Real-Time Iteration as a Connector Between Linear and Nonlinear MPC

Ali Vaziri, *Ph.D. Student at ISSL Laboratory, MPC Course Project*
Original paper is written by . Sebastien Gros, Mario Zanon, Rien Quirynen, Alberto Bemporad, and Moritz Dieh.

***Abstract*—This paper aims to clarify the similarities and differences between linear Model Predictive Control (MPC) and Nonlinear MPC (NMPC) based on the Real-Time Iteration (RTI) scheme. However, We tried to convey our understanding of the RTI algorithm for NMPC and the original paper played the role of tutorial for us to do so. We also has made a comparison between the RTI and SQP method in theory, not in the results. While linear MPC has been widely used due to recent advancements in algorithms for solving online Quadratic Programs (QPs), NMPC is effective in tackling problems with nonlinear constraints and dynamics. However, the computational time required for NMPC is longer than that of linear MPC. This paper proposes the RTI-based NMPC approach, which has been successfully tested and requires computational times comparable to linear MPC. In the original paper, particular, they focused on analysis on NMPC based on the Real-Time Iteration (RTI) scheme and the goal of the paper is to promote the understanding of RTI-based NMPC within the linear MPC community.**

***Index Terms*—linear MPC, real-time NMPC**

I. Introduction

LINEAR Model Predictive Control is an advanced control technique able to deal with multiple input multiple output constrained linear systems [1]. There have been pros and cons to using linear MPC. On the one hand, linear MPC solves the linear costs and constraints with an acceptable computational efficiency using methods such as Simplex, LQR, LQG, and QP. In some applications when we do not have constraints, we can also apply dynamic programming. On the other hand, linear MPC cannot deal with nonlinear costs or constraints. Nonlinear Model Predictive Control (NMPC) is an effective way to tackle nonlinear costs and dynamics. Some approaches which are able to properly solve NMPC are SQP, Interior-Point, Numerical methods, and Real Time Iteration method (RTI). In this report, we try to count the problem of conventional methods for solving NMPC and introduce RTI as a way to overcome those problems.
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]:

Ali Vaziri is with the Mechanical Engineering Department, University of Kansas, Lawrence, 66045, USA, (e-mail: alivaziri@ku.com).
Original paper can be found in [1]

1) The linearization 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 [2].
The paper is organized as follows. Sections II and III formulate, respectively, the linear and nonlinear MPC problems. Section IV describes in detail the RTI-based NMPC approach, establishing its connection to linear MPC. Section V describes numerical methods to obtain the discrete-time nonlinear prediction model from a continuous-time description of the system. Case study and simulation results are shown in Section VI and Conclusions are drawn in Section VII.

II. Linear Model Predictive Control (MPC)

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

$$QP_{MPC}(\mathbf{x}_i, \mathbf{x}_i^{ref}, \mathbf{u}_i^{ref}) =$$

$$\arg\min_{\mathbf{x_i},\mathbf{u_i}} \sum_{k=0}^{N-1} \|\mathbf{x}_{i,k} - \mathbf{x}_{i,k}^{ref}\|_{\mathbf{Q}}^2 + \|\mathbf{u}_{i,k} - \mathbf{u}_{i,k}^{ref}\|_{\mathbf{R}}^2, \tag{1a}$$

$$\text{s.t.} \quad \mathbf{x}_{i,k+1} = \mathbf{A}_{i,k}\mathbf{x}_{i,k} + \mathbf{B}_{i,k}\mathbf{u}_{i,k} \tag{1b}$$

$$h_{i,k}(\mathbf{x}_{i,k}, \mathbf{u}_{i,k}) \leq 0 \tag{1c}$$

where $\mathbf{x}_{i,k}^{ref}$, $\mathbf{u}_{i,k}^{ref}$ are the reference trajectories provided at time i, $\mathbf{x}_i$, $\mathbf{u}_i$ are optimization variables, constraint(1) is the cost function, constraint(1b) is the system dynamics, and constraint (1c) enforces path constraints which include e.g. actuator limitations, obstacle avoidance, etc. We use here and in the following the notation $\mathbf{x}_{i,k}^{ref}$ with $k = 0, \ldots, N-1$, to denote the k^{th} element of the reference trajectory $\mathbf{x}_{i,k}^{ref}$ provided at time i. We denote vectors and matrices in boldfaces. The same applies to $\mathbf{u}_{i,k}^{ref}$ with $k = 0, \ldots, N-1$. At every discrete-time instant i, the input to be applied to the system is given by $QP_{MPC}(\mathbf{x}_i, \mathbf{x}_i^{ref}, \mathbf{u}_i^{ref})$ resulting in $\mathbf{x}_i$, $\mathbf{u}_i$ from which we take $\mathbf{u}_{i,0}$, $\mathbf{x}_{i,0}$. Matrices $\mathbf{Q}$ and $\mathbf{R}$ are symmetric positive semidefinite. For the sake of brevity and without loss of generality, we omit the terminal cost in this section. Formulation (1) is a structured Quadratic Program

