Preprints, 18th IFAC Workshop on Control Applications of Optimization
July 18-22, 2022. Gif sur Yvette, France

Nearly Optimal Tunable MPC Strategies on Embedded Platforms

Karol Kiš, Peter Bakaráč, Martin Klaučo

Institute of Information Engineering, Automation, and Mathematics,
Slovak University of Technology in Bratislava, Slovakia,
(e-mail: {karol.kis,peter.bakarac,martin.klauco}@stuba.sk)

Abstract: We present an embeddable optimization-free application of a near-optimal MPC implementation with continuous tuning capabilities. We propose a strategy combining the advantages of explicit model predictive control with tunable properties that is implementable on embedded platforms with limited memory and computational resources. We consider a neural network (NN) learning procedure to mimic the control actions of an MPC strategy. While acknowledging limited guarantees on the constraints satisfaction with just the NN-based controller, we introduce an optimization-based corrector of the mimicked control action. Such a corrector then steers the control authority of the mimicked controller such that constraints on manipulated and process variables are enforced. To demonstrate the efficacy of the proposed control strategy, a case study implemented on an embedded platform is shown.

Keywords: Data-based optimal control; Real-Time Control Problems; Optimal Control

1. INTRODUCTION

The ability to change the aggressiveness of any controller during the operation plays a vital role in all industrial applications. It allows for a change in closed-loop performance without the need for extensive retuning procedures which often include system identification or other experiments. The retuning concept is especially resource-demanding in the application of Model Predictive Control (MPC), where choosing the weighing factors is often difficult for system operators and requires deeper knowledge about the controller structure (Wojsznis et al., 2003). For practical approaches, it is very convenient to have a specific interval in which the tuning factors can be changed. Naturally, the controller needs to provide satisfactory performance within these intervals. This manuscript aims precisely at this goal, where a control strategy improved by neural networks provides near-to-optimal control action subject to process measurements and choice of the tuning matrices.

Application of the model predictive control strategy on embedded platforms boils down to constructing an explicit solution via parametric programming. In order to arrive at a successful explicit solution in a reasonable time with a reasonable memory footprint, we are limited by the number of parameters, the length of the prediction horizon, and subsequently by the memory capabilities of the given embedded platform. Even though there exist several strategies how to mitigate the memory requirements of the explicit MPC, we are still limited to the fixed structure of the optimization problem, i.e., linear constraints and linear or quadratic performance criteria. Such controllers can not be considered when a change in tuning factors is required. Several scientific works address the problem of memory requirements for the explicit MPC, like (Kvasnica et al., 2013; Holaza et al., 2015) and references therein, but all motioned works are limited to the fixed structure of the optimization problem.

Of course, the problem with tunability does not arise in the implicit solution, where the change of the tuning factors does not affect the procedure of solving the problem. In the case of the explicit MPC, the parametric programming must be performed each time the tuning factors change. Another possibility would be to consider the tuning factors as parameters, but then we no longer keep the favorable structure of linear constraints and quadratic objective function.

One of the remedies to previous design obstacles is the MPC imitated by the neural networks (denoted as NN-controller). Such a concept is gaining much popularity in the control community in recent years. Among theoretical approaches, as presented by Lucia et al. (2021), a significant amount of application-based results has been published (Lohr et al., 2019; Karg and Lucia, 2020, 2018). In fact, the tunability aspects of the MPC can be easily adopted in the training procedure to obtain an NN-based controller in which input parametric space can cover not only state measurements but also values of the tuning factors (Kiš et al., 2020).

To consider a sole NN-controller as the main governing body raises an issue of constraint satisfaction. Even though the neural network is trained to imitate the MPC strategy, output constraint satisfaction is difficult. Therefore, we suggest adopting a mechanism called MPC-based cor-

⋆ The authors gratefully acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under the grant 1/0545/20, the Slovak Research and Development Agency under the projects APVV-20-0261, APVV-21-0019 and the STU Grant scheme for Support of Young Researchers.

rectors (Kvasnica et al., 2012), which steers the control authority of the NN-controller such that constraints on process variables are enforced. Unfortunately, the consideration of the optimization-based corrector decreases the applicability of the entire scheme on the embedded hardware. We propose to solve the corrector problem with the Random Shooting approach (Bakaráč and Kvasnica, 2018), which relies only on simple mathematical operations, that do not require any optimization solvers. Therefore, the entire implementation of the control strategy is free of optimization the explicit MPC construction, and moreover, we allow for continuous tuning during the operation of the control.

