} \mathbb{0} & F_v(x^+ + \frac{T_\mathrm{i}}{2} k_1, u_k) \end{bmatrix}$$

$$k_3 \leftarrow F(x^+ + \tfrac{T_\mathrm{i}}{2} k_2, u_k)$$

$$\begin{bmatrix} \frac{dk_3}{dx_k} & \frac{dk_3}{du_k} \end{bmatrix} \leftarrow F_s(x^+ + \tfrac{T_\mathrm{i}}{2} k_2, u_k) \begin{bmatrix} \left(A_k + \frac{T_\mathrm{i}}{2} \frac{dk_2}{dx_k}\right) & \left(B_k + \frac{T_\mathrm{i}}{2} \frac{dk_2}{du_k}\right) \end{bmatrix} + \begin{bmatrix} \mathbb{0} & F_v(x^+ + \frac{T_\mathrm{i}}{2} k_2, u_k) \end{bmatrix}$$

$$k_4 \leftarrow F(x^+ + T_\mathrm{i}\, k_3, u_k)$$

$$\begin{bmatrix} \frac{dk_4}{dx_k} & \frac{dk_4}{du_k} \end{bmatrix} \leftarrow F_s(x^+ + T_\mathrm{i}\, k_3, u_k) \begin{bmatrix} \left(A_k + T_\mathrm{i} \frac{dk_3}{dx_k}\right) & \left(B_k + T_\mathrm{i} \frac{dk_3}{du_k}\right) \end{bmatrix} + \begin{bmatrix} \mathbb{0} & F_v(x^+ + T_\mathrm{i}\, k_3, u_k) \end{bmatrix}$$

$$x^+ \leftarrow x^+ + \tfrac{T_\mathrm{i}}{6} \left(k_1 + 2k_2 + 2k_3 + k_4\right)$$

$$\begin{bmatrix} A_k & B_k \end{bmatrix} \leftarrow \begin{bmatrix} A_k & B_k \end{bmatrix} + \tfrac{T_\mathrm{i}}{6} \left(\begin{bmatrix} \frac{dk_1}{dx_k} & \frac{dk_1}{du_k} \end{bmatrix} + 2 \begin{bmatrix} \frac{dk_2}{dx_k} & \frac{dk_2}{du_k} \end{bmatrix} + 2 \begin{bmatrix} \frac{dk_3}{dx_k} & \frac{dk_3}{du_k} \end{bmatrix} + \begin{bmatrix} \frac{dk_4}{dx_k} & \frac{dk_4}{du_k} \end{bmatrix} \right)$$

end

return $x^+,\ A_k,\ B_k$

5.2.3 Implicit integration schemes

In case of a stiff system of differential equations, it is advised to use an implicit integration scheme instead [29]. In contrast to the previous methods, they require the solution of an implicit system of equations to perform each integration step. Their improved stability properties typically result in a better numerical accuracy for a given computational effort. An implicit integration scheme can readily be extended to deal with an implicit ODE system or with a DAE of index 1. To keep our discussion compact, we do not detail any implicit or semi-implicit methods [29] nor the computation of their sensitivities in this article. More information and a possible implementation can be found in [1, 30]. Additionally, auto generated implicit integrators with tailored sensitivity propagation are presented in [41].

5.3 *Exponential integrators: integration by linearisation*

In our discussion on how to compute the discretised and linearised dynamics from (11c), we distinguished between two seemingly different approaches. Inspired by linear systems theory, the first approach linearises the continuous ODE system and uses the matrix exponential to obtain its discrete time representation. However, this can be considered a special case of the more general family of exponential integrators [31]. This means that it can actually be considered part of the second approach in which the exact nonlinear dynamics (24) are numerically approximated.

