Repetitive neurocontroller with disturbance dual feedforward – choosing the right dynamic optimization algorithm

Bartlomiej Ufnalski and Lech M. Grzesiak
WARSAW UNIVERSITY OF TECHNOLOGY
Institute of Control and Industrial Electronics
75 Koszykowa St., Warsaw 00-662, Poland
Phone: +48 22 234-6138
Fax: +48 22 234-6023
Email: {bartlomiej.ufnalski, lech.grzesiak}@ee.pw.edu.pl
URL: http://www.ee.pw.edu.pl

Acknowledgments

The research was partially supported by the statutory fund of Electrical Drive Division within the Institute of Control and Industrial Electronics, Faculty of Electrical Engineering, Warsaw University of Technology, Poland.

Keywords

«repetitive control», «iterative learning control», «neurocontroller», «sine wave converter», «repetitive disturbance rejection», «dynamic optimization problem», «disturbance dual feedforward», «training algorithm», «non-local update rule», «global update rule»

Abstract

The paper presents a recently developed repetitive neurocontroller (RNC) that does not require additional filtering and/or forgetting to robustify it, i.e. to circumvent the long horizon stability issue present in the classic iterative learning control (ILC) scheme. Initially, the Levenberg–Marquardt (L–M) error backpropagation (BP) algorithm was used as a DOP(dynamic optimization problem)-capable search mechanism. At that time the choice of the training algorithm was made based on the frequently reported effectiveness of the L–M method in static optimization problems. However, there is an abundance of neural network training methods characterized, e.g., by different convergence rates, computational burden, noise sensitivity, etc. The performance of a particular optimization method is always problem specific. The case study of a constant-amplitude constant-frequency (CACF) voltage-source inverter (VSI) with an *LC* output filter is analysed here and some recommendations regarding the trade-off between convergence rate and computational complexity are made. The robustness to a measurement noise is also tested. The comparison is based on the results of numerical experiments. A couple of algorithms is then suggested for real-time implementation.

Introduction

Repetitive process control has gained noticeable attention during the last decade. This is mainly due to the constant pursuit of developing control systems characterized by nearly perfect reference tracking and disturbance rejection capabilities. Theoretically, it is possible to obtain such a perfect control signal for repetitive process by introducing the integral action in the pass to pass direction (here the k-direction):

$$u(p,k) = u(p,k-1) + k_{\mathrm{RC}} e(p,k-1) \ , \tag{1}$$

where u denotes the control signal, e is the control error, k_{RC} is the controller gain, k is the iteration (pass, trial, cycle) index, p is the time index along the pass ($1 \leq p \leq \alpha$, where α is the pass length). In practice, the formula (1) has to be altered to make the system stable in the long-time horizon. A majority of those modifications can be represented as special cases of

$$u(p,k) = \mathbf{Q}\left(z^{-1}\right) u(p,k-1) + k_{\mathrm{RC}} \mathbf{L}\left(z^{-1}\right) e(p,k-1) \ , \tag{2}$$

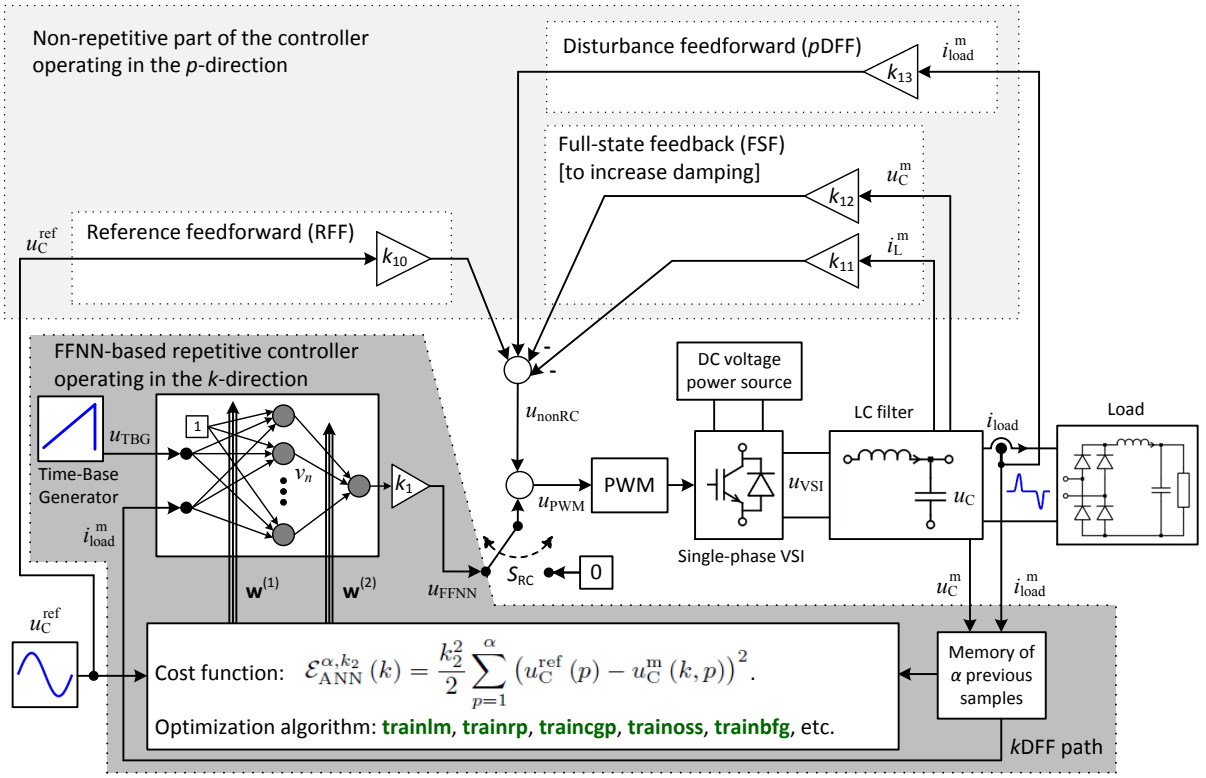


Fig. 1: Schematic diagram of the proposed repetitive neurocontrol system with a disturbance dual feedforward path (the block *Load* depicts an exemplary diode rectifier).

