Entropy **2013**, *15*, 4188-4198; doi:10.3390/e15104188

OPEN ACCESS

entropy

ISSN 1099-4300

www.mdpi.com/journal/entropy

Article

The Fractional Differential Polynomial Neural Network for Approximation of Functions

Rabha W. Ibrahim

Institute of Mathematical Sciences, University Malaya, Kuala Lumpur 50603, Malaysia;
E-Mail: rabhaibrahim@yahoo.com; Fax: +60 3-7967 3535

Received: 26 August 2013; in revised form: 5 September 2013 / Accepted: 24 September 2013 / Published: 29 September 2013

Abstract: In this work, we introduce a generalization of the differential polynomial neural network utilizing fractional calculus. Fractional calculus is taken in the sense of the Caputo differential operator. It approximates a multi-parametric function with particular polynomials characterizing its functional output as a generalization of input patterns. This method can be employed on data to describe modelling of complex systems. Furthermore, the total information is calculated by using the fractional Poisson process.

Keywords: fractional calculus; fractional differential equations; fractional polynomial neural network

1. Introduction

The Polynomial Neural Network (PNN) algorithm is one of the most important methods for extracting knowledge from experimental data and to locate its best mathematical characterization. The proposed algorithm can be utilized to analyze complex data sets with the objective to conclude internal data relationships and to impose knowledge about these relationships in the form of mathematical formulations (polynomial regressions). One of the most common types of PNN is the Group Method of Data Handling (GMDH) polynomial neural network created in 1968 by Professor Ivakhnenko at the Institute of Cybernetics in Kyiv (Ukraine).

Based on GMDH, Zjavka developed a new type of neural network called Differential Polynomial Neural Network (D-PNN) [1–4]. It organizes and designs some special partial differential equations, performing a complex system model of dependent variables. It makes a sum of fractional polynomial formulas, determining partial mutual derivative alterations of input variable combinations. This kind of

retreatment is based on learning generalized data connections. Furthermore, it offers dynamic system models a standard time-series prediction, as the character of relative data allow it to employ a wider range of input interval values than defined by the trained data. In addition, the advantages of differential equation solutions facilitate a major variety of model styles. The principle of this type is similar to the artificial neural network (ANN) construction [5,6].

Fractional calculus is a section of mathematical analysis that deals with considering real number powers or complex number powers of the differentiation and integration operators. The integrals are of convoluted form and exhibit power-law type kernels. It can be viewed as an experimenter for special functions and integral transforms [7–12]. It is well known that the physical interview of the fractional derivative is an open problem today. In [13], the author utilized fractional operators, in the sense of the Caputo differential operator, to define and study the stability of recurrent neural network (NN). In [14], Gardner employed a discrete fractional calculus to study Artificial Neural Network Augmentation. In [15], Almarashi used neural networks with a radial basis function method to solve a class of initial boundary values of fractional partial differential equations. Recently, Jalab *et al.*, applied the neural network method for finding the numerical solution for some special fractional differential equations [16]. Zhou *et al*. propoced a fractional time-domain identification algorithm based on a genetic algorithm [17], while Chen *et al*. studied the synchronization problem for a class of fractional-order chaotic neural networks [18].

Here, our aim is to introduce a generalization of the differential polynomial neural network utilizing fractional calculus. The fractional calculus is assumed in the sense of the Caputo differential operator. It approximates a multi-parametric function with particular polynomials characterizing its functional output as a generalization of input patterns. This method can be employed on data to describe modelling of complex systems [19].

2. Preliminaries

This section concerns with some basic preliminaries and notations regarding the fractional calculus. One of the most considerably utilized instruments in the theory of fractional calculus is provided by the Caputo differential operator.

Definition 2.1 The fractional (arbitrary) order integral of the function h of order $\beta > 0$ is defined by:

$$I_a^{\beta} h(t) = \int_a^t \frac{(t-\tau)^{\beta-1}}{\Gamma(\beta)} h(\tau) d\tau. \tag{1}$$

When $a = 0$, we write $I_a^{\beta} h(t) = h(t) * \chi_{\beta}(t)$, where $(*)$ denoted the convolution product (see [7]), $\chi_{\beta}(t) = \frac{t^{\beta-1}}{\Gamma(\beta)}, t > 0$ and $\chi_{\beta}(t) = 0, t \leq 0$ and $\chi_{\beta} \to \wp(t)$ as $\beta \to 0$ where $\wp(t)$ is the delta function.