Exponential integrators are based on a linearisation of the nonlinear ODE from Eq. (21) in a point $(\bar{x}, \bar{u})$

$$\Delta \dot{s}(t) = F_s(\bar{x}, \bar{u})\, \Delta s(t) + F_v(\bar{x}, \bar{u})\, \Delta v(t) + g(\Delta s(t), \Delta v(t)), \tag{27}$$

where $\Delta s(t) = s(t) - \bar{x}$, $\Delta v(t) = v(t) - \bar{u}$ and $g(\Delta s(t), \Delta v(t))$ denotes the nonlinear remainder. Because the linear part of the latter equations is integrated exactly, this type of methods are popular for stiff differential equations. The simplest numerical scheme holds the value of the function $g(\cdot)$ constant over the integration step resulting in the first order exponential Euler approximation. The linearisation based approach from Section 5.1 can also be obtained by applying the latter method directly to the nonlinear ODE, using the reference trajectory as a linearisation point. This directly shows the limitations of that approach, since only one integration step is performed and this in an offline manner instead of applying an integration method based on the actual current state of the system.

Summary of the section

In the literature, one can find many NMPC implementations using a special case of the more general framework of using an integration method to approximate (24) resulting in a discrete time model which can be linearised by propagating the corresponding sensitivities. In [34], the nonlinear system is for example discretised and linearised using a single step forward Euler integrator. In [18], a similar approach is used but the linearisation is done at the current initial state and it is kept constant throughout the prediction horizon in order to reduce computations.

It is important to note that:

- when the system is in steady state, the matrix exponential approach provides an exact linearisation of the nonlinear system dynamics. In all other cases, this linearisation however becomes inexact;
- numerical integration methods can provide an arbitrarily accurate approximation of the nonlinear discrete dynamics of the system for any given sampling time;
- the technique of Internal Numerical Differentiation in combination with AD allows one to efficiently propagate the sensitivities of any integrator to obtain the exact linearisation of the system's approximated discrete time dynamics;
- efficient numerical methods are available, which make it possible to simulate the system dynamics and sensitivities in extremely short times.

6. Reliable Implementation of RTI for Continuous Time Systems

Because the RTI scheme only takes one single full Newton step per sampling instant, this scheme is expected to work better for systems which are mildly nonlinear, while more nonlinear systems could be harder to stabilise. This remark is true for discrete-time systems, however for continuous-time systems a careful implementation of the algorithm makes it possible to control also highly nonlinear systems. Important tuning parameters are: (a) the sampling time, (b) the horizon length, (c) the integrator accuracy, (d) the use of a shifting strategy (e) passing the reference in a smart way, and (f) the cost tuning matrices.

The choice of the cost tuning matrices is usually done by trial and error, using knowledge of the system to be controlled. If the problem has a clear economic criterion, a cost design strategy has been proposed in [49, 50]. If the problem is instead of tracking nature, the cost can be chosen so as to not only stabilise the system, but also help convergence of the algorithm. Rough guidelines include weighting every state and control and avoiding large differences between the weights associated with each state or control.

In the following, we illustrate points (a)-(e) using as an example a pendulum mounted on top of a cart. The derivation of the model, as well as a tutorial on integrators for fast NMPC is given in [41]. The cart can only move on the (horizontal) x-axis and its position is given by w_0. The angle of the pendulum is denoted by θ, using the convention that $\theta = 0\,\mathrm{rad}$ corresponds to the pendulum hanging down in the negative (vertical) y direction. The system dynamics are given by the explicit ODE

$$\ddot{w}_0 = \frac{ml\sin(\theta)\dot{\theta}^2 + mg\cos(\theta)\sin(\theta) + u}{M + m - m(\cos(\theta))^2}, \quad \ddot{\theta} = -\frac{ml\cos(\theta)\sin(\theta)\dot{\theta}^2 + u\cos(\theta) + (M+m)g\sin(\theta)}{l(M + m - m(\cos(\theta))^2)},$$

where $M = 1\ \mathrm{kg}$, $m = 0.1\ \mathrm{kg}$, $l = 0.5\ \mathrm{m}$, $g = 9.81\ \mathrm{m/s^2}$ are respectively the mass of the cart, the mass

attached at the end of the massless pendulum rod, the rod length and the gravitational acceleration. The NMPC controller (9) has been set up using weighting matrices $W_k = \mathrm{diag}(\begin{bmatrix} 10 & 10 & 0.1 & 0.1 & 0.01 \end{bmatrix})$ and the terminal cost $x_N^\top \mathrm{diag}(\begin{bmatrix} 10 & 10 & 0.1 & 0.1 \end{bmatrix})x_N$. No path constraints have been introduced for simplicity. All other tuning parameters are specified separately for each simulation.

