"ALADIN" WEATHER MODEL LOCAL REVISIONS USING THE DIFFERENTIAL POLYNOMIAL NEURAL NETWORK

Ladislav Zjavka[*]

Abstract: The 48-hour "Aladin" forecast model can predict significant meteorological quantities in a middle scale area. Neural networks could try to replace some statistical techniques designed to adapt a global meteorological numerical forecast model for local conditions, described with real data surface observations. They succeed commonly a cut above problem solutions with a predefined testing data set, which provides bearing inputs for a trained model. Time-series predictions of the very complex and dynamic weather system are sophisticated and not any time faithful using simple neural network models entered only some few variables of their own next-time step estimations. Predicted values of a global meteorological forecast might instead enter a neural network locally trained model, for refine it. Differential polynomial neural network is a new neural network type developed by the author; it constructs and substitutes for an unknown general sum partial differential equation of a system description, with a total sum of fractional polynomial derivative terms. This type of non-linear regression is based on trained generalized data relations, decomposed into many partial derivative specifications. The characteristics of composite differential equation solutions of this indirect type of a function description can facilitate a much greater variety of model forms than is allowed using standard soft-computing methods. This adjective derivative model type is supposed to be able to solve much more complex problems than is usual using standard neural network techniques.

Key words: *Polynomial neural network, differential equation composition, sum relative derivative term, multi-parametric function approximation*

Received: September 13, 2013 **DOI:** 10.14311/NNW.2014.24.009
Revised and accepted: March 21, 2014

1. Introduction

The short-term numerical "Aladin" forecast model is a limited area version of a global French model ARPEGE and it needs to be forced by a global model which has to provide lateral boundary conditions. It refines the ARPEGE model on a

[*]Ladislav Zjavka, VŠB-Technical University of Ostrava, IT4innovations Ostrava, Czech Republic, E-mail: `lzjavka@gmail.com`

middle-scale target area using a more detailed time and spatial resolution of the interpolation and numerical integration of the system of differential equations, which describes atmospheric processes on account of global meteorological observations. Differential equations are generally able to describe most of physical or natural complex systems, which it is difficult to model by unique explicit functions. The solutions can apply power [2] or wave series [6], fractional calculus [11] [9] or standard soft-computing techniques [4] [12]. These make use in common of straight computational composing methods, which operate only within absolute interval values of input variables, e.g. genetic programming (GP) or fuzzy models can compose a searched function using collections of operators and terminals of a defined set to form symbolic tree expressions [7]. A common artificial neural network (ANN) pattern identification or function approximation is based on learned entire similarity relationships between new presented input patterns and the trained ones, regarding no straight elementary data relations, which multi-parametric polynomial functions can describe [10]. It is possible to express a general connection between input and output variables by means of the Volterra functional series, a discrete analogue of which is the Kolmogorov-Gabor polynomial (1).

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

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

The Group Method of Data Handling (GMDH) was created by a Ukrainian scientist Aleksey Ivakhnenko in 1968, when the back-propagation technique was not yet known. It forms a multi-layer polynomial neural network (PNN) in successive steps, adding one layer a time, which decomposes the complexity of a process into many simpler relationships, each described by low order polynomials (2) for every pair of the input values x_i, x_j [8].

$$y = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2 \tag{2}$$

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 defines the optimal structure of a complex system model with identifying non-linear relations between input and output variables. Typical GMDH network maps an input vector $\boldsymbol{x}$ to a scalar output y, which is an estimate of the true function $f(\boldsymbol{x})= y^t$ [10]. PNN can compose sum partial differential equation models describing discrete data observations. Differential polynomial neural network (D-PNN) is a new neural network type, designed by the author. The skeleton is formed by the GMDH-PNN; however its operating and constructing principles differ from those of GMDH, being based on Taylor-series expansions. This new neural network technique extends the PNN structure to generate sum series of polynomial relative derivative terms, which together define and substitute for (solve) a general partial differential equation of a complex function description. The adjective derivative type of real complex function models seem to be a preferable qualities to simple upright compositing computational techniques, which use a set of terminals and operators to form a tree-like structural solution. D-PNN can combine the PNN functionality with some math techniques of differential equation (DE) solutions. A complex function, which exact solution is problematic or impossible to get using a predetermined DE or direct composing techniques, is decomposed into many partial data relation specifications, defining a sum composed derivative model [14].