where $\mathbf{Q}$ and $\mathbf{L}$ denote filters (in some approaches reduced to a single gain). These filters can be designed as non-causal ones and it is required to make $\mathbf{Q}$ of zero-phase-shift type. Some common approaches include but are not limited to

a) $\mathbf{Q} = 1 - \gamma$ and $\mathbf{L} = 1$, where $\gamma \in (0, 1)$ is the forgetting factor,
b) $\mathbf{Q}$ and $\mathbf{L}$ denoting low-pass zero-phase non-causal filters based on IIR filters to prevent overlearning,
c) $\mathbf{Q} = 1$ and $\mathbf{L}$ being an FIR filter (in some research groups also known as "wave" and/or "non-local" control law [1]).

Probably the most significant obstacle in deploying (2) is the lack of effective recipes to select and tune those filters. At the same time a majority of theoretical analyses and stability proofs as well as numerical demonstrations (e.g. [2]) assumes that at least one out of four features of almost any physical repetitive control system is negligible: uncertainties due to a measurement noise and drift, identification errors, saturation on the plant side, and a repetitive disturbance. One of the consequences is limited applicability of those results in real systems. A practical ILC solution should address at least one common consequence of the above features – a non-zero steady state error in the pass to pass direction. The integration introduced by (1) may then in the long run have a destabilizing impact. That is why many solutions try to robustify the system by applying filters aimed to prevent overlearning in the upper frequency band, i.e. to stop integral action for those frequencies [3–5]. The main problem with those approaches is that due to practicalities, such as limited resources of microcontrollers, the attenuation in the stop band may be not sufficient to ensure stable operation in the long time horizon, e.g. typically tens of millions or even more than one hundred million repetitions in power electronic converters. The task of stabilizing the system becomes even more challenging if a repetitive external disturbance non-additive to the controlled output is anticipated to enter a given system. Such disturbances are inevitable in power electronic converters – usually in the form of load current. Moreover, it is often hard to determine the upper band of such an external disturbance, e.g. for a diode rectifier it changes notably with the rectifier's inductance.

To the best of the authors' knowledge, there is still some (if not plenty of) room for computational intelligence techniques in the iterative learning control field. Currently there is only one surefire way, namely the forgetting factor, to tackle the overlearning phenomenon in uncertain real-life systems. Obviously, this means that the pure integrator in the k-direction is replaced with the first order lag element, which in turn implies that, depending on the severity of the system uncertainties and the resulting minimum pace of forgetting, some or even most of the desired properties of the ILC scheme are lost [6]. Alternatively, it is also possible to include a penalty for control signal dynamics as in [7].

Table I: Selected parameters of the model

Component/Parameter	Description/Value
LC output filter	300 µH, 160 µF, $R_\mathrm{f} = 0.6\,\Omega$
Resonant frequency	726 Hz
Critical damping resistance	$R_\mathrm{crit} = 2.74\,\Omega$ (highly underdamped)
Reference output voltage	$f^\mathrm{ref} = 50$ Hz, $U^\mathrm{ref}_\mathrm{RMS} = 230$ V, sinusoidal
Sampling time	$T_\mathrm{s} = 100$ µs ($\alpha = 200$ points per pass)
Measurement noise	3% of 100 A or 325 V (band-limited white noise with 95% of its samples within the range)
Load-1	Resistive: 4 kW
Load-2	Diode rectifier: 500 µH, 3 mF, 6 kW, crest factor of ca. 2.5
Closed-loop system damping	3 times higher than in the open-loop system
Identified filter resistance	$\hat{R}_\mathrm{f} = 0.25 R_\mathrm{f}$ (significant identification error assumed to highlight dynamics of the repetitive part)
Number of hidden neurons	$N = 7$
Type of neurons	`tansig` (excl. the output `purelin` neuron)
Training method type	`trainlm`, `trainrp`, `traincgp`, `trainoss`, `trainbfg`, etc.
Learning parameters	Default settings, except `net.trainParam.epochs=1` for all traning methods and `net.trainParam.lr=200` for `traingd`, `traingdm`, `traingda` and `traingdx`; also `net.divideFcn='dividetrain'` always holds, which assigns all targets to the training set.
Weight constraints	Yes, in the interval [-30,30].

Local, non-local and global update rules

Reported ILC laws could be categorized into three groups: local, non-local and global. The very basic update law (1) is of local type – it uses only a single value of control error from the previous pass to update the current value of control signal. The control laws that incorporate any filtering in the along the pass direction become non-local. Their non-locality comes from the fact that more than one value of control error from the previous pass is used to update the current value of control signal. It seems that the members of these two groups, despite some formal proofs of their stability, are bound to be robustified in practice by introducing the forgetting factor. This equally applies to classic laws as well as to repetitive neurocontrollers proposed by other teams. For example, the practical implementation of the non-local training rule developed in [8] also includes the above-mentioned forgetting mechanism [9].

There is one recently proposed member of the third group that employs a global update rule – the repetitive neurocontroller [10, 11]. The solution is distinct from non-local ones in that it uses all values of control error from the previous pass to update current control signal sample. Moreover, this algorithm is global in terms of the objective function used in the definition of the update law, namely the mean squared control error calculated for the entire pass. The repetitive control task at hand has then been rearranged to pose a dynamic optimization problem. In consequence, gradient-based training algorithms developed for feedforward neural networks may serve as potential candidates to provide the iterative learning rule. It has been decided not to interfere in the already established nomenclature [1] and to keep the non-local group as it is. However, it should be clarified that the members of the non-local group use strictly local objectives when defining a control law – even if the length of the filter is equal to the length of the whole pass.

