Failure and Power Utilization System Models of Differential Equations by Polynomial Neural Networks

Ladislav Zjavka
National supercomputing center IT4innovations
VŠB-Technical University of Ostrava
Ostrava, Czech Republic
lzjavka@gmail.com

Ajith Abraham[1,2]
[1]Machine Intelligence Research Labs,Washington, USA
[2]IT4innovations, VŠB-Technical University of Ostrava, Czech Republic
ajith.abraham@ieee.org

***Abstract*—Reliability modeling of electronic circuits can be best performed by the stressor – susceptibility interaction model. A circuit or a system is deemed to be failed once the stressor has exceeded the susceptibility limits. Complex manufacturing systems often require a high level of reliability from the incoming electricity supply. Modern industrial time power quality monitoring systems can be used for the pre-fault load alarming. Neural networks can successfully model and predict the failure frame of critical electronic systems and power utilization in power plants described only a few input quantities. Differential polynomial neural network is a new type of neural network, which constructs and substitutes an unknown general sum partial differential equation with a total sum of fractional polynomial terms. The system model describes partial relative derivative dependent changes of some input combinations of variables. This type of non-linear regression is based on trained generalized data relations decomposed by partial low order polynomials of 2-input variables. Experimental results indicate that the proposed method is efficient.**

Keywords-reliability modeling; power utilization; polynomial neural network; differential equation composition; multi-parametric function approximation

I. INTRODUCTION

Differential equation solutions can describe a variety of real data observation problems, which a unique single exact model is hardly to specify. They can apply sum series [1], genetic programming (GP) techniques [2] or an artificial neural network (ANN) construction [3]. Standard soft-computing methods are generally direct computational techniques, operating only within absolute interval values of input variables, e.g. GP or fuzzy models can compose a searched function using collections of operators and terminals from a defined set to form some symbolic expressions. A common ANN operating principle is based on entire similarity relationships of new presented input patterns with the trained ones. If a modeled multi-parametric function involves some similar harmonic parts, then a benefit of ANN solution might become evident. ANN models are solid and simple but impossible to be expressed in the form of a math description, the models appear to the users as a “black box”.

Differential polynomial neural network (D-PNN) is a new neural network type designed by Zjavka [4], which results from the GMDH (Group Method of Data Handling), created by a Ukrainian scientist Aleksey Ivakhnenko in 1968, when the back-propagation technique was not known yet [5]. GMDH constructs in successive steps a polynomial neural network (PNN), adding one layer a time and calculating parameters. General connection between input and output variables are possible to express by the Volterra functional series, a discrete analogue of which is Kolmogorov-Gabor polynomial (1). This polynomial can approximate any stationary random sequence of observations and can be computed by either adaptive methods or system of Gaussian normal equations. GMDH decomposes the complexity of a process into many simpler relationships each described by low order polynomials (2) for every pair of the input values. Typical GMDH network maps a vector input $\boldsymbol{x}$ to a scalar output y, which is an estimate of the true function $f(\boldsymbol{x}) = y^t$. Each neuron of the PNN should fit its output to a desired value y^t for each input vector $\boldsymbol{x}$ from the training set. It defines an optimal structure of a complex system model with identifying non-linear relations between input and output variables [6].

$$y = a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} a_{ijk} x_i x_j x_k + \ldots \quad (1)$$

m – number of variables $X(x_1, x_2, \ldots, x_m)$
$A(a_1, a_2, \ldots, a_m), \ldots$ *- vectors of parameters*

D-PNN combines the PNN functionality with some math techniques of differential equation (DE) solutions. Its models are a boundary of neural network and exact computational techniques. D-PNN forms and resolves an unknown general DE description of a searched function approximation. A DE is substituted producing sum of fractional polynomial derivative terms, forming a system model of dependent variables. In contrast with the ANN functionality, each neuron can direct be a part in the total network output calculation, which is generated by the sum of active neuron output values. A real complex function is decomposed into many partial data relation specifications, defining a composite structural model, which exact solution is vague or impossible to get using a predetermined DE [7].