2. General sum differential equation composition

D-PNN forms and resolves a general partial DE (3), which can generally describe a system model of dependent variables, using summation derivative terms. The searched function $u$may be expressed in the form of sum series (4), consisting of series arising from derivative sum convergent term series (5) in the case of 2 input variables. The study tries to solve the sum partial DE (3) with substitutions of multi-parametric polynomial fraction terms (6), defining relative dependent derivative changes over some input variables combinations.

$$a + bu + \sum_{i=1}^{n} c_i \frac{\partial u}{\partial x_i} + \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \cdots = 0 \tag{3}$$

$$a, b, \boldsymbol{c}(c_1, c_2, \ldots, c_n), \boldsymbol{d}(d_{11}, dc_{12}, \ldots), \cdots - \textit{parameters}$$

$$u = \sum_{k=1}^{\infty} u_k \tag{4}$$

$$n - \textit{number of variables}$$

$\boldsymbol{x}(x_1, x_2, \ldots, x_n)$ – vector of input variables $\quad$ $u(\boldsymbol{x})$ – unknown function of n-variables

$$\left(\sum \frac{\partial u_k}{\partial x_1}, \sum \frac{\partial u_k}{\partial x_2}, \sum \frac{\partial^2 u_k}{\partial x_1^2}, \sum \frac{\partial^2 u_k}{\partial x_1 \partial x_2}, \sum \frac{\partial^2 u_k}{\partial x_2^2}\right) \tag{5}$$

The method of integral analogues, which is a part of the similarity model analysis applying various formal adaptations of a DE [5], provides an essential principle for the substitution of the partial polynomial DE (3) terms. It replaces mathematical operators and symbols in a DE by the ratio of corresponding variables. Derivatives are replaced by their integral analogues, i.e. derivative operators are removed and replaced by analogue or proportional signs in equations. Complete input polynomials of a specific order can replace the partial functions u_k of derivative terms (5), while the combination polynomials of denominators represent the alterative derivative parts (6).

$$u_i = \frac{\left(a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2 + \ldots\right)^{\frac{m}{n}}}{b_0 + b_1 x_1 + \ldots} = \frac{\partial^m f(x_1, \ldots, x_n)}{\partial x_1 \partial x_2 \ldots \partial x_m}$$

$$n - \textit{combination degree of a complete polynomial of n-variables}$$

$$m - \textit{combination degree of a denominator polynomial} \tag{6}$$

The partial DE (3), which describes time-series data observations, might be converted into an ordinary DE with respect to only time-dependent term derivatives

(7). The input variables $x_{t1}, x_{t2}, \ldots, x_{tn}$ are observations of the same variable x in specific time t interval sequences. A DE composition may as well take a form of the combined DE, which involves partial and ordinary-time derivatives together.

$$a + bf + \sum_{i=1}^{m} c_i \frac{df(t,x_i)}{dt} + \sum_{i=1}^{m}\sum_{j=1}^{m} d_{ij} \frac{d^2 f(t,x_i,x_j)}{dt^2} + \cdots = 0 \tag{7}$$

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

Blocks of the D-PNN (Fig. 1) consist of neurons, one for each fractional polynomial derivative combination (8), so each neuron is considered a summation DE term. Each block contains a single output polynomial (2) without derivative part. Neurons do not affect the block output but can participate directly in the total network output sum calculation of a DE composition. Each block has 1 and neuron 2 vectors of adjustable parameters $\boldsymbol{a}$,then $\boldsymbol{a}$, $\boldsymbol{b}$. The root function of a neuron's numerator decreases a combination degree of the input polynomial of a term (6) with respect to the denominator, in order to get the dimensionless values [5].

