Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Deterministic Global Nonlinear Model Predictive Control with Neural Networks Embedded

Danimir T. Doncevic [**,*] **Artur M. Schweidtmann** [*]
Yannic Vaupel [*] **Pascal Schäfer** [*] **Adrian Caspari** [*]
Alexander Mitsos [***,*,**]

[*] *Aachener Verfahrenstechnik - Process Systems Engineering, RWTH Aachen University, Aachen, Germany (e-mail: amitsos@alum.mit.edu).*
[**] *Institute of Energy and Climate Research - Energy Systems Engineering (IEK-10), Forschungszentrum Jülich GmbH, Wilhelm-Johnen-Straße, 52425 Jülich, Germany.*
[***] *JARA - Jülich Aachen Research Alliance, Center for Simulation and Data Science*

Abstract: Nonlinear model predictive control requires the solution of nonlinear programs with potentially multiple local solutions. Here, deterministic global optimization can guarantee to find a global optimum. However, its application is currently severely limited by computational cost and requires further developments in problem formulation, optimization solvers, and computing architectures. In this work, we propose a reduced-space formulation for the global optimization of problems with recurrent neural networks (RNN) embedded, based on our recent work on feed-forward artificial neural networks embedded. The method reduces the dimensionality of the optimization problem significantly, lowering the computational cost. We implement the NMPC problem in our open-source solver MAiNGO and solve it using parallel computing on 40 cores. We demonstrate real-time capability for the illustrative van de Vusse CSTR case study. We further propose two alternatives to reduce computational time: i) reformulate the RNN model by exposing a selected state variable to the optimizer; ii) replace the RNN with a neural multi-model. In our numerical case studies each proposal results in a reduction of computational time by an order of magnitude.

Keywords: Nonlinear process control; Model predictive and optimization-based control; Global optimization; Recurrent neural networks; Neural networks in process control.

1. INTRODUCTION

Dynamic operation of processes is becoming increasingly relevant as the shift to renewable energy supply and feedstocks introduces additional variability and motivates the use of nonlinear models (Mitsos et al. (2018)). For this, data-driven models constitute a popular choice due to recent advances in machine learning and increasing availability of process data (Lee et al. (2018)).

Among the variety of data-driven models, artificial neural networks (ANN) are very promising for control due to their capability to learn highly nonlinear multivariate relationships. For instance, they can constitute the nonlinear block of a Hammerstein-Wiener process model (e.g., Ławryńczuk (2013)) or be used to learn the control law in explicit MPC (e.g., Lucia and Karg (2018)). RNNs (Werbos (1988)) are a special type of ANN capable to represent sequential data thanks to preserving internal memory by feeding back the network outputs to the input layer. As such, they are suited to learn the input-output behavior of dynamic systems. In literature RNNs are commonly applied to model a plant and embedded in MPC. Wu et al. (2019) provide some mathematical background on MPC with RNNs regarding the stability of the controller assuming bounded prediction errors of the network.

As the activation function of RNNs is usually nonlinear, their use in MPC leads to nonlinear model predictive control (NMPC) problems. NMPC problems potentially exhibit multiple local optima. Finding the global optimum is thus desired. In contrast to local and stochastic global solvers, only deterministic global optimization methods can guarantee to identify global optima up to arbitrary tolerance within finite time. In previous literature, global dynamic optimization was examined, e.g., by Chachuat et al. (2006), but NMPC problems have rarely been solved with deterministic global optimization methods: Sriniwas and Arkun (1997) have proposed global NMPC for a system with two inputs and two outputs governed by a polynomial auto-regressive model with exogenous inputs (ARX) and solved the embedded problems with a primal-relaxed dual approach. Long et al. (2006) and Čižniar et al. (2008) examined global NMPC of the isothermal van de Vusse CSTR, a SISO system. The authors solved the resulting nonlinear nonconvex programs (NLP) using a branch-and-bound algorithm. Long et al. (2007) applied global NMPC to a system with hybrid models leading to mixed-integer

Copyright lies with the authors

NLPs and solved them with a decomposition-based algorithm. Finally, Wang et al. (2017) suggested global solution of NLPs within NMPC by normalized multiparametric disaggregation and applied the method to two variants of the van de Vusse CSTR. These studies found the computational effort of global NMPC to be prohibitive for real time applications to larger systems. Further, the longest horizons employed were a control horizon of 2 with a prediction horizon of 5 or a control horizon of 1 with a prediction horizon of 30 respectively. The highest number of manipulated variables used was 2. The authors reported solution times in the order of seconds.