For the following simulations, we consider a swing-up of the pendulum with a step in the reference for the angle θ from 0 to π occuring at $t = 2$ s. We used a prediction horizon $T_\mathrm{p} = 2$ s, a sampling time $T_\mathrm{s} = 0.1$ s, an explicit Runge Kutta integrator of order 4 (RK4) with a fixed stepsize $= 0.025$ s (see Algorithm 5) and we made use of the shifting strategy (19)-(20). Using the control provided by NMPC, the closed-loop trajectories have been simulated using the integrators with error control available in Matlab. In the following, we will study the effect of varying each one of the given parameters singularly, while keeping all others fixed. In all figures, we will plot the closed-loop solutions obtained using RTI and converged NMPC using thick dashed and thin continuous lines respectively, unless differently specified.

Sampling Time. When implementing NMPC for continuous-time systems, one can reduce the nonlinearities in the problem by choosing a sampling time which is short enough. For different sampling times, the closed-loop solutions are displayed in Figure 7. It can be seen that, as the sampling time gets larger, the RTI delivers closed-loop solutions which can be quite different from the ones obtained using converged NMPC. We also remark that, when this effect starts to become noticeable, the control performance of both RTI and converged NMPC deteriorates significantly. It can therefore be noted that, on the proposed example, RTI yields a better closed-loop response than fully converged NMPC when a shorter sampling time is chosen for the former. We remark that, in order to make the comparison fair, we set the terminal cost to 0, such that the cost functions approximate the same continuous-time cost functional.

Prediction Horizon. We illustrate now how the prediction horizon affects the controller performance by using a prediction horizon $T_\mathrm{p} = 0.5,\ 0.8,\ 1,\ 2,\ 4$ s. In order to better visualise the performance of the different NMPC controllers, we introduce the step in the reference at $t = 4$ s. The closed-loop solutions are displayed in Figure 8. It can be seen that, when the prediction horizon becomes too short, the controller performance visibly degradates and the RTI solutions start to diverge from the converged solutions.

Integrator Accuracy. Because the integrator accuracy determines the accuracy of the discrete-time model used by NMPC, one must choose an integrator which delivers predictions that are accurate enough to predict the evolution of the system in time. The closed-loop solutions are displayed in Figure 9 using a sampling time $T_\mathrm{s} = 0.15$ s and an explicit Euler integrator with a number of integration steps $N_\mathrm{S} = 20,\ 10,\ 5,\ 2,\ 1$ over one shooting interval. It can be seen that, as the accuracy becomes lower, the RTI solutions start to diverge from the converged solutions and the control performance worsens.

It is important to note that, using 2 steps of RK4 yields a closed-loop behaviour which is very close to that obtained by using 30 steps of explicit Euler, but its preparation phase takes only about 26% of the time needed when using 30 steps of explicit Euler. As described in Section 5, RK4 consists of 4 stages, while explicit Euler consists of 1 stage: this is reflected in the computational times for the preparation phase which become similar when using 2 steps of RK4 or 8 steps of explicit Euler. The closed-loop trajectories displayed in Figure 9 also highlight another important fact: the difference between using 10 or 20 steps of explicit Euler is marginal. Indeed, high integration accuracies are not always needed and the closed-loop trajectories become insensitive to integrator accuracy when it gets high enough. Moreover, when deploying NMPC on real systems, unmodeled dynamics and external perturbations dominate over the integration error, so that it can be preferrable to favour faster sampling times and rather use a reduced integration accuracy in order to meet tight timing constraints. For the considered scenario, 10 steps of explicit Euler or the cheaper choice of 1 step of RK4 could already yield an accurate enough integrator. We remark however that the studied example has been chosen for illustration and all the proposed schemes have an overall computational time well below 1 ms on a $2,3$ GHz Intel Core i7 with 16 Gb of RAM so that the computational effort is not a concern.