It is worth noticing that online trained neurocontrollers are widely discussed within the context of non-repetitive control systems and various motion control systems are reported [12–16]. However, there are few attempts to reformulate these algorithms to make them suitable within the context of repetitive control in power electronics and drives. These attempts focus on incorporating B-spline networks with non-local learning rules as reported in [9].

Feedforward neural network based repetitive controller

The developed repetitive neurocontroller [10, 11] for a constant-amplitude constant-voltage true sine inverter is sketched in Fig. 1. The overall control system contains two DFF paths, the classic one in the p-direction (the along the pass direction) and the novel one in the k-direction [17]. No low-pass filtering is required to robustify this control scheme. Also no forgetting factor is introduced in the global update

rule. For the purpose of controlling the output voltage of a CACF VSI, the following cost function is used

$$\mathcal{E}_{\mathrm{ANN}}^{\alpha,k_2}(k) = \frac{k_2^2}{2}\sum_{p=1}^{\alpha}\left(u_{\mathrm{C}}^{\mathrm{ref}}(p) - u_{\mathrm{C}}^{\mathrm{m}}(k,p)\right)^2 , \tag{3}$$

where $u_{\mathrm{C}}^{\mathrm{ref}}$ is the reference voltage, $u_{\mathrm{C}}^{\mathrm{m}}$ denotes the measured output filter capacitor voltage and k_2 is the error scaling factor. The functional (3) should be seen as a function of the neural weights

$$\mathcal{E}_{\mathrm{ANN}}^{\alpha,k_2}(k) = \mathcal{E}_{\mathrm{ANN}}(\mathbf{w}^{(1)}(k), \mathbf{w}^{(2)}(k)) \tag{4}$$

and the training algorithm is employed to continuously solve the dynamic optimization problem (DOP) of the form:

$$\begin{aligned} &\underset{\mathbf{w}^{(1)},\mathbf{w}^{(2)}}{\texttt{reduce iteratively}} && \mathcal{E}_{\mathrm{ANN}}(\mathbf{w}^{(1)}(k), \mathbf{w}^{(2)}(k)) \\ &\quad\texttt{subject to:} && \texttt{system nonlinearities and uncertainties}, \\ &&& \texttt{system nonstationarity}, \\ &&& N = \texttt{const}, \\ &&& \texttt{constrained weight space}, \end{aligned} \tag{5}$$

where N is the number of hidden neurons (here organized into a single layer). The resulting output signal of the neural network is summed with the non-repetitive path – here a reference feedforward (RFF) plus a full state feedback (FSF) plus a disturbance feedforward (pDFF) – to produce reference signal for the pulse width modulator (PWM).

As the proposed neurocontroller acts only in the k-direction, it cannot shape the dynamics of the response in the p-direction. The plant being an *RLC* series circuit is itself stable; however due to the highly underdamped natural characteristics of the plant (compare R_{f} with R_{crit} in Tab. I) the above-mentioned RFF+FSF+DFF non-repetitive controller acting in the p-direction is needed to shape the response in a sample-by-sample manner. For the purpose of this study, the FSF has been implemented to increase damping 3 times, i.e. FSF gains k_{11} and k_{12} have been determined using the pole placement procedure to shift closed-loop poles 3 times deeper into the left-half s-plane in respect to open-loop poles. This produces control signal

$$u_{\mathrm{FSF}} = -(k_{11}i_{\mathrm{L}}^{\mathrm{m}} + k_{12}u_{\mathrm{C}}^{\mathrm{m}}) \tag{6}$$

additive to the feedforward neural network repetitive controller (FFNNRC) output signal. Also, the standard RFF path

$$u_{\mathrm{RFF}} = (1 + k_{12})u_{\mathrm{C}}^{\mathrm{ref}} \tag{7}$$

is introduced to give a unity gain for the zero frequency [18]. Finally, the disturbance static feedforward (pDFF) path is included to compensate the resistive voltage drop (for the zero frequency) [19]

$$u_{\mathrm{DFF}} = (\hat{R}_{\mathrm{f}} + k_{11})i_{\mathrm{load}}^{\mathrm{m}}, \tag{8}$$

where $\hat{R}_{\mathrm{f}}$ is the identified resistance of the output filter and $i_{\mathrm{load}}^{\mathrm{m}}$ denotes the measured load current. A relatively high identification error is assumed in this study ($\hat{R}_{\mathrm{f}} = 0.25R_{\mathrm{f}}$) to accentuate the influence of the repetitive controller by producing a more significant control error for the FFNNRC and as a result to make the case scenario more illustrative. The resulting control signal

$$u_{\mathrm{PWM}} = u_{\mathrm{FFNN}} + \underbrace{u_{\mathrm{RFF}} + u_{\mathrm{FSF}} + u_{\mathrm{DFF}}}_{u_{\mathrm{nonRC}}} \tag{9}$$

passed to the modulator acts simultaneously in the along the pass direction and the pass to pass direction.

The FFNN based approach to repetitive control gives a significant flexibility not present in the classic ILC scheme in terms of the controller's input signal selection. The minimal implementation requires a time base generator (TBG) signal to be passed to the FFNN. This signal transforms the p-direction control signal synthesis problem into a function approximation one; however, the minimal realization has three major drawbacks:

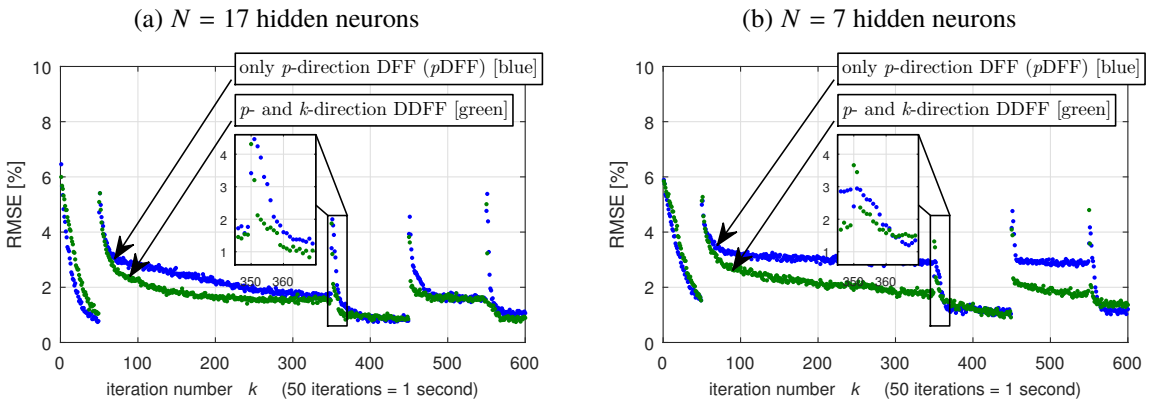


Fig. 2: Comparison of the root mean square error (RMSE) decay rates for the Levenberg–Marquardt BP (`trainlm`) in the case of the classic disturbance feedforward and the novel disturbance dual feedforward.

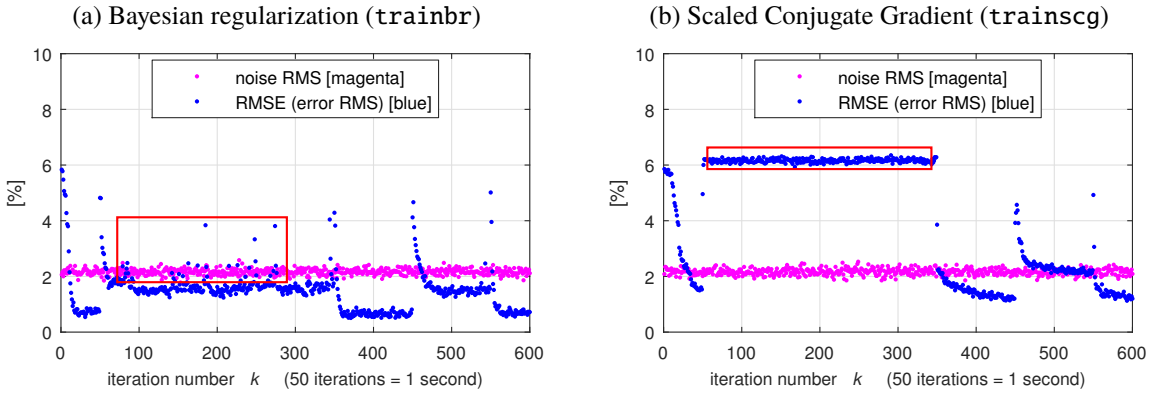


Fig. 3: Undesired transient states (emphasized using red frames) caused by the regularization mechanism in `trainbr`, and inconsistent behaviour of `trainscg` despite identical load variations at time instances 50 s and 450 s – the algorithm failed to track the optimum after the first change in load type.

a) the number of neurons has to be relatively high (ca. 20) to enable effective approximation of control signal needed to reject nonlinear load currents that usually do not resemble in shape the TBG signal; and this is an online trained neural network, thus maintaining a low computational burden of the algorithm, which grows with the number of neurons, is of paramount importance here;
b) the convergence rate is weakened due to the lack of any plant related information at the neural network inputs;
c) the number of neurons is a trade-off between a precise control signal shaping for significantly nonlinear loads and an absence of excessive overlearning for linear loads and no-load conditions; the minimal realisation makes working out this trade-off very challenging and the quality of the output voltage has to be compromised.

A dynamic response in the k-direction improves and the number of neurons can be significantly reduced after additionally introducing the load current signal at the inputs of the neurocontroller. This is illustrated in Figs. 2a and 2b. The name disturbance dual feedforward (DDFF) is used throughout the paper to highlight that the load current signal is exploited in both directions.

Choosing the right iterative DOP-capable learning algorithm

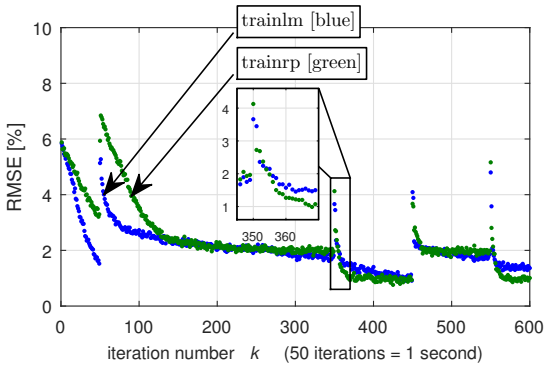
The main goal of the research reported in this paper is to identify learning algorithms for a real-time implementation. Initially, the Levenberg–Marquardt backpropagation (BP) method was employed and identified as being able to ensure fast error convergence and high voltage quality at a steady state. However, there exist less computationally demanding learning algorithms such as the resilient backpropagation or Broyden–Fletcher–Goldfarb–Shanno quasi-Newton algorithm. Taking into account that the performance of any optimization tool is always problem specific, the quality of the output waveform during transients and at steady states ought to be investigated in the specific system before selecting candidates for practical

Table II: Learning algorithms comparison and assessment

Algorithm	Acronym	Exec. time*	Transients	Steady states
Levenberg–Marquardt BP	`trainlm`	41.2 s	++	+
Resilient BP (RPROP)	`trainrp`	21.8 s	+	++
One-step secant BP	`trainoss`	24.8 s	++	++
BFGS quasi-Newton BP	`trainbfg`	24.0 s	++	+
Gradient descent (GD)	`traingd`	22.4 s	-+	-+
GD with momentum	`traingdm`	22.0 s	-+	-+
GD with momentum & adaptive LR	`traingdx`	24.0 s	-+	-+
GD with adaptive LR	`traingda`	22.9 s	-+	-+
Bayesian regularization	`trainbr`	48.2 s	-	+
Scaled Conjugate Gradient (CG)	`trainscg`	28.9 s	-	+
Polak–Ribiére CG (Conjugate Grad.)	`traincgp`	24.0 s	+	+
Fletcher–Powell CG	`traincgf`	24.8 s	+	++
CG with Powell/Beale restarts	`traincgb`	24.0 s	+	++