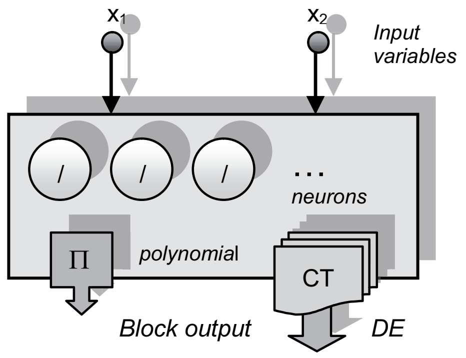

Fig. 1 *2-variable combination block of basic (/) and compound terms (CT) – neurons.*

D-PNN processes 2-combination square polynomials of blocks and neurons (DE terms), the same as those applied by the GMDH algorithm (2). This simple polynomial type has generally proved to yield the best results besides being easy to use. Each block so include 5 basic neurons with respect to derivatives x_1, x_2, x_1x_2, x_1^2 and x_2^2 of the 2nd order partial DE (3), of a searched 2-variable u partial function description, which might be expressed in the form of (8). This type of DE is preferably used to describe natural and physical non-linearities of natural and physical processes; it applies 2 linear (9), 2 square (10) and 1 combination (11) derivative term.

$$F\left(x_1, x_2, u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \frac{\partial^2 u}{\partial x_1^2}, \frac{\partial^2 u}{\partial x_1 \partial x_2}, \frac{\partial^2 u}{\partial x_2^2}\right) = 0 \tag{8}$$

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

$$y_1 = \frac{\partial f(x_1,x_2)}{\partial x_1} = w_1 \frac{\left(a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2 + a_4x_1^2 + a_5x_2^2\right)^{\frac{1}{2}}}{1.5 \cdot (b_0 + b_1x_1)} \tag{9}$$

$$y_3 = \frac{\partial^2 f(x_1,x_2)}{\partial x_2^2} = w_3 \frac{a_0 + a_1x_1 + a_2x_2 + a_3x_1x_2 + a_4x_1^2 + a_5x_2^2}{2.7 \cdot (b_0 + b_1x_2 + b_2x_2^2)} \tag{10}$$

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

3. Multi-layer backward differential polynomial neural network

Multi-layer PNN forms composite polynomial functions (Fig. 2); the previous layers form internal functions (12), which substitute for the next hidden layer input variables of neuron and block polynomials to define external functions (13). Compound DE terms, i.e. composite function derivatives with respect to the variables of previous layers, are calculated according to the partial function derivation rules (14)(15).

$$y_i = \varphi_i(X) = \varphi_i(x_1, x_2, \ldots, x_n) \qquad i = 1, \ldots, m \tag{12}$$

$$F(x_1, x_2, \ldots, x_n) = f(y_1, y_2, \ldots, y_m) = f(\phi_1(X), \mathrm{g}\phi_2(X), \ldots, \phi_m(X)) \tag{13}$$

$$\left.\begin{array}{l}
\frac{\partial F}{\partial x_1} = \frac{\partial f}{\partial y_1} \cdot \frac{\partial \varphi_1}{\partial x_1} + \frac{\partial f}{\partial y_2} \cdot \frac{\partial \varphi_2}{\partial x_1} + \ldots + \frac{\partial f}{\partial y_m} \cdot \frac{\partial \varphi_m}{\partial x_1} \\
\frac{\partial F}{\partial x_2} = \frac{\partial f}{\partial y_1} \cdot \frac{\partial \varphi_1}{\partial x_2} + \frac{\partial f}{\partial y_2} \cdot \frac{\partial \varphi_2}{\partial x_2} + \ldots + \frac{\partial f}{\partial y_m} \cdot \frac{\partial \varphi_m}{\partial x_2} \\
\ldots\ldots\ldots \\
\frac{\partial F}{\partial x_n} = \frac{\partial f}{\partial y_1} \cdot \frac{\partial \varphi_1}{\partial x_n} + \frac{\partial f}{\partial y_2} \cdot \frac{\partial \varphi_2}{\partial x_n} + \ldots + \frac{\partial f}{\partial y_m} \cdot \frac{\partial \varphi_m}{\partial x_n}
\end{array}\right\} \tag{14}$$