Shifting. As already highlighted in Section 3.2, constructing an initial guess by shifting the trajectory obtained at the previous sampling time can be very beneficial when implementing NMPC using the RTI

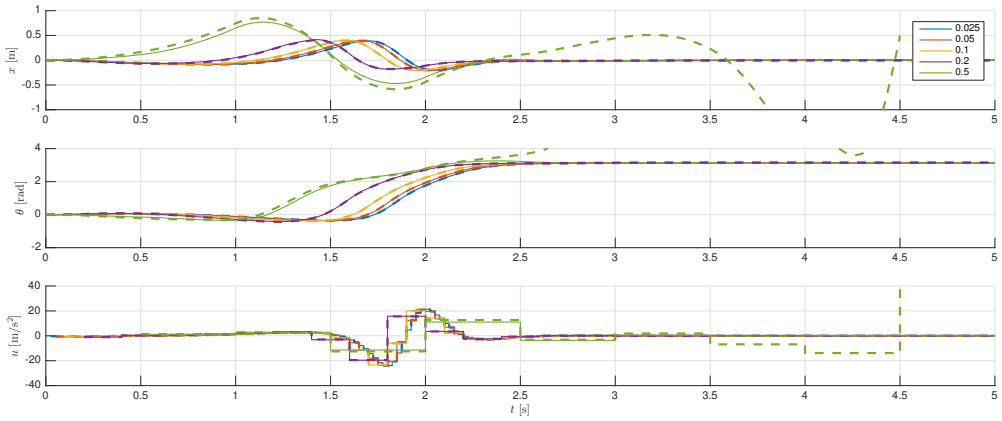


Figure 7. Closed-loop simulations of a pendulum swing-up using several sampling times $T_{\mathrm{s}} = 0.025,\ 0.05,\ 0.1,\ 0.2,\ 0.5$ s. The trajectories obtained using RTI and converged NMPC are plotted in thick dashed and thin continuous lines respectively.

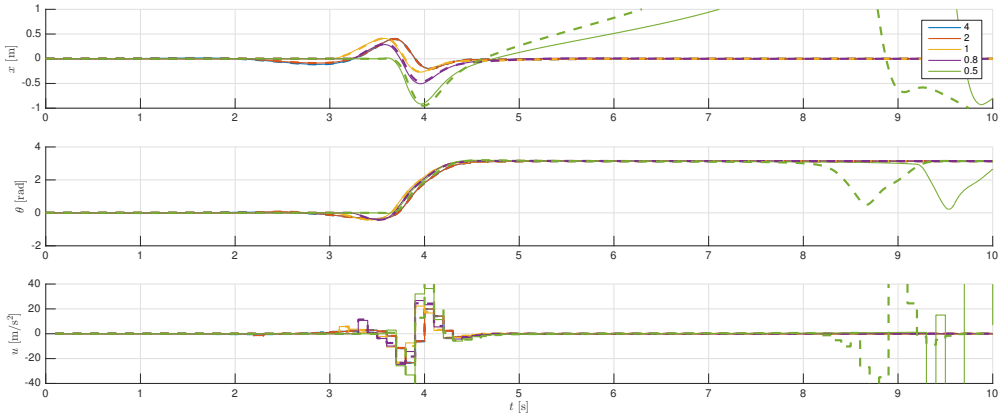


Figure 8. Closed-loop solutions of a pendulum swing-up using several prediction horizons $T_{\mathrm{p}} = 0.5,\ 0.8,\ 1,\ 2,\ 4$ s and a step in the reference occuring at $t = 4$ s. The trajectories obtained using RTI and converged NMPC are plotted in thick dashed and thin continuous lines respectively.

scheme. This fact is also highlighted in Figure 10, where it can be seen that, using a sampling time $T_{\mathrm{s}} = 0.05$ s the closed-loop trajectories obtained using RTI without shifting yield poor control performance. Instead, by using the shifting strategy, the RTI solution matches the converged NMPC solution very closely. We remark that, by using a sampling time $T_{\mathrm{s}} = 0.1$ s, the RTI solution without shifting becomes unstable, while the RTI solution with shifting still matches the converged NMPC solution very closely.