Definition 2.2 The Riemann-Liouville fractional derivative of the function h of order $0 \leq \beta < 1$ is defined by:

$$D_a^{\beta} \mathrm{h}(t) = \frac{d}{dt} \int_a^t \frac{(t-\tau)^{-\beta}}{\Gamma(1-\beta)} \mathrm{h}(\tau) d\tau = \frac{d}{dt} I_a^{1-\beta} \mathrm{h}(t). \tag{2}$$

Remark 2.1 [7]

$$D^{\beta} t^{\mu} = \frac{\Gamma(\mu+1)}{\Gamma(\mu-\beta+1)} t^{\mu-\beta}, \mu > -1;\ 0 < \beta < 1 \tag{3}$$

and:

$$I^{\beta}t^{\mu} = \frac{\Gamma(\mu+1)}{\Gamma(\mu+\beta+1)}t^{\mu+\beta}, \mu > -1;\ \beta > 0. \tag{4}$$

The Leibniz rule is:

$$\begin{aligned} D_a^{\beta}[f(t)g(t)] &= \sum_{k=0}^{\infty}\binom{\beta}{k} D_a^{\beta-k}f(t)D_a^{k}g(t) \\ &= \sum_{k=0}^{\infty}\binom{\beta}{k} D_a^{\beta-k}g(t)D_a^{k}f(t). \end{aligned} \tag{5}$$

Definition 2.3. The Caputo fractional derivative fractional derivative of order β>0 is defined, for a smooth function *f* by:

$$\begin{aligned} f^{(\beta)}(x) &:= \frac{\partial^{\beta} f}{\partial x^{\beta}} :=^{c} D_x^{\beta} f(x) \\ &= \frac{1}{\Gamma(n-\beta)}\int_0^x \frac{f^{(n)}(\tau)}{(x-\tau)^{\beta-n+1}}d\tau. \end{aligned} \tag{6}$$

The local fractional Taylor formula has been generalized by many authors [20–22]. This generalization admits the following formula:

$$\begin{aligned} f(x+\Delta x) = f(x) + {}^{c}D_x^{\beta} f(x)\frac{(\Delta x)^{\beta}}{\Gamma(\beta+1)} + {}^{c}D_x^{\beta}\,{}^{c}D_x^{\beta} f(x)\frac{(\Delta x)^{2\beta}}{\Gamma(2\beta+1)} + \cdots \\ + {}^{c}D_x^{n\beta} f(x)\frac{(\Delta x)^{n\beta}}{\Gamma(n\beta+1)}, \end{aligned} \tag{7}$$

where ${}^{c}D^{\beta}{}_{x}$ is the Caputo differential operator and:

$${}^{c}D^{n\beta}{}_{x} = \underbrace{{}^{c}D^{\beta}{}_{x}\,{}^{c}D^{\beta}{}_{x}\cdots{}^{c}D^{\beta}{}_{x}}_{n-times} \tag{8}$$

3. Results

3.1. Proposed Method

The fractional differential polynomial neural network (FD-PNN) is based on an equation of the form:

$$a + \sum_{i=1}^{n} b_i \frac{\partial^{\beta} u}{\partial x_i^{\beta}} + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}\frac{\partial^{\beta} u}{\partial x_i^{\beta}}\frac{\partial^{\beta} u}{\partial x_j^{\beta}} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} d_{ijk}\frac{\partial^{\beta} u}{\partial x_i^{\beta}}\frac{\partial^{\beta} u}{\partial x_j^{\beta}}\frac{\partial^{\beta} u}{\partial x_k^{\beta}} + \ldots = 0, \tag{9}$$

where $u := f(x_1, x_2, \ldots, x_n)$ is a function of all input variables, a, b_i, c_{ij}, d_{ijk} are the polynomial coefficients. Solutions of fractional differential equations can be expressed in term of the Mittag-Leffler function:

$$E_\alpha(z) = \sum_{n=0}^{\infty} \frac{z^n}{\Gamma(1+n\alpha)} \tag{10}$$

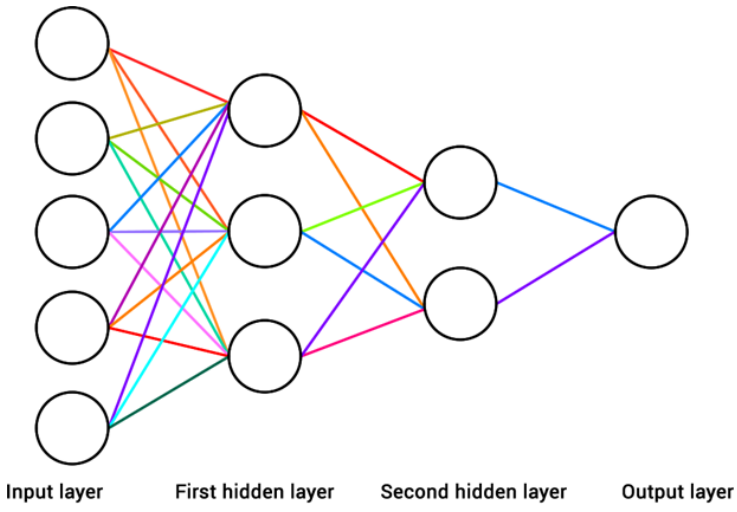
Recently, numerical routines for Mittag-Leffler functions have been developed, e.g., by Freed *et al.* [23], Gorenflo *et al.* [24] (with MATHEMATICA), Podlubny [25] (with MATLAB), Seybold and Hilfer [26].

We proceed to form sum derivative terms changing the fractional partial differential equation (9) by applying different math techniques, e.g. fractional wave series, [27]:

$$\begin{aligned} y_i^\beta &= \frac{(a_0 + a_1x_1 + a_2x_2 + \ldots + a_nx_n + a_{n+1}x_1x_2 + \ldots)^{\frac{m+\beta}{n}}}{b_0 + b_1x_1 + \ldots} \\ &= \frac{\partial^{m\beta} f(x_1, x_2, \ldots, x_n)}{\partial x_1^\beta \, \partial x_2^\beta \ldots \partial x_m^\beta}, \end{aligned} \tag{11}$$

where n refers to the combined degree of $n-$input variable polynomial of numerator; while m indicates to the combined degree of denominator w_t – weights of terms and y_i^β is the output neuron. Note that when $\beta \to 1$, Equation (11) reduces to Equation (4) in [4]. The fractional polynomials of fractional power (11), determining relations of n-input variables, appear summation derivative terms (neurons) of a fractional differential equation. The numerator of Equation (11) is a complete n-variable polynomial, which recognizes a new partial function u of Equation (9). The denominator of Equation (11) is a fractional derivative part, which implies a fractional partial change of some input variables combination. Equation (11) indicates a aingle output for fixed fractional power. Each layer of the FD-PNN contains blocks. These blocks stress fractional derivative neurons. For each fractional polynomial of fractional order formulates the fractional partial derivative depending on the change of some input variables. Each block implicates a unique fractional polynomial which forms its output access into the next hidden layer (Figure 1). For example of a system of the form : input layer, first hidden layer, second hidden layer and output layer; we may use $y_1^{1/4}$ to perform its output to the first layer; $y_2^{1/2}$ to execute its output to the second hidden layer and $y_3^{3/4}$ to carry out the last y of the system in the output layer.

Figure 1. GMDH-PNN.



Let there be a network with two inputs, formulating one functional output value y^β, then, for special values of β, the sum derivative terms is:

$$
\begin{aligned}
y^{1} &= w_1 \frac{a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2}{b_0 + b_1x_1} + w_2 \frac{a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2}{b_0 + b_1x_2}, (see\ [4]) \\
y^{3/4} &= w_1 \frac{(a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2)^{7/8}}{b_0 + b_1x_1} + w_2 \frac{(a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2)^{7/8}}{b_0 + b_1x_2} \\
y^{1/2} &= w_1 \frac{(a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2)^{3/4}}{b_0 + b_1x_1} + w_2 \frac{(a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2)^{3/4}}{b_0 + b_1x_2} \\
y^{1/4} &= w_1 \frac{(a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2)^{5/8}}{b_0 + b_1x_1} + w_2 \frac{(a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2)^{5/8}}{b_0 + b_1x_2};
\end{aligned} \tag{12}
$$

we realize that y^{β} includes only one block of two neurons, terms of both fractional derivative variables x_1 and x_2. Table 1 shows approximation errors (y-axis) of the trained network, *i.e.* differences of the true and estimated function, to random input vectors with dependent variables.

Table 1. Approximation values of $f(x_1, x_2) = x_1 + x_2$.

