Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 56-2 (2023) 11841–11846

Fast Reinforcement Learning Based MPC based on NLP Sensitivities [⋆]

Saket Adhau [*] Dirk Reinhardt [**] Sigurd Skogestad [*] Sébastien Gros [**]

[*] *Department of Chemical Engineering, Norwegian University of Science and Technology, 7491, Trondheim, Norway. (e-mail: {saket.adhau,sigurd.skogestad}@ntnu.no).*

[**] *Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7491 Trondheim, Norway (e-mail: {dirk.p.reinhardt, sebastien.gros}@ntnu.no)*

Abstract: This paper proposes a comprehensive approach to improve the computational efficiency of Reinforcement Learning (RL) based Model Predictive Controller (MPC). Although MPC will ensure controller safety and RL can generate optimal control policies, combining the two requires substantial time and computational effort, particularly for larger data sets. In a typical RL-based MPC and $Q-$learning workflow, two not-so-different MPC problems must be evaluated at each RL iteration, i.e. one for the action-value and one for the value function, which is time-consuming and prohibitively expensive in terms of computations. We employ nonlinear programming (NLP) sensitivities to approximate the action-value function using the optimal solution from the value function, reducing computational time. The proposed approach can achieve comparable performance to the conventional method but with significantly lower computational time. We demonstrate the proposed approach on two examples: Linear Quadratic Regulator (LQR) problem and Continuously Stirred Tank Reactor (CSTR).

Keywords: Reinforcement learning, Economic model predictive control, Nonlinear programming, Sensitivity

1. INTRODUCTION

Model predictive control (MPC) has been widely used for the control of dynamic systems due to its ability to optimize future control actions based on a mathematical model of the system (Qin and Badgwell, 2003). However, uncertainties in model parameters, measurement errors, stochasticity, unmodeled dynamics, and incomplete knowledge of the system can introduce errors, leading to suboptimal control performance. To address these challenges, researchers have proposed combining data-driven and model-based control. In particular, Gros and Zanon (2019), proposed an approach that leverages data-driven techniques to enhance the performance of MPC in complex systems with nonlinear dynamics and uncertainties, without relying solely on a precise mathematical model of the system. This approach will hereafter be termed as RL based MPC or simply RL-MPC.

The approach proposed by Gros and Zanon (2019) can achieve good control performance for complex systems with uncertain and nonlinear dynamics, but it is often computationally inefficient due to the iterative nature of updating the policy parameters. In their approach, an MPC scheme is used to support the parametrization to approximate the action-value function, which requires iterative updates of the policy parameters to converge to an optimal solution. This iterative process can be time-consuming and computationally expensive, especially for systems with high-dimensional state and action spaces. The usual workflow in this approach is to evaluate two optimal control problems (OCP) in a single RL iteration, as both the value function and action-value function are parameterized using an MPC scheme. As a result, this increases the computational burden, making the approach less efficient for real-time control applications. Additionally, RL algorithms require large amounts of data to train to an acceptable level of performance, further increasing the computational cost.

To mitigate these challenges, we propose a method to reduce the computational burden by approximating the action-value function using nonlinear programming (NLP) sensitivities derived from the optimal solution of the value function. By exploiting these sensitivities, we only need to evaluate one optimal control problem (OCP) in a single RL iteration, instead of two, resulting in a significant reduction of computational effort and time without compromising control performance. This approach is particularly well-suited for complex real-time control applications. We demonstrate our approach on two benchmark methods commonly used in the field of control engineering. We show that even using NLP sensitivities for action-value function approximation can provide comparable control

[⋆] This work was supported by the Research Council of Norway (RCN), under the IKTPLUSS program (Project number 299585) and through the project Safe Reinforcement Learning using MPC (SARLEM), grant number 300172.

performance to traditional RL methods. This highlights the potential of our method to improve the computational efficiency of RL-based control strategies.

The paper is organized as follows: In Section 2, we provide a brief overview of parametric nonlinear programming and sensitivity analysis. Section 3 introduces background on RL-based MPC. The proposed approach for approximating the OCP associated with the action-value function is presented in Section 4, followed by implementations and numerical examples in Section 5. Finally, in Section 6, we conclude the paper.

2. NLP SENSITIVITIES

Nonlinear model predictive control (NMPC) is a widely used control technique in various fields due to its ability to handle nonlinear and non-convex systems. However, it is also known to pose significant computational challenges due to the need to solve a sequence of online optimization problems repeatedly. Additionally, the presence of constraints and nonlinearities can further exacerbate these challenges (Santos et al., 2001). To tackle this problem, various real-time NMPC strategies have been proposed, including explicit MPC for e.g. (Bemporad et al., 2002), neighboring extremals for e.g. (Diehl et al., 2005a), Newton-type controllers for e.g. (Diehl et al., 2005b), and controllers based on NLP sensitivities for e.g. (Büskens and Maurer, 2001; Zavala et al., 2008).

The authors Zavala et al. (2008) and Zavala and Biegler (2009) suggest the use of NLP sensitivities to predict the future state of the plant based on the current control action. This allows for the approximation of the future optimal control problem (OCP) in advance and was later used to solve MPC problems with an economic objective function by Jäschke et al. (2014). Alternatively, the real-time iteration scheme proposed by Diehl et al. (2005a) approximates the next control law by solving a few quadratic programming (QP) problems, which are considerably faster to evaluate. This real-time iteration (RTI) scheme was also adapted as a function approximator for RL by Zanon et al. (2020). This integration of RTI scheme has enabled the implementation of the approach proposed in (Gros and Zanon, 2019) to a wide range of real-time systems. In the upcoming section, we will delve into a concise explanation of the theory behind NLP sensitivities.