Choice of the Reference. The choice of the reference trajectory is also a crucial element for ensuring a reliable implementation of the RTI scheme. In the ideal case, one would pre-compute a feasible trajectory so that the NMPC controller only needs to reject perturbations. Sometimes this is not possible and the NMPC controller needs to both reject perturbations and plan the trajectory that the system must follow. The simulations we performed in this section fall into the second category: we used a step in the reference that was passed to the RTI-NMPC controller. Because it is a large step, we decided to introduce it after 2 s instead of having it at the beginning of the horizon. This is beneficial because it progressively enters the NMPC prediction horizon and leaves the time to the RTI scheme to converge to the solution before the system starts to move. Indeed, reference changes which occur far from the beginning of the horizon do not affect the initial part of the predicted trajectory. This fact is illustrated in Figure 11, where we display the closed-loop solutions obtained by introducing the step in the reference at time $t = 0,\ 1,\ 2,$ s. It can be seen that, when the step enters at the end of the horizon, the RTI and converged solutions are indistinguishable. When the step is provided at the beginning of the horizon instead, the RTI solution is very far from the

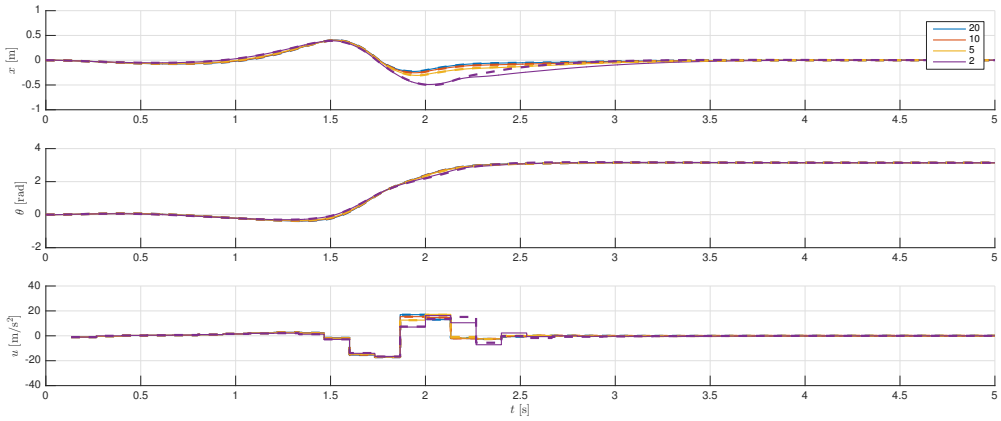


Figure 9. Closed-loop solutions of a pendulum swing-up using different integrator accuracies with a sampling time $T_{\mathrm{S}} = 0.15$ s and an explicit Euler integrator with a number of integration steps $N_{\mathrm{S}} = 20, 10, 5, 2, 1$ over one shooting interval. The trajectories obtained using RTI and converged NMPC are plotted in thick dashed and thin continuous lines respectively.

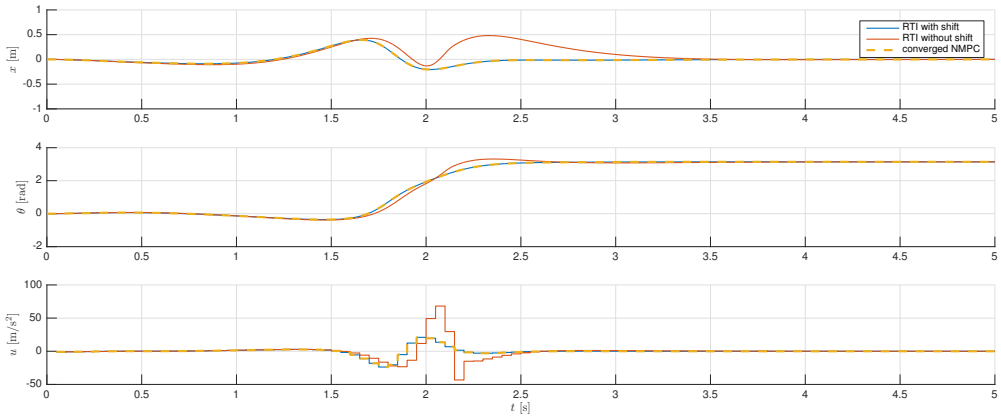


Figure 10. Closed-loop solutions of a pendulum swing-up using RTI with and without shifting as well as converged NMPC. The solutions obtained using RTI with and without shifting are displayed in continuous blue and red line respectively, while the converged NMPC solution is displayed in dashed yellow line.

converged one and its performance is very poor. When the step occurs at $t = 1$ s instead, the RTI solution differs from the converged NMPC solution, but its performance is still good.