$$y = a_0 + a_1x_i + a_2x_j + a_3x_ix_j + a_4x_i^2 + a_5x_j^2 \quad (2)$$

II. FAILURE AND POWER DEMAND SYSTEM MODELING

A. Failure modeling of electronic systems in power plants

The stressor-susceptibility interaction is a technique of failure predictions, which can be extended to simple electronic components or more complicated electronic circuits. Stressor is a physical entity influencing the lifetime of a component or circuit. A stressor, indicating a physical entity x will be denoted as ψ_x. Susceptibility of a component to a certain failure mechanism is defined as the probability function indicating the probability that a component will not remain operational for a certain time under a given combination of stressors. The susceptibility related to the failure mechanism *"y"* is usually defined as $S_y(t, \psi_p, \psi_q, \psi_r)$. Most components tend to have more than one failure mechanism, resulting in more than one "failure probability". It can be shown that there is a strong correlation between the various failure mechanisms existing within a component. Figure 1 illustrates the stressor - susceptibility interaction for a single failure mechanism. It is clear that the main source of problem is the overlap between stressor and susceptibility density.

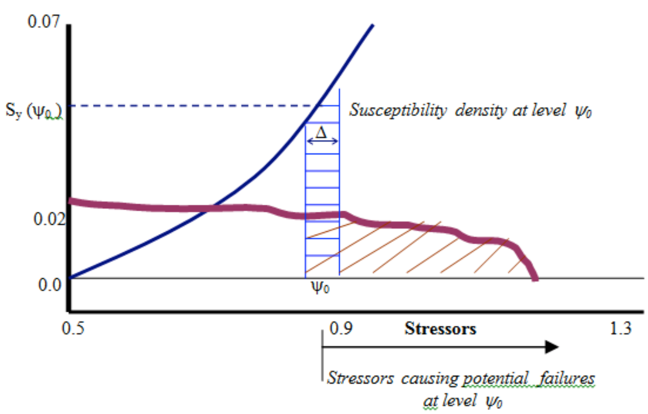

Figure 1. Stressor- Susceptibility interaction for single failure mechanism.

The failure probability can be calculated for this stressor distribution on a failure mechanism with a single time independent variable (3). To calculate the failure probability as a function of more complex susceptibility model, it will be necessary to calculate the failure probability of a part of the susceptibility model, for a certain stressor interval Δ, characterized by its mean value ψ_o and the corresponding susceptibility density function at that point $S_y(\psi_o)$.

$$f_{fail,y,\Psi}(\Psi_0) = \int_{\Psi_0}^{\infty} f_y(\Psi)\,d\Psi \quad (3)$$

As for large series of components, the physical structures of the individual components will be different for everyone, the survival probability of such a series of components will also show individual differences. The stress on a component may vary with time due to circuit behavior and circuit use. The circuit behavior will differ amongst a series of circuits due to physical differences in the individual circuit components, the physical structure of a circuit, the use of a circuit and the environment (electrical, thermal, etc.) of the circuit. To summarize the variety of effects it is useful to describe stressors as stochastic signals with properties depending on the influencing factors mentioned above.

There are two different categories of failure mechanisms applicable to electronic components. First are the failure mechanisms that are related to the electrical stress in a circuit. Second, failure mechanisms related to the intrinsic aspects of a component. There are two possible ways to obtain stressor sets for practical circuits. The First possibility involves usage of computer simulation models to derive all circuit signals using one single simulation. Second possibility is to derive stressor sets from practical measurements. In those cases where sufficient systems are available it is possible to do a statistical evaluation of the individual stressor functions existing in individual systems. As the stressor sets are dependent on the conditions of use and the operation modes of a system it is important that the measured stressor is based on all the possible operation modes of a circuit and all the possible transitions between the various operation modes. Accurate description of a stressor set will require a number of samples sufficient to cover all the different states of the system. As a signal has often more than one quasi-stationary states, each characterized by their stressor set, it is possible to derive the overall stressor set function from the individual state stressor sets using [8].

$$f_{str,y(x)} = \sum_{i=1}^{n} \frac{T_i}{T_{total}} f_{str,y,i(x)} \quad (4)$$

$f_{str,y\,(x)}$ is the stressor probability density function of quasi-stationary state i. T_i / T_{total} is the fraction of time that the stressor is in quasi-stationary state *i* (4). The derivation of stressor sets use Monte Carlo method, a logical model of the system being analyzed is repeatedly evaluated, each run using different values of the distributed parameters [9].