2.1 Parametric Nonlinear Optimization Problem

Parametric NLP involves solving a nonlinear optimization problem where the objective function and constraints depend on a set of parameters. In this context, we consider a general parametric NLP with both equality and inequality constraints, which can be written as:

$$\begin{aligned} \mathscr{P}(p) = \min_{\boldsymbol{w}} \quad & F(\boldsymbol{w}, p), && \text{(1a)} \\ s.t. \quad & c_i(\boldsymbol{w}, p) = 0, \quad \forall i \in \mathcal{E}, && \text{(1b)} \\ & g_i(\boldsymbol{w}, p) \leq 0, \quad \forall i \in \mathcal{I}, && \text{(1c)} \end{aligned}$$

where the decision variables are denoted by $\boldsymbol{w} \in \mathbb{R}^{n_w}$ and the parameter vector by $p \in \mathbb{R}^{n_p}$. Further, $F : \mathbb{R}^{n_w} \times \mathbb{R}^{n_p} \to \mathbb{R}$ is the scalar cost, $\mathcal{I}$ denotes the set of indices of inequality constraints, and $\mathcal{E}$ denotes the set of indices of equality constraints.

The Lagrangian function $\mathcal{L}$ for the problem $\mathscr{P}(p)$ is given by:

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}, p) := F(\boldsymbol{w}, p) + \boldsymbol{\lambda}^\top c(\boldsymbol{w}, p) + \boldsymbol{\mu}^\top g(\boldsymbol{w}, p), \quad (2)$$

where $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are the Lagrangian multipliers corresponding to the respective equality and inequality constraints. Suppose that $F(\cdot,\cdot)$, and $g(\cdot,\cdot)$ are continuously differentiable and $\boldsymbol{w}^\star$ is a local optimizer. If the linear independence constraint qualification (LICQ) condition holds at $\boldsymbol{w}^\star$, then the first-order optimality or Karush-Kuhn-Tucker (KKT) conditions for $\mathscr{P}(p)$ are given by

$$\begin{aligned} \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star, p) &= 0, && \text{(3a)} \\ c_i(\boldsymbol{w}^\star, p) &= 0, \quad \forall i \in \mathcal{E}, && \text{(3b)} \\ g_i(\boldsymbol{w}^\star, p) &\leq 0, \quad \forall i \in \mathcal{I}, && \text{(3c)} \\ \boldsymbol{\mu}_i^\star &\geq 0, \quad \forall i \in \mathcal{I}, && \text{(3d)} \\ \boldsymbol{\mu}_i^\star g_i(\boldsymbol{w}^\star, p) &= 0, \quad \forall i \in \mathcal{I}. && \text{(3e)} \end{aligned}$$

A point $\boldsymbol{z}^\star(p) := [\boldsymbol{w}^\star, \boldsymbol{\lambda}^\star, \boldsymbol{\mu}^\star]$ of primal-dual variables satisfying the KKT conditions (3) for a given initial parameter p, is called a KKT point for p. If the parameter p is clear from the context, we simply write $\boldsymbol{z}^\star$ instead of $\boldsymbol{z}^\star(p_0)$.

The set of indices of active constraints is denoted by $\mathcal{I}_{\mathbb{A}}$, and is defined as $\mathcal{I}_{\mathbb{A}} = \{i \in \mathcal{I} : g_i(\boldsymbol{w}^\star) = 0\} \cup \mathcal{E}$. However, for a local minimizer of (1) to be a KKT point, a constraint qualification is required (Nocedal and Wright, 1999). Specifically, the LICQ condition must hold at $\boldsymbol{w}$, which means that the vectors $\{\nabla g_i(\boldsymbol{w})\}_{i \in \mathcal{I}_{\mathbb{A}}}$ are linearly independent. Moreover, the strict complementarity condition holds if $\boldsymbol{\mu}_i > 0$ for all $i \in \mathcal{I}_{\mathbb{A}}$.

This set of KKT conditions is typically expressed as a system of nonlinear equations and inequalities in the primal and dual variables, and is used to solve parametric NLP problems numerically, using optimization algorithms such as sequential quadratic programming (SQP) and interior point methods.

$$\varphi(\boldsymbol{z}^\star(p), p) = \begin{bmatrix} \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{z}^\star(p), p) \\ c(\boldsymbol{w}^\star, p) \\ \boldsymbol{\mu}^{\star\top} g(\boldsymbol{w}^\star, p) \end{bmatrix} = 0. \quad (4)$$

Theorem 1. Let $F(\cdot,\cdot)$, $c(\cdot,\cdot)$ and $g(\cdot,\cdot)$ of the parametric NLP problem $\mathscr{P}(p)$ be twice continuous differentiable in a neighborhood of the KKT point $\boldsymbol{z}^\star(p_0)$. Further, assume strict complementarity (SC), linear independence constraint qualification (LICQ), and second order sufficient condition (SOSC) hold for the solution vector $\boldsymbol{z}^\star(p_0)$. Then,

- $\boldsymbol{z}^\star(p_0)$ is an isolated local minimizer of $\mathscr{P}(p_0)$ and the associated Lagrangian multipliers $\boldsymbol{\lambda}^\star$ are unique.
- For parametric perturbations Δp in the neighborhood of p_0, there exits a unique, continuous and differentiable vector function $\boldsymbol{z}^\star(p_0 + \Delta p)$ which is a local minimizer satisfying SOSC and LICQ for $\mathscr{P}(p_0 + \Delta p)$.
- For parametric perturbations Δp in the neighborhood of p_0, the set of active constraints remain unchanged.