(QP). In the case of matrices $\mathbf{A}$, $\mathbf{B}$ are constant, we refer to linear MPC based on a Linear Time-Invariant (LTI) model, otherwise to Linear Time-Varying (LTV) MPC.

III. Nonlinear MPC (NMPC)

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

$$NLP(\mathbf{x}_i, \mathbf{x}_i^{ref}, \mathbf{u}_i^{ref}) =$$

$$\arg\min_{\mathbf{x}_i, \mathbf{u}_i} \sum_{k=0}^{N-1} \|\mathbf{x}_{i,k} - \mathbf{x}_{i,k}^{ref}\|_{\mathbf{Q}}^2 + \|\mathbf{u}_{i,k} - \mathbf{u}_{i,k}^{ref}\|_{\mathbf{R}}^2, \tag{2a}$$

$$\text{s.t.} \quad \mathbf{x}_{i,k+1} = f(\mathbf{x}_{i,k}, \mathbf{u}_{i,k}) \tag{2b}$$

$$h_{i,k}(\mathbf{x}_{i,k}, \mathbf{u}_{i,k}) \leq 0 \tag{2c}$$

where (2b) is the nonlinear dynamics. 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[3] and additional care might be needed at the algorithmic level [4]. Similarly to Section II, for the sake of brevity, we omit the terminal cost in this Section. At every time instant i, problem (2) provides the NMPC control solutions for system (2b), given by$NLP(\mathbf{x}_i, \mathbf{x}_i^{ref}, \mathbf{u}_i^{ref})$ from which we can pick $\mathbf{u}_{i,0}$, $\mathbf{x}_{i,0}$. Problem (2) can be solved using different methods. We briefly review the SQP method which in each iteration we approximate linear system dynamics and linear constraints, and apply QP (1) until convergence is obtained.

A. Sequential Quadratic Programming (SQP) for NMPC:

In the SQP, we approximate problem (2) by QPs starting from the initial guess. At each time step i, we linearize the objective function and constraints in the available $\mathbf{x}_{i,k}^{guess}$, $\mathbf{u}_{i,k}^{guess}$, instead of linearizing it in $\mathbf{x}_{i,k}^{ref}$, $\mathbf{u}_{i,k}^{ref}$, and solve the following problem:

$$QP_{NMPC}(\mathbf{x}_i, \mathbf{x}_{i,k}^{guess}, \mathbf{u}_{i,k}^{guess}) =$$

$$\arg\min_{\mathbf{x}_i, \mathbf{u}_i} \sum_{k=0}^{N-1} \|\mathbf{x}_{i,k} - \mathbf{x}_{i,k}^{guess}\|_{\mathbf{Q}}^2 + \|\mathbf{u}_{i,k} - \mathbf{u}_{i,k}^{guess}\|_{\mathbf{R}}^2, \tag{3a}$$

$$\text{s.t.} \quad \mathbf{x}_{i,k+1} = \mathbf{A}_{i,k}\mathbf{x}_{i,k} + \mathbf{B}_{i,k}\mathbf{u}_{i,k} \tag{3b}$$

$$h_{i,k}(\mathbf{x}_{i,k}, \mathbf{u}_{i,k}) \leq 0 \tag{3c}$$

The NMPC solution is then obtained from the following:

$$(\mathbf{x}_i, \mathbf{u}_i) = SQP(\mathbf{x}_i, \mathbf{x}_i^{guess}, \mathbf{u}_i^{guess}, \mathbf{x}_i^{ref}, \mathbf{u}_i^{ref}) \tag{4}$$

where by (4) we denote the solution of $NLP(\mathbf{x}_i, \mathbf{x}_i^{ref}, \mathbf{u}_i^{ref})$ achieved by starting from the initial guess $(\mathbf{x}_i^{guess}, \mathbf{u}_i^{guess})$.