Summary of the section

The RTI scheme is able to closely track the converged NMPC solution, provided that the algorithm is implemented carefully. In particular, the sampling time should be chosen small enough, the prediction horizon long enough, the integrator should be accurate, the shifting strategy should be used and the reference should be chosen adequately. Moreover, tuning the cost appropriately is also important for guaranteeing good performance. Examples of successful implementations of the RTI scheme for nontrivial nonlinear systems can be found in [2, 9, 23, 24, 26, 47, 51].

7. Conclusions

In this paper, we have clarified the similarities and differences between linear MPC and the RTI-based NMPC approach. On the one hand, RTI can be seen as a straightforward extension of linear MPC, which makes use of an integrator to re-linearise the system dynamics and path constraints at the current prediction rather than on the reference. On the other hand, RTI is an SQP-type solver for NMPC, which, under mild

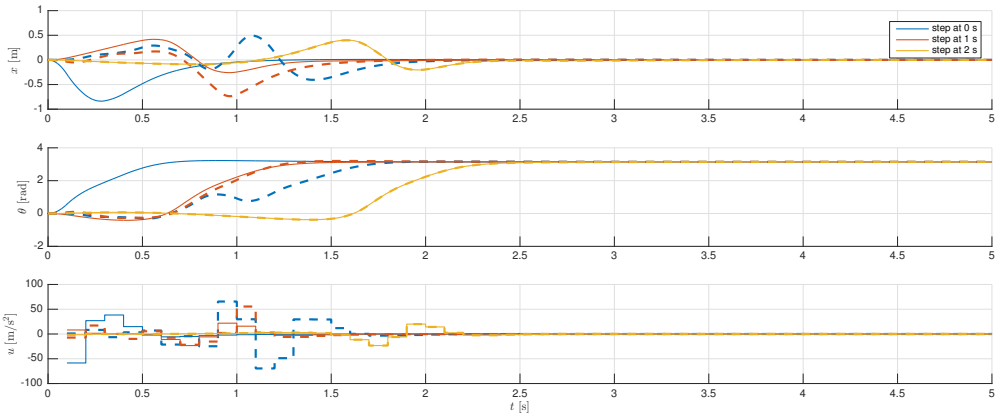


Figure 11. Closed-loop solutions of a pendulum swing-up using a step in the reference occuring at times $t = 0,\ 1,\ 2,$ s. The trajectories obtained using RTI and converged NMPC are plotted in thick dashed and thin continuous lines respectively.

assumptions, tracks the NMPC solution manifold. Therefore, in many cases, the RTI strategy can be deployed to implement a genuine NMPC scheme with a limited additional computational burden and coding effort compared to linear MPC.

References

[1] J. Albersmeyer and H.G. Bock. Sensitivity Generation in an Adaptive BDF-Method. In *Modeling, Simulation and Optimization of Complex Processes: Proceedings of the International Conference on High Performance Scientific Computing, 2006, Hanoi, Vietnam*, pages 15–24. Springer, 2008.

[2] T. Albin, D. Ritter, D. Abel, R. Quirynen, and M. Diehl. Nonlinear MPC for a two-stage turbocharged gasoline engine airpath. In *54th IEEE Conference on Decision and Control*, 2015.

[3] F. Allgöwer, Z. Nagy, and R. Findeisen. Nonlinear model predictive control: From theory to applications. In *Proc. Int. Symp. Design, Operation and Control of Chemical Plants (PSE)*, 2002.

[4] H.G. Bock. Recent advances in parameter identification techniques for ODE. In P. Deuflhard and E. Hairer, editors, *Numerical Treatment of Inverse Problems in Differential and Integral Equations*, pages 95–121. Birkhäuser, Boston, 1983.

[5] H.G. Bock, M. Diehl, E.A. Kostina, and J.P. Schlöder. Constrained Optimal Feedback Control of Systems Governed by Large Differential Algebraic Equations. In L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, editors, *Real-Time and Online PDE-Constrained Optimization*, pages 3–22. SIAM, 2007.

[6] H.G. Bock, M. Diehl, P. Kühl, E. Kostina, J.P. Schlöder, and L. Wirsching. Numerical Methods for Efficient and Fast Nonlinear Model Predictive Control. In *Proceedings of "Int. Workshop on assessment and future directions of Nonlinear Model Predictive Control"*. Springer, 2005.