Data	Actual Value	Approximate Value	Absolute Error
(1,0)	1	$y^1 = y^{3/4} = \frac{3}{4}$	0.25
		$y^{1/2} = y^{1/4} = \frac{3}{4}$	0.25
(0,1)	1	$y^1 = y^{3/4} = \frac{3}{4}$	0.25
		$y^{1/2} = y^{1/4} = \frac{3}{4}$	0.25
(1,1)	1	$y^1 = 1.5$	0.5
		$y^{3/4} = 1.3$	0.3
		$y^{1/2} = 1.1$	0.1
		$y^{1/4} = 0.99$	0.01
(1/2,1/2)	1	$y^1 = 1.66$	0.66
		$y^{3/4} = 1.6$	0.6
		$y^{1/2} = 1.57$	0.57
		$y^{1/4} = 1.53$	0.53
		$y^{0.1} = 1.4$	0.4

The 3-variable FD-PNN (Table 2) for linear true function approximation (e.g., $f(x_1, x_2, x_3) = x_1 + x_2 + x_3$) may involve one block of six neurons, FDE terms of all 1 and 2-combination derivative variables of the complete FDE, e.g.:

$$
\begin{aligned}
y_1^{1} &= w_1 \frac{(a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_1x_2 + a_5x_1x_3 + a_6x_2x_3 + a_7x_1x_2x_3)^{2/3}}{b_0 + b_1x_1}, (see\ [4]) \\
y_1^{3/4} &= w_1 \frac{(a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_1x_2 + a_5x_1x_3 + a_6x_2x_3 + a_7x_1x_2x_3)^{7/12}}{b_0 + b_1x_1} \\
y_1^{1/2} &= w_1 \frac{(a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_1x_2 + a_5x_1x_3 + a_6x_2x_3 + a_7x_1x_2x_3)^{1/2}}{b_0 + b_1x_1} \\
y_1^{1/4} &= w_1 \frac{(a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_1x_2 + a_5x_1x_3 + a_6x_2x_3 + a_7x_1x_2x_3)^{5/12}}{b_0 + b_1x_1};
\end{aligned} \tag{13}
$$

and:

$$\begin{aligned} y_4^1 &= w_2 \frac{a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_1x_2 + a_5x_1x_3 + a_6x_2x_3 + a_7x_1x_2x_3}{b_0 + b_1x_1 + b_2x_2 + b_3x_1x_2}, (see\ [4]) \\ y_4^{3/4} &= w_2 \frac{(a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_1x_2 + a_5x_1x_3 + a_6x_2x_3 + a_7x_1x_2x_3)^{11/12}}{b_0 + b_1x_1 + b_2x_2 + b_3x_1x_2} \\ y_4^{1/2} &= w_2 \frac{(a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_1x_2 + a_5x_1x_3 + a_6x_2x_3 + a_7x_1x_2x_3)^{5/6}}{b_0 + b_1x_1 + b_2x_2 + b_3x_1x_2} \\ y_4^{1/4} &= w_2 \frac{(a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_1x_2 + a_5x_1x_3 + a_6x_2x_3 + a_7x_1x_2x_3)^{3/4}}{b_0 + b_1x_1 + b_2x_2 + b_3x_1x_2}; \end{aligned} \quad (14)$$

Table 2. Approximation values of $f(x_1, x_2, x_3) = x_1 + x_2 + x_3$.

Data	Actual Value	Approximate Value	Absolute Error
(1,0,0)	1	$y_4^1 = y_4^{3/4} = \frac{1}{2}$	0.5
		$y_4^{1/2} = y_4^{1/4} = \frac{1}{2}$	0.5
(0,1,0)	1	$y_4^1 = y_4^{3/4} = \frac{1}{2}$	0.5
		$y_4^{1/2} = y_4^{1/4} = \frac{1}{2}$	0.5
(0,0,1)	1	$y_4^1 = y_4^{3/4} = 1$	0
		$y_4^{1/2} = y_4^{1/4} = 1$	
(1,1,0)	1	$y_4^1 = 1.125$	0.125
		$y_4^{3/4}$=1.025	0.025
		$y_4^{1/2} = 1.873$	0.12
		$y_4^{1/4} = 0.936$	0.063
(1,0,1)	1	$y_4^1 = 1.5$	0.5
		$y_4^{3/4}$=1.368	0.368
		$y_4^{1/2} = 1.249$	0.249
		$y_4^{1/4} = 1.1$	0.1
(1,1,1)	1	$y_4^1 = 1.6$	0.6
		$y_4^{3/4}$=1.488	0.488
		$y_4^{1/2} = 1.26$	0.26
		$y_4^{1/4} = 1.0755$	0.0755

We proceed to compute approximations for non-linear functions. We let $u := f(x_1, x_2)$ be a function with square power variables, then we have:

$$F(x_1, x_2, u, \frac{\partial^\beta u}{\partial x_1^\beta}, \frac{\partial^\beta u}{\partial x_2^\beta}, \frac{\partial^{2\beta} u}{\partial x_1^{2\beta}}, \frac{\partial^{2\beta} u}{\partial x_2^{2\beta}}, \frac{\partial^{2\beta} u}{\partial x_1^\beta\, \partial x_2^\beta}) = 0. \quad (15)$$

