Backpropagation algorithm with fractional derivatives

Zbigniew Gomolka[*]

University of Rzeszow, Faculty of Mathematics and Natural Sciences Department of Computer Engineering, ul. Pigonia 1, 35-959 Rzeszow, Poland

Abstract. The paper presents a model of a neural network with a novel backpropagation rule, which uses a fractional order derivative mechanism. Using the Grunwald Letnikow definition of the discrete approximation of the fractional derivative, the author proposed the smooth modeling of the transition functions of a single neuron. On this basis, a new concept of a modified backpropagation algorithm was proposed that uses the fractional derivative mechanism both for modeling the dynamics of individual neurons and for minimizing the error function. The description of the signal flow through the neural network and the mechanism of smooth shape control of the activation functions of individual neurons are given. The model of minimization of the error function is presented, which takes into account the possibility of changes in the characteristics of individual neurons. For the proposed network model, example courses of the learning processes are presented, which prove the convergence of the learning process for different shapes of the transition function. The proposed algorithm allows the learning process to be conducted with a smooth modification of the shape of the transition function without the need for modifying the IT model of the designed neural network. The proposed network model is a new tool that can be used in signal classification tasks.

1 Introduction

Here we briefly recall some approaches to neural networks connected with gradient methods of error minimalization. The algorithm of backpropagation was firstly pro-posed by the Finnish student Seppo Linnainmaa in 1970, but without any indication of its possible relationship with neural networks [1, 2]. In the history of the development of neural networks, this was the period directly after the 1969 publication of Minsky and Papert titled *Perceptrons*, in which they presented proof of the limitations of the network of linear perceptrons for solving linearly non-separable tasks, eg XOR [1, 3]. Paradoxically, this fact initiated a period of stagnation, in which the idea of neural networks development was frozen for several years, becoming an un-fulfilled dream of artificial intelligence applications. Fortunately at the beginning of the 1980s a novel approach was presented (in some aspects earlier by Grossberg in 1973 [3]) by Werbos in 1982, Parker 1985, and LeCun in 1985, which overcame the non-linear XOR problem by using the mechanism of errors

[*] Corresponding author: zgomolka@ur.edu.pl

minimization via gradient descent, specifically using complex derivative of error function [1,3 - 9]. The classical backpropagation algorithm that uses this mechanism of minimizing the error function assumes that the expression within the error function is differentiable. Because in such notation, the error function is a complex function, there is a need among other things to differentiate the transfer function of a single neuron, hence the key limitation of the possible applications of transient functions that must satisfy the condition of differentiability. In practice, more or less accurate approximations of transition functions are presumed, which in consequence leads to a simplification of the general model of the network. In addition, the assumed network structure forces the use of the same IT model, which includes the numerical implementation of the derivative of the transfer function of a single neuron, eg *sigmoid*, *tanh*, Gauss, log. In the author's opinion, this is a limitation which is a significant simplification of both the dynamics of individual neurons and the strategy of conducting their learning processes. Taking into account the above and assuming that biological neural cells can adopt any transitional character, the idea arose to design a mechanism that would allow for a smooth modeling of their dynamics. The concept of the smoothly changed transfer functions was already presented in works [10] and [11]. The main assumption is that the mathematical model of a neuron that uses base functions derived from a number of sigmoidal functions that assumes a dynamic change depends on the order of the used derivative, which in the general case may have a non-integer value.

