Agile Online-Trained Neural Network Models by Using Robust Fixed Point Transformations

Teréz A. Várkonyi*§, Vincenzo Piuri§, József K. Tar†, Annamária R. Várkonyi-Kóczy**, Imre J. Rudas†

*Doctoral School of Applied Informatics, John von Neumann Faculty of Informatics
Óbuda University
Budapest, 1034, Hungary
varkonyi.teri@phd.uni-obuda.hu

§Department of Computer Science
Universitá degli Studi di Milano
Crema, 26013, Italy
terez.varkonyi@unimi.it, vincenzo.piuri@unimi.it

**Donát Bánki Faculty of Mechanical and Safety Engineering
Óbuda University
Budapest, 1081, Hungary
varkonyi-koczy@uni-obuda.hu

†Institute of Applied Mathematics, John von Neumann Faculty of Informatics
Óbuda University
Budapest, 1034, Hungary
tar.jozsef@nik.uni-obuda.hu, rudas@nik.uni-obuda.hu

***Abstract*—Nowadays, in the field of information processing, neural networks (NNs) are very used, because they can learn adaptively how to behave in a desired way. In the field of adaptive control, NNs are the most beneficial when the system to be controlled is not known in advance, because the system can be modeled by NNs to predict its behavior. In this case, it is very important how accurate the estimation of the NN is, since if the approximation is too rough then extra calculations are needed to get better results. One of the possibilities for improving the NN model is on-line learning during the control process, but this option requires high computational time. Another possibility is the application of Robust Fixed Point Transformations (RFPT), since though, it can only guarantee local stability, RFPT has been developed to reduce the inaccuracy caused by modeling errors and it does not need high computational time. In this paper, a new combination of the two methods the on-line trained neural networks and RFPT is proposed to decrease the computational burden caused by the on-line adaptation and keep the results close to optimum.**

I. Introduction

Human recognition and control abilities far exceed those of complex intelligent control systems (e.g. robots, industrial machines, etc.). This has motivated scientists to analyze the human thinking to model and copy nervous systems and use artificial neural networks (NN) in many areas (e.g. image precessing, signal processing, and control) [1]. There are two main advantages of using NNs: one of them is that NNs can approximate the behavior of nonlinear systems. It can be useful since the controlled system is not always known to control accurately. The other advantage is the learning ability of NNs. Since the controlled systems can change any time (due to external influence), it worths to use NNs instead of identifying the model every time it changes.

Neural networks are widely used in the field of information processing, especially for control tasks. In the applications they are mainly used for two purposes: to identify a system (see e.g. [2], [3]), or to use as a controller (like in [4]). As controllers, NNs can be useful if, as it is mentioned, the controlled systems change during usage. To identify a controlled system, NNs can be very beneficial because the systems cannot be always modeled due to complexity or uncertainty issues. In this case, the systems are approximated by NN models which might capture the system and give useful prediction about the behavior of it, though, the approximation worsens the accuracy of the control.

One of the possibilities to decrease inaccuracy caused by the approximation is taking advantage of the learning ability of neural networks. By applying on-line adaptation to the system, which is not known a priori, it can be captured better and the accuracy of the NN model can be increased. Though, on-line learning has a main disadvantage: if it uses a large amount of data (which is needed for the exactitude) it needs very high computational time. To be able to keep the computational burden under a limit, additional techniques are required.

Another possibility to decrease inaccuracy caused by the approximation is the application of the so-called Robust Fixed Point Transformations (RFPT) method [5]. It is a manner that

can improve existing and well behaving controllers' results when an approximate model is used in the control process. As included in its name, RFPT is characterized by robustness and it has the ability to handle rough approximations. Meanwhile, it does not increase the controller's computational burden considerably which means that theoretically, the inaccuracy caused by the approximation of the neural network can be reduced without time-consuming solutions. The main disadvantage of RFPT is that it can guarantee only local stability thus, in that case if the controlled system changes considerably, some additional methods are needed to keep RFPT in the local stable region.

In this paper, a new neural network approach, called RFPT-NN is introduced. It is shown that both on-line training and RFPT can reduce the error caused by the incompleteness of the NN's adaptation, and both methods can reduce the disadvantages of the other ones: on-line training can keep RFPT in the local stable region and if the neural network is more-or less trained, then by skipping on-line adaptation, RFPT can keep the results close to optimum.