For example, for $\beta = 1$, we get [4]:

$$\begin{aligned} y_{10}^{1} &= w_{10}\frac{a_0+a_1x_1+a_2x_2+a_3x_1^2+a_4x_2^2+a_5x_1x_2}{b_0+b_1x_1+b_2x_1^2} \\ &= \frac{\partial^2 f(x_1,x_2)}{\partial x_1^2} \\ y_{01}^{1} &= w_{01}\frac{a_0+a_1x_1+a_2x_2+a_3x_1^2+a_4x_2^2+a_5x_1x_2}{b_0+b_1x_2+b_2x_2^2} \\ &= \frac{\partial^2 f(x_1,x_2)}{\partial x_2^2}. \end{aligned} \tag{16}$$

In general, for fractional power β, we have:

$$\begin{aligned} y_{10}^{\beta} &= w_{10}\frac{(a_0+a_1x_1+a_2x_2+a_3x_1^2+a_4x_2^2+a_5x_1x_2)^{\frac{1+\beta}{2}}}{b_0+b_1x_1+b_2x_1^2} \\ &= \frac{\partial^{2\beta} f(x_1,x_2)}{\partial x_1^{2\beta}} \\ y_{01}^{\beta} &= w_{01}\frac{(a_0+a_1x_1+a_2x_2+a_3x_1^2+a_4x_2^2+a_5x_1x_2)^{\frac{1+\beta}{2}}}{b_0+b_1x_2+b_2x_2^2} \\ &= \frac{\partial^{2\beta} f(x_1,x_2)}{\partial x_2^{2\beta}}. \end{aligned} \tag{17}$$

For example:

$$\begin{aligned} y_{10}^{3/4} &= w_{10}\frac{(a_0+a_1x_1+a_2x_2+a_3x_1^2+a_4x_2^2+a_5x_1x_2)^{\frac{7}{8}}}{b_0+b_1x_1+b_2x_1^2} \\ y_{10}^{1/2} &= w_{10}\frac{(a_0+a_1x_1+a_2x_2+a_3x_1^2+a_4x_2^2+a_5x_1x_2)^{\frac{3}{4}}}{b_0+b_1x_1+b_2x_1^2} \\ y_{10}^{1/4} &= w_{10}\frac{(a_0+a_1x_1+a_2x_2+a_3x_1^2+a_4x_2^2+a_5x_1x_2)^{\frac{5}{8}}}{b_0+b_1x_1+b_2x_1^2}. \end{aligned} \tag{18}$$

3.2. Modified Information Theory

In this section, we try to measure the learning of the neuron of the system in Figure 1. We wish to improve a applicable measure of the information we get from observing the appearance of an event having probability p. The approach depends on the probability of extinction, which describes by the fractional Poisson process as follows [28]:

$$P_{\beta}(N,y) = \frac{(\sigma y)^N}{N!}\sum_{n=0}^{\infty}\frac{(n+N)!}{n!}\frac{(-\sigma y^{\beta})^n}{\Gamma(\beta(n+N)+1)}, \tag{19}$$

where $\sigma \in R$ is a physical coefficient, $\beta \in (0,1]$. Let N be the number of neurons, I be the average information and further that the source emits the symbols with probabilities $P_1, P_2, \ldots, P_N$, respectively such that $P_i = P_\beta(i,y)$. Thus we may compute the total information as follows:

$$I = \sum_{i=1}^{N} (NP_i\ ) \log\left(\frac{1}{P_i}\right). \tag{20}$$

The last assertion is modified work due to Shannon [29]. For example, to compute the average information of the system with *N*=3, for the last fractional derivative in Table 3, we have:

$$\begin{aligned} I = \sum_{i=1}^{3} (NP_i\ ) \log\left(\frac{1}{P_i}\right) &= 3P_1 \log\left(\frac{1}{P_1}\right) + 3P_2\ \log\left(\frac{1}{P_2}\right) + 3P_3 \log\left(\frac{1}{P_3}\right) \\ &\simeq 0.2408 - 0.09 - 0.051 = 0.051, \end{aligned} \tag{21}$$

where P_i converged to a hypergeometric function, which computed with the help of Maple.

Table 3. The approximation errors for $f(x_1, x_2) = (x_1 + x_2)^2$.