2 Model of the Fractional Back-Prop network

Let us assume the following model of the L layered neural network, for which the model with fractional order derivative mechanism [13-15] will be given (see Fig.1.). Excluding transfer functions this model resembles the classic model of the feedforward network in which the input signal is presented by the matrix of input vectors: $P=\left[\vec{p}^{1},\ldots,\vec{p}^{q},\ldots,\vec{p}^{Q}\right]$ where $q\in\langle 1,Q\rangle$, Q denotes the number of vectors in the training set, whereas $A_L=\left[\vec{a}_L^{1},\ldots,\vec{a}_L^{q},\ldots,\vec{a}_L^{Q}\right]$ is the matrix of the network response vectors for the excitation $\vec{p}$ in the input layer. The input signal of the network is defined as: $\vec{p}^{q}=\left[p^{q}(1),\ldots,p^{Q}(R)\right]^T$ while the output signal in the l layer is: $\vec{a}_L^{q}=\left[a_L^{q}(1),\ldots,a_L^{q}(N)\right]^T$ where R denotes the number of receptors, N stands for the number of network outputs. The form of the expected value vector matrix was similarly defined as $T\left(\left[\vec{t}^{1},\ldots,\vec{t}^{q},\ldots,\vec{t}^{Q}\right]\right)$ where consecutive vectors $\vec{t}^{q}$ are of the form: $\vec{t}^{q}=\left[t^{q}(1),\ldots,t^{q}(N)\right]^T$. For the sake of notation simplicity, it was assumed that in a given learning step the q index can be omitted.

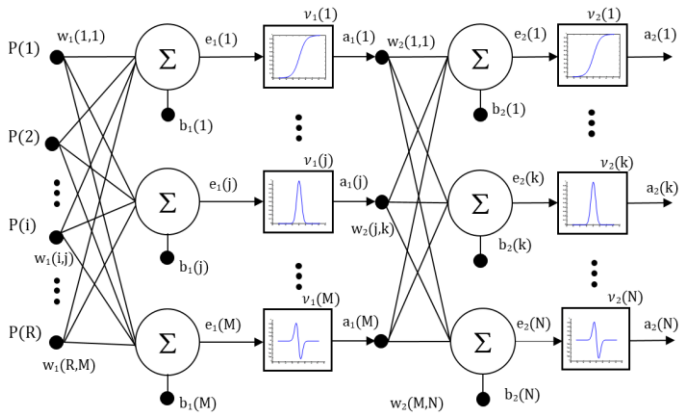

Fig. 1. Presumed model of the network

The flow of the signal within the network can be described as following. The input signal to the $j-th$ neuron in the first layer equals:

$$e_1(j)=\sum_{i=1}^{R} w_1(i,j)P(i)+b_1(j) \tag{1}$$

where: $w_1(i,j)$ - element of the matrix of weights connecting the receptor layer with the first layer of neurons, i - the number of the receptor on the input layer, j - the number of the neuron in the input layer, $p(i)$ - denotes $i-th$ element of the input vector, $b_1(j)$ - $j-th$ vector element of bias values in the first layer. Activation of neurons $a_1(j)$ in the first layer is expressed as follows:

$$a_1(j)=f\left(e_1(j)\right) \tag{2}$$

where: $f_B\left(e_1(j)\right)$ - is the neurons transfer function in the first layer. Similarly, the input signal for $k-th$ neuron in the second layer is equal to:

$$e_2(k)=\sum_{j=1}^{M} w_2(i,j)a_1(j)+b_2(k) \tag{3}$$

where: $w_2(j,k)$ - element of the matrix of weights connecting the first layer with the second layer, $b_2(k)$ - $k-th$ vector element of threshold values in the second layer. Similarly, neuron activation in the output layer is:

$$a_2(k)=f\left(e_2(k)\right) \tag{4}$$

Typically, the bias vector might be placed in the weight matrix, then the beginning of the i and j index notation changes respectively.

$$a_2(k)=f\left(\sum_{j=0}^{M} w_2(j,k)f\left(\sum_{i=0}^{R} w_1(i,j)p(i)\right)\right) \tag{5}$$