Proof. See Fiacco (1983).

The above results allow us to apply the implicit function theorem to (4) at $\boldsymbol{z}^\star(p_0)$, such that:

$$\frac{\partial}{\partial p}\varphi(\boldsymbol{z}^\star(p),p)\bigg|_{p=p_0} = \frac{\partial\varphi}{\partial \boldsymbol{z}}\frac{\partial \boldsymbol{z}^\star}{\partial p} + \frac{\partial\varphi}{\partial p} = 0\,. \tag{5}$$

Since the nominal solution satisfies both the SOSC and LICQ conditions, the KKT matrix (4) at the KKT point $\boldsymbol{z}^\star(p_0)$ is non-singular (Nocedal and Wright, 1999), and hence, can be used to calculate the sensitivity matrix from (5). The first-order estimates of the solution for neighboring problems can then be obtained as:

$$\tilde{\boldsymbol{z}}(p) \approx \boldsymbol{z}^\star(p_0) + \frac{\partial \boldsymbol{z}^\star(p_0)}{\partial p}\Delta p\,, \tag{6}$$

where $\tilde{\boldsymbol{z}}(p)$ is the approximate primal-dual solution of the optimization problem $\mathscr{P}(p_0+\Delta p)$. This is a computationally efficient approach, as the cost of solving this linear system is typically much lower than that of solving $\mathscr{P}(p_0+\Delta p)$ through a conventional optimization routine, especially when the number of decision variables is large, provided the solution $\boldsymbol{z}^\star(p_0)$ is available. This makes sensitivity analysis a powerful tool in optimizing and controlling complex systems.

Similarly, the first order Taylor expansion of the optimal cost for the problem $\mathscr{P}(p_0+\Delta p)$ can be expressed as

$$F(\boldsymbol{z}(p)) \approx F(\boldsymbol{z}^\star(p_0)) + \frac{\partial F(\boldsymbol{z}^\star(p_0))}{\partial p}\Delta p\,, \tag{7}$$

where the first order sensitivity derivative of the objective function is given by (Büskens and Maurer, 2001)

$$\frac{\partial}{\partial p}F\left(\boldsymbol{z}^\star(p),p\right)\bigg|_{p=p_0} = \frac{\partial}{\partial p}\mathcal{L}(\boldsymbol{z}^\star(p_0)), \tag{8}$$

and the second order Taylor expansion of the optimal cost is given as

$$F(\boldsymbol{z}(p)) \approx F(\boldsymbol{z}^\star(p_0)) + \frac{\partial F(\boldsymbol{z}^\star(p_0))}{\partial p}\Delta p + \frac{1}{2}(\Delta p)^\top \frac{\partial^2 F(\boldsymbol{z}^\star(p_0))}{\partial p^2}\Delta p. \tag{9}$$

3. REINFORCEMENT LEARNING BACKGROUND

Consider a real system having state transition dynamics described by a Markov Process (MP) with state $\boldsymbol{s}$ and action $\boldsymbol{a}$ and a state transition $\boldsymbol{s},\boldsymbol{a}\rightarrow\boldsymbol{s}_+$ described by a probability density

$$\mathbb{P}[\boldsymbol{s}_+|\boldsymbol{s},\boldsymbol{a}]\,. \tag{10}$$

To model the system as a Markov Decision Process (MDP), we augment this model with a stage cost function $\ell(\boldsymbol{s},\boldsymbol{a})$ and a discount factor $0\le\gamma\le 1$.

Let us consider a deterministic policy that delivers the control action $\boldsymbol{a}=\pi(\boldsymbol{s})$, giving a Markov Chain distribution τ^π. The ultimate goal of RL is to find the best policy $\pi^\star$ by evaluating the cumulative cost of the policy π, i.e. by solving,

$$\pi^\star := \arg\min_\pi J(\pi) := \mathbb{E}_{\tau^\pi}\left[\sum_{k=0}^{\infty}\gamma^k\ell\left(\boldsymbol{s}_k,\pi(\boldsymbol{s}_k)\right)\right]. \tag{11}$$

The optimal action-value function $Q^\star$, value function $V^\star$, and optimal policy $\pi^\star(\boldsymbol{s})$ associated to the MDP, are defined by the Bellman equations (Bertsekas, 2005):

$$Q^\star(\boldsymbol{s},\boldsymbol{a}) = \ell(\boldsymbol{s},\boldsymbol{a}) + \gamma\mathbb{E}\left[V^\star(s_+)\,|\,\boldsymbol{s},\boldsymbol{a}\right], \tag{12a}$$
$$V^\star(\boldsymbol{s}) = Q^\star(\boldsymbol{s},\pi^\star(\boldsymbol{s})) = \min_a Q^\star(\boldsymbol{s},\boldsymbol{a})\,. \tag{12b}$$

Various RL methods have been proposed in the literature to generate the optimal policy $\pi^\star$. However, in this paper, we will be specifically focusing on the classical $Q-$-learning algorithm.