Data	Actual Value	Approximate Value	Absolute Error
(1,0)	2	$y_{10}^{1} = 2$	0
		$y_{10}^{3/4} = 1.834$	0.166
		$y_{10}^{1/2} = 1.681$	0.319
		$y_{10}^{1/4} = 1.5422$	0.4577
(0,1)	2	$y_{10}^{1} = 2$	0
		$y_{10}^{3/4} = 1.834$	0.166
		$y_{10}^{1/2} = 1.681$	0.319
		$y_{10}^{1/4} = 1.5422$	0.4577
(1,1)	2	$y_{10}^{1} = 2.49$	0.49
		$y_{10}^{3/4} = 2.04$	0.04
		$y_{10}^{1/2} = 1.665$	0.335
		$y_{10}^{1/4} = 1.336$	0.633

4. Discussion

The presented 2-variable FD-PNN (Table 1) is able to approximate any linear function, e.g., the simple sum $f(x_1, x_2) = x_1 + x_2$. The comparison processes with respect to D-PNN (normal case) showed that the proposed method converged to the exact values rapidly. For example, the case (1,1) implied ABE=0.01 at $y^{1/4}$. In this experiment, we let $b_0 = 1, w_1 = w_2 = 1$. Figure 2 shows the approximation of the fractional derivative for the function $f(x_1, x_2) = x_1 + x_2$. The x-axis represents to the values when $x_1 = x_2$. It is clear that the interval of convergence is [0.2,1]. The endowed 3-variable FD-PNN (Table 2) is qualified to approximate any linear function e.g. simple sum $f(x_1, x_2, x_3) = x_1 + x_2 + x_3$. The comparison procedure with respect to D-PNN displayed that the proposed method, of 3-variables, is converged swiftly to the exact values. For example, the case (1,1,0), with $w_2 = 3/2$ and (1,1,1), with $w_2 = 1$ yield ABE=0.063 and 0.0755 respectively at $y^{1/4}$. Furthermore, Figure 3 shows the interval of convergence at [0.3,1]. Here, we let $x_1 = x_2 = x_3$. Comparable argument can be concluded from the non-linear case, where Table 3 computes approximation values, by utilizing FD-PNN. For example, the data (1,1) give the best approximation at

$y^{3/4}$ when $w_{10} = 1.5$. In Figure 4, the x-axis performs to the value when $x_1 = x_2$. Obviously, the interval of convergence is [0.4,2].

Figure 2. Selected fractional approximation derivative of $f(x_1,x_2) = x_1 + x_2$.

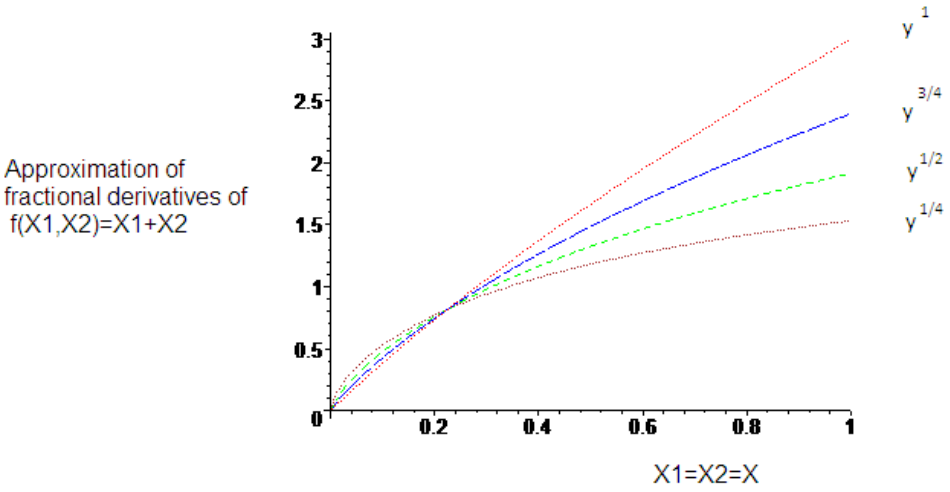


Figure 3. The fractional approximation y_4 of the function $f(x_1,x_2,x_3) = x_1 + x_2 + x_3$.

