2018 26th Mediterranean Conference on Control and Automation (MED)
Zadar, Croatia, June 19-22, 2018

Adaptive One-Step Model Predictive Control using Lyapunov Theory-Based Deep Neural Networks

Plakias Spyridon[1] and Yiannis S. Boutalis[1]

Abstract— **Neural network model predictive control has been widely used in the field of automatic control. However, the off-line training of the neural predictor and the appearance of disturbances and parameter variations, during the control phase, affect the system output by inducing steady state error. This paper tries to overcome this problem by proposing an adaptive deep neural network model predictive controller. To ensure the convergence of the adaptive process, we use Lyapunov stability theory, which guides the updating of the weights of the deep neural network. Likewise, a Lyapunov based algorithm guides the updating of the control signal of the one-step model predictive controller. Simulation results for two cases demonstrate the effectiveness of the proposed control scheme.**

I. Introduction

Model Predictive Control (MPC) is considered to be an advanced technique with several industrial applications in the field of automatic control over the last decades [1], [2]. The core of MPC strategy is a mathematical model, which describes the dynamic controlled process. MPC techniques produce control actions by solving on-line an optimization problem with constraints, over a finite prediction horizon. The objective function of the optimization problem relies on the mathematical model of the system. Hence, the performance of the MPC controller depends on the reliability of the model of the controlled system which acts as a predictor.

On the other hand, neural networks have been extensively used for the identification and modeling of unknown systems with uncertainties, due to their impressive approximation ability [3]. So, a straightforward strategy is to combine them with MPC, and use neural network model predictive control(NNMPC). There are a lot of NNMPC applications in the literature. A control scheme for the multi-variable nonlinear steel pickling process has been built using NNMPC in [4]. An application of single input single output Neural Generalized MPC for three-joint robotic manipulator with a cubic trajectory and random disturbances is presented in [5]. Also, in [6] the robust control of a pH neutralization process is applied using NNMPC.

In most cases, the training of the neural network as identifier of the controlled system is done off-line. In such cases, parameters variations and uncertainties during the control process affect the output, causing the increase of the tracking error. In this paper, we use the straightforward solution of on-line training and therefore correction of the neural predictor, compensating in this way the systems variations and uncertainties.

Recently, Deep Learning (DL) Architectures are hot topic with great success in the fields of computer vision, natural language processing, automatic speech and audio recognition, machine translation and others [7], [8]. A deep neural network (DNN) consists of multiple nonlinear hidden layers, achieving hierarchical representation of learning at each layer. So, DNNs learn more abstract relationships within the training data and obtain better generalization. In this work, we use a deep neural network, as a non-linear dynamic identifier, to exploit their powerful representation ability.

The stability of the adaptive on-line training of the deep neural model predictor is ensured by using a back-propagation algorithm based on Lyapunov stability theory [9], [10]. At first, we linearize the neural model locally by Taylor series expansion. Then, we use as Lyapunov candidate $V(k)$ an energy function, that depends on the square norm of the tracking error. At each time step we update the weights of the neural network in such a way that handles singularities and guarantees that the term $\Delta V(k)$ is negative. The mathematical analysis of the convergence of the neural weights, using the proposed update law with singularities, is included.

Also, it is important to mention that the DNN predictor is linearized only during when it is updated. In this way, we achieve to ensure the local stability of the process around the current operating point and at the same time we use the capability of the DNN to express nonlinear systems.

In NNMPC techniques, the updated control signal is calculated by the minimization of an objective function over a prediction horizon with respect the input signal at each time step. One advantage of the above optimization problem is the ability to set constraints for control states and inputs. On the other hand, the intense computational effort for solving the problem is a drawback of the MPC technique. Also, stability issues may be presented in applications of NNMPC, using the above technique.

In this paper, the control signal at next time step is calculated using one step model predictive control combined with the theory of Lyapunov stability. The procedure is similar with the one that used at on-line updating of the weight vector. Firstly, the output of the DNN predictor is linearized around the previous control signal and subsequently we calculate the next one by maintaining the Lyapunov stability condition for the selected energy function. The contribution of this paper is described by the above briefly analysis.