3.1 ENMPC as a Function Approximator

The use of an ENMPC scheme as a generic function approximator for RL has been proposed by Gros and Zanon (2019). The authors, have demonstrated that an ENMPC scheme can support the parametrization of the action-value function Q_θ, V_θ and the policy π_θ. The central result is that with some modifications in the stage cost, terminal cost, and constraints, ENMPC scheme can deliver optimal control policy even when the model underlying the MPC scheme is incorrect. We briefly summarize the main results of Gros and Zanon (2019) in the following.

Let us use a parameter vector $\boldsymbol{\theta}$ and consider the following parametrization of an ENMPC scheme to approximate the value function:

$$V_\theta(\boldsymbol{s}) = \min_{\boldsymbol{x},\boldsymbol{u}}\ \lambda_\theta(\boldsymbol{x_0}) + \gamma^N V_\theta^{\mathrm{f}}(\boldsymbol{x}_N) + \sum_{k=0}^{N-1}\gamma^k\ell_\theta(\boldsymbol{x}_k,\boldsymbol{u}_k) \tag{13a}$$
$$s.t.\quad \boldsymbol{x}_0 = \boldsymbol{s}, \tag{13b}$$
$$\boldsymbol{x}_{k+1} = \mathbf{f}_\theta(\boldsymbol{x}_k,\boldsymbol{u}_k), \tag{13c}$$
$$\boldsymbol{g}(\boldsymbol{u}_k)\le 0, \tag{13d}$$
$$\boldsymbol{h}_\theta(\boldsymbol{x}_k,\boldsymbol{u}_k)\le 0,\quad \boldsymbol{h}_\theta^{\mathrm{f}}(\boldsymbol{x}_N)\le 0, \tag{13e}$$

where the stage and terminal cost $\ell_\theta, V_\theta^{\mathrm{f}}$, the system dynamics $\mathbf{f}_\theta$ and the constraints $\boldsymbol{h}_\theta,\boldsymbol{h}_\theta^{\mathrm{f}}$ and the storage cost λ_θ are parametric functions of $\boldsymbol{\theta}$. To ensure feasibility of (13), Gros and Zanon (2019) propose to use an exact relaxation of the state-dependent constraints. We define the policy

$$\pi_\theta(\boldsymbol{s}) = \boldsymbol{u}^\star_,, \tag{14}$$

where $\boldsymbol{u}_0^\star$ is the first element of the optimal input sequence $\boldsymbol{u}_0^\star,\dots,\boldsymbol{u}_{N-1}^\star$ solution of (13) for a given state $\boldsymbol{s}$. Further, we define the action-value function $Q_\theta(\boldsymbol{s},\boldsymbol{a})$ as

$$Q_\theta(\boldsymbol{s},\boldsymbol{a}) = \min_{\boldsymbol{u},\boldsymbol{x}}\quad (13a), \tag{15a}$$
$$\text{s.t.}\quad (13b)-(13e), \tag{15b}$$
$$\boldsymbol{u}_0 = \boldsymbol{a}, \tag{15c}$$

Note that the two problems i.e. (13) and (15) differ only with an additional constraint $\boldsymbol{u}_0=\boldsymbol{a}$ in (15). The proposed parametrization satisfies the fundamental equalities underlying the bellman equations, i.e.,

$$\pi_\theta(\boldsymbol{s}) = \arg\min_{\boldsymbol{a}} Q_\theta(\boldsymbol{s},\boldsymbol{a}),\quad V_\theta(\boldsymbol{s}) = \min_{\boldsymbol{a}} Q_\theta(\boldsymbol{s},\boldsymbol{a}). \tag{16}$$

Additionally, the Lagrange function underlying the parameterized ENMPC scheme in (15) is given as:

$$\begin{aligned}\mathcal{L}_\theta(\boldsymbol{y}) =& \lambda_\theta(\boldsymbol{x}_0) + \gamma^N V_\theta^{\mathrm{f}}(\boldsymbol{x}_N) + \boldsymbol{\chi}_0^\top(\boldsymbol{x}_0-\boldsymbol{s}) + \boldsymbol{\mu}_N^\top\boldsymbol{h}_\theta^{\mathrm{f}}(\boldsymbol{x}_N)\\ &+\sum_{k=0}^{N-1}\boldsymbol{\chi}_{k+1}^\top\left(\boldsymbol{f}_\theta(\boldsymbol{x}_k,\boldsymbol{u}_k)-\boldsymbol{x}_{k+1}\right) + \boldsymbol{\nu}_k^\top\boldsymbol{g}_\theta(\boldsymbol{u}_k)\\ &+\gamma^k\ell_\theta(\boldsymbol{x}_k,\boldsymbol{u}_k) + \boldsymbol{\mu}_k^\top\boldsymbol{h}_\theta(\boldsymbol{x}_k,\boldsymbol{u}_k) + \boldsymbol{\zeta}^\top(\boldsymbol{u}_0-\boldsymbol{a}),\end{aligned} \tag{17}$$

where $\boldsymbol{\chi}, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\zeta}$ are the multipliers associated to the constraints (15b)-(15c), and $\boldsymbol{y} = (\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\chi}, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\zeta})$ are the primal-dual variables associated to (15). Note that for $\boldsymbol{\zeta} = 0$, $\mathcal{L}_\theta(\boldsymbol{y})$ is the Lagrange function associated to the problem (13), or $V_\theta(\boldsymbol{s})$. Using the results in (8), we observe that,

$$\frac{\partial}{\partial\theta} V_\theta(\boldsymbol{s}) = \frac{\partial}{\partial\theta}\mathcal{L}_\theta(\boldsymbol{y}^\diamond), \tag{18}$$

where $\boldsymbol{y}^\diamond$ is the primal-dual solution of (13).