A speed-up of global NMPC would facilitate the utilization of longer control and prediction horizons and the application to bigger systems. In order to achieve this speed-up, we propose a deterministic global NMPC framework based on RNN process models. In our previous research, we have demonstrated that feed-forward ANNs can be efficiently optimized using a reduced-space formulation (Schweidtmann and Mitsos (2019)).

In this work, we extend the reduced-space optimization to RNNs and solve the NMPC problem with our open-source deterministic global solver MAiNGO (Bongartz et al. (2018)). We put emphasis on improving the computational cost of global optimization in NMPC with surrogate models embedded. In particular, we train RNNs on simulated process data and implement the NMPC controller with the RNN as a system model. Then, we propose a neural multi-model (Ławryńczuk and Tatjewski (2010)) as an alternative surrogate model. It is a collection of ANNs, where each ANN predicts one output of a sequence of outputs. We compare the solution time for global optimization of a test problem using the two surrogate models. Finally, we show that the computational cost of the problem using the RNN model can further be reduced by a reformulation of the problem, i.e., the exposition of a state variable to the optimizer. The remainder of this article is structured as follows: Section 2 introduces the isothermal van de Vusse case study. Section 3 describes the methods used for data generation, training of neural networks, and the implementation of the NMPC controller. The results of the *in silico* study are discussed in Section 4. First, the efficacy of the controller is validated and then the computational cost of applying global optimization is analyzed. Finally, Section 5 draws a conclusion.

2. CASE STUDY

In this work, we study the isothermal van de Vusse CSTR (Kravaris and Daoutidis (1990)). Herein, a reactant A is converted into product B with side reactions taking place. The CSTR is described by the following ODE-system:

$$\frac{dC_A}{dt} = -k_1 C_A - k_3 C_A^2 + \frac{\dot{V}}{V_r}\left(C_{A0} - C_A\right), \tag{1a}$$

$$\frac{dC_B}{dt} = k_1 C_A - k_2 C_B - \frac{\dot{V}}{V_r} C_B. \tag{1b}$$

The system exhibits input multiplicity and inverse response making its control a challenging task. The manipulated variable $\mathbf{u}$ is the dimensionless flow rate $\dot{V}/V_r$ and the controlled variable $\mathbf{y}$ corresponds to the concentration of product B, C_B. The reaction rates are $k_1 = 50\ \mathrm{h}^{-1}$ for the main reaction, and $k_2 = 100\ \mathrm{h}^{-1}$, $k_3 = 10\ \mathrm{L/mol \cdot h}^{-1}$, for a first-order and second-order side reaction respectively. The feed concentration of substrate is $C_{A0} = 10\ \mathrm{mol}/L$. Initially, the CSTR is at steady-state with $\dot{V}/V_r = 180\ \mathrm{h}^{-1}$, $C_A = 6.1706$ mol/L, and $C_B = 1.1019$ mol/L. Full state feedback is assumed and no delays are taken into account. The control objective is to drive C_B along a given setpoint trajectory $\mathbf{y^{ref}}$. Ultimately, a discretized optimal control problem (OCP) of the following form has to be solved in every control step:

$$\min_{\mathbf{u_k}} \quad \sum_{k=1}^{N_p} \left(\mathbf{y_k} - \mathbf{y_k^{ref}}\right)^{\mathbf{T}} \mathbf{Q_k} \left(\mathbf{y_k} - \mathbf{y_k^{ref}}\right) + \sum_{k=0}^{N_u-1} \mathbf{\Delta u_k}^{\mathbf{T}} \mathbf{R_k} \mathbf{\Delta u_k} \tag{2a}$$

$$\text{s.t.} \quad \mathbf{y_k} = \mathbf{f}\left(\mathbf{y_{k-1}}, \ldots, \mathbf{y_{k-L_y}}, \mathbf{u_{k-1}}, \ldots, \mathbf{u_{k-L_u}}\right), \quad \forall k = 1, \ldots, N_p, \tag{2b}$$

$$\mathbf{u^L} \preceq \mathbf{u_k} \preceq \mathbf{u^U}, \qquad \forall k = 0, \ldots, N_u - 1, \tag{2c}$$

$$\mathbf{y_0} = \mathbf{y_{init}}. \tag{2d}$$

The system equations (2b) model the plant output as an explicit function of the previous L_u inputs and the previous L_y outputs of the system. The manipulated variable is restricted by the following bounds: $10 \leq \dot{V}/V_r \leq 200$. N_u and N_p denote the control and prediction horizons respectively.