B. Power demand modeling in power plants

Modern complex manufacturing systems rely heavily on computer numerical controlled machines, variable speed drives and robotic devices which often require a high level of reliability from the incoming electricity supply. Due to the widespread usage of non-linear loads there has been a significant increase in the harmonic content of the 3-phase supply, raising serious power quality issues [10]. Predicting the power factor and active power demand is possible to automate the control of reactive power load and to better utilize the volt-amperes (VA) inflow. Efficient usage of the VA loading will not only improve the overall grid condition but also reduce the consumer's industrial tariffs. Depending on the estimated power factor, power factor corrective measures could be turned on or off to control the VA inflow into the plant. The ratio of active power P [W] to the apparent power S [VA] is termed the power factor (5).

$$\text{Power factor} = \cos(\varphi) = \frac{\mathrm{P}}{\mathrm{S}} = \frac{\text{resistance (R)}}{\text{impedance (Z)}} \qquad (5)$$

The power factor is lagging when the current lags the supply voltage and leading when the current leads the supply voltage. This means that the supply voltage is regarded as the reference quantity, a majority of loads served by a power utility draw current at a lagging power factor. When the power factor of the load is unity, active power equals apparent power (P = S). But, when the power factor of the load is less than unity, say 0.6, the power utilized is only 60%. This means that 40% of the apparent power is being utilized to supply the reactive power demand of the system. It is therefore clear that the higher the power factor of the load, the greater the utilization of the apparent power [11]. For the generating and transmission stations, lower the power factor the larger must be the size of the source to generate that power, and greater must be the cross-sectional area of the conductor to transmit it, i.e. the greater is the cost of generation and transmission of the power. Moreover, lower power factor will also increase the I^2R (I denotes current) losses in lines/equipment as well as result in poor voltage regulation.

III. General Differential Equation Composition

The basic idea of the D-PNN is to compose and substitute a general sum partial differential equation (6), which is not known in advance and can describe a system of dependent variables, with a sum of fractional relative multi-parametric polynomial derivative terms (7).

$$a + \sum_{i=1}^{n} b_i \frac{\partial u}{\partial x_i} + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \ldots = 0 \qquad u = \sum_{k=1}^{\infty} u_k \qquad (6)$$

$u = f(x_1, x_2, \ldots, x_n)$ – searched function of all input variables
$a, B(b_1, b_2, \ldots, b_n), C(c_{11}, c_{12}, \ldots)$ – polynomial parameters

Partial DE terms are formed according to the adapted integral analogues method, which is a part of similarity model analysis. It replaces mathematical operators and symbols of a DE by ratio of corresponding values. Derivatives are replaced by their integral analogues, i.e. derivative operators are removed and simultaneously with all operators are replaced by similarly or proportion signs in equations to form dimensionless groups of variables [12].

$$u_i = \frac{\left(a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2 + a_4x_1^2 + a_5x_2^2 + \ldots\right)^{m/n}}{b_0 + b_1x_1 + \ldots} = \frac{\partial^m f(x_1, \ldots, x_n)}{\partial x_1 \partial x_2 \ldots \partial x_m} \qquad (7)$$

n – combination degree of a complete polynomial of n-variables
m – combination degree of denominator variables

The fractional polynomials (7) define partial relations of *n*-input variables. The numerator of a DE term (7) is a polynomial of all *n*-input variables and partly defines an unknown function *u* of eq. (6). The denominator is a derivative part, which includes an incomplete polynomial of the competent combination variable(s). The root function of numerator takes the polynomial into competent combination degree to get the dimensionless values. In a case of time-series data application an ordinary differential equation is formed with only time derivatives, the partial DE (6) might become form of (8).