The rest of the paper is organized as follows. In section

[1] S. Plakias and Y. Boutalis are with the Department of Electrical and Computer Engineering, Democritus University of Thrace, 67100 Xanthi, Greece splakias@ee.duth.gr, ybout@ee.duth.gr

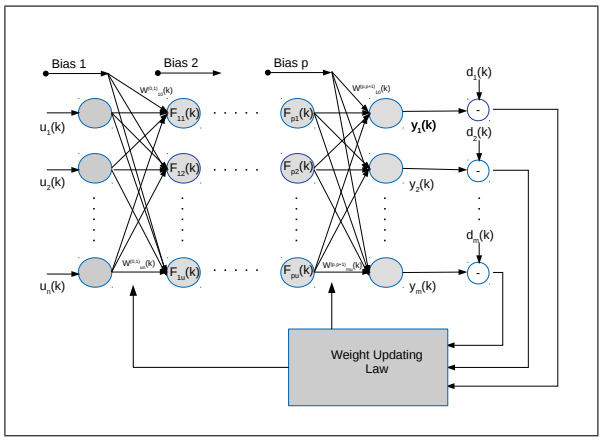

Fig. 1. Architecture of Deep Neural Network

II we describe the architecture of deep neural predictor and its linear approximation around the current estimate of the weight vector. The proposed updating law, which is based on Lyapunov stability theory and the proof of the convergence of the training process is discussed and presented in section II. Section IV describes the model predictive control scheme and formulates the calculation of next time step control signal. In section V we demonstrate the efficiency of the proposed scheme with two simulation examples and finally we have the conclusions and future work.

II. Formulation of Deep Neural Network & Linearization

Figure 1 illustrates the architecture of the conventional deep neural network (DNN) with n inputs, m outputs and p hidden layers. The output vector of each hidden layer $F_i(k), i = 1..p$ and the output vector $Y(k)$ of the deep neural network at time step k are calculated by the following equations

$$\begin{aligned} F_1(k) &= f(W^{(0,1)}(k)U(k)) \\ F_i(k) &= f(W^{(i-1,i)}(k)F_i(k-1)), i = 1,..,p \\ Y(k) &= f(W^{p,p+1}(k)F_p(k)) \end{aligned} \tag{1}$$

where $f()$ is the nonlinear activation function, $U(k) \in \Re^n$ is the input of the system and $W^{(i-1,i)}$ the weight matrix between layer $i-1$ and layer i.

We define as $\theta(k)$ the vector that holds all the DNN weights at time step k. Also, we consider the DNN system as a nonlinear mapping $G(.)$ from $\Re^n$ to $\Re^m$ as

$$Y(k) = G(U(k), \theta(k)). \tag{2}$$

Lets assume that for input $U(k)$, the desired output vector is $Y_d(k)$ and that the estimate weight vector at time $k-1$ is $\hat{\theta}(k-1)$. Then, the goal of on-line training of the DNN is the update of the estimate weight vector in such a way that the actual output of the system to be closer to the target output.

Firstly, we linearize the DNN system [11] by expanding the vector function $G(U(k), \theta(k))$ into a Taylor series around the current estimate weight vector $\hat{\theta}(k-1)$ as

$$\begin{aligned} Y(k) =& G(U(k), \hat{\theta}(k-1)) + \\ & J_G(\theta(k))|_{\theta(k)=\hat{\theta}(k-1)}(\theta(k) - \hat{\theta}(k-1)) + \xi(k) \end{aligned} \tag{3}$$

where $J_G(\theta(k))$ is the Jacobian matrix of function $G(.)$ with respect the vector $\theta(k)$ and $\xi(k)$ is the higher order term. The term $J_G(\theta(k))|_{\theta(k)=\hat{\theta}(k-1)}$ is an observation matrix and is depended on the current estimate of the weights.

Then, the output $Y(k)$ and the linearized output $\tilde{Y}(K)$ of the DNN system are expressed as

$$\begin{aligned} Y(K) =& H_k(k)\theta(k) + \eta(k) + \xi(k) \\ \tilde{Y}(K) =& H_k(k)\theta(k) \end{aligned} \tag{4}$$

where

$$\begin{aligned} H_k &= J_G(\theta(k))|_{\theta(k)=\hat{\theta}(k-1)} \\ \eta(k) &= G(U(k), \hat{\theta}(k-1)) - H_k\hat{\theta}(k-1) \end{aligned} \tag{5}$$

and can be calculated at time step k given the current estimate of the weight vector $\hat{\theta}(k-1)$.

The calculation of the matrix H_k is straightforward using the chain rule and equations (1) and is omitted in this paper. By linearizing the neural system, we achieve mathematical simplicity of the model. Thus, it is easy to update the weights of the system, so that we ensure the stability of the linear DNS in the sense of Lyapunov.