In the presented model (differently than in the classic approach), the transition function of a single neuron is taken as a Grünwald- Letnikov fractional derivative (GL) [16- 20] of log base function. Based on the definition of the integer derivative and the fractional derivative, the GL derivative is described by the formula:

$${}_{e_0}D_e^v f(e)=\lim_{h\to 0}\frac{1}{h^v}\sum_{n=0}^{\left[(e-e_0)/h\right]}(-)^n\binom{v}{n}f_B(e-nh) \tag{6}$$

where $\binom{v}{n}$ denotes the Newton binomial, v – order of fractional derivative of basis function $f_B(x)$, e_0 – the interval range, h - step of discretization. For the given discrete function $f(e)$ of real variable e defined on the interval $\langle e_0, e\rangle$, where $0 \le e_0 \le e$, we assume the backward difference of fractional order v ${}_{e_0}\Delta_e^{(v)}$ where $v \in R^+$:

$$ {}_{e_0}\Delta_e^{(v)} f(e) = \sum_{n=0}^{(e-e_0)/h} a_n^{(v)} f_B(e - nh) \tag{7} $$

where consecutive coefficients $a_n^{(v)}$ are defined as follow:

$$ a_n^{(v)} = \begin{cases} 1 & dla \quad n = 0 \\ (-1)^n \dfrac{v(v-1)(v-2)\ldots(v-n+1)}{n!} & dla \quad n = 1,2,3,\ldots,N \end{cases} \tag{8} $$

N stands for the number of discrete measurements and $f_B(e)$, which might be defined as:

$$ f_B(e) = \left(\log\left(1 + \exp(\beta e)\right)\right) \tag{9} $$

where $\beta = 1$ denotes the inclination coefficient. The set of the possible basis functions as well as their retrieval has been shown in [11]. When $h \to 0$ and $v = 1$ this GL derivative of $f_B(e)$ becomes:

$$ {}_{e_0}D_e^1 f_B(e) = \frac{1}{1 + \exp(-e)} \tag{10} $$

The BP learning method is based on the minimization of the error function, which takes the form for the presented architecture as follow:

$$ E = \frac{1}{2}\sum_{kq}\left[t^q(k) - {}_{e_0}D_e^1 f_B\left(\sum_{j=0}^{M} w_2^q(j,k)\, {}_{e_0}D_e^1 f_B\left(\sum_{i=0}^{R} w_1^q(i,j)\, p^q(i) \right) \right) \right]^2 \tag{11} $$

where q - means the number of the input vector, for which the appropriate vector is desired at the output of the network t^q. This is the normal equation of the least squares method (LSM). The expected change in the weight values, which minimize the error, is expressed:

$$ w^{(q+1)} - w^{(q)} = \Delta w^{(q)} = -\eta \frac{\partial E^{(q)}}{\partial w} \tag{12} $$

where: η – the speed of learning. Thus, starting from the output layer, respectively, we can rewrite:

$$ \Delta w_2^q(j,k) = -\eta \frac{\partial E}{\partial w_2^q(j,k)} = -\eta \frac{\partial E}{\partial a_2^q} \frac{\partial a_2^q}{\partial e_2^q} \frac{\partial e_2^q}{\partial w_2^q(j,k)} \tag{13} $$

$$ \frac{\partial E}{\partial a_2^q} = -\left[t^q(k) - a_2^q(k) \right] \tag{14} $$

$$ \frac{\partial a_2^q}{\partial e_2^q} = {}_{e_0}D_e^{v+1} f_B(e) \tag{15} $$

$$ \frac{\partial e_2^q}{\partial w_2^q(j,k)} = a_1^q(j) \tag{16} $$

By introducing additional notations $\delta_2^q(k)$

$$\delta_2^q(k)=\left[t^q(k)-a_2^q(k)\right]\frac{\partial a_2^q}{\partial e_2^q} \tag{17}$$

We can write the final equation of weight correction in the output layer as:

$$\Delta w_2^q(j,k)=-\eta\frac{\partial E}{\partial w_2^q(j,k)}=\eta a_1^q(j)\delta_2^q(k) \tag{18}$$