The paper is organized as follows: Section II summarizes the basics of classical feedback control and Robust Fixed Point Transformations as bases of the proposed method. Section III introduces the new RFPT-NN approach plus the proposed strategy is shown. In Section IV, some illustrative simulation results are presented. Finally, in Section V, the conclusions are summarized.

II. Classical feedback control task and Robust Fixed Point Transformations

The idea of the paper is based on the classical feedback control tasks which are built as follows (see Fig. 1): There is a prescribed or "desired" behavior r^d for an existing system. The existing system has some kind of "excitation", for example some kind of torque or a control signal u which forces the system (with mapping φ) to produce the desired response. The control task can be formulated by the following equation

$$r^r = \varphi(u) \tag{1}$$

which describes the correspondence of the control signal (u) and the actual system response r^r (after applying u on it). The main difficulty here is that the controlled system is not exactly known. For the proper system response computation the determination of the proper control force u^d is needed so that $r^d = \varphi(u^d)$, for which only approximate models (φ_{appr}^{-1}) can be of help:

$$r^r = \varphi\left(\varphi_{appr}^{-1}\left(r^d\right)\right) \tag{2}$$

The approximation results in an error because the controller treats the approximate model as it was the desired one. The desired system response could be achieved only by using an exact inverse system model. So applying r^d on the approximate inverse model (which results in $u^d_{appr} \neq u^d$) and then its output to the system, gives the realized response. The correspondence between the realized (r^r) and the desired response (r^d) is

$$r^r \equiv \varphi(\varphi_{appr}^{-1}(r^d)) \equiv f(r^d) \neq r^d \tag{3}$$

Since the controlled system is not exactly known, neither can we determine function $f = \varphi_{appr}^{-1} \circ \varphi$. All we can do is measure its output and based on it build a strategy to decrease the error.

One of the possible options to reduce the error caused by the model approximation can be the application of some kind of on-line adaptable model as approximate inverse model. Since there is the possibility of modifying it real-time, the accuracy of the approximation can be increased by the help of which the accuracy of the control actions are also improved. There are several options for such models, e.g. traditional statistical approaches like Autoregressive-moving-average (ARMA) [6] and their nonlinear generalization NARMA [7] models, but more sophisticated results can be get by connectionist (e.g. neural network) techniques. For instance, Multi Layer Perceptrons are proved to be universal approximators [8] which makes them ideal for system approximation.

Another possibility to reduce the error is a method called the Robust Fixed Point Transformations which makes improvement on a different dimension. It is based on an idea of using a function (G) which transforms u^d_{appr} so that it gets closer to u^d: $|G(u^d_{appr}) - u^d| < |u^d_{appr} - u^d|$. In [9] authors show an iterative fixed point searching algorithm to prove that if

1) G is differentiable and
2) $|G'| \leq M < 1$ ($M \in \mathbb{R}$)

then sequence $\{u_0, u_1 = G(u_0), u_2 = G(u_1), ..., u_{n+1} = G(u_n)\}$ is convergent and the fixed point of G equals $\{u_n\}$'s limit value. They also suggest a function which provides quick convergence:

$$\begin{gathered} G\left(u, u^d_{appr}\right) = (u + K) \times \\ \times \left(1 + B \tanh\left(A\left(h\left(u\right) - u^d_{appr}\right)\right)\right) - K \end{gathered} \tag{4}$$

where $h(x) = \varphi_{appr}^{-1}(\varphi(x))$, $h\left(u^d\right) = u^d_{appr}$, and free parameters A, B, and K have to be set to fulfill the constraints.

III. The RFPT-NN

In this section, a novel method is shown how Robust Fixed Point Transformations and on-line training of neural networks can mutually improve the control of a partially known nonlinear system. Both can be used as strategy for the error reduction. In the following, it is shown how the idea of RFPT can be inserted into the logic of the on-line trained feedforward neural networks and the advantages of the mutual use are shown.