Proposed control strategy offers 4 distinct advantages compared to known control schemes including explicit model predictive control:

(1) implementation on embedded platforms (explicit nature of the NN-controller with evaluation time on a micro-second level),
(2) tunability of the controller in terms of varying weighing factors,
(3) input, and output constraint satisfactions with nearly optimal control performance,
(4) drastic reduction of memory requirements compared to explicit MPC.

We acknowledge, that such a control strategy does not enforce any stability clauses, and the total quality of the performance is largely determined by the quality of the training process of the NN-based controller. On the other hand, experimental results clearly support the validity of the proposed strategy. Furthermore, we open the range of applicability of NN-controllers to other fields, especially to processes with extremely fast dynamics.

2. CONTROL PROBLEM AND PRELIMINARIES

2.1 Control Problem

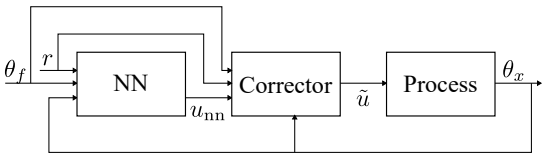
We aim to construct a two-step control strategy, involving a substitution of the MPC and a corrector term. The main part of the control strategy involves a neural network, which mimics the behavior of an MPC, hence its substitution. Then, a corrector term is designed, which ensures, that the joint control authority of the neural network and the corrector satisfy constraints imposed on the controlled process. The proposed control strategy is visualized in Fig. 1(a).

The controller in the form of a neural network is formally given by

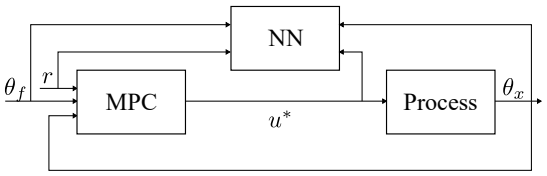
$$u_{\mathrm{nn}} = \mathcal{C}_{\mathrm{NN}}(\theta_{\mathrm{x}}, \theta_{\mathrm{f}}), \tag{1}$$

where u_{nn} denotes the control action provided by neural network controller, vector θ_{x} represents measurements, while the θ_{f} denotes the varying tuning factors. Concretely, we store diagonal values of the weighing factors in the vector θ_{f}. Specifics of the neural networks are presented in the Section 2.3 Then, an optimization-based corrector is synthesized as a copy of the MPC, later describe in Section 2.2. To ease the notation, lets formally enclose the corrector term into

$$\widetilde{u} = \mathcal{C}_{\mathrm{C}}(u_{\mathrm{nn}}, \theta_{\mathrm{f}}, \theta_{\mathrm{x}}). \tag{2}$$



(a) Closed loop with neural network and corrector controllers.



(b) Training of neural network controller.

Fig. 1. Integration of the closed-loop environment and scheme of the neural network training.

Note, that both controllers, i.e., $\mathcal{C}_{\mathrm{NN}}$ and $\mathcal{C}_{\mathrm{C}}$ are evaluated subsequently at each time instant.

Both steps of this control strategy heavily rely on the casted model predictive controller. The first step, the NN-based controller, imitates the MPC behavior with a near-optimal performance as an explicit function similar to explicit MPC. In the second step, the MPC solved with the random shooting in principle can arrive to a good nearly optimal control action, but we use the NN-based control action as the warm-start for the RS algorithm. We will show that such an initialization of the RS algorithm reduces the overall evaluation time of the control law and notably improves the performance of the RS implementation. Recall that random shooting has been successfully used to solve the model predictive control problems Dyer et al. (2002), but results presented here focus on *tunable* and *embeddable* aspects of the control strategy; hence the need for improved warm-starting is called for.