3.2 $Q-$Learning for ENMPC

In $Q-$learning, the action-value function is parameterized as $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$, where $\boldsymbol{\theta}$ is a vector of parameters. The classical $Q-$learning algorithm updates these parameters based on temporal differences, with instantaneous policy updates.

$$\delta_k = \ell(\boldsymbol{s}_k, \boldsymbol{a}_k) + \gamma \min_{\boldsymbol{a}_{k+1}} Q_\theta(\boldsymbol{s}_{k+1}, \boldsymbol{a}_{k+1}) - Q_\theta(\boldsymbol{s}_k, \boldsymbol{a}_k), \tag{19a}$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\delta_k \nabla_\theta Q_\theta(\boldsymbol{s}_k, \boldsymbol{a}_k), \tag{19b}$$

where the scalar $\alpha > 0$ is the learning rate.

4. SENSITIVITY ANALYSIS

To use ENMPC as a function approximator in RL using the $Q-$learning method, one needs to evaluate $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$, $V_\theta(\boldsymbol{s})$ in (19a) and its parametric sensitivities, i.e $\nabla_\theta Q_\theta$ in (19b). The terms $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$, $V_\theta(\boldsymbol{s})$ are conventionally obtained by evaluation of two parametric nonlinear programming problems, i.e. (13), and (15), of which, both need to be solved at each iteration. The evaluation of these two NLP problems can be computationally demanding and on a closer observation, (13) and (15) differ only with a single constraint, i.e. $\boldsymbol{u}_0 = \boldsymbol{a}$. Hence, $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ can be approximated with the help of parametric NLP sensitivities using the optimal solution from (13). Approximation of $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ from the optimal value function $V_\theta(\boldsymbol{s})$, avoiding the evaluation of one extra NLP, and reducing the computational time required for a single RL iteration.

4.1 Approximation of $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ and $\nabla_\theta Q_\theta$ using NLP Sensitivities

In this section, we detail on how to approximate the action-value function $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ and its gradient $\nabla_\theta Q_\theta$ from the optimal solution of (13) using the results presented in Section 2.1. More specifically, we propose to solve problem (13) to full convergence and avoid solving (15). Using (9), the second order Taylor expansion of the optimal value function can be written as

$$Q_\theta(\boldsymbol{s}, \boldsymbol{a}) \approx V_\theta(\boldsymbol{s}) + \left.\frac{\partial Q_\theta(\boldsymbol{s}, \boldsymbol{a})}{\partial \boldsymbol{a}}\Delta\boldsymbol{a}\right|_{\boldsymbol{a}=\pi_\theta(\boldsymbol{s})} + \frac{1}{2}(\Delta\boldsymbol{a})^\top \left.\frac{\partial^2 Q_\theta(\boldsymbol{s}, \boldsymbol{a})}{\partial \boldsymbol{a}^2}\Delta\boldsymbol{a}\right|_{\boldsymbol{a}=\pi_\theta(\boldsymbol{s})}, \tag{20}$$

where $\Delta\boldsymbol{a} = \boldsymbol{a} - \pi_\theta(\boldsymbol{s})$ and $\pi_\theta(\boldsymbol{s})$ is the policy (14). Note that the first order Taylor expansion cannot be used because the term,

$$\left.\frac{\partial Q_\theta(\boldsymbol{s}, \boldsymbol{a})}{\partial \boldsymbol{a}}\right|_{\boldsymbol{a}=\pi_\theta(\boldsymbol{s})} = \frac{\partial}{\partial \boldsymbol{a}}\mathcal{L}_\theta(\boldsymbol{y}^\star) = 0, \tag{21}$$

where $\mathcal{L}_\theta(\boldsymbol{y}^\star)$ and $\boldsymbol{y}^\star$ are associated to the problem (15).

Next, we detail on how to approximate the gradient of $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ as required in (19b) using Taylor expansion:

$$\frac{\partial}{\partial\theta} Q_\theta(\boldsymbol{s}, \boldsymbol{a}) \approx \frac{\partial}{\partial\theta} V_\theta(\boldsymbol{s}) + \left.\frac{\partial^2 Q_\theta(\boldsymbol{s}, \boldsymbol{a})}{\partial\theta\partial\boldsymbol{a}}(\boldsymbol{a} - \pi(\boldsymbol{s}))\right|_{\pi(\boldsymbol{s})}, \tag{22}$$

where,

$$\begin{aligned}\frac{\partial^2 Q_\theta(\boldsymbol{s}, \boldsymbol{a})}{\partial\theta\partial\boldsymbol{a}} &= \frac{\mathrm{d}}{\mathrm{d}\theta}\frac{\partial Q_\theta(\boldsymbol{s}, \boldsymbol{a})}{\partial\boldsymbol{a}} = \frac{\mathrm{d}}{\mathrm{d}\theta}\frac{\partial \mathcal{L}_\theta(\boldsymbol{y}^\star)}{\partial\boldsymbol{a}} \\ &= \frac{\partial^2 \mathcal{L}_\theta(\boldsymbol{y}^\star)}{\partial\theta\partial\boldsymbol{a}} + \frac{\partial^2 \mathcal{L}_\theta(\boldsymbol{y}^\star)}{\partial\boldsymbol{a}\partial\boldsymbol{y}}\frac{\partial\boldsymbol{y}}{\partial\theta}.\end{aligned} \tag{23}$$