As the on-line training of NNs can be a tool to transform the approximate inverse model, RFPT can be used to transform the output of the model to get more accurate results. Since feedforward neural networks can be used as universal approximators, they can be applied to estimate the behavior of nonlinear systems. So by 1. creating a neural network and training it based on a primitive, rough approximation of the system, 2. training the NN on-line during the control process (on the basis of the actual system), and 3. applying RFPT, the error can be reduced significantly. Thus, in order to achieve higher accuracy, we attach an RFPT module to the NN and the new model is called RFPT-NN. The proposed control scheme

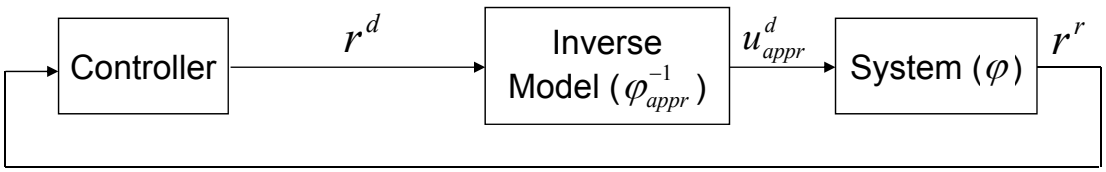

Fig. 1. The block scheme of the classical feedback control.

with the on-line training and the RFPT part is shown in Fig. 2. The proposed scheme contains two neural networks that have to be identical so that function G results in proper output.

The main disadvantages of the proposed combination are the computational burden caused by the on-line training and the possible instability generated by RFPT. To save time it worths to stop on-line learning after a while since when the on-line adaptation is applied the computational time can increase tenfold (depending on the amount of data, the type of the NN, and the training algorithm). The other reason for stopping real-time training is that if the controlled system does not change and disturbances do not effect the system, the neural network can be trained properly so additional adaptation is not needed for getting accurate results. The possible instability of RFPT can be prevented by the on-line training, since if the system and the model are similar enough, RFPT can be kept in the locally stable region.

The main advantage of the mutual usage of RFPT and on-line trained neural networks is that (after a very short initial period) RFPT can reduce the error caused by the inaccuracy of the neural network. Then, when the NN is trained properly, both the on-line adaptation and RFPT can be skipped. To save more time it is also possible to stop on-line training when the real time learning is not complete, but in this case RFPT is needed in the following calculations, to avoid the increase in the tracking error. If the system does not change suddenly, RFPT is able to keep the error close to optimum.

IV. Simulation results

In this section, some illustrative simulation results are presented without limiting the generality of the proposed method. The simulation task is to control a van der Pol oscillator (see [10]) which can be described by the following equation:

$$m\ddot{x} - \mu\left(1 - x^2\right)\dot{x} + \omega_0^2 x + \alpha x^3 + \lambda x^5 = Q \qquad (5)$$

in which m corresponds to some inertia, the term $-\mu\left(1-x^2\right)\dot{x}$ symbolizes some nonlinear viscosity (i.e. dissipation for $|x| > 1$ and energy input for $|x| < 1$), ω_0^2 corresponds to some spring constant, while the remaining terms may describe further nonlinearities of this spring. The control force applied on the system is denoted by Q.

The van der Pol oscillator is ideal for testing the effectiveness of the proposed combined approach since the system's equation of motion contains higher order terms which makes its approximation more difficult. Thus, if its inverse is approximated by e.g. a damped string

$$Q = \hat{m}\ddot{x} + \hat{k}x \qquad (6)$$

where $\hat{m}$ and $\hat{k}$ are free parameters and the neural network is pretrained based on this equation then there will be significant difference between the system and the model. Because of the difference the improving techniques detailed above are needed to get acceptable results.

The simulations are made in MATLAB environment [11]. In the simulations, the van der Pol oscillator has to follow a nominal trajectory which is a sinusoidal wave of amplitude 1.5, frequency $3\,rad/s$, and bias $1\,rad$. The neural networks applied as inverse models are two identical feedforward multi-layer perceptrons supervised by the actual system. They have one hidden layer with twenty neurons and have three inputs: x, $\dot{x}$, and $\ddot{x}$. From these values they calculate the control force (Q^d_{appr} and $h(Q)$, respectively). They are trained by Levenberg-Marquardt backpropagation [12]. Their training- and test ratios are both set to 0.5. Their goal parameters are set to 10^{-5}. As controller, a PID-type law is used:

$$\ddot{x}^d = \ddot{x}^n + 3\Lambda\dot{e} + 3\Lambda^2 e + \Lambda^3 \int e \qquad (7)$$

where x^n denotes the nominal trajectory for the system, $e = x^n - x^r$ stands for the tracking error (x^r marks the realized trajectory), and $\Lambda = 5/s$ is the free parameter of the controller.