B. Warm-started SQP for NMPC:

SQP needs the initial guess as input. Selecting a proper initial guess is important for getting a reliable convergence after iterations are operated. Not only is it important to avoid infeasible solutions but also it allows for fast convergence. 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 builds $\mathbf{x}_i^{guess}$, $\mathbf{u}_i^{guess}$ for time i by using the output of the SQP in the previous time step $i-1$. The shifting procedure then acts as follows:

$$\mathbf{x}_{i,k}^{guess} = \mathbf{x}_{i-1,k+1} \qquad k = 0, \ldots, N-1 \tag{5a}$$

$$\mathbf{u}_{i,k}^{guess} = \mathbf{u}_{i-1,k+1} \qquad k = 0, \ldots, N-2 \tag{5b}$$

$$\mathbf{x}_{i,N}^{guess} = f(\mathbf{x}_{i,N-1}^{guess}, \mathbf{u}_{i,N-1}^{guess}) \tag{5c}$$

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 [1]. 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 for the Real-Time Iteration approach, described in Section IV. Before presenting it in detail, we first review the $real - time\ dilemma$.

C. The real-time dilemma:

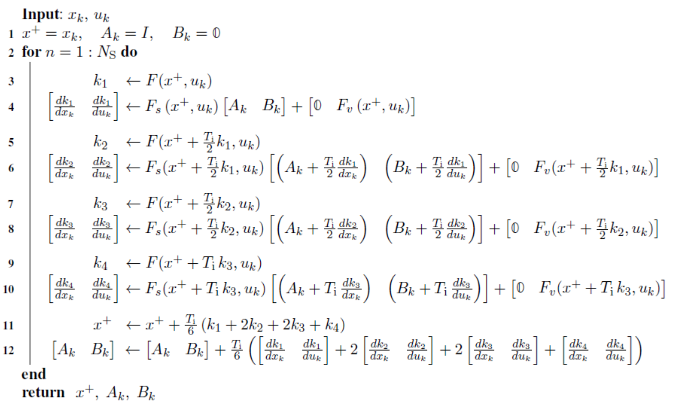
After obtaining a new estimate of the system's state, the SQP procedure can begin. However, there is a problem with real-time implementation. While the SQP iterations are being performed, the physical system continues to evolve, making the information used to compute the state estimate outdated. One way to address this issue is to use a prediction of the system state at the time the SQP algorithm will be completed, rather than directly using the current state estimate. However, even with this prediction approach, there is still a delay between gathering measurements and delivering the corresponding control action since updating the control policy requires the completion of the QP until convergence. Therefore, the SQP algorithm is still based on outdated system information. The main concept behind the RTI procedure, which was first introduced in[5] and described in detail in Section IV, is to continuously integrate the most current information about the system's evolution into the iterations that compute the NMPC solutions. However, this approach creates a real-time dilemma where there is a choice between using an exact solution that is based on outdated information or using an approximate solution that is based on the most up-to-date information available.

IV. The Real-Time Iteration (RTI)

This section serves as a reminder of the RTI approach that was initially presented in [5]. It is worth noting that numerous real-time NMPC approaches have been suggested in the literature such as $Newton - Type\ Controller$ [6], $Continuation/GMRES\ Method$ [7]. In this paper, we