$$a + bf + \sum_{i=1}^{m} c_i \frac{df(t, x_i)}{dt} + \sum_{i=1}^{m} \sum_{j=1}^{m} d_{ij} \frac{d^2 f(t, x_i, x_j)}{dt^2} + \ldots = 0 \qquad (8)$$

$f(t,\mathbf{x})$ – function of time t and independent variables $\mathbf{x}(x_1, x_2, \ldots, x_m)$

Blocks of the D-PNN (Figure 2.) consist of derivative neurons with the same inputs, one for each fractional polynomial derivative combination, so each neuron is considered a summation DE term (7). Each block contains a single output polynomial (2), without derivative part. Neurons do not affect the block output but can participate direct in the total network output sum calculation of a DE composition. Each block has *1* and neuron *2* vectors of adjustable parameters ***a***, resp. ***a, b***.

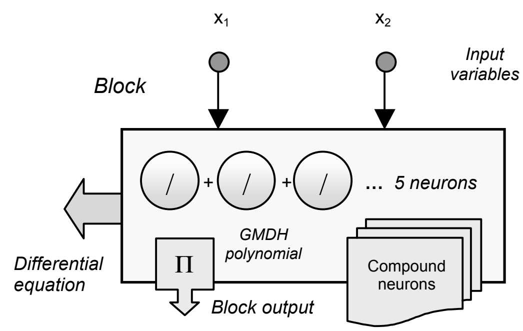

Figure 2. D-PNN block of basic and compound neurons (DE terms).

In the case of 2 input variables the 2nd odder partial DE can be expressed in the form (9), which involves all derivative terms with respect to variables applied by the GMDH polynomial (2). D-PNN's blocks and neurons process these 2-combination square polynomials, which form competent DE terms of a DE definition (6). Each block so include *5* simple neurons of derivatives $x_1, x_2, x_1x_2, x_1^2, x_2^2$ of the 2nd order partial DE (9), which is most often used to specify models of physical or natural systems.

$$F\left(x_1, x_2, u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \frac{\partial^2 u}{\partial x_1^2}, \frac{\partial^2 u}{\partial x_1 \partial x_2}, \frac{\partial^2 u}{\partial x_2^2}\right) = 0 \qquad (9)$$

where $F(x_1, x_2, u, p, q, r, s, t)$ is a function of 8 variables

D-PNN's basic form, using only 2-input variables comprises 1 block of at most 5 simple derivative neurons of a DE solution (9) (Figure 2). Denominator coefficients balance a length variety of the derivative polynomials (10)(11)(12).

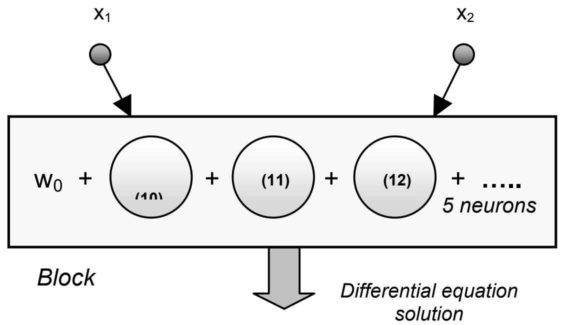

Figure 3. 1-block D-PNN with 2-inputs and 5 basic neurons (DE terms).

$$y_1 = \frac{\partial f(x_1, x_2)}{\partial x_1} = w_1 \frac{\left(a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2 + a_4x_1^2 + a_5x_2^2\right)^{1/2}}{1.5 \cdot (b_0 + b_1x_1)} \qquad (10)$$

$$y_4 = \frac{\partial^2 f(x_1, x_2)}{\partial x_2^2} = w_4 \frac{a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2 + a_4x_1^2 + a_5x_2^2}{2.7 \cdot (b_0 + b_1x_2 + b_2x_2^2)} \qquad (11)$$

$$y_5 = \frac{\partial^2 f(x_1, x_2)}{\partial x_1 \partial x_2} = w_5 \frac{a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2 + a_4x_1^2 + a_5x_2^2}{2.3 \cdot (b_0 + b_1x_{11} + b_2x_{12} + b_3x_{11}x_{12})} \qquad (12)$$