5. SIMULATIONS

In this section, we demontsrate the capabilities of our proposed method by applying it to two benchmark problems. We apply the approach to a simple LQR case study and an example from the process industry, namely continuously stirred tank reactor (CSTR). These simulations serve to illustrate the potential of our approach for practical applications in economic MPC.

5.1 LQR case

Consider the trivial linear dynamics and quadratic stage cost of a regulator LQR problem.

$$s_{k+1} = 2s_k - a_k, \quad L(s, a) = s^2 + a^2 + 4sa \ngeq 0 \tag{24}$$

The optimal steady-state, denoted as $(s_s, a_s) = (0, 0)$, can be easily calculated for the given linear dynamics and quadratic stage cost. Using the Riccati equation, the exact optimal value function and policy can be given as:

$$V^\star(s) = 11.7446s^2, \quad \pi^\star(s) = 1.6861s$$

The parameterized stage cost $\hat{\ell}_\theta$, the terminal cost T_θ and the storage function λ_θ are selected as

$$\lambda_\theta(s) = \theta_1 s^2, \quad T_\theta(s) = \theta_2 s^2, \tag{25a}$$

$$\hat{\ell}_\theta(s) = \begin{bmatrix} s \\ a \end{bmatrix}^\top \begin{bmatrix} \theta_3 & \theta_4 \\ \theta_5 & \theta_6 \end{bmatrix} \begin{bmatrix} s \\ a \end{bmatrix}, \tag{25b}$$

where the vector of parameters $\boldsymbol{\theta} = \theta_1, \ldots, \theta_6$ can be adjusted using $Q-$learning. We initialize these parameters as $\boldsymbol{\theta}_0 = [0.1, 1, 1, 0, 0, 0]^\top$. For our simulations, we set the prediction horizon to $N = 5$, learning rate to $\alpha = 5e^{-3}$ and the discount factor to $\gamma = 1$.

The iterative updates of the parameters $\boldsymbol{\theta}$ using $Q-$learning are compared between two methods: solving $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ to full convergence and approximating it using NLP sensitivities, as shown in Fig.1. The results indicate that the update steps are similar for both methods. Furthermore, Fig.2 demonstrates that $Q-$learning can capture the optimal value function even when $Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ and $\nabla_\theta Q_\theta(\boldsymbol{s}, \boldsymbol{a})$ are approximated using NLP sensitivities.

5.2 CSTR Case Study

We demonstrate the proposed method using the isothermal Van de Vusse reaction in a continuous stirred tank reactor as an illustrative example (Chen et al., 1995; Klatt and Engell, 1993). This model includes both material and

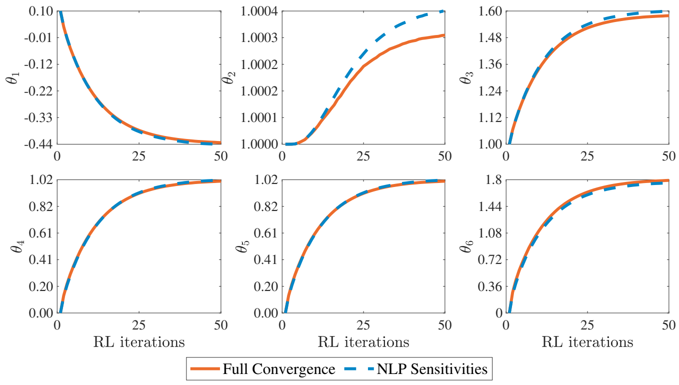

Fig. 1. Comparison of the iterative update of parameter vector $\boldsymbol{\theta}$ using $Q-$learning when the action-value function $Q_\theta(\boldsymbol{s},\boldsymbol{a})$ is solved until full convergence vs. when it is approximated using NLP sensitivities. The plots show the progress of the parameter updates over time for both methods. Note that the scales are different.

enthalpy balances, and is described by the following equations:

$$\dot{c_A} = r_A(c_A, T) + [c_{in} - c_A]u_1 \tag{26a}$$
$$\dot{c_B} = r_B(cA, c_B, T) - c_B u_1, \tag{26b}$$
$$\dot{T} = h(c_A, c_B, T) + \varphi[T_c - T] + [T_{in} - T]u_1, \tag{26c}$$
$$\dot{T_c} = \beta[T - T_c] + \varsigma u_2, \tag{26d}$$

with $c_A(t), c_B(t)$ the concentrations of A and B respectively, $T(t)$ and $T_c(t)$ the temperatures in the reactor and in the cooling jacket. The reaction rates r_A and r_B, and the reaction enthalpy contribution h are described by

$$r_A(c_A, T) = -k_1(T)c_A - k_2(T)c_A^2, \tag{27}$$
$$r_B(c_A, c_B, T) = k_1(T)[c_A - c_B], \tag{28}$$
$$h(c_A, c_B, T) = -\vartheta[k_1(T)[c_A\Delta H_{AB} + c_B\Delta H_{BC}] + k_2(T)c_A^2\Delta H_{AD}], \tag{29}$$

with the functions $k_1(T)$ and $k_2(T)$ of the Arrhenius type:

$$k_i(T) = k_{i0}\exp\left(\frac{-E_i}{T + 273.15}\right), \quad i = 1, 2. \tag{30}$$