Algorithm 1 RTI for NMPC at discrete time i

```
   Preparation phase performed over the time interval $[t_{i-1}, t_i)$
1: Inputs: previous NMPC solution $(\mathbf{x}_{i-1}^{guess}; \mathbf{u}_{i-1}^{guess})$, and
           reference $(\mathbf{x}_{i-1}^{ref}; \mathbf{u}_{i-1}^{ref})$
2: Apply shifting method to build $(\mathbf{x}_{i}^{guess}; \mathbf{u}_{i}^{guess})$
3: Evaluate $\mathbf{A}_{i,k}, \mathbf{B}_{i,k}$ and other parameters for constraints at
   $(\mathbf{x}_{i}^{guess}; \mathbf{u}_{i}^{guess})$
4: prepare all possible computations for QP
5: return QP (wait until state estimate $x_i$ becomes available)
   Feedback phase performed at time $t_i$ upon availability of
   state estimate $x_i$
6: Inputs: Prepared QP and state estimate $x_i$
7: Solve QP
8: Output: $(\mathbf{x}_i; \mathbf{u}_i)$
```



```
   Input: $x_k, u_k$
1  $x^+ = x_k, \quad A_k = I, \quad B_k = \mathbb{0}$
2  for $n = 1 : N_S$ do
3      $k_1 \leftarrow F(x^+, u_k)$
4      $\begin{bmatrix} \frac{dk_1}{dx_k} & \frac{dk_1}{du_k} \end{bmatrix} \leftarrow F_s(x^+, u_k) \begin{bmatrix} A_k & B_k \end{bmatrix} + \begin{bmatrix} \mathbb{0} & F_v(x^+, u_k) \end{bmatrix}$
5      $k_2 \leftarrow F(x^+ + \frac{T_i}{2} k_1, u_k)$
6      $\begin{bmatrix} \frac{dk_2}{dx_k} & \frac{dk_2}{du_k} \end{bmatrix} \leftarrow F_s(x^+ + \frac{T_i}{2} k_1, u_k) \left[ \left( A_k + \frac{T_i}{2} \frac{dk_1}{dx_k} \right) \quad \left( B_k + \frac{T_i}{2} \frac{dk_1}{du_k} \right) \right] + \begin{bmatrix} \mathbb{0} & F_v(x^+ + \frac{T_i}{2} k_1, u_k) \end{bmatrix}$
7      $k_3 \leftarrow F(x^+ + \frac{T_i}{2} k_2, u_k)$
8      $\begin{bmatrix} \frac{dk_3}{dx_k} & \frac{dk_3}{du_k} \end{bmatrix} \leftarrow F_s(x^+ + \frac{T_i}{2} k_2, u_k) \left[ \left( A_k + \frac{T_i}{2} \frac{dk_2}{dx_k} \right) \quad \left( B_k + \frac{T_i}{2} \frac{dk_2}{du_k} \right) \right] + \begin{bmatrix} \mathbb{0} & F_v(x^+ + \frac{T_i}{2} k_2, u_k) \end{bmatrix}$
9      $k_4 \leftarrow F(x^+ + T_i k_3, u_k)$
10     $\begin{bmatrix} \frac{dk_4}{dx_k} & \frac{dk_4}{du_k} \end{bmatrix} \leftarrow F_s(x^+ + T_i k_3, u_k) \left[ \left( A_k + T_i \frac{dk_3}{dx_k} \right) \quad \left( B_k + T_i \frac{dk_3}{du_k} \right) \right] + \begin{bmatrix} \mathbb{0} & F_v(x^+ + T_i k_3, u_k) \end{bmatrix}$
11     $x^+ \leftarrow x^+ + \frac{T_i}{6} (k_1 + 2k_2 + 2k_3 + k_4)$
12     $\begin{bmatrix} A_k & B_k \end{bmatrix} \leftarrow \begin{bmatrix} A_k & B_k \end{bmatrix} + \frac{T_i}{6} \left( \begin{bmatrix} \frac{dk_1}{dx_k} & \frac{dk_1}{du_k} \end{bmatrix} + 2 \begin{bmatrix} \frac{dk_2}{dx_k} & \frac{dk_2}{du_k} \end{bmatrix} + 2 \begin{bmatrix} \frac{dk_3}{dx_k} & \frac{dk_3}{du_k} \end{bmatrix} + \begin{bmatrix} \frac{dk_4}{dx_k} & \frac{dk_4}{du_k} \end{bmatrix} \right)$
   end
   return $x^+, A_k, B_k$
```

Fig. 1: Runge-Kutta 4 algorithm, is an integrator method using AD techniques and propagated forward through the integration algorithm using the chain rule [1].

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[1]. In RTI, when we are at time $i-1$, we use shifting procedure to construct $\mathbf{x}_i^{guess}$, $\mathbf{u}_i^{guess}$, and simultaneously building $\mathbf{A}_{i,k}, \mathbf{B}_{i,k}$ as well as constraints. When the state estimate becomes available, we apply the QP *once* (not until the convergence) which gives us the new guess for the next time step $\mathbf{x}_i^{guess}, \mathbf{u}_i^{guess}$. The algorithm performing RTI method is shown in algorithm 1. The RTI scheme (see Algorithm 1) thus proposes to split the operation between:

1) A $preparation\ phase$, performing the computations involved in the steps prior to obtaining the new state estimate.
2) A $feedback\ phase$, performing the computations involved in steps 6 and 4 7 upon obtaining the latest state estimate.