$$\frac{\partial F}{\partial x_k} = \sum_{i=1}^{m} \frac{\partial f(y_1, y_2, \ldots, y_m)}{\partial y_i} \cdot \frac{\partial \phi_i(X)}{\partial x_k} k = 1,\ \ldots,\ n \quad c \tag{15}$$

The blocks of the 2nd and following hidden layers (Fig. 2) are additionally extended with compound terms (neurons), which form composite function derivatives with respect to the output and input variables of the back connected previous layer blocks. For example the 1st block of the last (3rd) hidden layer forms linear compound neurons (16, 17). The amount of neurons of blocks, which involve composite functions, doubles in each previous back-connected layer. As the couples of input variables of the internal functions $\phi_1(x_1, x_2)$ and $\phi_2(x_3, x_4)$ (15) can differ from each other, the partial derivations are calculated separately with respect to each block variables. This way the sums (14, 15) consist of only 1 derivative term, which represents a single neuron [14].

$$\begin{aligned} y_2 &= \frac{\partial f(x_{21}, x_{22})}{\partial x_{11}} = w_2 \frac{(a_0 + a_1x_{21} + a_2x_{22} + a_3x_{21}x_{22} + a_4x_{21}^2 + a_5x_{22}^2)^{\frac{1}{2}}}{1.6 \cdot x_{22}} \cdot \\ &\cdot \frac{(x_{21})^{\frac{1}{2}}}{1.5 \cdot (b_0 + b_1x_{11})} \end{aligned} \tag{16}$$

$$\begin{aligned} y_3 &= \frac{\partial f(x_{21}, x_{22})}{\partial x_1} = w_3 \frac{(a_0 + a_1x_{21} + a_2x_{22} + a_3x_{21}x_{22} + a_4x_{21}^2 + a_5x_{22}^2)^{\frac{1}{2}}}{1.6 \cdot x_{22}} \cdot \\ &\cdot \frac{(x_{21})^{\frac{1}{2}}}{1.6 \cdot x_{12}} \cdot \frac{(x_{11})^{\frac{1}{2}}}{1.5 \cdot (b_0 + b_1x_1)} \end{aligned} \tag{17}$$