In the examples, the numerical parameters of the van der Pol oscillator are set to $m = 1$, $\mu = 0.4$, $\omega_0 = 0.46$, $\alpha = 1$, and $\lambda = 0.1$, the model parameters are $\hat{m} = 2$ and $\hat{k} = 3\omega_0^2$, while in function G they are $K = 7000$, $A = 10^{-4}$, and $B = -1$. The initial settings are $x = 1$, $\dot{x} = 0$, and $\ddot{x} = 0$. The duration time of the simulations is $t = 10s$

In the simulations, seven cases are investigated: C1: Offline train without RFPT; C2: Off-line train with RFPT; C3: On-line train without RFPT; C4: On-line train with RFPT; C5: On-line train with RFPT, but both are stopped when adaptation is achieved (t = 3075ms); C6: On-line train with RFPT, but both are stopped before adaptation is achieved (t = 1000ms); C7: On-line train with RFPT and only learning is stopped when adaptation is not achieved yet (t = 1000ms). The results can be compared in Figs. 3 and 4. It can be seen that with off-line learning (C1) the error, which is a periodical signal thanks to the PID-type control actions, remains relatively big. If at least one of the improving techniques is used (C2 or C3), the error is reduced by more than two orders of magnitude, though the on-line adaptation increases the computational time ninefold. It is also well observable when the neural network model achieves adaptation because after that the error becomes periodical. If both RFPT and on-line training are applied (C4), the error in the initial period is reduced significantly, otherwise remains similar to that achieved by on-line learning (C3). The signal becomes periodical earlier because RFPT reduces the error

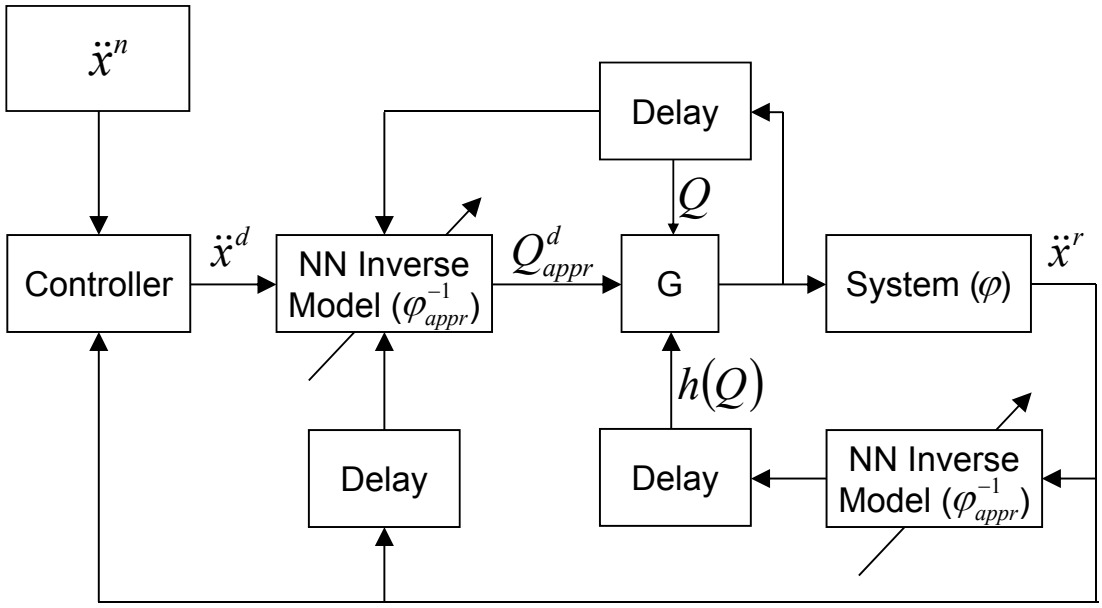

Fig. 2. The block scheme of the proposed method: the on-line trained NNs and RFPT are used mutually.

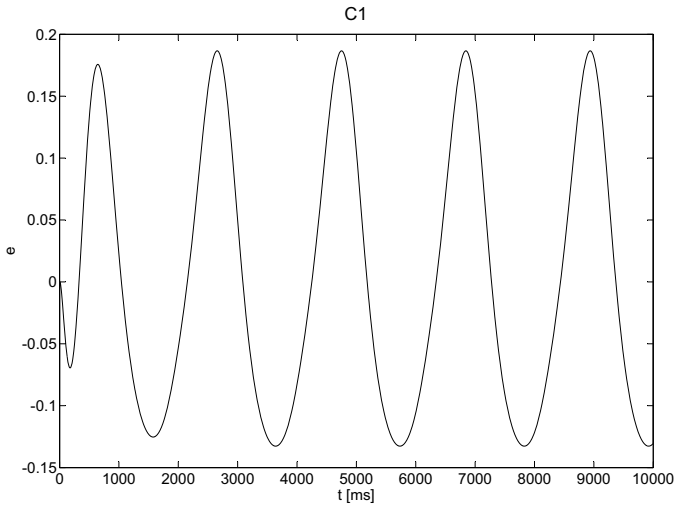

Fig. 3. Tracking error achieved in case C1. Though the error oscillates around zero, it remains relatively high.