IV. DIFFERENTIAL POLYNOMIAL NEURAL NETWORK

Multi-layered networks forms composite polynomial functions (Figure 4.). Compound terms (CT), i.e. derivatives in respect to variables of previous layers, are calculated according to the composite function partial derivation rules (13)(14). They are formed by products of partial derivatives of external and internal functions [7].

$$F(x_1, x_2, \dots, x_n) = f(y_1, y_2, \dots, y_m) = f(\phi_1(X), \phi_2(X), \dots, \phi_m(X)) \qquad (13)$$

$$\frac{\partial F}{\partial x_k} = \sum_{i=1}^{m} \frac{\partial f(y_1, y_2, \dots, y_m)}{\partial y_i} \cdot \frac{\partial \phi_i(X)}{\partial x_k} \qquad k=1, \dots, n \qquad (14)$$

2nd and following hidden layers blocks are additionally extended with compound terms (neurons), which form composite derivatives with respect to outputs and inputs of back connected previous layer blocks. The 1st block of the last (3rd) hidden layer (Figure 4.) forms CT e.g. (15)(16). The square and combination derivative terms are calculated analogously, according to the partial derivation rules (14).

$$y_2 = \frac{\partial f(x_{21}, x_{22})}{\partial x_{11}} = w_2 \frac{(a_0 + a_1x_{21} + a_2x_{22} + a_3x_{21}x_{22} + a_4x_{21}^2 + a_5x_{22}^2)^{1/2}}{1.6 \cdot x_{22}} \cdot \frac{(x_{21})^{1/2}}{1.5 \cdot \left(b_0 + b_1x_{11}\right)} \qquad (15)$$

$$y_3 = \frac{\partial f(x_{21}, x_{22})}{\partial x_1} = w_3 \frac{(a_0 + a_1x_{21} + a_2x_{22} + a_3x_{21}x_{22} + a_4x_{21}^2 + a_5x_{22}^2)^{1/2}}{1.6 \cdot x_{22}} \cdot \frac{(x_{21})^{1/2}}{1.6 \cdot x_{12}} \cdot \frac{(x_{11})^{1/2}}{1.5 \cdot \left(b_0 + b_1x_1\right)} \qquad (16)$$

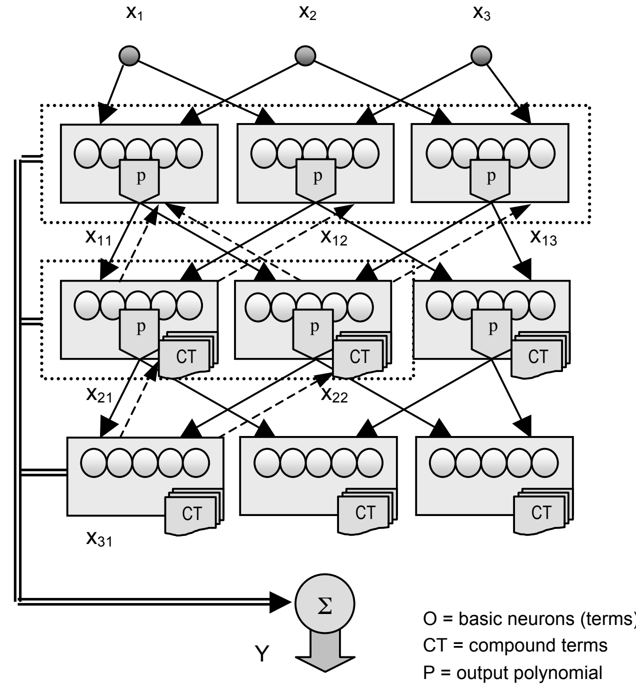

Figure 4. 3-variable multi-layered backward D-PNN

The best-fit neuron selection is the initial phase of the DE composition, which may apply a proper genetic algorithm (GA). Parameters of polynomials might be adjusted by means of difference evolution algorithm (EA), supplied with sufficient random mutations [13]. The parameter optimization is performed simultaneously with the GA neuron combination search, which may result in a quantity of local or global error solutions. There would be welcome to apply an adequate gradient descent method too, which parameter updates result from partial derivatives of polynomial DE terms in respect with the single parameters [14]. The number of network hidden layers should coincide with a total amount of input variables.