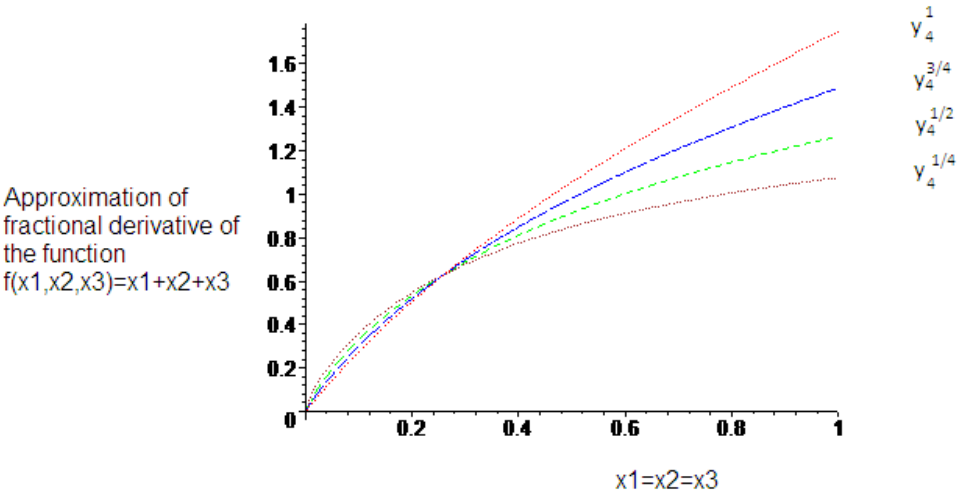
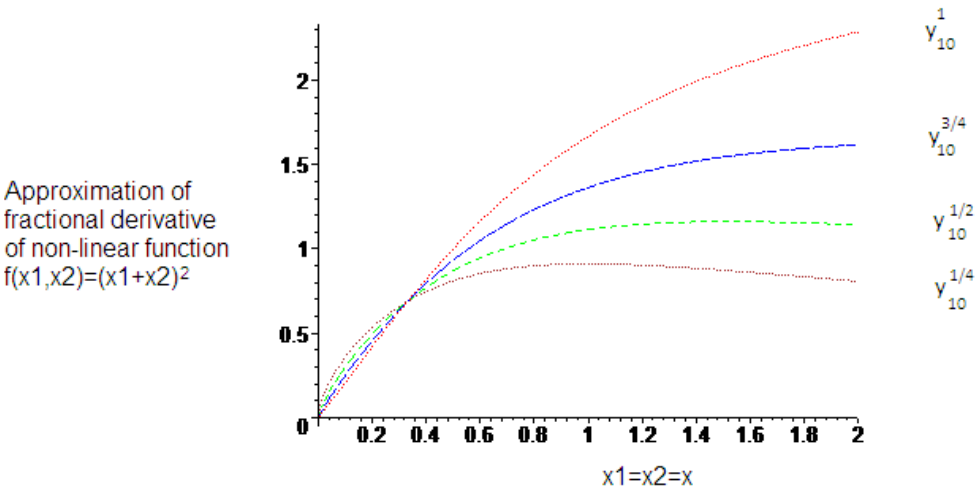


Figure 4. The fractional approximation y_{10} of the function $f(x_1,x_2) = (x_1 + x_2)^2$.



5. Conclusions

Based on GMDH-PNN (Figure 1) and modifying the work described in [4], we suggested a generalized D-PNN, called FD-PNN. The experimental results showed that the proposed method satisfies a quick approximation to the exact value comparison with the normal method. The generalization depended on the Riemann-Liouville differential operator. This method can be employed on data to describe modelling of complex systems. Next step, our aim is to modify this work by utilizing mixed D-PNN and FD-PNN, e.g. one can consider a function of the form:

$$F(x_1, x_2, u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \ldots \frac{\partial^{\beta} u}{\partial x_1^{\beta}}, \frac{\partial^{\beta} u}{\partial x_2^{\beta}}, \frac{\partial^{2\beta} u}{\partial x_1^{2\beta}}, \frac{\partial^{2\beta} u}{\partial x_2^{2\beta}}, \frac{\partial^{2\beta} u}{\partial x_1^{\beta}\, \partial x_2^{\beta}}, \ldots) = 0 \quad (22)$$