*for 600 calls using Intel® Core™ i5-3210M CPU @ 2.50GHz

(a) `trainlm` vs. `trainrp`



(b) `trainlm` vs. `trainoss`

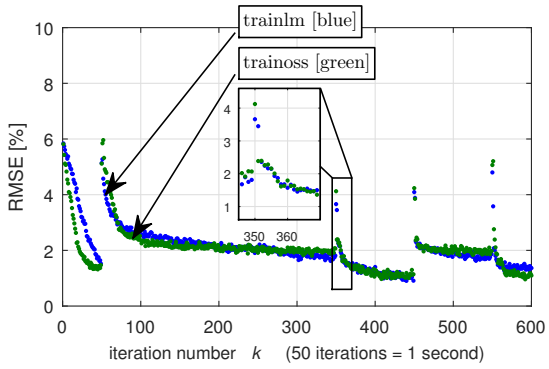
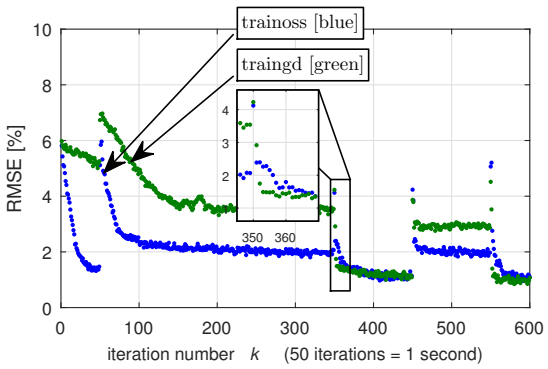


Fig. 4: Root mean square error decay rate for Levenberg–Marquardt BP (`trainlm`), resilient BP (`trainrp`) and one-step secant BP (`trainoss`) algorithms (Load-1 and Load-2 are switched cyclically).

(a) `trainoss` vs. `traingd`



(b) `trainoss` vs. `traingdx`

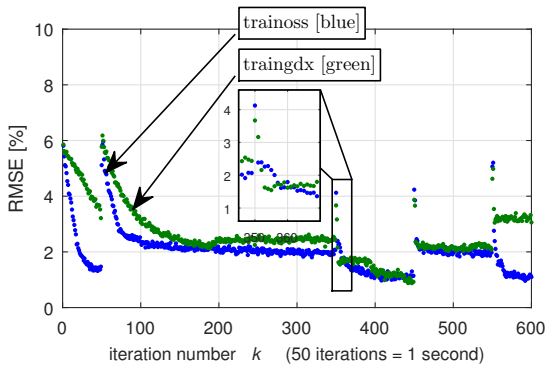


Fig. 5: Root mean square error decay rate for one-step secant BP (`trainoss`), gradient descent (`traingd`) and gradient descent with momentum and an adaptive learning rate (`traingdx`) algorithms (Load-1 and Load-2 are switched cyclically).

implementation. It should be stressed that these learning algorithms are to be used in a noisy environment and the optimization task is of a dynamic type. The latter is due to variable load conditions. There is a rich variety of learning algorithms designed for FFNNs training. A majority of them was originally developed as static optimization problem solvers. Nevertheless, most of them can be used in dynamic optimization problems without the need of any alterations. Over a dozen off-the-shelf training algorithms available in the Neural Network Toolbox from MathWorks [20] are tested and compared here. The comparison is by no means a definite one – it refers to only one plant and involves visual assessment of the performance of the control system. Consequently, only an exemplary decision process is demonstrated and several recommendations are made.

Selected parameters of the model are collated in Tab. I. It should be noted that all measurements are corrupted by noise. This makes the search task challenging due to the jagged optimization landscape. Moreover, the task at hand is of a dynamic type, i.e. each training vector (each pass) is presented to the controller only once (`net.trainParam.epochs=1`) and then is forgotten completely. The training algorithm directly shapes the dynamics of the controller in the k-direction. Even at a steady state of the system the optimisation landscape is still dynamic because of the measurement noise. There are several requirements for the weight adaptation mechanism, i.e. the learning algorithm, to be met in order to facilitate correct operation of the controller under variable load conditions and in the presence of a measurement noise:

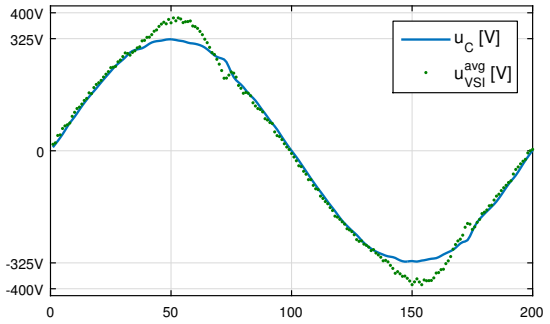
a) the search algorithm should be relatively noise-immune – the effective operation under at least 3 % unfiltered measurement noise has to be possible and higher noise levels (up to 10 %) should not render the search impossible;
b) the algorithm should not stick to an outdated optimum after an abrupt load change, i.e. should be able to track sudden movements of the optimum;
c) a transition to the new optimal control signal should be smooth to ensure acceptable output voltage quality (u_C) during transients;
d) the algorithm should generate consistent results, i.e. a given change in load conditions should always yield similar behaviour of the system.