3. METHODOLOGY

In this section, the implementation of the NMPC controller and the employed optimizer are described. Then, the data generation and subsequent training of RNNs is outlined. For clarity, we distinguish the output predicted by the surrogate model, $\hat{y}$, from the actual plant output y. Furthermore, it is now assumed that k is the fixed current timestep and m denotes a sample within the prediction horizon: $m \in \{1, \ldots, N_p\}$. An overline denotes the full sequence of controls or states within the prediction horizon, e.g., $\overline{\mathbf{u}}_\mathbf{k} = \{\mathbf{u_{k+m}}\}$ with $m = \{1, \ldots, N_p\}$.

3.1 NMPC Framework

Fig. 1 shows the developed global NMPC framework. At each sample k, the optimizer computes an optimal control sequence $\overline{\mathbf{u}}_\mathbf{k}^\star$. The plant model (Equation (2b)) is represented by a RNN, predicting a sequence of future states $\hat{\overline{\mathbf{y}}}_\mathbf{k+1}$. The first control move of the optimal sequence, $\mathbf{u_k^\star}$, is applied to the controlled system in each sample of the controller. Then, k is increased by 1.

The sequencing of the presented NMPC framework is implemented in Python. The optimizer MAiNGO is written in C++. The controlled system is simulated in Dymola with default integrator settings.

3.2 Optimizer

MAiNGO performs deterministic global optimization via a branch-and-bound method using McCormick relaxations in a reduced-space (McCormick (1976); Mitsos et al. (2009); Bongartz and Mitsos (2017)). This formulation

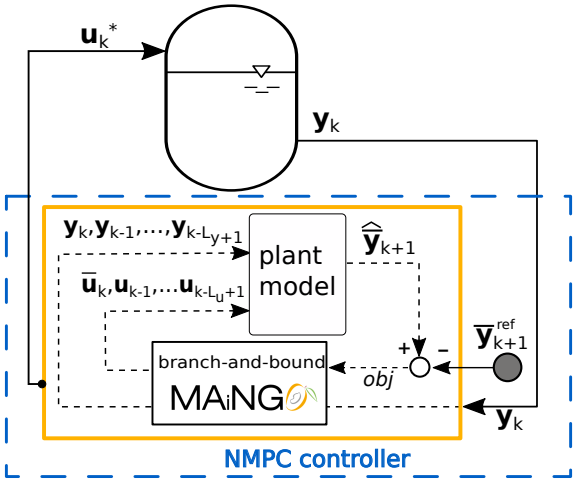


Fig. 1. Implemented NMPC framework.

inserts all equality constraints describing the RNN, see Equation (3), into the objective function and propagates the convex relaxations through these explicit functions (Schweidtmann and Mitsos (2019)). This eliminates intermediate state variables and equality constraints from the optimization problem reducing its dimensionality significantly. Only the discretized controls, assumed piecewise constant, remain as decision variables for the optimizer. As shown in our previous work (e.g., Schweidtmann and Mitsos (2019)), the reduced-space formulation can significantly speed up global optimization with neural networks embedded. However, the propagation of relaxations through such complex expressions can also lead to weak relaxations.

In contrast, the full-space formulation is most commonly used for global optimization. Here, all intermediate state variables and equality constraints are visible to the branch-and-bound solvers. The full-space formulation leads to larger problem sizes but can potentially also lead to tighter relaxations. In some studies, the most efficient problem formulation is a hybrid between the two formulations described above (e.g., Bongartz and Mitsos (2019)). Therefore, we also consider a hybrid formulation by exposing a state variable at timestep m_v as a decision variable ξ to the optimizer. This is achieved by adding ξ to the decision variables vector and linking it with the output $\hat{\mathbf{y}}_{\mathbf{m_v}}$ via the simple equality constraint, i.e., $\hat{\mathbf{y}}_{\mathbf{m_v}} = \xi$. In a similar approach, Diedam and Sager (2018) exploited additional state variables for global optimal control.