For a detailed overview on the RTI scheme, including a proof of nominal stability, we refer to [8–10]. Several important points should be noted regarding the RTI approach:

1) The delay introduced by feedback time can be addressed by including a corresponding prediction in the state estimate.
2) The total time spent in solving both the feedback phase and the preparation phase limits the overall sampling time $(t_i - t_i - 1)$ that can be achieved by the RTI scheme.
3) The time required for the feedback phase is equal to the time required to solve the linear MPC problem.
4) Part of the computations related to the QP solution can often be moved to the preparation phase.
5) Ideally, the feedback time should be a fraction of the overall sampling time.
6) The preparation phase can often fill out the time between the feedback phase and the next state estimate.

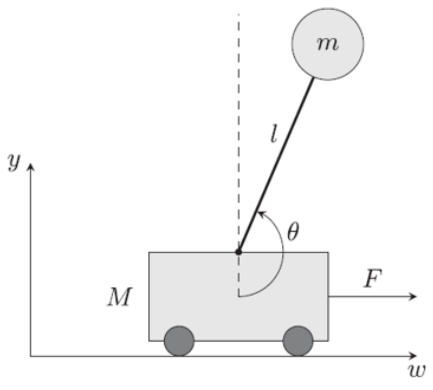
The only remaining part to be mentioned is when we are working with the continuous-time system. In this case, in order to obtain the sensitivities $(\mathbf{A}_{i,k}, \mathbf{B}_{i,k})$, we need to discretize and then linearize dynamics of the system. In the next section we will show how we can achieve this.

V. MPC and NMPC for continuous-time systems

Having a continuous-time system necessitates us to change the problem in a way, so we can apply our solvers. We can first do liniearization and then discretization, but it only gives us a good approximation when we are in the steady state. Therefore, it is better to first discretize and then linearize. A generally applicable approach is to numerically approximate the discrete dynamics, in order to obtain a linearization with a specific desired accuracy. In this article, we will limit our discussion to the category of one-step methods, which is a significant family of methods that includes Runge-Kutta (RK) methods. One-step methods are advantageous over multistep schemes because they do not necessitate any start-up procedure, which makes them particularly well-suited for short simulation times. In the context of numerical methods for solving differential equations, a start-up procedure is a set of initial computations or steps that must be performed before the actual solution process can begin. This is typically required for certain types of methods, such as multistep schemes, to establish an initial set of values for the variables involved in the computation. In contrast, one-step methods, like Runge-Kutta methods, do not require a start-up procedure as they are designed to calculate the solution directly from the initial values of the variables. In addition, we will consider the integration over one shooting interval using a fixed step size T_i resulting in $N_S = T_s/T_i$ 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 [1]. We applied numerical methods using the techniques of Algorithmic Differentiation (AD) [1] and propagated forward through the integration algorithm using the chain rule as detailed in Fig. 1, where we denote the identity matrix by I. in the Fig. 1 we show RK4 algorithm.

VI. RTI NMPC implementation on the inverted pendulum

The RTI scheme is designed to take only one full Newton step per sampling instant, making it more suitable for controlling systems that are mildly nonlinear. However, for highly nonlinear systems, it may be more difficult to achieve stabilization. While this is true for discrete-time systems,



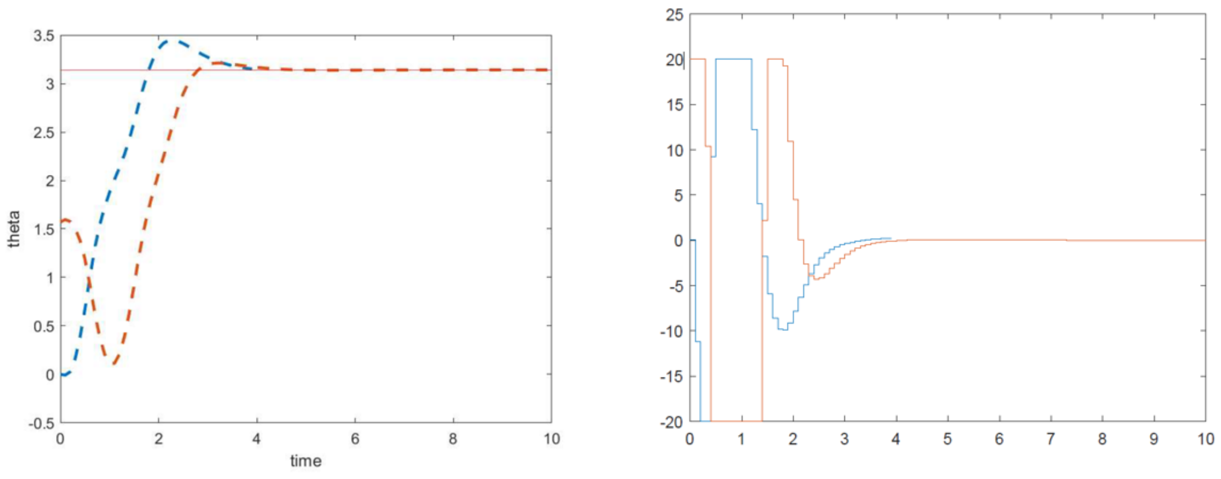
(a)