2.2 Model Predictive Control

We consider a reference tracking formulation of the model predictive controller with a discrete time state space model. The MPC is given as

$$\min \ ||x_N||_{Q_{\mathrm{N}}} + \sum_{k=0}^{N-1} ||y_k - r_k||_{Q_{\mathrm{y}}} + \sum_{k=0}^{N-1} ||\Delta u_k||_{Q_{\mathrm{du}}} \tag{3a}$$

$$\text{s.t.} \ \ x_{k+1} = Ax_k + Bu_k, \quad \forall k \in \mathbb{N}_0^{N-1}, \tag{3b}$$

$$y_k = Cx_k + Du_k, \quad \forall k \in \mathbb{N}_0^{N-1}, \tag{3c}$$

$$\Delta u_k = u_k - u_{k-1}, \quad \forall k \in \mathbb{N}_0^{N-1}, \tag{3d}$$

$$x_{k+1} \in \mathcal{X}, u_k \in \mathcal{U}, \Delta u_k \in \mathcal{U}_{\mathrm{du}}, \quad \forall k \in \mathbb{N}_0^{N-1}, \tag{3e}$$

$$x_0 = x(t), u_{-1} = u(t - T_{\mathrm{s}}). \tag{3f}$$

The objective function is in the quadratic form, where $||z||_Q = z^\intercal Q z$, with terminal penalty, penalization of control action deviations and control error in the term $(y_k - r_k)$, where the r_k denotes reference at given prediction step. All state variables are enforced for entire length of the prediction horizon. The formulation of the MPC in (3) ensures offset-free reference tracking given the

assumption that the prediction model in (3b), (3c) is identical to the controller process. Otherwise an extension with disturbance modeling is necessary to satisfy the offset free control performance (Pannocchia, 2003). The MPC is initialized with

$$\theta_{\mathrm{x}} = [x(t)^\intercal, u(t-T_{\mathrm{s}})^\intercal, r(t)^\intercal]^\intercal, \tag{4}$$

where $x(t)$ represents the state measurement and the $u(t-T_{\mathrm{s}})$ represent the previous control action.

The MPC presented in this section plays two roles in the manuscript. First, it is used to generate data for neural network controller, and second, it is used as as baseline for evaluating the control performance of proposed tunable strategy.

2.3 Neural Networks

We design the neural network such that it maps measurements and tuning factors $\theta = [\theta_{\mathrm{x}}^\intercal, \theta_{\mathrm{f}}^\intercal]^\intercal$ to the value of the control action u_{nn}, which should be as close as possible with the optimal control action provided by (3). The training of the neural network is posed as a regression problem over m data points. The results of the regression gives the optimal parameters of the neural network controller. The training procedure relies on solving the following optimization problem with minimization of the square error (MSE),

$$\min_{W,B} \quad \sum_{k=1}^{m} (u_{\mathrm{nn},k} - u_k^\star)^2 \tag{5a}$$

$$\text{s.t.} \quad u_{\mathrm{nn},k} = \mathcal{C}_{\mathrm{NN}}(\theta_k), \quad \forall k \in \mathbb{N}_1^m \tag{5b}$$

where the neural network model is given as a mesh of interconnected M neurons expressed as activation functions arranged in L layers. In our case we considered the rectified linear units (ReLU) as activation functions

$$\mathrm{ReLU}(v) = \max(0, w \cdot v + b), \tag{6}$$

where v stands for the input to the neuron from neurons in previous layers and coefficients w and b are optimized by (5). Note, that for convenience we stack all coefficients w and b for each neuron into an aggregated vector W and B respectively.

Training and testing data for the neural network construction are generated via repeatedly solving the optimal control problem (3) m-times for different values of initial conditions of parameters θ_{x} and θ_{f}. We considered equidistant gridding of the feasible input space to generate the input set of initial conditions. Even though issues with data generation are not directly related to this paper, we acknowledge that the equidistant approach is applicable only for systems with a low number of parameters. For a large-scale system, a set with reasonable spaced points in various combinations is necessary to cover (Liu and Bellet, 2019). Since the optimal input by the MPC is applied in the receding horizon fashion, we similarly structure the training data. The $i-$th row of the training set $\mathcal{D}$ consist of vector tuples given by particular value θ_{x}, θ_{f} and the optimal control action $u_0^\star$ associated with these particular values of input parameters θ. The process of learning an approximate controller (Karg and Lucia, 2021) is summarized in the following algorithm After the training, the local minimizer $W^\star, B^\star$ is found, and the resulting neural network in form of a controller $\mathcal{C}_{\mathrm{NN}}$ is only an approximation of the real MPC feedback law, which means that the solution is sub-optimal with a small approximation error. Due to the simplicity of the structure of the neural network, we do assume that there are minor constraints-violations presented.