III. Proposed Update Law & Proof of Convergence

The estimation error $e(k)$ of the linear DNN is

$$e(k) = \tilde{Y}(K) - D(k) = H_k\theta(k) - Y_d(k) \tag{6}$$

where matrix H_k is given by equation (5). Also, we denote as $a(t)$ the a priori estimation error of the system, which is calculated by the following equation

$$\alpha(t) = H_k\hat{\theta}(k-1) - Y_d(k). \tag{7}$$

The following theorem provides the updating law for the determination of the DNN weights.

Theorem 1: The update law of the weight vector $\theta(k)$, which guarantees the Lyapunov stability during on-line training, is expressed as

$$\Delta\theta(k) = \begin{cases} -H_k^T(H_kH_k^T + \lambda I)^{-1}\zeta(k) \text{ if } \det H_kH_k^T = 0 \\ -H_k^T(H_kH_k^T)^{-1}\zeta(k) \text{ otherwise} \end{cases} \tag{8}$$

where $\lambda > \left\|H_kH_k^T\right\|$ and signal $\zeta(k)$ is calculated by the following formula

$$\zeta(k) = \alpha(k) - \beta^{-k/2}e(k-1). \tag{9}$$

Also, we denote $\varepsilon = \left\|H_kH_k^T\right\|$ and $\gamma = \frac{\lambda}{\lambda-\varepsilon}$.

Proof: The Lyapunov function for the linear DNN predictor is selected as

$$V(k) = \beta^k \|e(k)\|^2 = \beta^k e^T(k)e(k) \tag{10}$$

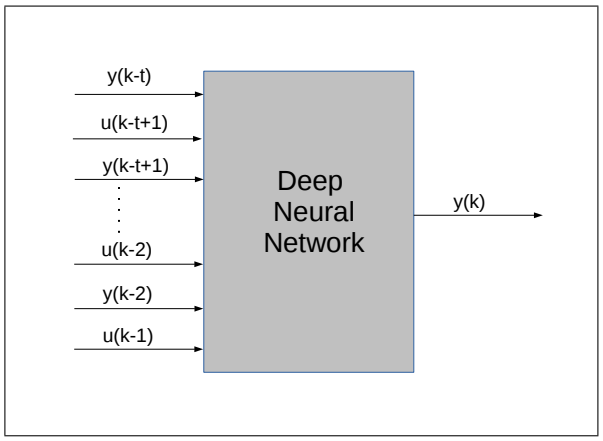

Fig. 2. Input-Output Mapping of Deep Neural Network

where $e(k)$ is given by equation (6) and constant $\beta > 1$. Further, from equation (6) and (7) we have

$$\begin{aligned} e(k) =& H_k(\Delta\theta(k) + \hat{\theta}(k-1)) - Y_d(k) \\ =& \alpha(k) + H_k\Delta\theta(k) \end{aligned} \tag{11}$$

In case that $H_k H_k^T$ is nonsingular ($\det H_k H_k^T \neq 0$), by using the corresponding update law in (8), equation (11) becomes

$$e(k) = \alpha(k) - \zeta(k) = \beta^{-k/2} e(k-1). \tag{12}$$

Furthermore, using equations (10) and (12) we obtain

$$\begin{aligned} \Delta V(k) &= V(k) - V(k-1) \\ &= \beta^k \|e(k)\|^2 - \beta^{k-1} \|e(k-1)\|^2 \\ &= \beta^k \left\| \beta^{-k/2} e(k-1) \right\|^2 - \beta^{k-1} \|e(k-1)\|^2 \\ &= (1-\beta^{k-1}) \|e(k-1)\|^2 < 0 \end{aligned} \tag{13}$$

for constant $\beta > 1$. So, in case that $H_k H_k^T$ is nonsingular, the error will asymptotically converge to zero according to Lyapunov stability theorem. The parameter β acts as error convergence rate and must be $\beta \to 1^+$.

Similarly, from equation (11) and for the case of a singular $H_k H_k^T$ we get

$$\begin{aligned} e(k) &= a(k) - H_k H_k^T (H_k H_k^T + \lambda I)^{-1} \zeta(k) \\ &= a(k) - \zeta(k) + \lambda I (H_k H_k^T + \lambda I)^{-1} \zeta(k) \\ &= a(k) - \zeta(k) + \lambda (H_k H_k^T + \lambda I)^{-1} \zeta(k). \end{aligned} \tag{14}$$

Also, we have from the last equation using the triangle inequality for vectors

$$\begin{aligned} \|e(k)\| &\leqslant \|a(k) - \zeta(k)\| + \lambda \left\| (H_k H_k^T + \lambda I)^{-1} \right\| \|\zeta(k)\| \\ &\leqslant \|a(k) - \zeta(k)\| + \left\| (\frac{1}{\lambda} H_k H_k^T + I)^{-1} \right\| \|\zeta(k)\|. \end{aligned} \tag{15}$$