[7] I. Bonis, W. Xie, and C. Theodoropoulos. A linear model predictive control algorithm for nonlinear large-scale distributed parameter systems. *AIChE J.*, 58:801–811, 2012.

[8] M. Caracotsios and W.E. Stewart. Sensitivity analysis of initial value problems with mixed ODEs and algebraic equations. *Computers and Chemical Engineering.*, 9:359–365, 1985.

[9] F. Debrouwere, M. Vukov, R. Quirynen, M. Diehl, and J. Swevers. Experimental Validation of Combined Nonlinear Optimal Control and Estimation of an Overhead Crane. In *Proceedings of the 19th World Congress of the International Federation of Automatic Control*, 2014.

[10] S. Di Cairano, D. Yanakiev, A. Bemporad, I. V. Kolmanovsky, and D. Hrovat. Model Predictive Idle Speed Control: Design, Analysis, and Experimental Evaluation. *IEEE Transactions on Control Systems and Technology*, 20:84–97, 2012.

[11] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*. PhD thesis, Universität Heidelberg, 2001.

[12] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*, volume 920 of *Fortschr.-Ber. VDI Reihe 8, Meß-, Steuerungs- und Regelungstechnik*. VDI Verlag, Düsseldorf, 2002.

[13] M. Diehl, H.G. Bock, and J.P. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5):1714–1736, 2005.

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

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

[16] M. Diehl, R. Findeisen, F. Allgöwer, H.G. Bock, and J.P. Schlöder. Nominal Stability of the Real-Time Iteration Scheme for Nonlinear Model Predictive Control. *IEE Proc.-Control Theory Appl.*, 152(3):296–308, 2005.

[17] A. Domahidi, A. Zgraggen, M.N. Zeilinger, M. Morari, and C.N. Jones. Efficient Interior Point Methods for Multistage Problems Arising in Receding Horizon Control. In *IEEE Conference on Decision and Control (CDC)*, pages 668 – 674, Maui, HI, USA, December 2012.

[18] P. Falcone, M. Tufo, F. Borrelli, J. Asgari, and H. E. Tseng. A Linear Time Varying Model Predictive Control Approach to the Integrated Vehicle Dynamics Control Problem in Autonomous Systems. In *Proceedings of the 46th IEEE Conference on Decision and Control*, 2007.

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

[20] J. V. Frasch, S. Sager, and M. Diehl. A Parallel Quadratic Programming Method for Dynamic Optimization Problems. *Mathematical Programming Computations*, 7(3):289–329, 2015.

[21] G. Frison, H.B. Sorensen, B. Dammann, and J.B. Jorgensen. High-performance small-scale solvers for linear model predictive control. In *Proc. 2014 European Control Conference (ECC)*, pages 128–133, June 2014.

[22] A. Griewank and A. Walther. *Evaluating Derivatives*. SIAM, 2 edition, 2008.

[23] S. Gros, R. Quirynen, and M. Diehl. Aircraft Control Based on Fast Nonlinear MPC & Multiple-shooting. In *Conference on Decision and Control*, 2012.

[24] S. Gros, R. Quirynen, and M. Diehl. An Improved Real-time NMPC Scheme for Wind Turbine Control using Spline-Interpolated Aerodynamic Coefficients. In *Conference on Decision and Control*, 2014.

[25] S. Gros, M. Vukov, and M. Diehl. A Real-time MHE and NMPC Scheme for the Control of Multi-Mega Watts Wind Turbines. In *Conference on Decision and Control*, 2013.

[26] S. Gros, M. Zanon, and M. Diehl. Control of Airborne Wind Energy Systems Based on Nonlinear Model Predictive Control & Moving Horizon Estimation. In *European Control Conference*, pages 1017–1022, 2013.

[27] S. Gros, M. Zanon, M. Vukov, and M. Diehl. Nonlinear MPC and MHE for Mechanical Multi-Body Systems with Application to Fast Tethered Airplanes. In *Proceedings of the 4th IFAC Nonlinear Model Predictive Control Conference, Noordwijkerhout, The Netherlands*, pages 86–93, 2012.