As an additional advantage of MAiNGO, it can solve problems in parallel using a master/slave approach. We solve all problems on an Intel Skylake Platinum 8160 with 40 cores in parallel with 2.10 GHz each. The computation time in Section 4 is nonetheless always given as the corresponding time for a single core, i.e., summed computational time over all cores.

3.3 Data Generation

We generate a data set of 60,000 samples for the input $\dot{V}/V_r$ and output C_B. The training data set is generated by simulating the dynamic model of the CSTR with a hybrid input sequence. This technique is frequently used to obtain data of a dynamic system (e.g., Lightbody and Irwin (1997)) and described in the following: The input sequence is designed by superposition of a low-frequency signal with high-frequency noise. The low-frequent signal is constructed by appending points sampled from the input domain of the plant, which is defined by the manipulated variables and their bounds. The transitions between step changes are slow enough to allow the plant to settle. Thus, this signal explores the response behaviour with regard to step changes within the full input domain of the plant. It is overlaid by high-frequent noise, which has a much smaller amplitude, exciting the dynamics of the plant within each region. An exemplary hybrid input sequence is depicted in Fig. 2. The neural networks trained on this data learn the discrete input-output behavior of the system. Some of their properties are thus set by the input sequence: The discretization timestep of an ANN, trained to predict future states of a system based on this data, is bounded by the sampling time of the noise. Further, the system behavior is learned with a piece-wise constant control profile.

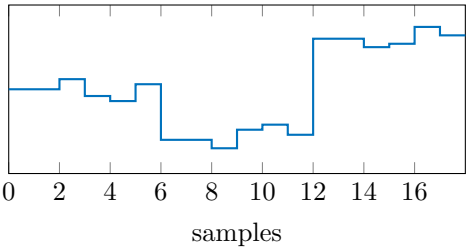


Fig. 2. Exemplary excerpt from a hybrid input sequence: Step changes occur at samples 6 and 12. Noise is applied after each step change. See, e.g., samples 8–12.

3.4 Neural Network Training

The RNN used herein is characterized by feeding back its own lagged outputs to the network input vector together with external input values. It is thus able to learn the system equations (2b). In particular, we use two lags of the input and output each to learn the following time-explicit mapping of the predicted output at timestep $k+m$, $\hat{\mathbf{y}}_{\mathbf{k+m}}$:

$$\hat{\mathbf{y}}_{\mathbf{k+m}} = \mathbf{f}\left(\mathbf{y}_{\mathbf{k+m-1}}, \mathbf{y}_{\mathbf{k+m-2}}, \mathbf{u}_{\mathbf{k+m-1}}, \mathbf{u}_{\mathbf{k+m-2}}\right) \quad (3)$$

This constitutes a nonlinear auto-regressive model with external inputs (NARX). During the training of RNNs, the past plant outputs $\mathbf{y}_{\mathbf{k+m-1}}$ and $\mathbf{y}_{\mathbf{k+m-2}}$ are known at each sample and used as past feedback outputs. The trained RNN is employed as the plant model for NMPC, i.e., the system equation (2b) in Problem (2) is substituted by Equation (3). The prediction sequence $\hat{\mathbf{y}}_{\mathbf{k+1}}$ is generated by successive evaluation of the trained RNN model. The predicted output is propagated through the future timesteps. Thus, for $m > 2$, the past RNN predictions $\hat{\mathbf{y}}_{\mathbf{k+m-1}}$ and $\hat{\mathbf{y}}_{\mathbf{k+m-2}}$ are used in Equation (3) instead of the past plant outputs. This introduces nested functions, the depth of which increases with each additional timestep. We further train an alternative surrogate model structure: a neural multi-model Ławryńczuk and Tatjewski (2010). Here, independent ANNs are trained for each sampled timestep k of the prediction horizon. Thus, no predicted outputs $\hat{\mathbf{y}}$ have to be reused for consecutive predictions.