For $v=1$ and $h\to 1$, regarding (10) and additivity property of derivatives [21 - 26] we have:

$${}_{e_0}D_e^{v+1}f_B(e)={}_{e_0}D_e^{v}\left({}_{e_0}D_e^{1}f_B(e)\right)={}_{e_0}D_e^{1}f_B(e)\left(1-{}_{e_0}D_e^{1}f_B(e)\right) \tag{19}$$

or directly:

$${}_{e_0}D_e^{v+1}f_B(e)={}_{e_0}D_e^{2}f_B(e) \tag{20}$$

For the first layer, the considerations are analogous

$$\Delta w_1^q(i,j)=-\eta\frac{\partial E}{\partial a_1^q}\frac{\partial a_1^q}{\partial e_1^q}\frac{\partial e_1^q}{\partial w_1^q(i,j)} \tag{21}$$

with the exception of the following component described by the measure of the next layer error:

$$\frac{\partial E}{\partial a_1^q}=\sum_k w_2^q(j,k)\left(\left[t^q(k)-a_2^q(k)\right]\frac{\partial a_2^q}{\partial e_2^q}\right) \tag{22}$$

$$\frac{\partial a_1^q}{\partial e_1^q}={}_{e_0}D_e^{v+1}f_B(e) \tag{23}$$

$$\frac{\partial e_1^q}{\partial w_1^q(i,j)}=p^q(i) \tag{24}$$

$$\begin{aligned}\Delta w_1^q(i,j)&=\eta p^q(i)\left\{\sum_k w_2^q(j,k)\left(\left[t^q(k)-a_2^q(k)\right]\frac{\partial a_2^q}{\partial e_2^q}\right)\frac{\partial a_1^q}{\partial e_1^q}\right\}=\\ &\eta p^q(i)\left(\sum_k w_2^q(j,k)\delta_2^q(k)\frac{\partial a_1^q}{\partial e_1^q}\right)\end{aligned} \tag{25}$$

Introducing additional similar designation $\delta_1^q(j)$ we have:

$$\delta_1^q(j)=\sum_k w_2^q(j,k)\frac{\partial a_1^q}{\partial e_1^q}\delta_2^q(k) \tag{26}$$

Inserting into the expression defining ${}_{\Delta}w_1^q(i,j)$ we get finally:

$${}_{\Delta}w_1^q(i,j)=-\eta\frac{\partial E}{\partial w_1^q(i,j)}=\eta p^q(i)\delta_1^q(j) \tag{27}$$

Based on the presented considerations, the general formula for weight modification for L layered neural network with modified transition function can be given:

$$w_1^q(i,j)=w_1^{q-1}(i,j)+\eta\delta_j^l a_i^l \tag{28}$$

where the value δ_j^i, while presenting the $q-th$ pattern, will be written as:

$$\delta_j^i=\begin{cases}\left(t_j-a_j^l\right){}_{e_0}D_e^{v+1}f_B(e) & for\quad l=L\\ {}_{e_0}D_e^{v+1}f_B(e)\sum_{k=0}^{N^{l+1}}\delta_k^{l+1}w_{l+1}(j,k) & for\quad l<L\end{cases}\quad then\quad j\equiv k \tag{29}$$

where: l - means the number of the considered layer, j - the number of the neuron in layer l, k - is the number of neurons in the layer $l+1$, N - is the number of neurons in layer l. In the above considerations, it was assumed that in the presented network architecture the parameter v is the same in each layer and for individual neurons because of the first stage of the presented investigations.

2.1 FBP net under the XOR problem

For the accepted network model with fractional backpropagation (FBP) mechanism, experiments were carried out to examine the convergence of the proposed learning algorithm. The XOR problem was assumed as the input task for the neural network.