The test scenario assumes abrupt switchings between two load types: Load-1 and Load-2 (see Tab. I for details). The sequence is as follows: Load-1 in the time interval 0–50 s, Load-2 in 50–350 s, Load-1 in the interval 350–450 s, Load-2 in 450–550 s, and at the end again Load-1. If voltage quality deteriorates significantly, i.e. beyond noise level, from pass to pass within intervals, this is caused solely by the learning algorithm itself. Algorithms that offer a near-monotonic decay of the RMSE are preferable. The consistency of the controller responses is verified using multiple (>3) re-evaluations in the above-mentioned test scenario. All noise generators have random seeds (`rng('shuffle')`) and also initial weights are not reproduced from test to test due to the random element of Nguyen-Widrow initialization [21].

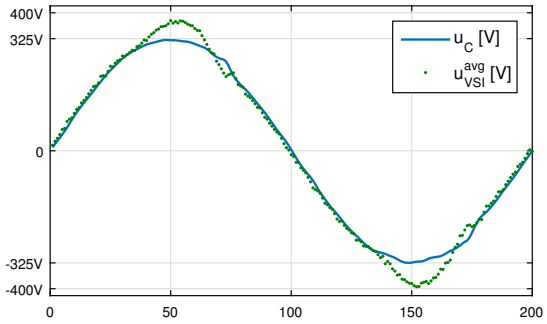
Practical implementation of any online trained neurocontroller requires weights to be constricted within a certain range; thus, a mechanism that can effectively prevent weights from overflowing is indispensable. The training with Bayesian regularisation (`trainbr`) introduces soft limits on weights; however, the controller equipped with this algorithm tends to produce undesirable transients illustrated in Fig. 3a. In the case of all other algorithms discussed here, hard limits have been imposed on weights. Also the controller incorporating the scaled conjugate gradient method (`trainscg`) failed to produce reproducible results in a noisy environment, which is illustrated in Fig. 3b. The rest of the tested algorithms perform consistently throughout experiments and selected observed features are summarized in Tab. II.

Often the RPROP is suggested as the first-hand choice for on-line trained non-repetitive neurocontrollers [22] if the cost function applied is just the squared current control error sample, i.e. weights are updated after each presentation of a single control error sample (incremental training). The RPROP is reported as providing learning spread equally over the network and, thanks to adaptation affected only by the sign of the partial derivative, also less prone to fail due to environment noisiness. It also has the lowest computational complexity. However, other algorithms such as the one-step secant BP (see Fig. 4a) or the BFGS quasi-Newton BP have similar execution time and manifest fast convergence when applied to the discussed control task. The gradient descent methods struggle to operate effectively in noisy measurement environment (see Tab. I for the specific noise level) as demonstrated in Fig. 5. All methods tested here are configured as batch ones, i.e. errors are accumulated and all of the weights' updates are made at once at the end of a pass. Further study will also include sequential learning algorithms. However, already at this point it can be concluded that as far as the repetitive neurocontroller with the global update law is considered there is no definitive winner. The commonly recommended RPROP has the lowest computational complexity but can be surpassed in terms of convergence rate, e.g, by the one-step memory-less secant method or other full gradient based (in contrast to only gradient sign based) methods as shown in Fig. 4. This may suggest that the related dynamic optimization landscape is only moderately challenging and standard learning methods are sufficient to solve the relevant DOP (Fig. 6).

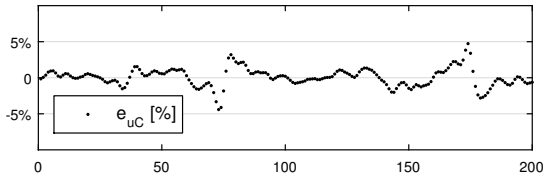
(a) Output capacitor voltage and PWM converter average voltage (for `trainoss`)



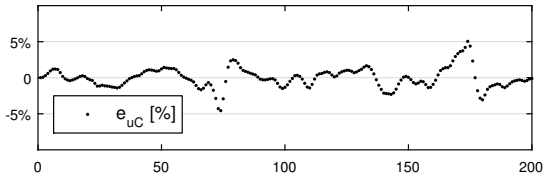
(b) Output capacitor voltage and PWM converter average voltage (for `trainrp`)



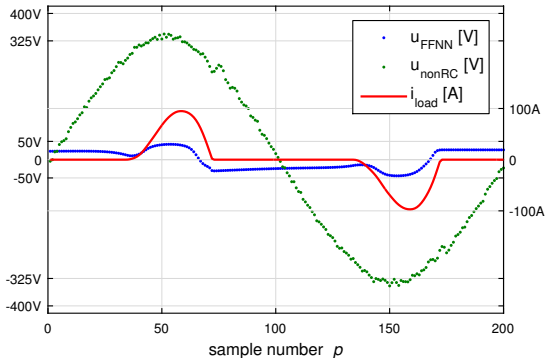
(c) Control error (for `trainoss`)



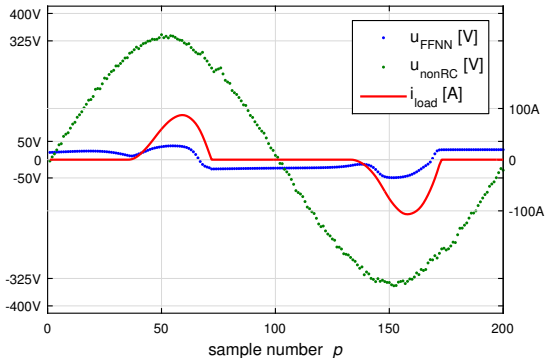
(d) Control error (for `trainrp`)



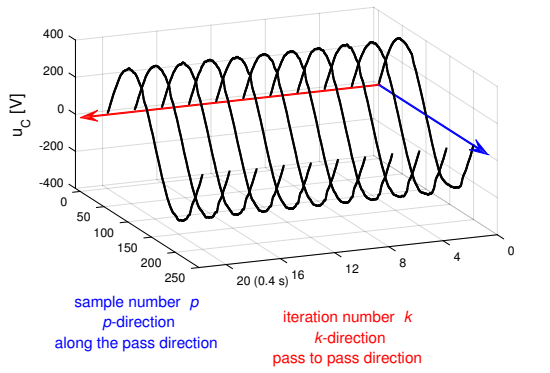
(e) Control signal components and load current (for `trainoss`)