When $\frac{1}{\lambda} \left\| H_k H_k^T \right\| < 1$, using the inequality of the Appendix, we conclude that

$$\left\| (\frac{1}{\lambda} H_k H_k^T + I)^{-1} \right\| < \frac{1}{1 - \frac{1}{\lambda} \left\| H_k H_k^T \right\|}. \tag{16}$$

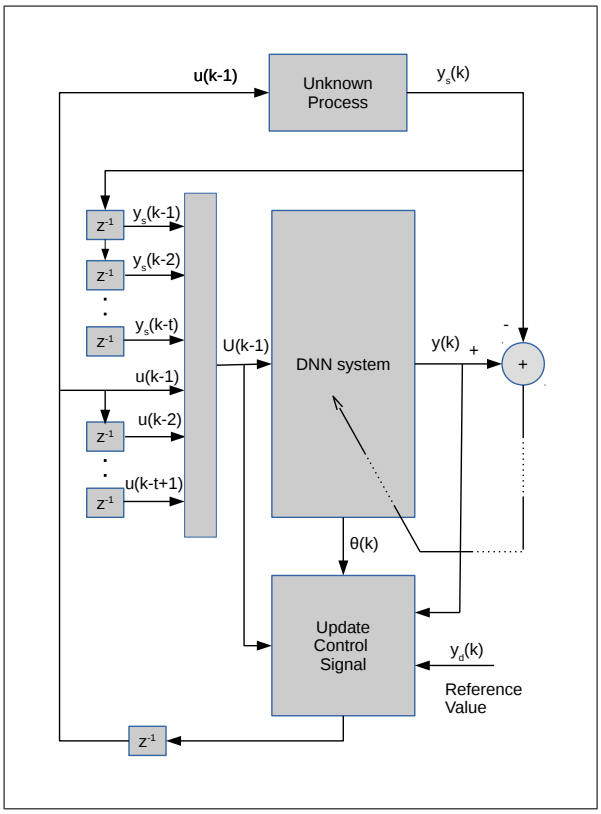

Fig. 3. Architecture of Model Predictive Control System

Combining (15) and (16) we obtain

$$\begin{aligned} \|e(k)\| &\leqslant \|a(k) - \zeta(k)\| + \frac{1}{1 - \frac{1}{\lambda} \left\| H_k H_k^T \right\|} \|\zeta(k)\| \\ &\leqslant \|a(k) - \zeta(k)\| + \frac{\lambda}{\lambda - \varepsilon} \|\zeta(k)\| \\ &\leqslant \left\| \beta^{-k/2} e(k-1) \right\| + \gamma \left\| \alpha(k) - \beta^{-k/2} e(k-1) \right\| \\ &\leqslant \beta^{-k/2} \|e(k-1)\| + \gamma(\|\alpha(k)\| + \beta^{-k/2} \|e(k-1)\|) \\ &\leqslant \beta^{-k/2} (\gamma+1) \|e(k-1)\|) + \gamma \|\alpha(k)\|. \end{aligned} \tag{17}$$

Finally, by using inequality (19) and with simple algebra we get

$$\Delta V(k) \leqslant \Delta \bar{V}(k) \tag{18}$$

where

$$\begin{aligned} \Delta \bar{V}(k) =& ((\gamma+1)^2 - \beta^{k-1}) \|e(k-1)\|^2 + \beta^k \gamma^2 \|\alpha(k)\| \\ &+ 2\beta^{k/2} \gamma(\gamma+1) \|\alpha(k)\| \|e(k-1)\|. \end{aligned} \tag{19}$$

For large values of k, $\Delta \bar{V}(k)$ is a concave down parabolic function and has two roots, $r_1 = -\frac{\beta^{k/2}\gamma\|\alpha(k)\|}{(\gamma+1)+\beta^{(k-1)/2}} < 0$ and $r_2 = \frac{\beta^{k/2}\gamma\|\alpha(k)\|}{\beta^{(k-1)/2}-(\gamma+1)} > 0$. When $\|e(k-1)\| > r_2$ we have that $\Delta V(k) \leqslant \Delta \bar{V}(k) < 0$. Finally, in this case of singularity the error will converge inside the attractive ball with center at the origin of the error space and radius r_2. ■

So, at each time step we update the weight vector $\theta(k)$ of the DNN, ensuring the stability in the sense of Lyapunov and achieving to model the unknown dynamics of the process.