Acknowledgments

The author would like to thank the reviewers for their comments on earlier versions of this paper. This research has been funded by the University of Malaya, under Grant No. RG208-11AFR.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Zjavka, L. Generalization of patterns by identification with polynomial neural network. *J. Elec. Eng.* **2010**, *61*, 120–124.
2. Zjavka, L. Construction and adjustment of differential polynomial neural network. *J. Eng. Comp. Inn.* **2011**, *2*, 40–50.
3. Zjavka, L. Recognition of generalized patterns by a differential polynomial neural network. *Eng. Tech. Appl. Sci. Res.* **2012**, *2*, 167–172.
4. Zjavka, L. Approximation of multi-parametric functions using the differential polynomial neural network. *Math. Sci.* **2013**, *7*, 1–7.
5. Giles, C.L. Noisy time series prediction using recurrent neural networks and grammatical inference. *Machine Learning* **2001**, *44*, 161–183.
6. Tsoulos, I.;. Gavrilis, D.; Glavas, E. Solving differential equations with constructed neural networks. *Neurocomputing* **2009**, *72*, 2385–2391.
7. Podlubny, I. *Fractional Differential Equations*; Academic Press: New York, NY, USA, 1999.
8. Hilfer, R. *Application of Fractional Calculus in Physics*; World Scientific: Singapore, 2000.
9. West, B.J.; Bologna, M.; Grigolini, P. *Physics of Fractal Operators*; Academic Press: New York, NY, USA, 2003.
10. Kilbas, A.A.; Srivastava, H.M.; Trujillo, J.J. *Theory and Applications of Fractional Differential Equations*; Elsevier: Amsterdam, The Netherland, 2006.
11. Sabatier, J.; Agrawal, O.P.; Machado, T. Advance in Fractional Calculus: Theoretical Developments and Applications in Physics and Engineering; Springer: London, UK, 2007.

12. Lakshmikantham, V.; Leela, S.; Devi, J.V. *Theory of Fractional Dynamic Systems*; Cambridge Scientific Pub.: Cambridge, UK, 2009.
13. Jalab, J.A.; Ibrahim R.W. Stability of recurrent neural networks. *Int. J. Comp. Sci. Net. Sec.* **2006**, *6*, 159–164.
14. Gardner, S. Exploring fractional order calculus as an artifficial neural network augmentation. Master's Thesis, Montana State University, Bozeman, Montana, April 2009.
15. Almarashi, A. Approximation solution of fractional partial differential equations by neural networks. *Adv. Numer. Anal.* **2012**, *2012*, 912810.
16. Jalab, H.A.; Ibrahim, R.W.; Murad, S.A.; Hadid, S.B. Exact and numerical solution for fractional differential equation based on neural network. *Proc. Pakistan Aca. Sci.* **2012**, *49*, 199–208.
17. Zhou, S.; Cao, J.; Chen, Y. Genetic algorithm-based identification of fractional-order systems. *Entropy* **2013**, *15*, 1624–1642.
18. Chen, L.; Qu, J.; Chai Y.; Wu, R.; Qi, G. Synchronization of a class of fractional-order chaotic neural networks. *Entropy* **2013**, *15*, 3265–3276.
19. Ivachnenko, A.G. Polynomial Theory of Complex Systems. *IEEE Trans. Sys. Man Cyb.* **1971**, *4*, 364–378.
20. Kolwankar, K.M.; Gangal, A.D. Fractional differentiability of nowhere differentiable functions and dimensions. *Chaos*, **1996**, *6*, 505–513.
21. Adda, F.B.; Cresson, J. About non-differentiable functions. *J. Math. Anal. Appl.* **2001**, *263*, 721–737.
22. Odibat, Z.M.; Shawagfeh, N.T. Generalized Taylor's formula. *Appl. Math. Comp.* **2007**, *186*, 286–293.
23. Freed, A.; Diethelm, K.; Luchko, Y. Fractional-order viscoelasticity (FOV): Constitutive development using the fractional calculus. In *First Annual Report NASA/TM-2002-211914*; NASA's Glenn Research Center: Cleveland, OH, USA, 2002.
24. Gorenflo, R.; Loutchko, J.; Luchko, Y. Computation of the Mittag-Leffler function $E_{\alpha,\beta}(z)$ and its derivative. *Frac. Calc. Appl. Anal.* **2002**, *5*, 491–518.
25. Podlubny, I. Mittag-Leffler function, The MATLAB routine. http://www.mathworks.com/matlabcentral/fileexchange (accessed on 25 March 2009).
26. Seybold, H.J.; Hilfer, R. Numerical results for the generalized Mittag-Leffler function. *Frac. Calc. Appl. Anal.* **2005**, *8*, 127–139.
27. Ibrahim, R.W. Fractional complex transforms for fractional differential equations. *Adv. Diff. Equ.* **2012**, *192*, 1–11.
28. Casasanta, G.; Ciani, D.; Garra, R. Non-exponential extinction of radiation by fractional calculus modelling. *J. Quan. Spec. Radi. Trans.* **2012**, *113*, 194–197.
29. Shannon, C.E. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *Volume*, 379–423.