[28] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I*. Springer Series in Computational Mathematics. Springer, Berlin, 2nd edition, 1993.

[29] E. Hairer, S.P. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations II – Stiff and Differential-Algebraic Problems*. Springer Series in Computational Mathematics. Springer, Berlin, 2nd edition, 1996.

[30] A.C. Hindmarsh, P.N. Brown, K.E. Grant, S.L. Lee, R. Serban, D.E. Shumaker, and C.S. Woodward. SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers. *ACM Transactions on Mathematical Software*, 31:363–396, 2005.

[31] Marlis Hochbruck and Alexander Ostermann. Exponential integrators. *Acta Numerica*, 19:209–286, 5 2010.

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

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

[34] T. Keviczky and G. J. Balas. Software-Enabled Receding Horizon Control for Autonomous Unmanned Aerial Vehicle Guidance. *Jounal of Guidance, Control, and Dynamics*, 29:680–694, 2006.

[35] W.C. Li and L.T. Biegler. Multistep, Newton-Type Control Strategies for Constrained Nonlinear Processes. *Chem. Eng. Res. Des.*, 67:562–577, 1989.

[36] David Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12):2967 – 2986, 2014.

[37] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2 edition, 2006.

[38] T. Ohtsuka. A Continuation/GMRES Method for Fast Computation of Nonlinear Receding Horizon Control. *Automatica*, 40(4):563–574, 2004.

[39] P. Patrinos and A. Bemporad. An accelerated dual gradient-projection algorithm for embedded linear model predictive control. *Automatic Control, IEEE Transactions on*, 59(1):18–33, Jan 2014.

[40] R. Quirynen, B. Houska, M. Vallerio, D. Telen, F. Logist, J. Van Impe, and M. Diehl. Symmetric Algorithmic Differentiation Based Exact Hessian SQP Method and Software for Economic MPC. In *Conference on Decision and Control*, pages 2752–2757, 2014.

[41] R. Quirynen, M. Vukov, M. Zanon, and M. Diehl. Autogenerating Microsecond Solvers for Nonlinear MPC: a Tutorial Using ACADO Integrators. *Optimal Control Applications and Methods*, 36:685–704, 2014.

[42] J. B. Rawlings, D. Angeli, and C. N. Bates. Fundamentals of economic model predictive control. In *51st IEEE Conference on Decision and Control*, 2012.

[43] J.B. Rawlings and D.Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill, 2009.

[44] M. Vukov, A. Domahidi, H. J. Ferreau, M. Morari, and M. Diehl. Auto-generated Algorithms for Nonlinear Model Predictive Control on Long and on Short Horizons. In *Proceedings of the 52nd Conference on Decision and Control (CDC)*, 2013.

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

[46] Andrea Walther. Automatic differentiation of explicit Runge-Kutta methods for optimal control. *Computational Optimization and Applications*, 36(1):83–108, 2006.

[47] M. Zanon, J. V. Frasch, M. Vukov, S. Sager, and M. Diehl. Model Predictive Control of Autonomous Vehicles. In *Proceedings of the Workshop on Optimization and Optimal Control of Automotive Systems*, pages 41–57. 2014.

[48] M. Zanon, S. Gros, and M. Diehl. Model Predictive Control of Rigid-Airfoil Airborne Wind Energy Systems. In U. Ahrens, M. Diehl, and R. Schmehl, editors, *Airborne Wind Energy*. Springer, 2013.

[49] M. Zanon, S. Gros, and M. Diehl. Indefinite Linear MPC and Approximated Economic MPC for Nonlinear Systems. *Journal of Process Control*, 24:1273–1281, 2014.

[50] M. Zanon, S. Gros, and M. Diehl. A Tracking MPC Formulation that is Locally Equivalent to Economic MPC. *Journal of Process Control*, 2016. (accepted).

[51] M. Zanon, G. Horn, S. Gros, and M. Diehl. Control of Dual-Airfoil Airborne Wind Energy Systems Based on Nonlinear MPC and MHE. In *European Control Conference*, pages 1801–1806, 2014.

[52] V. M. Zavala and L.T. Biegler. The Advanced Step NMPC Controller: Optimality, Stability and Robustness. *Automatica*, 45:86–93, 2009.