IV. Model Predictive Control Scheme & Calculation of Control Signal

In order to identify the unknown system, nonlinear autoregressive exogenous modeling (NARX) is used [12]. The input vector of the DNN is composed as

$$U(k-1) = [y(k-p)^T, u(k-p+1)^T, y(k-p+1)^T, .., u(k-1)^T]^T \quad (20)$$

where $u(k-p)$ and $y(k-p)$ are vectors that represent the past inputs and outputs of the system at time step $k-p$. So, the DNN model tries to predict the output of the system at the next time step, using past input-output information (figure 2).

The architecture of the one-step model predictive controller is shown in figure 3. During the control phase, the DNN system is trained based on Lyapunov theory, using the error between the actual output of the unknown system and the output of the DNN. So, we identify the unknown process by the on-line adaptation of the DNN, ensuring at the same time the convergence of the weight vector.

Also, at each time step we have to update the control signal based on the identifier of the unknown process, the past known input-output information of the system and the desired reference output. Similarly, we linearize the DNN predictor using Taylor series but now around the previous control signal $\bar{U}(k-1)$. During the calculation of the next control signal, we consider that the weight vector of the DNN remains constant. Further, we update the control signal ensuring that the error between the output of the DNN predictor and the desired output will converge to zero according to Lyapunov theory of stability. Therefore, we are based on the ability of the DNN identifier to model the unknown dynamics of the system.

The expansion of the DNN model into a Taylor series around the previous control signal $\bar{U}(k-1)$ is presented by the equation

$$\begin{aligned} Y(k) =& G(\bar{U}(k-1), \theta(k)) + \\ & J_G(U(k))|_{U(k)=\bar{U}(k-1)}(U(k)-\bar{U}(k-1)) + \tau(k) \end{aligned} \quad (21)$$

where $J_G(U(k))$ is the Jacobian matrix of function $G(.)$ with respect the vector $U(k)$ and $\tau(k)$ is the higher order term, which is considered as zero in the following mathematical analysis.

We assume that for small sampling period, the reference outputs between two successive time steps are equal

$$Y_d(k) = Y_d(k-1). \quad (22)$$

By using equations (21) and (22), the difference between the output of the DNN system and the desired output of the system at time step k become

$$\begin{aligned} e_c(k) =& Y(k) - Y_d(k) = G(\bar{U}(k-1), \theta(k)) + \\ & J_G(U(k))|_{U(k)=\bar{U}(k-1)}(U(k)-\bar{U}(k-1)) - Y_d(k) \\ =& Y(k-1) - Y_d(k-1) + H_U \Delta U(k) \\ =& e_c(k-1) + H_U \Delta U(k) \end{aligned} \quad (23)$$

where $H_U = J_G(U(k))|_{U(k)=\bar{U}(k-1)}$.

Similarly, we select as candidate Lyapunov function the energy function $V_c(k) = \beta_c^k \|e_c(k)\|^2$ with $\beta_c \to 1^+$. If we set

$$H_U \Delta U(k) = (\beta_c^{-k/2} - 1) e(k-1) \quad (24)$$

then the difference $\Delta V_c(k)$ becomes

$$\begin{aligned} \Delta V_c(k) =& V_c(k) - V_c(k-1) \\ =& (1-\beta_c^k) \|e_c(k-1)\|^2 < 0. \end{aligned} \quad (25)$$

So, we guarantee the system convergence in the sense of Lyapunov stability by calculating the next control signal based on equation (24).

Furthermore, if we denote as $U_c(k)$ the input of DNN system at time step k, we have

$$\Delta V_c(k) = U_c(k) - \bar{U}(k-1). \quad (26)$$

and so from equation (25) we obtain

$$H_U U_c(k) = (1-\beta_c^k) \|e_c(k-1)\|^2 + H_U \bar{U}(k-1) = \bar{C}. \quad (27)$$

The input $U_c(k)$ of DNN system at time step k has the form

$$U_c(k) = [\tilde{U}(k-1)^T, u_{mpc}(k)^T]^T, \quad (28)$$

where vector $\tilde{U}(k-1)$ contains the known past input-output information of the system and $u_{mpc}(k)$ denotes the updated control signal. From the last equation we have

$$H_U[:, -l:] u_{mpc}(k) = \bar{C} - H_U[:, :-l] \tilde{U}(k-1) = \bar{D}, \quad (29)$$

where l is the length of control input vector. The notation $H_U[:, -l:]$ indicates that from the matrix H_U we take the last l columns.