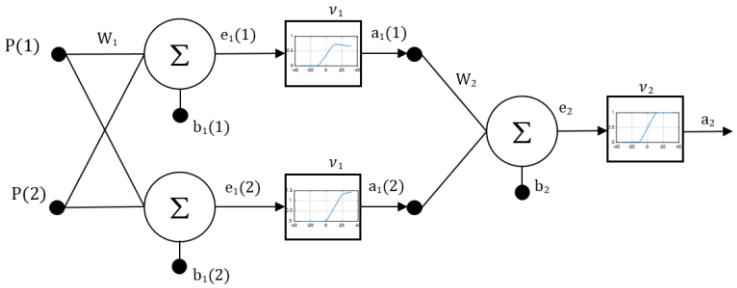

Fig. 2. FBP network architecture for the XOR problem.

The diagram of the network designed for this purpose is shown in Fig. 2. It has been assumed that for a randomly selected set of weights and bias $S\{w_1, b_1, w_2, b_2\}$ a learning process will be performed for successive smoothly changing values of parameter ν in the presumed range $0 \le v \le 1.1$ for comparisons in series of tests. The network uses a discrete approximation of the fractional GL derivative of the log base (9) function f_B in case of shape and derivative acquiring respectively:

$$ {}_{e_0}D_e^{(v)} f_B(e) = \lim_{\substack{h\to 0 \\ e-e_0=kh}} \frac{1}{h^v} \left[a_0^{(v)} \cdots \; {}_N^{(v)} \right] \begin{bmatrix} f_B(e) \\ f_B(e-h) \\ \vdots \\ f_B(e-kh) \end{bmatrix} \tag{30} $$

$$ {}_{e_0}D_e^{(v+1)} f_B(e) = \lim_{\substack{h\to 0 \\ e-e_0=kh}} \frac{1}{h^{v+1}} \left[a_0^{(v+1)} \cdots \right] \begin{bmatrix} f_B(e) \\ f_B(e-h) \\ \vdots \\ f_B(e-kh) \end{bmatrix} \tag{31} $$

The initial values of $S\{w_1, b_1, w_2, b_2\}$ have been shown in Table 1 and these values are common to each subsequent learning process. Those values have been obtained with standard randomizing procedure within the Matlab environment.

Table 1. Initial values of weights and biases.

w_1	b_1	w_2	b_2
$\begin{bmatrix} -0.6437 & 0.3816 \\ 0.2959 & -0.5189 \end{bmatrix}$	$\begin{bmatrix} 0.3594 \\ 0.9453 \end{bmatrix}$	$\begin{bmatrix} 0.4676 \\ 0.8049 \end{bmatrix}$	$\begin{bmatrix} -0.3847 \end{bmatrix}$

3 Results

In the experimental part of the work, it was assumed that the proposed FBP model will be tested in the task of solving the XOR problem. This is a task in which both the ability of the network to solve non-linearly separable tasks can be demonstrated as well as the convergence of its learning process can be tested. Using the initial values of the weights and biases, a number of learning processes were carried out and the obtained weights with biases are presented in Table 2.

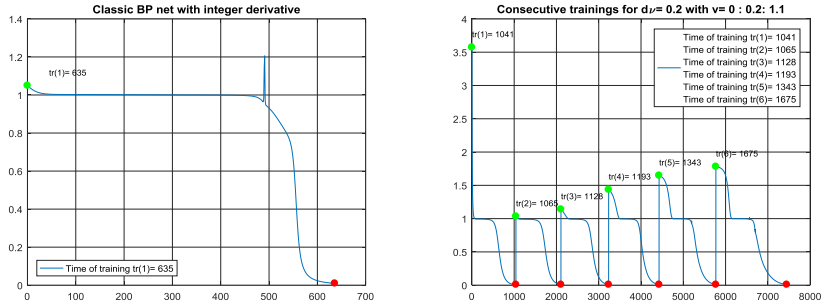

Fig. 3. The process of learning in classic BP network and FBP network.