Algorithm 1 Neural Network based MPC.

1: Design the MPC Controller (3)
2: **for** i=1$\ldots m$ **do**:
3: Solve MPC feedback law for set of initial values θ_{x}, θ_{f}.
4: Add the pair $(\theta_i, u_i^\star)$ to the dataset $\mathcal{D}$.
5: **end for**
6: Solve (5) with (L layers, M neurons) and obtain $W^\star$ and $B^\star$.
7: Validate MSE criterion from (5a), potentially repeat the step 6 with different L, M.
8: The approximate controller $\mathcal{C}_{\mathrm{NN}}$ is defined by $M, L, W^\star, B^\star$.

2.4 Random Shooting

The Random Shooting (RS) control strategy slightly differs from the standard optimal MPC control policy. Instead of performing the optimization process, only simple mathematical operations are involved in a control input evaluation. A large number of control scenarios are investigated while the control inputs are generated randomly. During the testing of individual scenarios, several factors influencing decision-making are considered, such as the value of the objective function the fulfillment of all constraints. It is based on a stochastic approach where no regularity conditions are required. Therefore, the performance index or model of the system is allowed to be non-linear, and constraints can be non-convex. Despite the apparent sub-optimality, there is a strong tendency to converge the result to the neighborhood of the optimum. This claim is supported not only by results presented in this manuscript but also in works by (Dyer et al., 2002; Piovesan and Tanner, 2009).

The detailed algorithm performed each sampling period to obtain the control input value is deeply described in (Bakaráč and Kvasnica, 2018) and is conceptually visualized in Fig. 2. Briefly, the algorithm consists of three main steps

(1) the control input sequence is randomly generated,
(2) the performance index along the whole prediction window is quantified and compared with so far the best one. If the current value of performance index J_{c} is lower, it is stored the best value J_{best},
(3) in addition to the evaluation of the performance index, the satisfaction of the constraints is also examined.

This procedure is cyclically repeated N_{RS} times. This parameter affects the quality of the result and the time required for the calculation. The higher value of the parameter N_{RS}, the better is the result and the longer the evaluation time.

In Section 2.3, the neural network has been used as the approximation of optimal MPC strategy. Despite all the

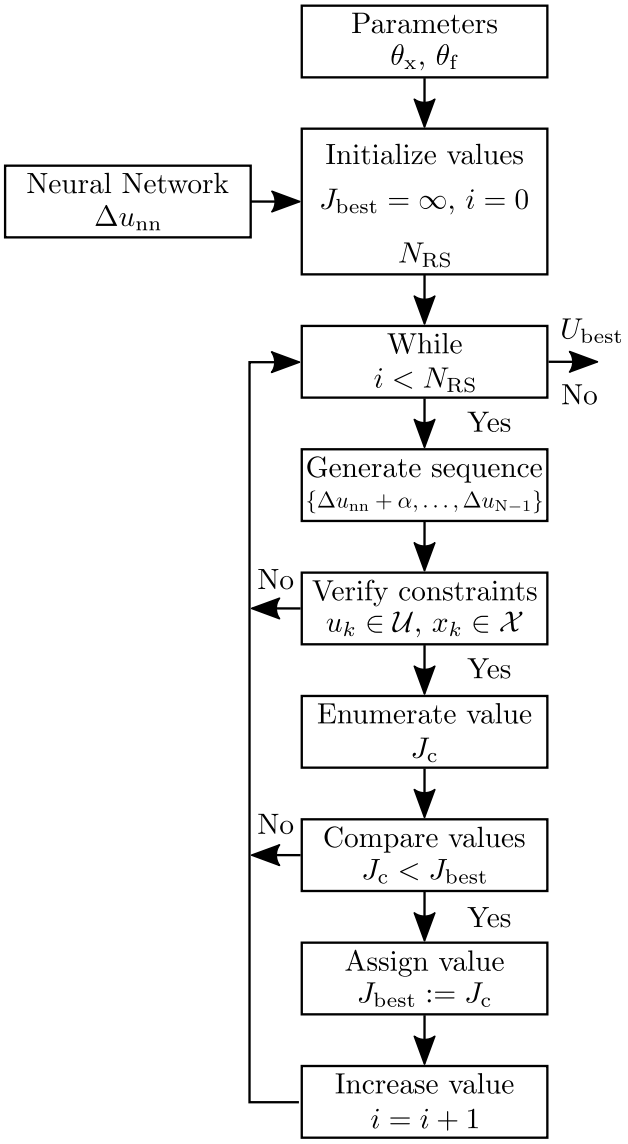


Fig. 2. Integration of NN-controller with and corrector solved by Random Shooting method.