$$Y = \frac{\sum_{i=1}^{k} y_i}{k} \qquad k = \text{amount of active neurons} \qquad (17)$$

Only some of all potential combination DE terms (neurons) may participate in the DE composition, in despite of they have an adjustable term weight (w_i). D-PNN's total output Y is the sum of all active neuron outputs, divided by their amount k (17).

$$MSE = \frac{\sum_{i=1}^{M} \left(y_i^d - y_i\right)^2}{M} \rightarrow \min \qquad (18)$$

M = number of data samples, y^d = desired output

The mean square error (MSE) method (18) was applied for the polynomial parameter optimization and neuron combination selection. D-PNN is trained only with a small set of input-output data samples likewise the GMDH algorithm does [6].

V. Experimental Results

Reliability models of electronic circuits apply 2 input variables of current and voltage, normalized into *<0, 1>*. The D-PNN model is very simple, consisting only 1 block of at most 5 neurons (Figure 3), i.e. derivative terms of the 2^{nd} order partial DE solution (9), forming the total sum output. The failure probability [%] model applied all 5 neurons (Figure 5), leakage current [Amps] only 2 neurons (Figure 6) and junction temperature [^{o}C] consists of 2 the same blocks of 4 neurons totally (Figure 7). All models, entered by 2 input variables, estimate a next time output value of a 1-shifted time step onward. The failure and leakage testing errors might probably get any lower.