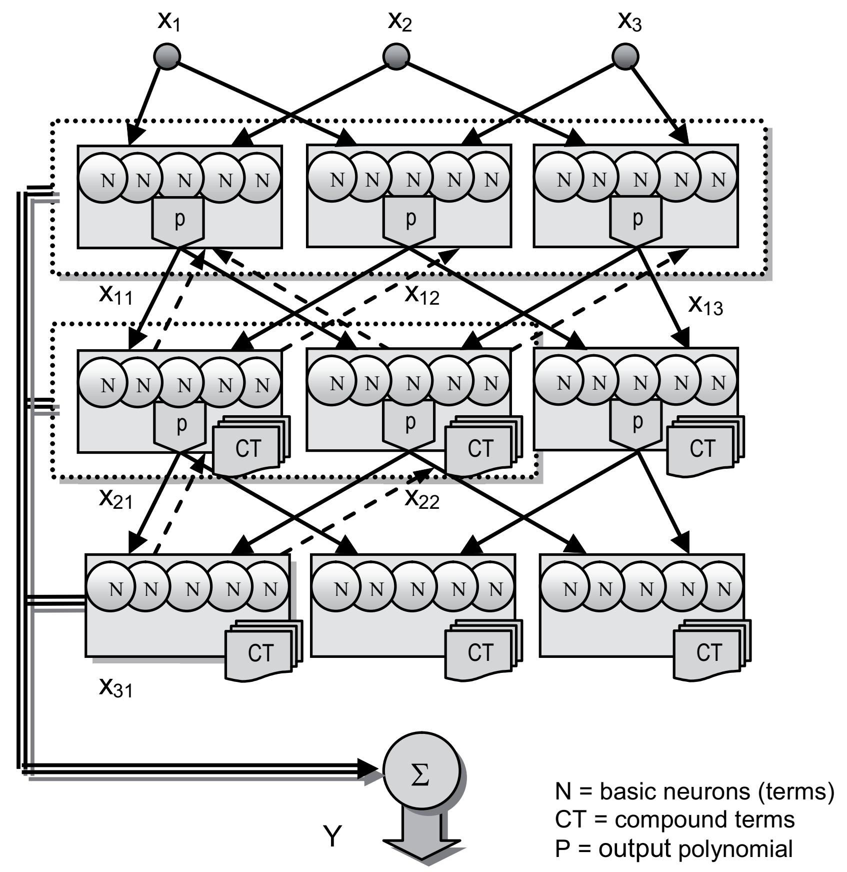

Fig. 2 *3-variable multi-layer D-PNN with 2-variable combination blocks. Backward connections of the 3rd layer 1st block (dashed lines).*

Square and combination compound derivative terms are also calculated according to the composite function derivation rules (18, 19, 20). The back-calculation of the composite function derivatives is well done by a recursive algorithm in the tree-like network structure.

$$F(x_1, x_2) = f(u, v) = f[\varphi(x_1, x_2), \psi(x_1, x_2)] \quad (18)$$

$$\frac{\partial^2 F}{\partial x_1^2} = \frac{\partial^2 f}{\partial u^2}\left(\frac{\partial \varphi}{\partial x_1}\right)^2 + 2\frac{\partial^2 f}{\partial u \partial v}\frac{\partial \varphi}{\partial x_1}\cdot\frac{\partial \psi}{\partial x_1} + \frac{\partial^2 f}{\partial v^2}\left(\frac{\partial \psi}{\partial x_1}\right)^2 + \frac{\partial f}{\partial u}\cdot\frac{\partial^2 \varphi}{\partial x_1^2} + \frac{\partial f}{\partial v}\cdot\frac{\partial^2 \psi}{\partial x_1^2} \quad (19)$$

$$\begin{aligned}\frac{\partial^2 F}{\partial x_1 \partial x_2} &= \frac{\partial^2 f}{\partial u^2}\cdot\frac{\partial \varphi}{\partial x_1}\cdot\frac{\partial \varphi}{\partial x_2} + \frac{\partial^2 f}{\partial u \partial v}\left(\frac{\partial \psi}{\partial x_1}\frac{\partial \varphi}{\partial x_2} + \frac{\partial \varphi}{\partial x_1}\frac{\partial \psi}{\partial x_2}\right) + \frac{\partial^2 f}{\partial v^2}\cdot\frac{\partial \psi}{\partial x_1}\cdot\frac{\partial \psi}{\partial x_2} + \\ &+ \frac{\partial f}{\partial u}\cdot\frac{\partial^2 \varphi}{\partial x_1 \partial x_2} + \frac{\partial f}{\partial v}\cdot\frac{\partial^2 \psi}{\partial x_1 \partial x_2} \quad (20)\end{aligned}$$

The number of hidden layers in the network should coincide with the total number of input variables, to enable the D-PNN to form all the possible combination derivative terms of a sum DE solution. Only some of all the potential derivative terms (neurons) may participate in the DE composition, even though they have an adjustable term weight w_i. A proper neuron combination, which can substitute for a DE solution, is not able to accept a disturbing effect of the rest of the neurons (which may compose other solutions) on the parameter optimization. The total output Y of the D-PNN is the arithmetical mean of all the active neuron output values (21) so as to prevent a changeable number of active neurons (of a combination) from influencing the total network output value.

$$Y = \frac{\sum_{i=1}^{k} y_i}{k} \qquad k = \textit{actual number of active neurons (DE terms)} \tag{21}$$