Finally, we estimate the control signal at next time step as

$$u_{mpc}(k) = (H_{UD}^T H_{UD} + \mu I)^{-1} H_{UD}^T \bar{D}, \quad (30)$$

where $H_{UD} = H_U[:, -l:]$ and μ is a small positive constant. At the same time, by using the above equation, we handle singularities.

One problem that arises is how to take into consideration the constraints of control signals. We handle the above problem by selecting the simplest solution, with the projection of the control signal into the feasible region.

V. Simulations Cases

We demonstrate the feasibility of the proposed scheme in two simulation cases. We build a DNN with 5 hidden layers, to exploit the powerful representation of DL, in both of them. Also, each hidden layer consists of 50 nodes and we select the rectified linear unit ($Relu, f(z) = max(z, 0)$) as the nonlinear activation function of the neural predictor. The sampling period, during on-line training, is equal to 0.1 sec and the parameter β equals 1.1 for both cases.

At an early stage of the control process, the rate of the change of the control signal is affected proportionally by the value of the parameter β_c, when β_c is close to 1. So, due to the absence of off-line training of the identifier and to the low level of on-line training initially, we use smaller value for the parameter β_c. In that way, we don't influence the

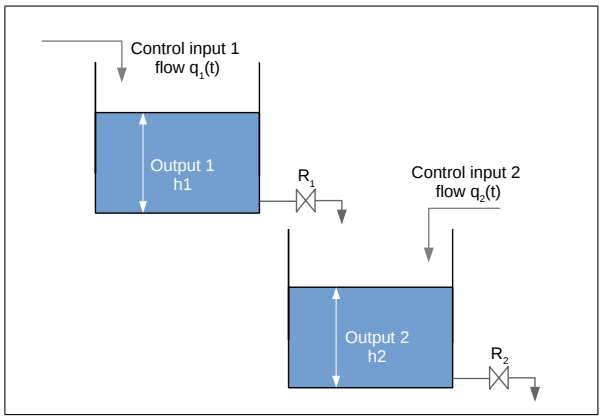

Fig. 4. Non-interacting liquid level control system

TABLE I

LIQUID LEVEL CONTROL SYSTEM PARAMETERS

Parameter	Description	Value
A_1	Cross Sectional Area of Tank 1	25 cm^2
A_2	Cross Sectional Area of Tank 2	100 cm^2
R_1	Linear flow resistance through valve 1	0.4 sec/cm^2
R_2	Linear flow resistance through valve 2	0.5 sec/cm^2
q_{max}	Maximum flow rate through pumps	20 cm^3/sec
q_{min}	Minimum flow rate through pumps	-20 cm^3/sec

output of the system with a negative impact on the initial stages of on-line training.

We use the python library of theano [13] to build the simulation code and to accelerate the simulations using parallel computing of GPU. Off-line training is passed over and the initial weights of the DNN systems are selected randomly for both cases. Also, in both cases the input of the neural identifier consists from past input-output system information of length 3 ($\tau = 3$).

A. Simulation Case 1

The two tank non-interacting level control system is presented in figure 4. The inputs of the system are the fluid flaw rates $q_1(t)$ and $q_2(t)$. Also, the outputs are the heights h_1 and h_2 of the tanks. The dynamics of the system can be described by the following state-space equations [14]:

$$\begin{aligned} \frac{dh_1}{dt} &= -\frac{\sqrt{h_1}}{A_1 R_1} + \frac{1}{A_1} q_1(t) \\ \frac{dh_2}{dt} &= \frac{\sqrt{h_1}}{A_2 R_1} - \frac{\sqrt{h_2}}{A_2 R_2} + \frac{1}{A_2} q_2(t) \end{aligned} \tag{31}$$

where the physical meaning and the values are shown in table I.

The parameter β_c that affects the update of the control signal is set to 1.001. Simulation results are shown in figure 5, where we observe that the heights of both tanks are close to the desired values for each testing case. Therefore, the proposed control scheme can move the system to any desired operating point.