Instead, only the initial system state $\mathbf{y_k}$ and all control moves made during the prediction horizon are taken into account as the network input. We expect that the latter approach yields tighter relaxations for the predicted output. The following mapping now results for $\hat{\mathbf{y}}_{\mathbf{k+m}}$:

$$\hat{\mathbf{y}}_{\mathbf{k+m}} = \mathbf{f}\left(\mathbf{y_k}, \mathbf{u_{k+m-1}}, \mathbf{u_{k+m-2}}, \ldots, \mathbf{u_k}\right) \tag{4}$$

Note that in contrast to Equation (3), the size of the input vector of ANNs in Equation (4) grows with m.
All networks are trained using MATLAB. RNNs with one or two hidden layers are used. We train the RNN with different hidden layer sizes, given in Table 1. Also, we test both, the tanh function and the identity function, as output layer activation function. As training algorithms we apply the Bayesian Regularization backpropagation (BR) algorithm or the Levenberg-Marquardt (LM) algorithm with early stopping. In total, 200 RNNs are trained on the generated data set. 40 different configurations result from combination of the parameters given in Table 1 and each configuration is trained 5 times.

Table 1. Parameters used for the training of RNNs.

hidden layer size	{[15 15], [10 10], [7 7], [5 5] [25], [15], [10], [7], [5] }
training algorithm	{LM, BR}
output activation function	{tanh, identity function}

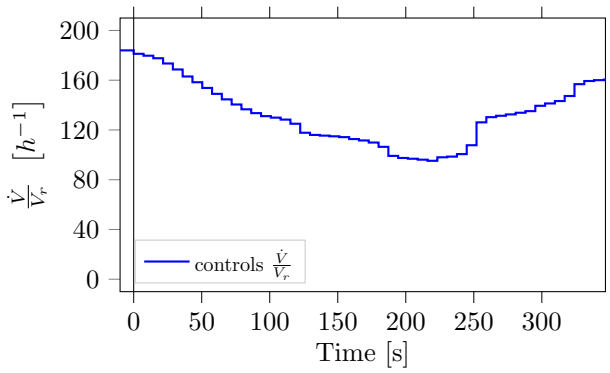
For the ANNs of the neural multi-model, we always use two hidden layers with 7 neurons each and the identity function as output layer activation function. The ANNs are larger in size than the smallest RNNs used to model the system. The choice is made to accommodate the growing input vector associated with longer prediction horizons in Equation (4). Training is performed with the LM algorithm together with early stopping. The same data set as for the RNNs is used. All ANNs forming the neural multi-model exhibit a high accuracy in the training data set.
The implementation of RNNs for MAiNGO that is used in this work can be found in our tool MeLOn (Schweidtmann et al. (2020)).

4. RESULTS & DISCUSSION

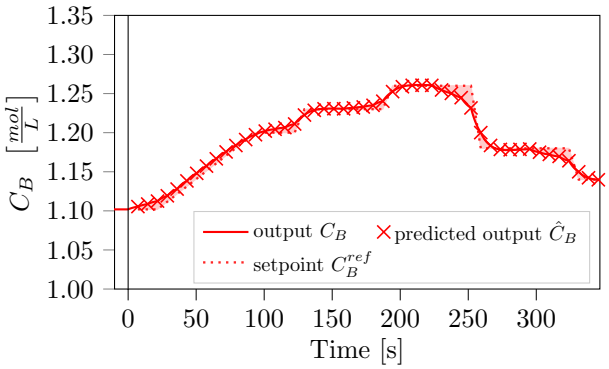
This section presents the results of our study. First, the closed-loop performance of the controller is demonstrated. Then, the computational performance of the proposed framework is analyzed.

4.1 Controller Performance

Fig. 3 shows a closed-loop simulation of the implemented controller. The sampling time is chosen as 7.2 s in accordance with the work of Čižniar et al. (2008) and Wang et al. (2017). The RNN used for this specific simulation has one hidden layer with 5 neurons and identity function in the output layer and was trained with the LM algorithm. It is selected because, of all RNNs trained, it exhibits the fastest solution times on an illustrative test problem, i.e., a single step of the closed-loop simulation. It should be noted, that all trained RNNs were able to accurately predict the process dynamics with the mean squared error of predictions varying between the order of 10^{-7} and 10^{-9}.