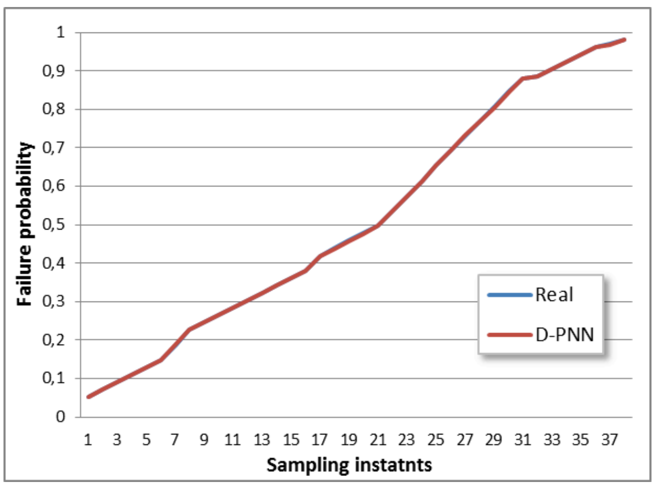

Figure 5. Failure probability MSE_{Test} = 0.000000838, MSE_{Train} = 0.00001

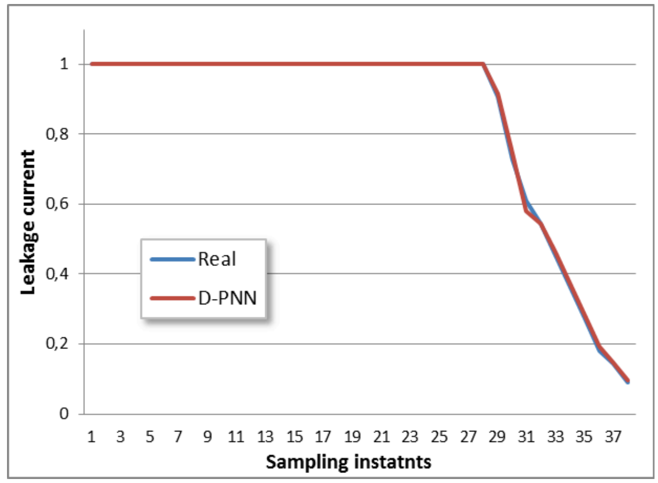

Figure 6. Leakage current MSE_{Test} = 0.0000439, MSE_{Train} = 0.00001

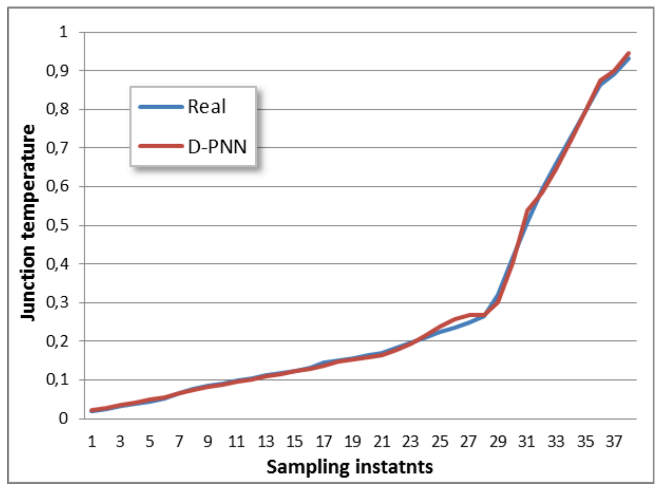

Figure 7. Junction temperature MSE_{Test} = 0.0000926, MSE_{Train} = 0.000036

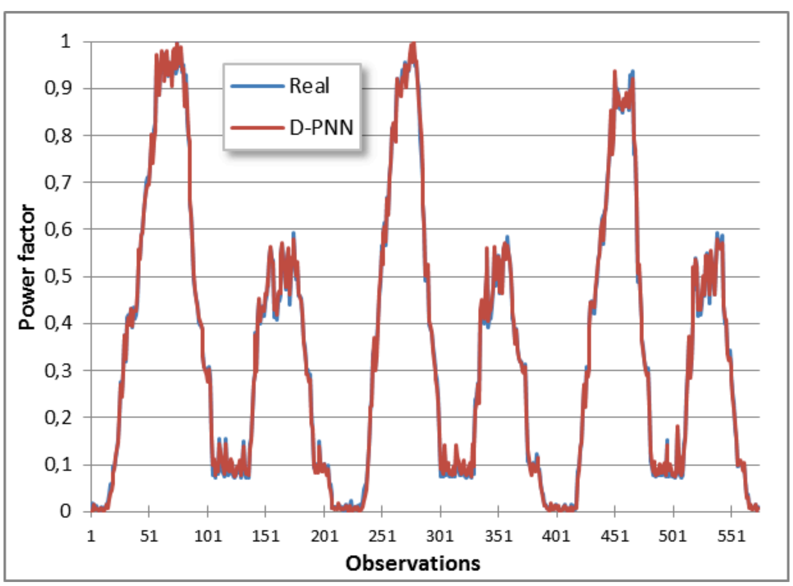

Figure 8. Power factor MSE_{Test} = 0.000124, MSE_{Train} = 0.000158

A heavy engineering manufacturing plant was considered for the prediction of power factor. Power demand (utilization) model applied 3-time series of 2 input parameters - the voltage and current again, i.e. 6 input vector variables totally. It used a 3-layered D-PNN structure, consisting of 2 interconnected networks of Figure 3., which disables it to define all possible data relations. The network was trained using 70% of the simulated data and the remaining 30% data was used for testing and validation (Figure 8.).

An increasing number of variables using PNN cause a combinatorial explosion of higher layers, i.e. an impossibility of a full faultless D-PNN data relation description (formation a complete DE model). Some few block outputs might merge (sum) to form 1 block of an auxiliary layer, to keep the previous hidden layer total amount of blocks and thus prevent from an undesirable outsize exponential combination growth. This complex merge network structure is still not resolved in a satisfactory manner [15].