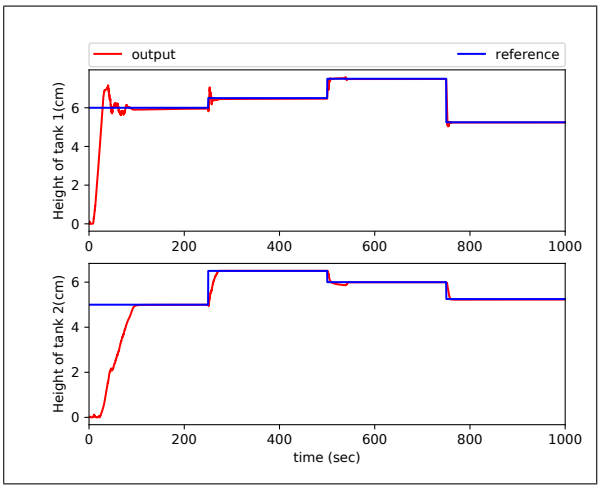

Fig. 5. Liquid Level System Output

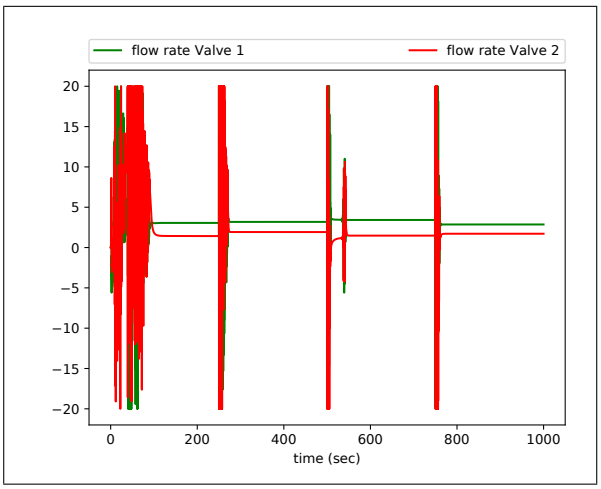

Fig. 6. Liquid Level System Control Effort

B. Simulation Case 2

Equations (32) describe the nonlinear MIMO system with two inputs(u_1 and u_2) and two outputs (x_1 and x_2). The objective of the control effort is the tracking of the sum of two sinosoids for both outputs.

$$\begin{aligned} \frac{dx_1}{dt} &= 0.15 \exp^{-0.5x_1} - 0.1 \sin(x_2) + \sin(u_1) \\ \frac{dx_2}{dt} &= 0.12 \sin(x_1) cos(-x_2) + 0.1 u_2 \end{aligned} \tag{32}$$

The parameter β_c equals 1.0001 and the range of the control signal for both inputs is set to $[-10, 10]$. Figure 7 illustrates the effective tracking for both outputs and figure 8 shows the control effort. We notice that after the initial stage of on-line training, the controller drives the system to the desired operation state.

VI. CONCLUSIONS

In this paper a one step adaptive model predictive controller was proposed, using as predictor a deep neural network. In this way, we exploit the approximation and

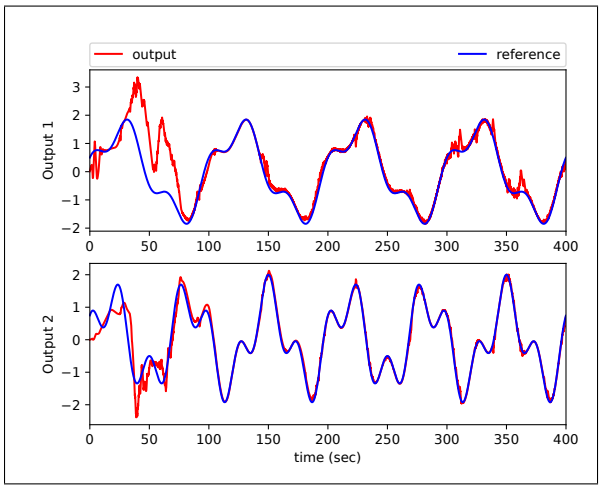

Fig. 7. MIMO System Output

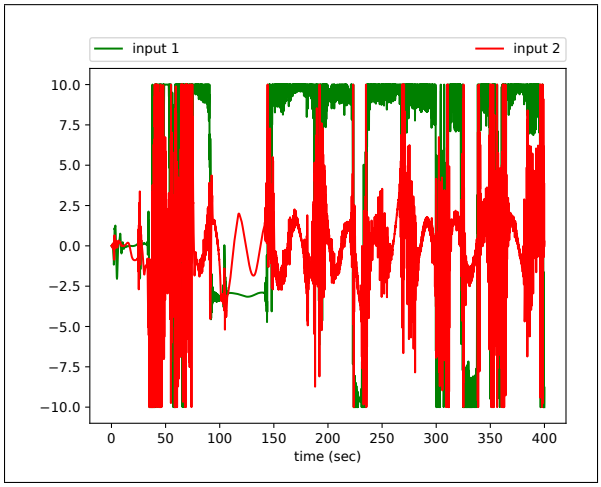

Fig. 8. MIMO control signals

representation ability of deep neural networks, establishing at the same time robustness to system variations and disturbances. The used update laws of the neural weights guarantee the stability of on-line training in the sense of Lyapunov. Ensuring the exact and satisfactory identification of the unknown controlled system by applying Lyapunov guidance of the process, we calculate the update control signal by making secure that the tracking error between the output of the neural identifier and the desired output will converge to zero according to Lyapunov theory. The simulation results for two cases are satisfactory and call for future work in the control of more complex nonlinear dynamical systems. Also, someone can test the performance of the proposed controller on system parameter variations and disturbances during the control process.