benefits, the resulting control input sequence can lead the system behind the constraints and violate them. There is no guarantee of the constraints satisfaction. On the other hand, the neural network result is potentially closed to the optimal one. The RS strategy suffers from the increasing interval of control input values but provides a closed-loop stabilizing result which guarantees the satisfaction of the system constraints. Therefore, the combination of the; neural network and Random Shooting strategy is proposed. More specifically, the idea is to provide the output from the neural network as a kind of warm-start for RS strategy. By doing that, the probability of superior results is significantly increased. Moreover, the satisfaction of all system constraints is guaranteed.

2.5 Metrics for Control Performance Evaluation

The performance of proposed control strategy is evaluated with standardizes integral quality criteria, given by

$$\mathrm{IAE} = \frac{1}{T}\sum_{k=1}^{T_\mathrm{f}} |y_k - r_k| \tag{7}$$

in which T_f denotes the length of the simulation window, y_k represents the measurement at specific sampling instant k, and the r_k is the reference value at time instant k.

Next, we evaluate constraints violation by proposed near-optimal approximation methods. In this part of the performance evaluation, we focus on the number of samples, where the near-optimal strategy crosses the limits, and the amount of which the limits are crossed. This evaluation is focused on violation in the state variable x, and Δu signal. The actual control inputs u is not considered in the constraints violation investigation since we employ simple clipping prior to feeding the signal to the process.

3. SIMULATION RESULTS ON EMBEDDED PLATFORM

3.1 Comparisons of Control Performance

To demonstrate the control performance we considered a double integrator simulation model, with discrete time dynamics with $T_\mathrm{s} = 0.1$s,

$$x_{k+1} = \begin{bmatrix} 1 & T_\mathrm{s} \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} \frac{T_\mathrm{s}^2}{2} \\ T_\mathrm{s} \end{bmatrix} u_k, \tag{8a}$$

$$y_k = [1 \;\; 0]\,, \tag{8b}$$

subject to constraints

$$\begin{bmatrix} -5 \\ -5 \end{bmatrix} \le x_k \le \begin{bmatrix} 5 \\ 5 \end{bmatrix}, \tag{9a}$$

$$-1 \le u_k \le 1, \tag{9b}$$

$$-0.25 \le \Delta u_k \le 0.25. \tag{9c}$$

We are interested in reference tracking (i.e. position tracking) by solving the MPC problem in (3) with the prediction horizon $N = 20$, $Q_\mathrm{y} = I$ and the interval for $Q_\mathrm{du} \in [0.01, 200]$. To train this controller we have collected $5{\cdot}10^4$ data points. The NN controller $\mathcal{C}_\mathrm{NN}$ consists of $L = 3$ layers of $M = 30$ fully connected neurons. Neural network model was trained with randomly initialized weights using Adam optimizer (Kingma and Ba, 2014) with the learning rate of 0.001. The training was performed on $2 \cdot 10^3$ epochs with early stopping based on development set MSE (5a) as the loss function.

To demonstrate the online tunability of controllers and the ability to track the reference we choose the simulation scenario with varying reference r and varying weight matrix Q_du. The control scenario is as follows

- at time $t = 0\,\mathrm{s}$ $x_0 = [0, 0]^\intercal$,
- the reference r is switched periodically each 5 s between values 4 and -4,
- the weighting matrix Q_du is changed each 10 s from values $0.012 \to 1.2 \to 120$.