(a) Manipulated variable $\frac{\dot{V}}{V_r}$.



(b) Controlled variable C_B: The crosses indicate the one-step-ahead prediction of the RNN employed. The predictions made by the RNN are accurate and the controlled variable follows the setpoint.

Fig. 3. Closed-loop simulation of the controller.

Thus, they also lead to effectively the same controller behaviour.
The controller is simulated over 50 timesteps and the control sequence is computed by deterministic global optimization in each timestep. As a convergence criterion, the relative optimality gap ϵ_{rel} is set to 0.01. This threshold is within the accuracy of the RNN and chosen conservatively to obtain accurate optimality guarantees. The prediction and control horizons are set to $N_u = 2$ and $N_p = 5$ respectively. As depicted in Fig. 3, the controller is able to track the reference trajectory of the controlled variable with a slight tracking error. Furthermore, Fig. 3 demonstrates that the one-step-ahead predictions of the RNN are accurate and display a mean squared error in the order of 10^{-8}. This observation is in good agreement with the accuracy measured during training of the RNNs. The proposed controller computes the control sequence in less than 0.421 CPU seconds for each sample, while Wang et al. (2017) reported a maximum solution time of 3.452 CPU seconds for the isothermal van de Vusse case study with identical horizons. It should be noted, that in most but not all instances herein, the global optimum is also found by IPOPT (Wächter and Biegler (2006)).

4.2 Scaling of Computational Performance

In this section, we study the computational cost of solving a NMPC problem with different neural network models and varying control and prediction horizons to global optimality. First, we examine the 200 RNNs trained with the parameters in Table 1. As a test problem, we consider

the nested problem of the first timestep of the previously shown closed-loop controller simulation and solve it globally. We increase the prediction horizon to 8 and elevate the relative optimality gap for convergence to 0.1 for this particular study, because we observe that the separation between the RNNs is clearer for a longer prediction horizon. The 200 RNNs trained differ substantially in computational performance. The mean solution time of the problem accumulated over all 40 cores is 12,408 CPU seconds with a standard deviation of 7,110 CPU seconds. Yet, the best-performing RNN, described in the previous paragraph, results in a solution time of 0.919 CPU seconds on this problem. The large variation is caused by tightness of the McCormick relaxations of the RNN output, which are strongly influenced by the network weight distribution. A detailed analysis is not within the scope of this contribution. Note, that the best-performing RNN is one of the smallest RNNs trained, as smaller networks tend to yield more tractable global optimization problems.

RNN performance is compared with the neural multi-model in the following. We use the same test problem but allow varying control and prediction horizons. Fig. 4 depicts the resulting development of the computational cost, comparing the RNN surrogate model and the neural multi-model. An increase in horizon length is accompanied by a higher solution time. For the control horizon this is probably caused by an increase of decision variables. With the exception of a control horizons of 1, the computational cost for the neural multi-model scales more favorably with an increasing prediction horizon than it does for the RNN. The resulting difference in computational cost for the two different models manifests itself clearly at about a prediction horizon of 12. For the longest prediction horizon considered, 20, the computational cost is reduced by two orders of magnitude when using the neural multi-model over the RNN. The higher computational effort that is caused by a longer prediction horizon originates from the weakened McCormick relaxations of the network output for multi-step-ahead prediction. This behavior was also observed for implied state bounds of nonlinear ODEs for global dynamic optimization by Singer and Barton (2006). Our results indicate that the neural multi-model suffers less from this effect than the RNN. For short prediction horizons, the neural multi-model exhibits slightly longer computational times than the RNN in our study. We believe this occurs because the networks used therein are larger than the RNN.

4.3 Optimization with an Exposed State Variable

We further explore the exposition of a state variable to the optimizer for the RNN model, that was discussed as an alternative problem formulation in Section 3.2. As a benchmark, we set $N_p = 15$ and select control horizons $N_u = \{1, 2, 3, 4, 5\}$. For each of those, we vary the placement m_v of the additional exposed variable between N_u and $N_p - 1$. We compare the resulting computational times with the times previously recorded for reduced-space formulation at $N_u = \{1, 2, 3, 4, 5\}$ and $N_p = 15$ in Figure 4. Table 2 gives the maximum reduction of CPU time achievable by this adjustment and the corresponding timestep m_v^*, at which the exposed variable is placed. For control horizons bigger than 1 a significant speed-up of the optimization is achieved by this technique. It is in the same order of magnitude as that achieved by using the neural multi-model. Our results favor exposing the state variable at the 6-th timestep of the prediction horizon for the considered problem.