APPENDIX

If the norm of a matrix A is less or equal 1 ($\|A\| \leq 1$) and the inverse of $I+A$ exists then $\left\|(I+A)^{-1}\right\| \leq \frac{1}{1-\|A\|}$.

Proof: Let matrix S equals

$$S = \sum_{n=0}^{\infty} (-1)^n A^n = I - A + A^2 - A^3 + \ldots \qquad (33)$$

Then, we can derive very easily the equation $S(I+A) = I$. So, we have

$$\left\|(I+A)^{-1}\right\| = \|S\| = \left\| \sum_{n=0}^{\infty} (-1)^n A^n \right\| \leq \sum_{n=0}^{\infty} \|A^n\| . \qquad (34)$$

Finally, from equation (34) we obtain

$$\left\|(I+A)^{-1}\right\| \leq \frac{1}{1-\|A\|} \qquad (35)$$

taking into account the submultiplicative property of matrices ($\|A^n\| \leq \|A\|^n$) and the formula of geometric sum $\sum_{n=0}^{\infty} \|A\|^n = \frac{1}{1-\|A\|}$ for $\|A\| \leq 1$.

REFERENCES

[1] S. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733 – 764, 2003.

[2] F. Allgöwer, T. A. Badgwell, J. S. Qin, J. B. Rawlings, and S. J. Wright, *Nonlinear Predictive Control and Moving Horizon Estimation — An Introductory Overview*. London: Springer London, 1999, pp. 391–449.

[3] Y. Calleecharan, "Nonlinear identification and control: a neural network approach, g.p. liu, advances in industrial control monograph series, springer, london, 2001. no. of pages: xx + 210. price: 82.50. hardcover. isbn 1-85233-342-1," *International Journal of Adaptive Control and Signal Processing*, vol. 21, no. 10, pp. 911–912, 2007. [Online]. Available: http://dx.doi.org/10.1002/acs.953

[4] P. Kittisupakorn, P. Thitiyasook, M. Hussain, and W. Daosud, "Neural network based model predictive control for a steel pickling process," *Journal of Process Control*, vol. 19, no. 4, pp. 579 – 590, 2009.

[5] F. Temurtas, H. Temurtas, and N. Yumusak, "Application of neural generalized predictive control to robotic manipulators with a cubic trajectory and random disturbances," *Robotics and Autonomous Systems*, vol. 54, no. 1, pp. 74 – 83, 2006.

[6] B. M. kesson, H. T. Toivonen, J. B. Waller, and R. H. Nystrm, "Neural network approximation of a nonlinear model predictive controller applied to a ph neutralization process," *Computers & Chemical Engineering*, vol. 29, no. 2, pp. 323 – 335, 2005.

[7] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85 – 117, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608014002135

[8] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 5 2015.

[9] Z. Man, H. R. Wu, S. Liu, and X. Yu, "A new adaptive backpropagation algorithm based on lyapunov stability theory for neural networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1580–1591, Nov 2006.

[10] K. H. Lim, K. P. Seng, L. M. Ang, and S. W. Chin, "Lyapunov theory-based multilayered neural network," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 56, no. 4, pp. 305–309, April 2009.

[11] S. C. Douglas and T. H. Y. Meng, "Linearized least-squares training of multilayer feedforward neural networks," in *IJCNN-91-Seattle International Joint Conference on Neural Networks*, vol. i, Jul 1991, pp. 307–312 vol.1.

[12] H. Sahoo, P. Dash, and N. Rath, "Narx model based nonlinear dynamic system identification using low complexity neural networks and robust h filter," *Applied Soft Computing*, vol. 13, no. 7, pp. 3324 – 3334, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494613000616

[13] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. P. Turian, D. Warde-Farley, and Y. Bengio, "Theano: A cpu and gpu math compiler in python," 2011.

[14] H. Pan, H. Wong, V. Kapila, and M. S. de Queiroz, "Experimental validation of a nonlinear backstepping liquid level controller for a state coupled two tank system," *Control Engineering Practice*, vol. 13, no. 1, pp. 27 – 40, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0967066103002946