The simulation results obtained with the micro-controller implementation are presented in Fig. 3. Here, Fig. 3(a) and 3(b) shows the time profile of state variables. Recall that the first state x_1 is the controlled variable. The profiles are generated under the control authority (cf. 3(c)) of the (i) baseline MPC, (ii) pure NN-based controller, (iii)

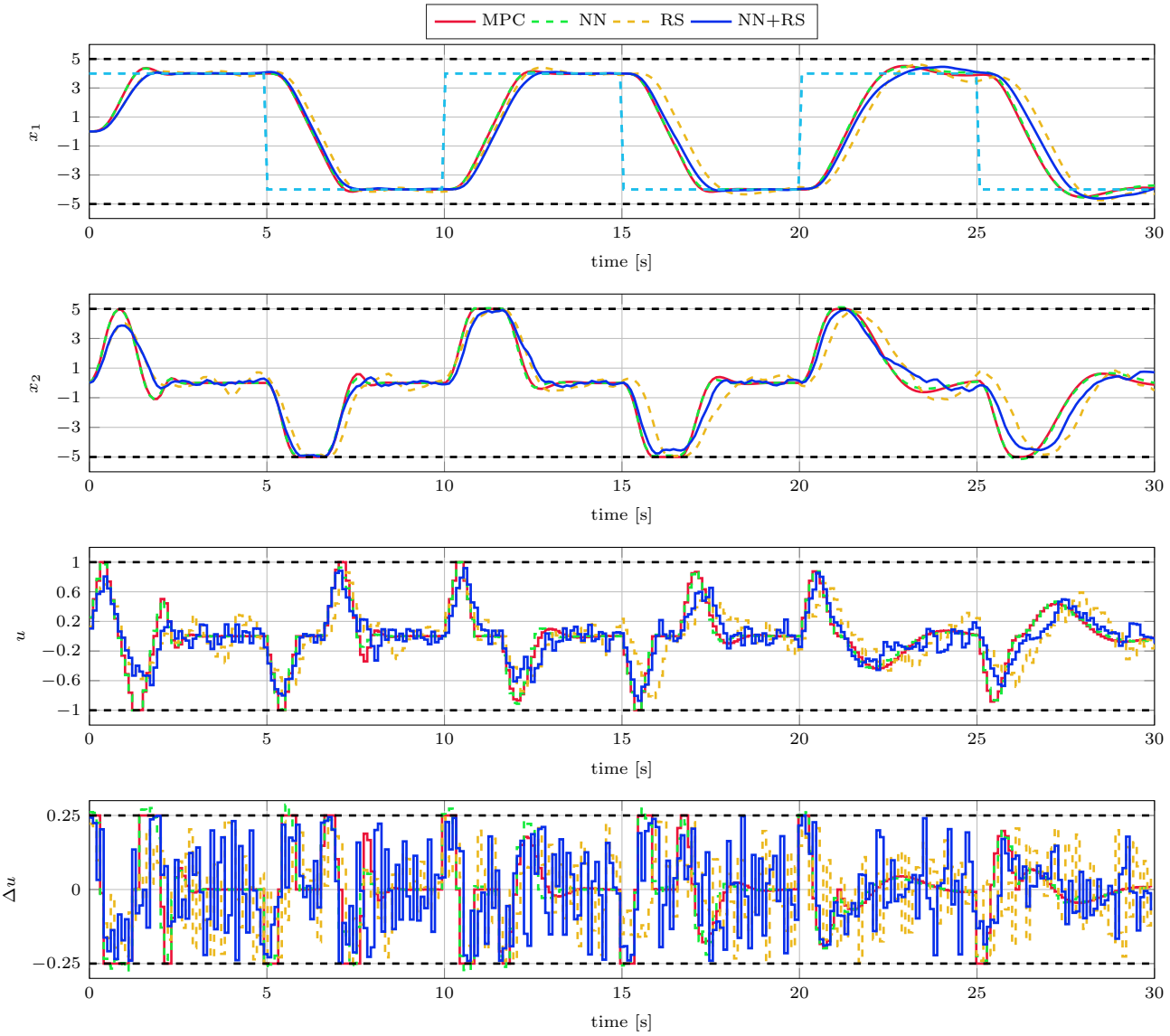


Fig. 3. Comparison of control performance of individual near-optimal strategies. Dashed blue line presents the reference, solid red covers the baseline MPC profiles, dashed green shows pure $\mathcal{C}_{\mathrm{NN}}$ controller, dashed yellow depicts the solution via random shooting, while the solid blue represents the joint strategy $\mathcal{C}_{\mathrm{NN}} \rightarrow \mathcal{C}_{\mathrm{C}}$.