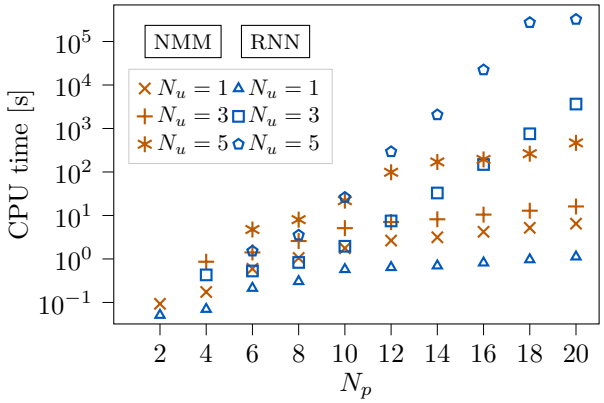


Fig. 4. Computational cost of solving a test problem for increasing control and prediction horizons N_u and N_p. The CPU time is given for a single core. A comparison between the RNN surrogate model (RNN) and the neural mutli-model (NMM) is made.

The results of the subsection show that the exposition of a state variable indeed reduces the CPU time required to solve the examined problem. In future endeavors, the technique has to be employed to problems with varying prediction horizons. Further, multiple exposed state variables should be tested.

Table 2. Reduction in CPU time for a benchmark OCP with $N_p = 15$ by exposing a state variable to the solver at timestep m_v^*.

N_u	m_v^*	CPU time reduction [–]
1	–	–
2	6	60.3%
3	5	91.3%
4	6	96.7%
5	6	98.4%

5. CONCLUSION

We propose the first deterministic global NMPC framework based on RNNs and the first one based on a neural multi-model. The neural network models learn the discrete dynamic input-output behavior of a plant from simulation data. The controller is validated on an illustrative van de Vusse CSTR case study. We apply global NMPC in real-time.

A comparison of computational times for two network structures shows that the neural multi-model leads to a faster convergence of global optimization with long prediction horizons than the RNN model. However, we demonstrate that a similar reduction of computational cost can be achieved for the RNN model by exposition of a state variable to the optimizer. Both enhancements facilitate global NMPC for bigger control and prediction horizons than those previously used for the van de Vusse CSTR in literature. If additional speed-up by MAiNGO's parallel computing facilities is taken into account, the real-time capability threshold of the controller can be further

elevated. In the examined van de Vusse case study, 40 cores allow to solve NMPC problems with a control horizon of 4 and a prediction horizon of 20 in the order of seconds. In the future, a training algorithm for the construction of RNNs and ANNs with tight McCormick relaxations subject to a pre-defined level of accuracy is desired. Finally, the application of the controller to modern MPC formulations like economic MPC and to case studies of larger scale is desirable.

ACKNOWLEDGEMENTS

Funded by the Excellence Initiative of the German federal and state governments. The authors gratefully acknowledge the financial support of the Kopernikus project SynErgie by the Federal Ministry of Education and Research (BMBF) and the project supervision by the project management organization Projektträger Jülich. This work was partially performed as part of the Helmholtz School for Data Science in Life, Earth and Energy (HDS-LEE) and received funding from the Helmholtz Association of German Research Centres. The authors further thank Dominik Bongartz, Jaromił Najman, and Susanne Sass for their work on MAiNGO. Simulations were performed with computing resources granted by RWTH Aachen University under projects rwth0422 and thes0612. Finally, we thank the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

Bongartz, D. and Mitsos, A. (2017). Deterministic global optimization of process flowsheets in a reduced space using mccormick relaxations. *Journal of Global Optimization*, 69(4), 761–796.

Bongartz, D. and Mitsos, A. (2019). Deterministic global flowsheet optimization: Between equation-oriented and sequential-modular methods. *AIChE Journal*, 65(3), 1022–1034.