The above mentioned concentrations and temperatures constitute the state $x = [c_A, c_B, T, T_c]$ with dynamics given by (26a)–(26d). The control inputs $u = [u_1, u_2]$ are normalized input flow rate $u_1(t) > 0$ through the reactor and the cooling power $u_2(t) < 0$ applied in the cooling jacket. The model parameters are summarized in Table 1.

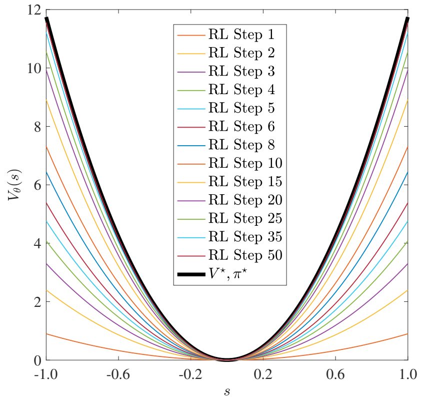

Fig. 2. Convergence of the approximated value function to the optimal $V^\star$ using the proposed approach..

Table 1. Model Parameters and the considered setpoints of the CSTR system (Chen et al., 1995).

φ	$= 30.8285\,\mathrm{h}^{-1}$	β	$= 86.688\,\mathrm{h}^{-1}$
ϑ	$= 3.556 \times 10^{-4}\,\mathrm{m^3K/kJ}$	ς	$= 0.1\,\mathrm{K\,kJ^{-1}}$
k_{10}	$= (1.287) \times 10^{12}\,\mathrm{h}^{-1}$	E_1	$= 9758.3$
k_{20}	$= (9.043) \times 10^{6}\,\mathrm{m^3/mol\,h}$	E_2	$= 8560$
ΔH_{AB}	$= 4.2\,\mathrm{kJ\,mol^{-1}}$		
ΔH_{BC}	$= -11\,\mathrm{kJ\,mol^{-1}}$		
ΔH_{AD}	$= -41.85\,\mathrm{kJ\,mol^{-1}}$		
c_{in}	$= 5100 \pm 600\,\mathrm{mol/m^3}$	T_{in}	$= 104.9\,\mathrm{K}^{-1}$
c_{A_0}	$= 3517.5\,\mathrm{mol/m^3}$	c_{B_0}	$= 740\,\mathrm{mol/m^3}$
T_0	$= 87\,\mathrm{K}$	T_{c_0}	$= 79.8\,\mathrm{K}$

We consider the problem of maximizing the production rate of c_B:

$$F = -c_B u_1, \tag{31}$$

and the ENMPC scheme is designed as depicted in (13), with $N = 12$ min and sampling time of $T_s = 6$ s,

$$\min_{x,u} \quad \lambda_\theta(x_0) + \gamma^N V_\theta^{\mathrm{f}}(x_N + \omega^\top\sigma_{\mathrm{f}}) \tag{32a}$$
$$+ \sum_{k=0}^{N-1}\gamma^k\left(F(x_k, u_k) + \omega^\top\sigma_k\right) \tag{32b}$$
$$s.t. \quad x_{k+1} = f_\theta(x_k, u_k), \quad x_0 = s, \tag{32c}$$
$$g(u_k) \le 0, \qquad h_\theta(x_k) \le \sigma_k. \tag{32d}$$

In this work, for formulating the MPC problem, we use CasADi (Andersson et al., 2019) with IPOPT as the Nonlinear Programming (NLP) solver. For learning the optimal policy, we use $Q-$learning with a learning rate of $\alpha = 10^{-3}$. The discount factor, was set to $\gamma = 0.99$. All the simulations have been performed on a MacBook Pro with Intel Core i7 running at 2.6 GHz and 16 GB of memory.

We conducted a closed-loop simulation to compare the performance of the proposed method with the conventional approach, wherein both optimization problems are solved. The experimental results of state evaluation for c_B, T, and the control inputs u_1 and u_2 are depicted in Figure 3. As observed, the performance of both methods is remarkably similar, thereby establishing that the proposed approach would not impede the closed-loop performance.

Table 2 presents the computational time required to solve problem (15) to full convergence and to approximate problem (13) using NLP sensitivities in a single RL iteration. Notably, the proposed approach substantially reduces the average CPU time needed for a single RL iteration, without compromising the average cost.

Table 2. Comparison of average cost and computation time for the CSTR system when solving (13) until full convergence and when approximating it using NLP sensitivities.