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

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

(b)

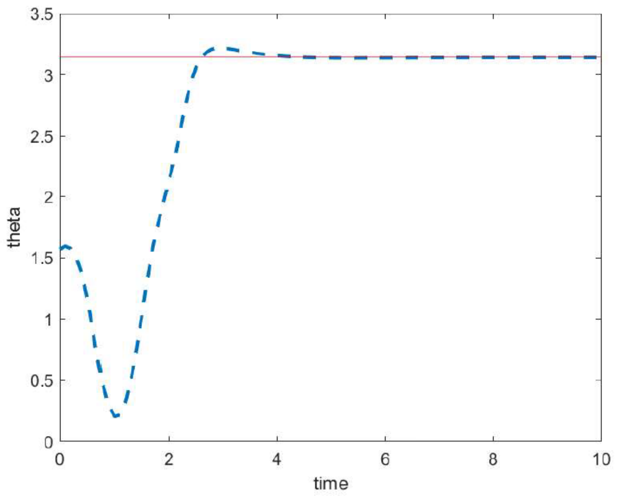
Fig. 2: (a) Schematic of the inverted pendulum. (b) System of ODEs for the described inverted pendulum in [1].



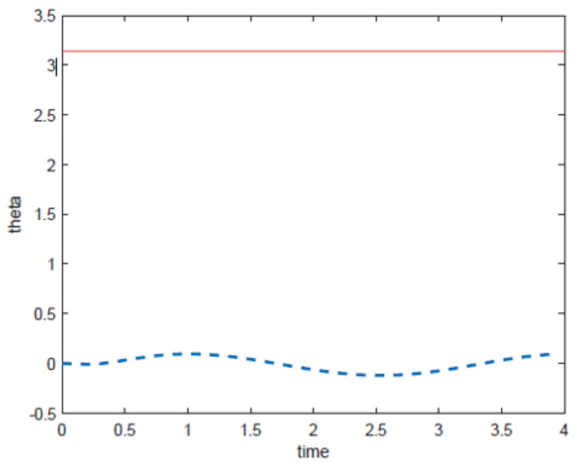
(a) (b)

Fig. 3: These results show the (a) trajectory of θ (b) control inputs u of the RTI method. The red line is for the initial condition of $\theta = pi/2$ and the blue line is for $\theta = 0$.

it is possible to control highly nonlinear continuous-time systems with a careful implementation of the algorithm. The effectiveness of the RTI scheme can be adjusted by tuning certain parameters such as the sampling time, horizon length, integrator accuracy, the use of a shifting strategy, passing the reference in a smart way, and cost tuning matrices. We considered as an example a pendulum mounted on top of a cart. The cart can only move on the horizontal axis (x) and its position is given by w_0. The angle of the pendulum is denoted by θ using the convention $\theta = 0$ rad that corresponds to the pendulum hanging down in the negative vertical (y) direction. The system dynamics are given by the explicit ODE 2b where $M = 1\ kg$; $m = 0.1\ kg$; $l = 0{:}5\ m$; $g = 9{:}81\ m/s^2$ 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 has been set up (with terminal cost) using weighting matrices $Q = diag([10, 10, 0.1, 0.1])$, $R = 0.01$, and terminal cost $\mathbf{x_N}^T diag([10, 10, 0.1, 0.1])\mathbf{x_N}$. No path constraints have been introduced for simplicity. All other tuning parameters are specified separately for each simulation. We used a prediction horizon $T_p = 2\ s$, a sampling time $T_s = 0.1\ s$, an explicit Runge Kutta integrator of order 4 (RK4) with a fixed stepsize $T_i = 0{:}025\ s$ and we made use of the shifting strategy (5). Using the control provided by the RTI-NMPC (see Algorithm 1), the closed-loop trajectories have been simulated using the integrators in MATLAB. We have used ACADO toolkit in MATLAB for our implementation. We assumed to have the lower bound of -20 and -2 with uper bound of 20 and 2 for control inputs and position of the cart respectively. The prediction horizon was 20 and RK4 number of steps was 5. You can see in Fig. 3 that with different initial conditions we can obtain the convergence. The problem turned out to be infeasible when we apply larg amount of mass for the cart