pure $\mathcal{C}_{\mathrm{C}}$ controller, and (iv) by joint strategy $\mathcal{C}_{\mathrm{NN}} \rightarrow \mathcal{C}_{\mathrm{C}}$. We can observe that all three versions of approximated controllers track the reference very closely while respecting the constraints in general. Minor violations are observed with a pure NN-based controller. Note that even such performance might be acceptable given the fact that the evaluation time of $\mathcal{C}_{\mathrm{NN}}$ requires only less than 1 millisecond (time evaluation comparison is presented in Section 3.2). On the other hand, considering a pure RS solution to the MPC results in worsening the tracking performance. The main benefit of the RS method is that it ensures the constraints satisfaction by compromising on the performance. To improve on that performance, we direct the reader to the performance of the joint strategy, where the warm-starting by $\mathcal{C}_{\mathrm{NN}}$, actually moves the performance of the pure RS method almost as close to the pure $\mathcal{C}_{\mathrm{NN}}$, which actually tracks the optimal MPC solution.

The numerical evaluation of the performance criteria shows that we gain extremely fast control law evaluation by considering pure NN-based controller, with less than 1.5% of performance deterioration and slight constraints violations. If we consider the pure RS approach, we gain constraint satisfaction, but the performance deteriorates even further, up to 25%. However, by joining these two approaches, we improve the evaluation time, we enforce constraint satisfaction, and we improve the overall control performance. The total performance degradation in this joint case is less than 12%. Specific values are reported in the table 1. Moreover, this table summarizes the number of violated constraints on a given signal, denoted by $\mathcal{N}(\cdot)$, and maximum value of constraints violation $\mathcal{V}_{\mathrm{m}}(\cdot)$.

3.2 Implementation Aspects

The control algorithms have been implemented on the embedded hardware to illustrate the practical aspects of the

Table 1. Qualitative evaluation of control performance.

	IAE (7)	constraints violation			
		$\mathcal{N}(x_2)$	$\mathcal{V}_{\mathrm{m}}(x_2)$	$\mathcal{N}(\Delta u)$	$\mathcal{V}_{\mathrm{m}}(\Delta u)$
MPC	574	0	0	0	0
NN	580	19	0.140	29	0.036
RS	717	0	0	0	0
RS+NN	647	0	0	0	0

Table 2. Comparison of evaluation times. All reported values are in ms.

	t_{w}	t_{b}	t_{m}	t_{a}
Pure NN-controller	0.142	0.131	0.131	0.132
Pure RS-controller	303.509	7.417	23.255	39.798
RS warm-started with NN	142.296	1.288	11.556	17.132

proposed strategy. The objective of the experiment is to prove that both control approaches, both $\mathcal{C}_{\mathrm{NN}}$ and $\mathcal{C}_{\mathrm{C}}$, are easily deployable on hardware with limited computational resources without compromising the control performance. The micro-control unit chosen for experimental tests is a 32-bit microprocessor *ESP32*. It is equipped with 520 KB of RAM memory that is used to store the internal variables and 4 MB of flash memory designed to store the source code of algorithms and libraries. The clock frequency of the chip is 240 MHz.

The particular implementation of the *random shooting* algorithm consists only of simple mathematical operations thus the compilation of the source code did not require any additional libraries. The parameters N_{RS} is set to 50.

To successfully deploy the neural network controller, $\mathcal{C}_{\mathrm{NN}}$, in the micro-controller unit, the EloquentTinyML [1] library has been added into the source code. Once the warm-starting was provided to the $\mathcal{C}_{\mathrm{C}}$, we managed to decrease $N_{\mathrm{RS}} = 10$. Hence the overall time to arrive at the value of the control input was drastically reduced.

The time required to evaluate the control input has been measured for both control algorithms separately and for the joint strategy as well. Specifically we evaluated the worst-case time t_{w}, best-evaluation time t_{b}, most common evaluation time t_{m}, and average time of control action evaluation t_{a}. Particular times required for control action evaluation are reported in the Table 2.