CSTR	Average Cost	Average CPU time [s]
Full Convergence	28463	9.57
NLP Sensitivities	28943	3.48

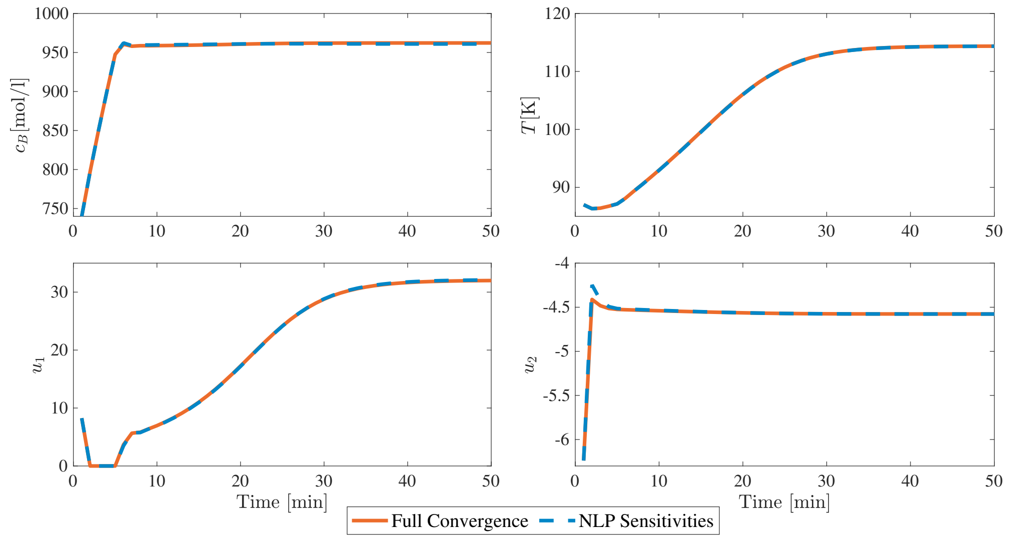

Fig. 3. Performance comparison of two approaches for states c_B, T, and control inputs u_1, u_2. The first approach involves solving (13) until full convergence, while the second approach approximates (13) using NLP sensitivities

6. CONCLUSION

In this paper, we have presented a simple yet effective approach to enhance the computational efficiency of RL-based MPC by utilizing NLP sensitivities. Our proposed method approximates the action-value function by exploiting similarities between consecutive nonlinear programming problems and leveraging NLP sensitivities. Through numerical simulations on two benchmark problems, we have demonstrated that our approach is capable of significantly reducing computational time without compromising controller performance. Our results suggest that the proposed framework offers a promising direction for future research aimed at further improving the computational efficiency of RL-based MPC.

REFERENCES

Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., and Diehl, M. (2019). CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11, 1–36.

Bemporad, A., Morari, M., Dua, V., and Pistikopoulos, E.N. (2002). The explicit linear quadratic regulator for constrained systems. *Automatica*, 38, 3–20.

Bertsekas, D.P. (2005). Dynamic programming and optimal control, volume 1 of optimization and computation series. *Athena Scientific, Belmont, MA, USA, 3rd edition. Cited on*, 2.

Büskens, C. and Maurer, H. (2001). Online optimization of large scale systems, chapter sensitivity analysis and real-time optimization of parametric nonlinear programming problems. *Berlin Heidelberg, Berlin, Heidelberg*, 3–16.

Chen, H., Kremling, A., and Allgöwer, F. (1995). Nonlinear predictive control of a benchmark CSTR. *Proceedings of 3rd European control conference*, 3247–3252.

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

Diehl, M., Findeisen, R., Allgöwer, F., Bock, H.G., and Schlöder, J.P. (2005b). Nominal stability of real-time iteration scheme for nonlinear model predictive control. *IEE Proceedings-Control Theory and Applications*, 152, 296–308.

Fiacco, A.V. (1983). *Introduction to sensitivity and stability analysis in nonlinear programming*, volume 165. Academic press.

Gros, S. and Zanon, M. (2019). Data-driven economic NMPC using reinforcement learning. *IEEE Transactions on Automatic Control*, 65, 636–648.

Jäschke, J., Yang, X., and Biegler, L.T. (2014). Fast economic model predictive control based on NLP-sensitivities. *Journal of Process Control*, 24, 1260–1272.

Klatt, K. and Engell, S. (1993). Kontinuierlicher rührkesselreaktor mit neben–und folgereaktionen. *Nichtlineare Regelung–Methoden, Werkzeuge, Anwendungen, VDI–Berichte*, 101–108.

Nocedal, J. and Wright, S.J. (1999). *Numerical optimization*. Springer.

Qin, S.J. and Badgwell, T.A. (2003). A survey of industrial model predictive control technology. *Control engineering practice*, 11, 733–764.

Santos, L.O., Afonso, P.A., Castro, J.A., Oliveira, N.M., and Biegler, L.T. (2001). On-line implementation of nonlinear MPC: an experimental case study. *Control Engineering Practice*, 9, 847–857.

Zanon, M., Kungurtsev, V., and Gros, S. (2020). Reinforcement learning based on real-time iteration NMPC. *IFAC-PapersOnLine*, 53, 5213–5218.

Zavala, V.M. and Biegler, L.T. (2009). The advanced-step NMPC controller: Optimality, stability and robustness. *Automatica*, 45, 86–93.

Zavala, V.M., Laird, C.D., and Biegler, L.T. (2008). Fast implementations and rigorous models: Can both be accommodated in NMPC? *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 18, 800–815.