thus, the NN model can get more precise data in the initial period. The drawback is that the computational time is very high in this case. If both two methods are used until proper adaptation of the system is achieved (at the $3075.\,ms$; C5) the error does not increase considerably and much computational time is saved. If both two methods are stopped before proper adaptation of the system (at the $1000.\,ms$; C6), the error stays low at the initial period, but after that it can become huge. The most beneficial case seems to be the last one (C7) when only on-line training is stopped before proper adaptation, but RFPT is still used. In this case, the error is optimal in the initial period, after that remains close to optimum, and the most computational time is saved.

V. Conclusions

Neural networks are widely used to investigate the behavior of unknown systems. In this paper, a new combination of Robust Fixed Point Transformations and on-line trained neural networks is introduced. It is shown that they can complete each other in a beneficial way, since in the initial period of the control process RFPT can reduce the error caused by the inaccurate training. After that the on-line adaptation can be stopped because it causes great computational burden and RFPT is able to keep the error close to optimum. Though on-line learning significantly increases the computational burden, with the help of Robust Fixed Point Transformations this extra time can be minimized.

Acknowledgment

This work was sponsored by the Hungarian National Scientific Fund (OTKA 78576).

References

[1] M. Akay, "Handbook of Neural Engineering," Wiley-IEEE Press, 2007.

[2] K. Chen, L. Wang, "Trends in Neural Computation," Berlin: Springer, 2007.

[3] R. Andoga, L. Madarász, T. Karol', L. Főző, V. Gašpar, "Intelligent Supervisory System for Small Turbojet Engines," *Topics in Intelligent Engineering and Informatics 2 : Aspects of Computational Intelligence: Theory and Applications*, Berlin Heidelberg : Springer-Verlag, pp. 85–104, 2013.

[4] E. N. Sanchez, A. Y. Alanís, A. G. Loukianov, "Discrete-Time High Order Neural Control Trained with Kalman Filtering," Berlin: Springer-Verlag, 2008.

[5] J. K. Tar, J F. Bitó, L. Nádai, and J. A. Tenreiro Machado, "Robust Fixed Point Transformations in Adaptive Control Using Local Basin of Attraction," *Acta Polytechnica Hungarica*, Vol. 6, No. 1, pp. 21–37, 2009.

[6] G. E. Box, J. E. Jenkins, "Time series analysis: Forecasting and control," Holden Day, San-Francisco, CA, USA, 1970.

[7] K. S. Narendra, S. Mukhopadhyay, "Recurrent Neural Networks and Robust Time Series Prediction," *IEEE Transactions on Neural Networks*, Vol. 5, No. 2, pp. 240–254, 1994.