(f) Control signal components and load current (for `trainrp`)



(g) Evolution of the output voltage waveform after connecting the diode rectifier (for `trainoss`)



(h) Evolution of the output voltage waveform after connecting the diode rectifier (for `trainrp`)

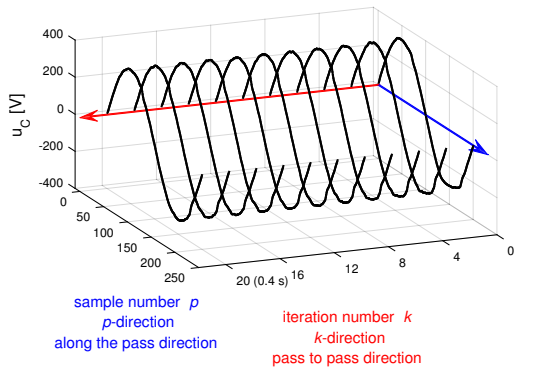


Fig. 6: Steady-state waveforms (a)-(f) under diode rectifier load for two algorithms comparable in terms of their performance and the evolution of output voltage (g)-(h) after switching from resistive load to diode rectifier load.

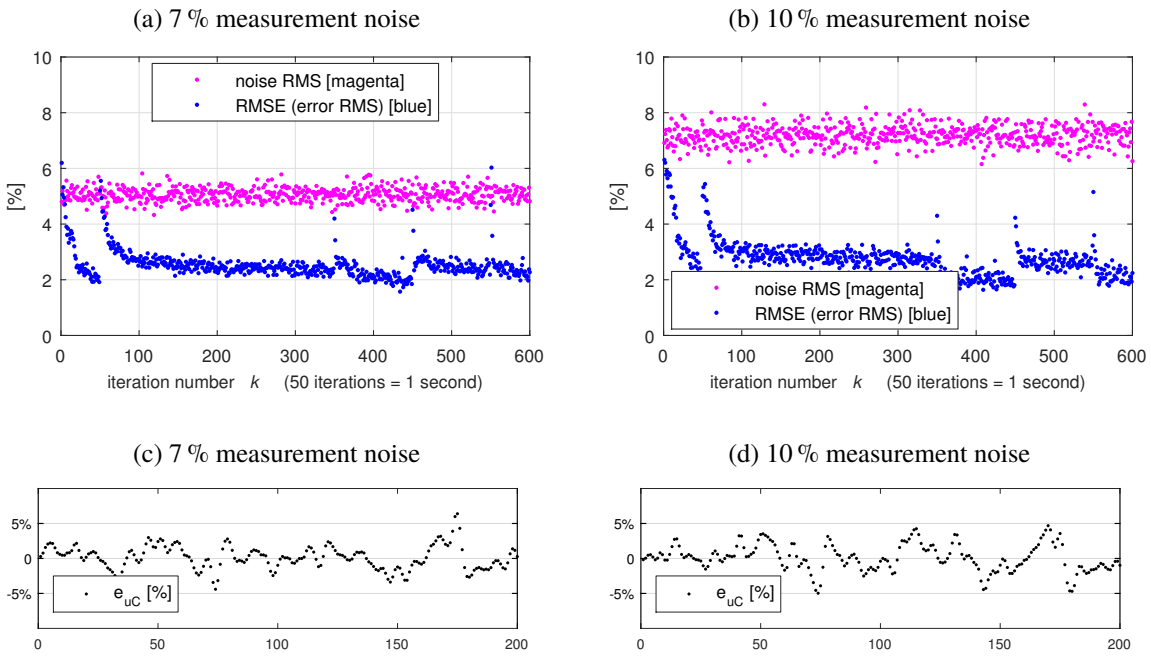


Fig. 7: Evolution of root mean squared error during transients and steady state performance of the converter under heavy noise conditions – the steady state error along the pass recorded at the time instance of 300 s (the diode rectifier load).

Noise robustness of the repetitive neurocontroller

The proposed k-direction neurocontroller needs a stable sufficiently damped plant to be plugged into it. The p-direction controller designed using the pole placement method determines the level of damping in the closed-loop system. The stability of the overall scheme then comes from the convergent optimization algorithm. The cost function as well as FSF are here the crucial elements shaping the noise robustness of the system. However, a more serious noise may require the p-direction controller to be retuned and this in turn affects the k-direction controller that should also be readjusted, i.e. the learning rate should be modified to work out a new compromise between the responsiveness of the controller and the steady state performance. The k-direction controller has a strong averaging capability due to the character of the cost function (3), which is desirable in the case of white noise. On the other hand, the p-direction controller is quite sensitive to noise if its poles are moved far to the left, i.e. high FSF controller gains are introduced to get strong damping. The highly damped plant makes the optimization task less challenging and hence results in faster convergence – but only if subject to no measurement noise. It is hard to propose a definitive analysis to determine operational noise limits. Nevertheless it has been tested that the dynamic neural optimization is still effective under severe noise of 10 % (Figs. 7b and 7d), which is way above noise levels encountered in practical converters. No measurement signal conditioning is introduced in any of the discussed case scenarios. Clearly, in a steady state the RMSE can drop far below the RMS of noise (see e.g. Fig. 7a or Fig. 7b); this happens due to the low-pass characteristics of the plant itself as well as thanks to the averaging nature of the employed functional. Further study will also include a methodical comparison of levels of immunity against extreme cases of noise for all mentioned algorithms.