In the initial phase of the experiments, we checked in general whether or not it is possible to obtain the convergence of the FBP algorithm for non-integer values of base function derivatives. The left part of Figure 3 shows the course of the learning process for the classical BP algorithm performed with the set of initial weights and biases set from Table 1. At the right part the course of the learning process in the FBP network for different values of the factor ν is shown. The BP network used for com-parisons uses the momentum mechanism and adaptive change of the learning rate coefficient.

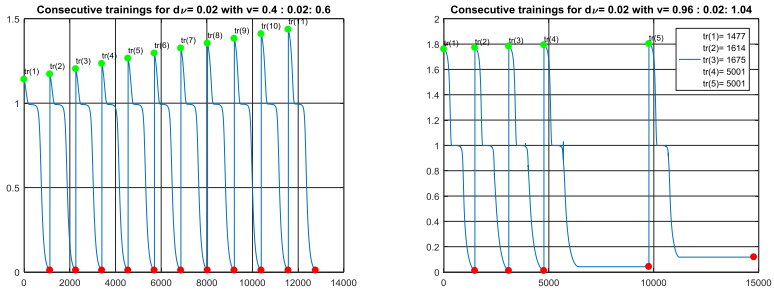

Fig. 4. The process of learning in FBP network with changing values of ν

The green circle marks denotes the beginning of the learning process, while the red circles indicates the end of the learning process. Figure Fig. 4 presents a juxtaposition of the

learning processes of the designed network while changing values of the order of the fractional derivative.

Table 2. Exemplary set of output weights and biases

v	w_1	b_1	w_2	b_2
0	$\begin{bmatrix} -1.7680 & 1.9924 \\ 1.7593 & -2.0166 \end{bmatrix}$	$\begin{bmatrix} 0.2737 \\ 0.7265 \end{bmatrix}$	$\begin{bmatrix} 4.8772 \\ 3.7517 \end{bmatrix}$	$\begin{bmatrix} -10.9198 \end{bmatrix}$
0.4	$\begin{bmatrix} -2.4017 & 2.5566 \\ 2.4331 & -2.6003 \end{bmatrix}$	$\begin{bmatrix} 0.5253 \\ 0.9094 \end{bmatrix}$	$\begin{bmatrix} 24.6784 \\ 21.2607 \end{bmatrix}$	$\begin{bmatrix} -11.6261 \end{bmatrix}$
0.8	$\begin{bmatrix} -6.4156 & 6.7210 \\ 6.4970 & -6.8104 \end{bmatrix}$	$\begin{bmatrix} 1.6580 \\ 1.8947 \end{bmatrix}$	$\begin{bmatrix} 109.1401 \\ 103.2894 \end{bmatrix}$	$\begin{bmatrix} -16.6733 \end{bmatrix}$
0.95	$\begin{bmatrix} -9.2876 & 9.6286 \\ 9.3702 & -9.7175 \end{bmatrix}$	$\begin{bmatrix} 2.4140 \\ 2.6122 \end{bmatrix}$	$\begin{bmatrix} 178.4342 \\ 171.7803 \end{bmatrix}$	$\begin{bmatrix} -19.5039 \end{bmatrix}$
1.00	$\begin{bmatrix} -10.9679 & 11.3125 \\ 11.0490 & -11.4003 \end{bmatrix}$	$\begin{bmatrix} 2.6919 \\ 2.8678 \end{bmatrix}$	$\begin{bmatrix} 216.6709 \\ 209.9594 \end{bmatrix}$	$\begin{bmatrix} -21.0725 \end{bmatrix}$
1.05	$\begin{bmatrix} -13.0336 & 13.3789 \\ 13.1125 & -13.4647 \end{bmatrix}$	$\begin{bmatrix} 2.9994 \\ 3.1545 \end{bmatrix}$	$\begin{bmatrix} 262.5784 \\ 255.8864 \end{bmatrix}$	$\begin{bmatrix} -22.7550 \end{bmatrix}$

In the initial phase of the experiments, two exemplary variation ranges of v were adopted. In the first case for $v \in \langle 0.4, dv, ..., 0.6 \rangle$, where $dv = 0.02$. In the second case the network convergence process for $v \in \langle 0.96, dv, ..., 1.04 \rangle$ is shown. The process of BP and FBP networks training was carried out by assuming the following general learning parameters: $ME = 5000$, $EG = 0.01$, $\eta = 0.01$, where ME is the maximum epochs number, EG network SSE error, η the learining rate coefficient.