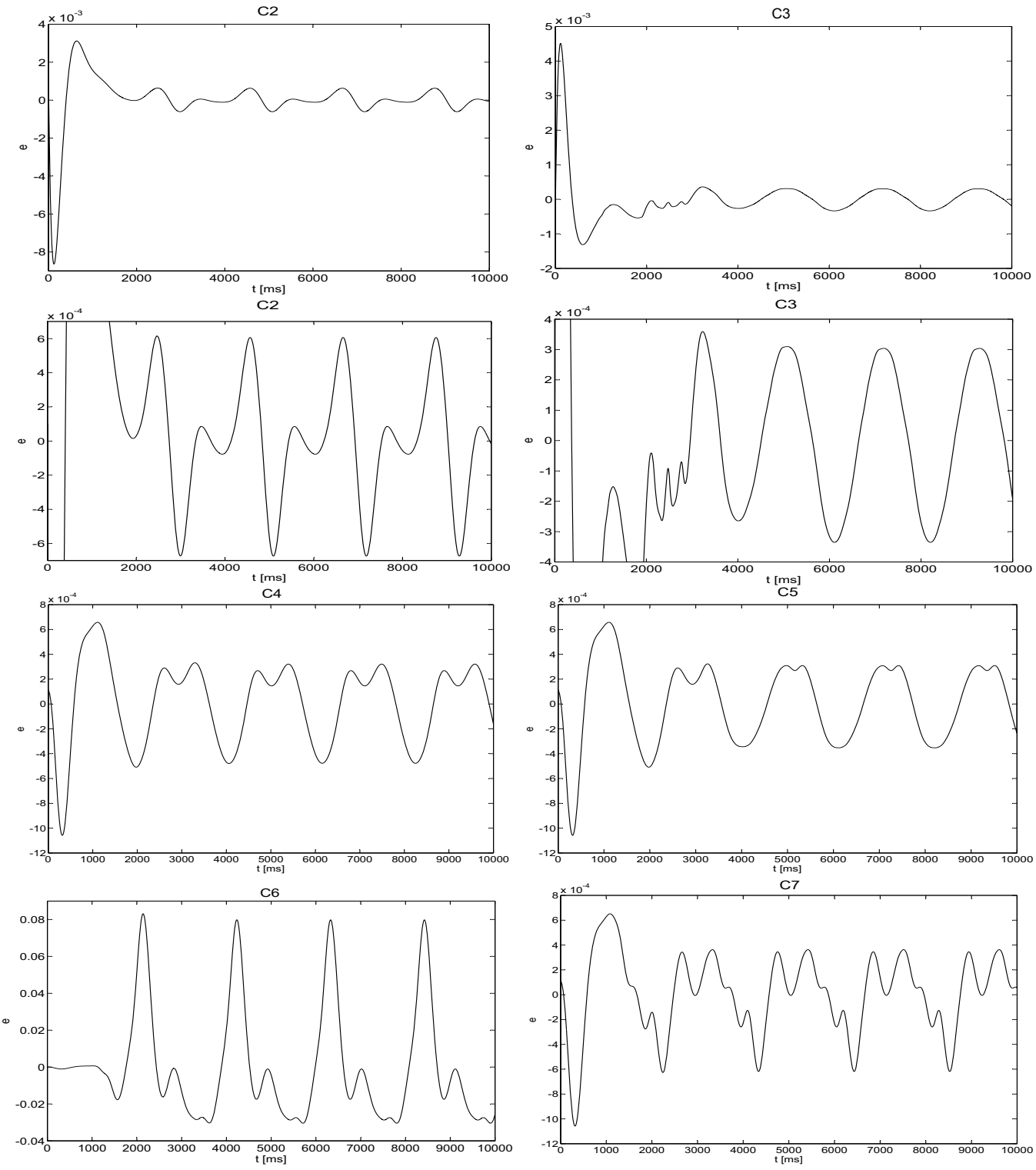

Fig. 4. Tracking errors achieved in case C2: normal (upper left) and zoomed (second from the top, left); C3: normal (upper right) and zoomed (second from the top, right); C4 (third from the top, left); C5 (third from the top, right); C6 (lower left); C7 (lower right). If we consider only accuracy then on-line training is more effective than RFPT. The most accurate results are got when both RFPT and on-line learning are applied (C4), though, if they are stopped after achieving adaptation ($t = 3075ms$, C5), the error does not increase considerably. If both on-line training and RFPT are stopped before proper adaptation ($t = 1000ms$, C6), the tracking error becomes huge (see the lower left figure). If only training is stopped before proper adaptation (C7) then the tracking error remains close to optimum and the most computational time is saved.

[8] K. Hornik, M. Stinchcombe, H. White, "Multilayer Feedforward Neural Networks are Universal Approximators," *Neural Networks*, Vol. 2, No. 5, pp. 359–366, 1989.

[9] J. K. Tar, I. J. Rudas, and K. R. Kozłowski, "Fixed Point Transformations-Based Approach in Adaptive Control of Smooth Systems," *Lecture Notes in Control and Information Sciences 360 (Eds.: M. Thoma and M. Morari), Robot Motion and Control 2007 (Ed.: Krzysztof R. Kozłowski)*, Springer Verlag London Ltd., pp.157–166, 2007.

[10] B. van der Pol, "Forced Oscillations in a Circuit with Non-Linear Resistance (Reception with Reactive Triode)," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science Ser.*, 7(3), pp. 65–80, 1927.

[11] MATLAB, http://www.mathworks.com.

[12] H. Yu, B. M. Wilamowski, "LevenbergMarquardt Training," *Industrial Electronics Handbook Vol. 5 Intelligent Systems*, 2nd Edition, Chapter 12, pp.**12-**1–**12-**16, 2010.