VI. CONCLUSIONS

The reliability estimations of electronic circuits (i.e. failure probability, temperature and leakage current) using the new type of PNN called differential polynomial neural network obtained very good results, despite of the applied 1-block models of DE solutions are too simple. The problem modeling using stressor – susceptibility interaction method can be widely applied to a wide range of electronic circuits or systems. The power factor model obtained satisfactory results, however an impossibility of using more input vector variables (more than 3), with a corresponding number of network hidden layers, disallows it to get better results. 6 input variables would require 6 hidden layers of blocks, to enable the D-PNN to form all eventual combination derivative terms. Random values of input parameter voltage (i.e. +/- 2.5% of the normal value) were considered to test the learning ability of the neural network during worst conditions in the grid voltage regardless of the plant load. The performance could have been even better if the observed rather than simulated values of voltage were used. The D-PNN's adjustment was highly time-consuming, as the applied methods are still not finished.

D-PNN forms and resolves an unknown general DE with a composition and substitution of sum fractional derivative terms, describing a system model of dependent variables. Its non-linear regression is based on a generalization of data relations. The characteristics of differential equation solutions can facilitate a greater variety of model forms than allowed by standard soft computing methods. Its models can combine the PNN functionality with exact computational techniques of differential equation solutions, so there would be possible to define and form derivative terms in different ways, using some alternative methods (e.g. Fourier's series). D-PNN's operating and constructing principles differ by far from other common neural network techniques.

ACKNOWLEDGMENT

The article has been elaborated in the framework of the IT4Innovations Centre of Excellence project, reg. no. CZ.1.05/1.1.00/02.0070 funded by Structural Funds of the European Union and state budget of the Czech Republic and in the framework of the project Opportunity for young researchers, reg. no. CZ.1.07/2.3.00/30.0016, supported by Operational Programme Education for Competitiveness and co-financed by the European Social Fund and the state budget of the Czech Republic.

REFERENCES

[1] J. Chaquet and E. Carmona, "Solving differential equations with fourier series and evolution strategies", Applied Soft Computing 12 (2012) 3051–3062.

[2] H. Iba, "Inference of differential equation models by genetic programming", Information Sciences 178 (4) (2008) 4453–4468.

[3] I. Tsoulos, D. Gavrilis and E. Glavas, "Solving differential equations with constructed neural networks", Neurocomputing 72 (2009) 2385–2391.

[4] L.Zjavka, "Generalization of patterns by identification with polynomial neural network", Journal of Electrical Engineering No. 2/2010, Vol.61, p.120-124.

[5] A. G. Ivakhnenko, „Polynomial theory of complex systems", *IEEE Transactions on systems*, Vol. SMC-1, No.4.

[6] N.Y. Nikolaev and H. Iba, *Adaptive Learning of Polynomial Networks*. *Springer*, New York 2006.

[7] L. Zjavka, "Recognition of Generalized Patterns by a Differential Polynomial Neural Network", *Engineering, Technology & Applied Science Research* Vol. 2, No 1 (2012).

[8] A.C. Brombacher, *Reliability by Design*, Wiley, 1995.

[9] P.D. O'Connor, *Practical Reliability Engineering*, 3rd edn., Wiley 1988.

[10] P. Lynch, "An active approach to harmonic filtering", IEE Review, Volume 45(3), May 1999

[11] T.J. Miller, "Reactive Power Control in Electric Systems", Wiley – Interscience, 1982.

[12] K. Chan and W. Y. Chau, "Mathematical theory of reduction of physical parameters and similarity analysis", International Journal of Theoretical Physics 18 (1979) 835–844.

[13] S. Das, A. Abraham, A. Konar, *Particle swarm optimization and Differential evolution algorithms*, Studies in Computational Intelligence 116, 1-38, Springer-Verlag Berlin 2008.

[14] N.Y. Nikolaev, H. Iba, "Polynomial harmonic GMDH learning networks for time series modelling", Neural Networks 16, 2003, 1527–1540. Science Direct.

[15] L. Zjavka, "Combined differential polynomial neural network", Journal of Electrical and Control Engineering Vol.2. No.3. (2012) 15–19.