The selection of a fit neuron combination is the principal part of a DE composition and it may apply the simulated annealing (SA) method [3] combined with a standard genetic algorithm [1] in the initial composing phase. The root mean square error (RMSE) method (22) was applied for the polynomial parameter optimization and neuron combination selection process. The model results were evaluated using normalized $\mathrm{RMSE_R}$ to the range of observed data (23) and normalized $\mathrm{RMSE_M}$ to the mean of observed data (24). D-PNN is trained with only a small set of input-output data samples in a similar way to the GMDH algorithm [10].

$$RMSE = \sqrt{\frac{\sum_{i=1}^{M} \left(y_i^d - y_i\right)^2}{M}} \to \min \tag{22}$$

y_i^d = *desired output* $\qquad y_i$ = *estimated output for* i^{th} *training vector*

$$NRMSE_R = NR = \frac{RMSE}{y_{\max}^d - y_{\min}^d} \tag{23}$$

$$NRMSE_M = NM = \frac{RMSE}{y_{mean}^d} \tag{24}$$

4. "Aladin" forecast model revisions

The Czech hydro-meteorological institute (CHMI) provides the free 48-hour "Aladin" chart numerical forecast model, which involve temperature, wind speed/direction, relative humidity/precipitations, static pressure and 3-level cloudiness prognoses at a selected locality [17]. Global forecast models based upon the differential equations for atmospheric dynamics do not perfectly determine weather conditions near the ground. Statistical corrections were developed to attempt to resolve this problem, based upon the 3-dimensional fields produced by numerical weather models, surface observations, and the climatological conditions for specific locations. These statistical models are collectively referred to as model output statistics (MOS), which neural network models referred to other predicted variables could try to replace. The local model is trained with real data time-series observations in the past few days to define a true multi-parametric function relation, exactly actual for this time interval and in part also (more or less) valid for the same variables of the prediction model. After it can form new 24-hour corrections of some prognosis (e.g. relative humidity), applying the real model trained with several previous day time-series and input variables of the "Aladin" forecast. In the case of an overnight dramatic change in the weather from the training to the forecasting day(s), the short-term correction model is not true, as it holds for wrong past weather conditions, however this trend is not very frequent. The quality of new estimations also depends on prediction accuracies of other numerical model outcomes that are applied, which are not entirely valid and enter the input vector. These input variables, quite independent from the neural network locally trained model, are outputs of a different exact numerical model type, which benefits the new revised estimations.

The presented network (Fig. 2) relative humidity models were trained only with 3 current state vector variables: wind speed, temperature and sea level pressure, measured at one station locality (Ostrava-Mošnov) [18]. The models can roughly estimate the time and amount of precipitations and also indicate a cloudiness progress (Fig. 3–Fig. 6). The relative humidity values increase at night hours (along with temperature decrease), upswing or steep daily grows can indicate precipitation (Fig. 5); easy or slight slope curve courses can imply a changeable cloudiness (Fig. 4). The D-PNN models are quite sensitive to the rain-fall so a modeled precipitous quantity growth might spuriously intensify the testing errors in some cases. For example the revised model predicted exactly on time a big storm, which "Aladin" referred a few before, not giving it the real strong intensity (Fig. 5). The D-PNN local revisions of the relative humidity numerical forecast models may succeed significantly in some days (Fig. 4) of the settled weather however in the case of an overnight weather change they fail or are less successful (Fig. 6) as the trained model still holds for wrong past weather conditions.