4. CONCLUSION

We proposed a novel strategy to formulate an approximated embeddable near-optimal controller with on-the-fly tunable capabilities. The near-optimal behavior is enforced by constructing a neural network trained from optimal control actions generated by the model predictive controller. Since the training procedure does not guarantee constraint satisfaction with the pure NN-based controller in operation, a control action corrector has been designed to mitigate the violations. The corrector term is solved with a random shooting algorithm. The entire implementation of the control strategy is free of optimization and relies only on mere function evaluations. The control strategy was implemented on embedded hardware with limited computational and memory resources. Experimental results with micro-controller implementation demonstrate that the combination of trained neural network and corrector solved with random shooting provides a minimal decrease in the optimality of the control performance. On the other hand, it allows for continuous changes in weighting factors, guarantees both input and output constraints satisfaction, and the entire evaluation of the control law requires less than a few milliseconds.

[1] https://github.com/eloquentarduino/EloquentTinyML/

REFERENCES

Bakaráč, P. and Kvasnica, M. (2018). Fast nonlinear model predictive control of a chemical reactor: a random shooting approach. *Acta Chimica Slovaca*, 11(2), 175–181.

Dyer, M., Kannan, R., and Stougie, L. (2002). A simple randomised algorithm for convex optimisation. *Mathematical Programming*, 147. doi:10.1007/s10107-013-0718-0.

Holaza, J., Takács, B., Kvasnica, M., and Cairano, S.D. (2015). Nearly optimal simple explicit mpc controllers with stability and feasibility guarantees. *Optimal Control Applications and Methods*, 35(6). doi:10.1002/oca.2131.

Karg, B. and Lucia, S. (2021). Model predictive control for the internet of things. *Recent Advances in Model Predictive Control: Theory, Algorithms, and Applications*, 485, 165.

Karg, B. and Lucia, S. (2018). Deep learning-based embedded mixed-integer model predictive control. In *2018 European Control Conference (ECC)*, 2075–2080. doi:10.23919/ECC.2018.8550234.

Karg, B. and Lucia, S. (2020). Efficient representation and approximation of model predictive control laws via deep learning. *IEEE Transactions on Cybernetics*, 50(9), 3866–3878. doi:10.1109/tcyb.2020.2999556.

Kingma, D.P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kiš, K., Klaučo, M., and Mészáros, A. (2020). Neural network controllers in chemical technologies. In *2020 IEEE 15th International Conference of System of Systems Engineering*, 397–402. IEEE. doi:10.1109/SoSE50414.2020.9130425.

Kvasnica, M., Gondhalekar, R., and Fikar, M. (2012). A hierarchical design methodology for implementing safety-critical constrained controllers with guaranteed stability and failure detection. In *IEEE Conference on Decision and Control*, 1214–1219. Maui, Hawaii.

Kvasnica, M., Hledík, J., Rauová, I., and Fikar, M. (2013). Complexity reduction of explicit model predictive control via separation. *Automatica*, 49(6), 1776–1781. doi:10.1016/j.automatica.2013.02.018.

Liu, K. and Bellet, A. (2019). Escaping the curse of dimensionality in similarity learning: Efficient frank-wolfe algorithm and generalization bounds. *Neurocomputing*, 333, 185–199.

Lohr, Y., Klaučo, M., Kalúz, M., and Mönnigmann, M. (2019). Mimicking predictive control with neural networks in domestic heating systems. In M. Fikar and M. Kvasnica (eds.), *Proceedings of the 22nd International Conference on Process Control*, 19–24. Slovak University of Technology in Bratislava, Slovak Chemical Library, Štrbské Pleso, Slovakia.

Lucia, S., Navarro, D., Karg, B., Sarnago, H., and Lucía, s. (2021). Deep learning-based model predictive control for resonant power converters. *IEEE Transactions on Industrial Informatics*, 17(1), 409–420. doi:10.1109/TII.2020.2969729.

Pannocchia, G. (2003). Robust disturbance modeling for model predictive control with application to multivariable ill-conditioned processes. *Journal of Process Control*, 13(8), 693 – 701.

Piovesan, J.L. and Tanner, H.G. (2009). Randomized model predictive control for robot navigation. In *2009 IEEE International Conference on Robotics and Automation*, 94–99. doi:10.1109/ROBOT.2009.5152468.

Wojsznis, W., Gudaz, J., Blevins, T., and Mehta, A. (2003). Practical approach to tuning MPC. *ISA Transactions*, 42(1), 149–162. doi:https://doi.org/10.1016/S0019-0578(07)60121-9.