Conclusion

A novel robust repetitive neurocontroller with a truly global iterative learning law has been proposed. The advantages of a recently developed disturbance dual feedforward concept have been briefly summarised. The performance of the controller has been studied within the context of a pure sine wave inverter and recommendations regarding training algorithms have been made. It has been demonstrated that the resilient backpropagation algorithm – a frequent winner in the case of non-repetitive online-trained neurocontrollers – is not a definitive winner in the case of the repetitive neurocontroller. Other learning algorithms can operate with similar effectiveness in terms of execution time and offer potentially faster convergence rates in this particular application. Important aspects for future study will include the identification of operational limits in unusually noisy environments and an experimental verification of the concept.

References

[1] Cichy, B., Galkowski, K., Rogers, E., and Kummert, A.: Control law design for discrete linear repetitive processes with non-local updating structures, *Multidimensional Systems and Signal Processing*, vol. 24, no. 4, 2013, pp. 707–726.

[2] Cichy, B., Galkowski, K., and Rogers, E.: 2D systems based robust iterative learning control using noncausal finite-time interval data, *Systems & Control Letters*, vol. 64, no. 0, 2014, pp. 36–42.

[3] Longman, R. W.: Iterative/repetitive learning control: learning from theory, simulations, and experiments, *Encyclopedia of the Sciences of Learning*, Springer US, 2012, pp. 1652–1657.

[4] Elci, H., Longman, R., Phan, M., Juang, J.-N., and Ugoletti, R.: Simple learning control made practical by zero-phase filtering: applications to robotics, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 49, no. 6, 2002, pp. 753–767.

[5] Shi, Y.: *Robustification in repetitive and iterative learning control*, PhD thesis, Columbia University, USA, 2013.

[6] Verwoerd, M. H. A.: *Iterative learning control – a critical review*, PhD thesis, University of Twente, The Netherlands, 2005.

[7] Ufnalski, B. and Grzesiak, L. M.: A performance study on synchronous and asynchronous update rules for a plug-in direct particle swarm repetitive controller, *Archives of Electrical Engineering*, vol. 63, no. 4, 2014, pp. 635–646.

[8] Chen, Y. Q., Moore, K., and Bahl, V.: Learning feedforward control using a dilated B-spline network: frequency domain analysis and design, *IEEE Transactions on Neural Networks*, vol. 15, no. 2, 2004, pp. 355–366.

[9] Deng, H., Oruganti, R., and Srinivasan, D.: Neural controller for UPS inverters based on B-spline network, *IEEE Transactions on Industrial Electronics*, vol. 55, no. 2, 2008, pp. 899–909.

[10] Ufnalski, B. and Grzesiak, L. M.: Artificial neural network based voltage controller for the single phase true sine wave inverter – a repetitive control approach, *Electrical Review (Przegląd Elektrotechniczny)*, vol. 89, no. 4, 2013, pp. 14–18.

[11] Ufnalski, B. and Grzesiak, L. M.: Particle swarm optimization of an online trained repetitive neurocontroller for the sine-wave inverter, *39th IECON Annual Conference of the IEEE Industrial Electronics Society*, 2013, pp. 6003–6009.

[12] Kaminski, M., Orlowska-Kowalska, T., and Szabat, K.: Neural speed controller based on two state variables applied for a drive with elastic connection, *16th International Power Electronics and Motion Control Conference and Exposition (PEMC)*, 2014, pp. 610–615.

[13] Orlowska-Kowalska, T. and Kaminski, M.: Adaptive Neurocontrollers for Drive Systems: Basic Concepts, Theory and Applications, *Advanced and Intelligent Control in Power Electronics and Drives*, ed. by Orlowska-Kowalska, T., Blaabjerg, F., and Rodriguez, J., vol. 531, Studies in Computational Intelligence, Springer International Publishing, 2014, pp. 269–302.

[14] Ufnalski, B., Grzesiak, L. M., and Kaszewski, A.: Advanced Control and Optimization Techniques in AC Drives and DC/AC Sine Wave Voltage Inverters: Selected Problems, *Advanced and Intelligent Control in Power Electronics and Drives*, ed. by Orlowska-Kowalska, T., Blaabjerg, F., and Rodriguez, J., vol. 531, Studies in Computational Intelligence, Springer International Publishing, 2014, pp. 303–333.

[15] Pajchrowski, T. and Zawirski, K.: Application of artificial neural network for adaptive speed control of PMSM drive with variable parameters, *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, vol. 32, no. 4, 2013, pp. 1287–1299.

[16] Ufnalski, B. and Grzesiak, L. M.: Particle swarm optimization of artificial-neural-network-based on-line trained speed controller for battery electric vehicle, *Bulletin of the Polish Academy of Sciences: Technical Sciences*, vol. 60, no. 3, 2012, pp. 661–667.

[17] Ufnalski, B.: *Repetitive Neurocontroller with Disturbance Feedforward*, 2014, URL: `www.mathworks.com/matlabcentral/fileexchange/47867-repetitive-neurocontroller-with-disturbance-feedforward`.

[18] Franklin, G., Powell, D., and Workman, M.: *Digital control of dynamic systems*, 3rd, Prentice Hall, 1997.

[19] Kaszewski, A., Ufnalski, B., and Grzesiak, L. M.: An LQ controller with disturbance feedforward for the 3-phase 4-leg true sine wave inverter, *IEEE International Conference on Industrial Technology (ICIT)*, 2013, pp. 1924–1930.

[20] MathWorks: *Neural Network Toolbox (MATLAB/Simulink)*, 2015, URL: `www.mathworks.com/help/nnet`.

[21] MathWorks: *Nguyen-Widrow layer initialization function*, 2015, URL: `www.mathworks.com/help/nnet/ref/initnw.html`.

[22] Pajchrowski, T., Zawirski, K., and Nowopolski, K.: A neural speed controller trained on-line by means of modified RPROP algorithm, *IEEE Transactions on Industrial Informatics*, vol. 11, no. 2, 2015, pp. 560–568.