The modification of weights was performed with the use of the *Quickprop* mechanism, i.e. adaptive change of the learning rate coefficient combined with the momentum mechanism. Table 2 presents a sample result sets of weights and biases respectively, which were obtained as a consequence of successive learning processes.

4 Conclusion

The presented paper presents the FBP network model using a fractional order derivative mechanism. The network uses a GL derivative to obtain a base function and to calculate a discrete approximation of its derivative in individual layers. The proposed network learning model is a new approach that overcomes the limitations associated with properties of the single neuron transfer function known in the current subject literature. For the proposed FBP network model, simulations of network convergence under the XOR task were performed. The courses of the learning process illustrated in Fig. 3 and Fig. 4 allow the conclusion to be drawn that it is possible to carry out the learning process using a derivative of fractional order. The resulting weight sets presented in Table 2 indicate that it is possible to achieve the same minimum error function for different values of the derivative order and the assumed shapes of the base function. The accuracy of the fractional derivative approximation has a key influence on the accuracy of the determination of the weights and biases sets. The new algorithm of the error function minimization can be used for various base functions without the need to modify the IT model of the NNet. As a continuation of the ongoing research, experiments should be undertaken to determine the optimal selection

of derivative approximation parameters for individual base functions. It is also necessary to examine the possibility of selecting specific base functions according to the class of the input signal being analyzed.

References

1. J.L. McClelland, Explorations in Parallel Distributed processing: A Handbook of Models, Programs, and Exercises, (2015)
2. J.Schmidhuber, Who Invented Backpropagation?, 2014 (updated 2015) http://people.idsia.ch/~juergen/who-invented-backpropagation.html
3. M Minsky, S Papert., Perceptrons: An Introduction to Computational Geometry, Expanded Edition Paperback, December 28, The MIT Press, Cambridge MA, ISBN 0-262-63022-2, (1987)
4. G. Drałus: Global models of dynamic complex systems – modelling using the multilayer neural networks, Annales UMCS Sectio AI Informatica Vol. 7, No 1 pp. 61-71, (2007)
5. E. Dudek-Dyduch: Algebraic Logical Meta-Model of Decision Processes - New Metaheuristics, Artificial Intelligence and Soft Computing Volume, ICAISC, pp 541-554, (2015)
6. Z. Gomolka, Neural networks in the analysis of fringe images (in Polish), PhD thesis, (2000)
7. Y. Kondratenko, V. Kondratenko: Soft Computing Algorithm for Arithmetic Multiplication of Fuzzy Sets Based on Universal Analytic Models, Information and Communication Technologies in Education, Research, and Industrial Application. Communications in Computer and Information Science, Springer International Publishing Switzerland, pp. 49-77, (2014)
8. T. Kwater, J. Bartman: Application of artificial neural networks in non-invasive identification of electric energy receivers, Progress in Applied Electrical Engineering (PAEE), Koscielisko, pp. 1-6, (2017)
9. B. Twaróg, Z. Gomółka, E Żesławska.: Time Analysis of Data Exchange in Distributed Control Systems Based on Wireless Network Model, Analysis and Simulation of Electrical and Computer Systems, Vol. 452, pp. 333-342,(2018)
10. Z. Gomolka, E. Dudek-Dyduch, Y.P Kondratenko.: From Homogeneous Network to Neural Nets with Fractional Derivative Mechanism, In: Artificial Intelligence and Soft Computing. ICAISC 2017. Lecture Notes in Computer Science, vol 10245, (2017)
11. Z. Gomolka, Neurons' Transfer Function Modeling with the Use of Fractional Derivative. In: Zamojski W., Mazurkiewicz J., Sugier J., Walkowiak T., Kacprzyk J. (eds) Contemporary Complex Systems and Their Dependability. DepCoS-RELCOMEX 2018. Advances in Intelligent Systems and Computing, vol 761. Springer, Cham, (2019)
12. U. Ghosh, S Sarkar., S. Das: Solution of system of linear fractional differential equations with modified derivative of Jumarie type, American Journal of Mathematical Analysis 3, No 3, pp. 72–84, (2015)
13. I. Moret: Shift-and-Invert Krylov Methods for Time-Fractional Wave Equations, Numerical Functional Analysis and Optimization, Vol. 36, Issue 1, pp. 86-103,(2015)
14. M. D. Ortigueira: Riesz potential operators and inverses via fractional centered derivatives, Int. J. Math. Math. Sci, Article ID 48391, pp. 1–12, (2006)