Bongartz, D., Najman, J., Sass, S., and Mitsos, A. (2018). Maingo: Mccormick-based algorithm for mixed-integer nonlinear global optimization.

Chachuat, B., Singer, A.B., and Barton, P.I. (2006). Global methods for dynamic optimization and mixed-integer dynamic optimization. *Industrial & Engineering Chemistry Research*, 45(25), 8373–8392.

Čižniar, M., Fikar, M., and Latifi, M.A. (2008). Design of constrained nonlinear model predictive control based on global optimisation. In *18th European Symposium on Computer Aided Process Engineering*, volume 25 of *Computer Aided Chemical Engineering*, 563–568.

Diedam, H. and Sager, S. (2018). Global optimal control with the direct multiple shooting method. *Optimal Control Applications and Methods*, 39(2), 449–470.

Kravaris, C. and Daoutidis, P. (1990). Nonlinear state feedback control of second-order nonminimum-phase nonlinear systems. *Computers & Chemical Engineering*, 14(4–5), 439–449.

Ławryńczuk, M. (2013). Practical nonlinear predictive control algorithms for neural wiener models. *Journal of Process Control*, 23(5), 696–714.

Ławryńczuk, M. and Tatjewski, P. (2010). Nonlinear predictive control based on neural multi-models. *International Journal of Applied Mathematics and Computer Science*, 20(1), 7–21.

Lee, J.H., Shin, J., and Realff, M.J. (2018). Machine learning: Overview of the recent progresses and implications for the process systems engineering field. *Computers & Chemical Engineering*, 114, 111–121.

Lightbody, G. and Irwin, G.W. (1997). Nonlinear control structures based on embedded neural system models. *IEEE transactions on neural networks*, 8(3), 553–567.

Long, C.E., Polisetty, P.K., and Gatzke, E.P. (2006). Nonlinear model predictive control using deterministic global optimization. *Journal of Process Control*, 16(6), 635–643.

Long, C.E., Polisetty, P.K., and Gatzke, E.P. (2007). Deterministic global optimization for nonlinear model predictive control of hybrid dynamic systems. *International Journal of Robust and Nonlinear Control*, 17(13), 1232–1250.

Lucia, S. and Karg, B. (2018). A deep learning-based approach to robust nonlinear model predictive control. *IFAC-PapersOnLine*, 51(20), 511–516.

McCormick, G.P. (1976). Computability of global solutions to factorable nonconvex programs: Part i — convex underestimating problems. *Mathematical Programming*, 10(1), 147–175.

Mitsos, A., Asprion, N., Floudas, C.A., Bortz, M., Baldea, M., Bonvin, D., Caspari, A., and Schäfer, P. (2018). Challenges in process optimization for new feedstocks and energy sources. *Computers & Chemical Engineering*, 113, 209–221.

Mitsos, A., Chachuat, B., and Barton, P.I. (2009). Mccormick-based relaxations of algorithms. *SIAM Journal on Optimization*, 20(2), 573–601.

Schweidtmann, A.M. and Mitsos, A. (2019). Deterministic global optimization with artificial neural networks embedded. *Journal of Optimization Theory and Applications*, 180(3), 925–948.

Schweidtmann, A.M., Netze, L., and Mitsos, A. (2020). Melon: Machine learning models for optimization. `https://git.rwth-aachen.de/avt.svt/public/MeLOn/`.

Singer, A.B. and Barton, P.I. (2006). Bounding the solutions of parameter dependent nonlinear ordinary differential equations. *SIAM Journal of Scientific Computing*, 27(6), 2167– 2182.

Sriniwas, R.G. and Arkun, Y. (1997). A global solution to the nonlinear model predictive control algorithms using polynomial arx models. *Computers & Chemical Engineering*, 21(4), 431–439.

Wächter, A. and Biegler, L.T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1), 25–57.

Wang, X., Mahalec, V., and Qian, F. (2017). Globally optimal nonlinear model predictive control based on multi-parametric disaggregation. *Journal of Process Control*, 52, 1–13.

Werbos, P.J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4), 339–356.

Wu, Z., Tran, A., Rincon, D., and Christofides, P.D. (2019). Machine learning–based predictive control of nonlinear processes. part i: Theory. *AIChE Journal*.