(a)



(b)

Fig. 4: These results show the (a) convergence with $-2 =< u <= 2$ and $M = 0.1$ and (b) infeasibility of the MPC problem when $-2 =< u <= 2$ and we have larger mass for the cart, $M = 1$.

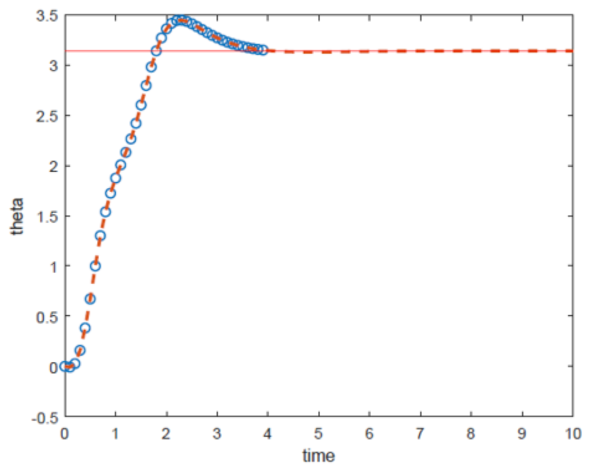


Fig. 5: One step RK4.

and could be solve by reducing the mass M or increasing the control input (see Fig. 4). We also reduced the number of RK4 steps to 1 and interestingly we found out that the larger prediction horizons can compensate the error in the approximation of the nonlinear dynamics, i.e. RK4 output (see Fig. 5).

VII. Conclusion

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. The original paper has provided a clear explanation of the similarities and differences between linear model predictive control (MPC) and the real-time iteration (RTI)-based NMPC approach. The RTI method can be viewed as a natural extension of linear MPC, where the system dynamics and path constraints are re-linearized using an integrator **at the current prediction rather than the reference**. However, RTI is also an SQP-type solver for NMPC, which can track the NMPC solution manifold under mild assumptions such as solving the QP just once at each time step and using the shifting procedure to build the initial guess for the next time step. As a result, in many cases, the RTI method can be used to implement an NMPC scheme with a limited additional computational burden and coding effort compared to linear MPC. RTI method also solved the problem of having the delay between real state of the physical system and the state estimated by the optimization solver, which we frequently observe in other solvers such as SQP.

References

[1] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear mpc: bridging the gap via the real-time iteration," *International Journal of Control*, vol. 93, no. 1, pp. 62–80, 2020.

[2] 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*, vol. 43, no. 5, pp. 1714–1736, 2005.

[3] 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 *53rd IEEE Conference on Decision and Control*, pp. 2752–2757, 2014.

[4] J. B. Rawlings, D. Angeli, and C. N. Bates, "Fundamentals of economic model predictive control," in *2012 IEEE*

51st IEEE Conference on Decision and Control (CDC), pp. 3851–3861, 2012.

[5] "Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations," *Journal of Process Control*, vol. 12, no. 4, pp. 577–585, 2002.

[6] "A continuation/gmres method for fast computation of nonlinear receding horizon control," *Automatica*, vol. 40, no. 4, pp. 563–574, 2004.

[7] "The advanced-step nmpc controller: Optimality, stability and robustness," *Automatica*, vol. 45, no. 1, pp. 86–93, 2009.

[8] M. Diehl, "Real-time optimization for large scale nonlinear processes," Ph.D. dissertation, 2001.

[9] M. Diehl, "Real-time optimization for large scale nonlinear processes, volume 920 of fortschr.-ber. vdi reihe 8, meß-, steuerungs-und regelungstechnik," 2002.

[10] M. Diehl, R. Findeisen, F. Allgöwer, H. G. Bock, and J. P. Schlöder, "Nominal stability of real-time iteration scheme for nonlinear model predictive control," *IEE Proceedings-Control Theory and Applications*, vol. 152, no. 3, pp. 296–308, 2005.