15. A. Oustaloup, F. Levron, B. Mathieu, F. M. Nanot: Frequency-band complex noninteger differentiator: characterization and synthesis, IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications I, Vol. 47, pp. 25–39,(2000)
16. D., Cao Labora, R. Rodriguez-Lopez From fractional order equations to integer order equations, Fractional Calculus and Applied Analysis, Vol. 20, No 6, pp. 1405-1423,(2017)
17. Y. Q. Chen, K. L. Moore: Discretization Schemes for Fractional-Order Differentiators and Integrators, IEEE Trans. On Circuits and Systems - I: Fundamental Theory and Applications, Vol. 49, No. 3, pp. 363–367, (2002)
18. H., Li, C. Ding: High-order algorithms for Riesz derivative and their applications, Numerical Methods for Partial Differential Equations, Vol. 33, Issue 5, (2017)
19. W.-H. Luo, C. Li, T.-Z Huang., Gu X.-M., Wu G.-C.: A High-Order Accurate Numerical Scheme for the Caputo Derivative with Applications to Fractional Diffusion Problems, Numerical Functional Analysis and Optimization, Vol. 39, Issue 5, pp. 600-622, (2018)
20. D. Xue: Computational Aspect of Fractional-Order Control Problems, Tutorial Workshop on Fractional Order Dynamic Systems and Controls, Proceedings of the WCICA'2010, China, (mechatronics.ece.usu.edu/foc/cdc10tw/code-matlab-simulink /xdy_foccode.rar), (2010)
21. D. Hengfei, L. Changpin, Ch. YangQuan: High-Order Algorithms for Riesz Derivative and Their Applications, Journal of Computational Physics, Vol. 293, 15, 2015, pp. 218-237
22. M. D. Ortigueira, J.A. T. Machado: What is a fractional derivative?, Journal of Computational Physics, Volume 293, 15 July 2015, pp. 4-13, (2015)
23. I. Petraš: Fractional Derivatives, Fractional Integrals, and Fractional Differential Equations in Matlab, Computer and Information Science – Engineering Education and Research Using MATLAB, ISBN 978-953-307-656-0, pp. 239-264, 2011
24. I. Podlubny: Matrix approach to discrete fractional calculus, Fractional Calculus and Applied Analysis, Vol. 3, pp. 359–386, (2000)
25. H. Sheng, Y. Li, Y. Q. Chen: Application of numerical inverse Laplace transform algorithms in fractional calculus, J. Franklin Inst., Vol. 348, pp. 315–330, (2011)
26. B. M. Vinagre, I Podlubny., Hernández A., Feliu, V.: Some approximations of fractional order operators used in control theory and applications, Fractional Calculus and Applied Analysis, Vol. 3, pp. 231–248, (2000)