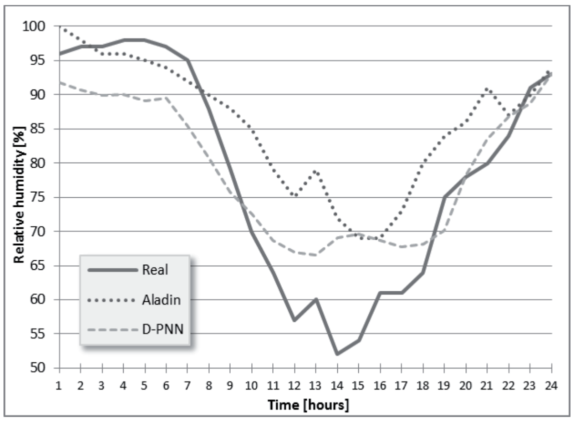

Fig. 3 *8. 6. 2013: Aladin (NR = 0.228, NM = 0.133), D-PNN (NR = 0.162, NM = 0.095).*

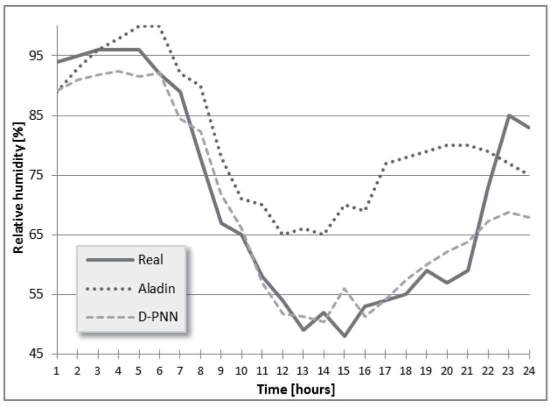

Fig. 4 *9. 6. 2013: Aladin (NR = 0.285, NM = 0.192), D-PNN (NR = 0.121, NM = 0.081).*

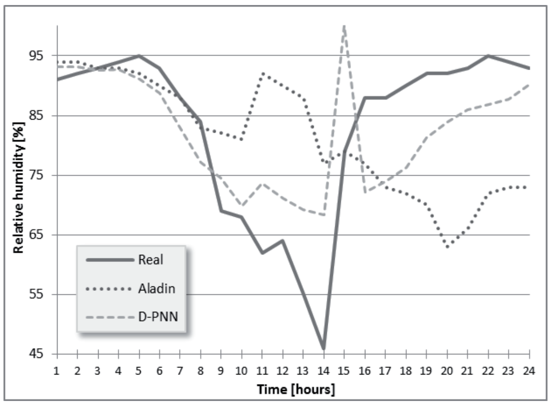

Fig. 5 *10. 6. 2013: Aladin (NR = 0.374, NM = 0.220), D-PNN (NR = 0.206, NM = 0.121).*

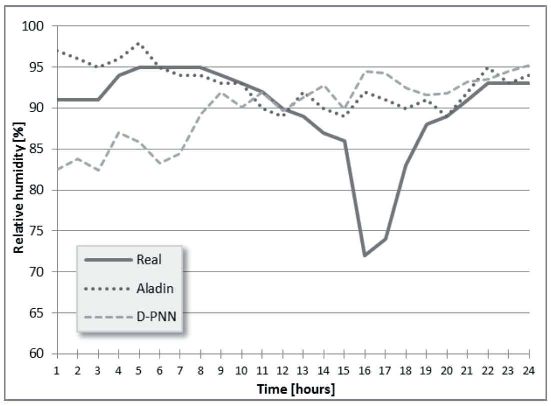

Fig. 6 *11. 6. 2013: Aladin (NR = 0.263, NM = 0.067), D-PNN (NR = 0.368, NM = 0.094).*

The neural network local revisions of the "Aladin" wind speed forecast models (Fig. 7–Fig. 10) apply 3 time-series of 3 input state parameters: temperature, relative humidity and sea level pressure, i.e. 9 input vector variables in total. Thus a combination of 1-parametric functions of time-series with further multi-parametric data relations is allowed, which seems to yield optimal solutions. D-PNN applies only to a network structure with 3 hidden layers of blocks, i.e. 3 inter-connected presented networks (Fig. 2) in principle, which does not allow all possible combination DE terms (a complete DE solution) to be formed [13]. An overnight weather change from 11. 6. 2013 (Fig. 10) corresponds to the failed revision of the relative humidity model (Fig. 6), however the "Aladin" wind speed prognosis was

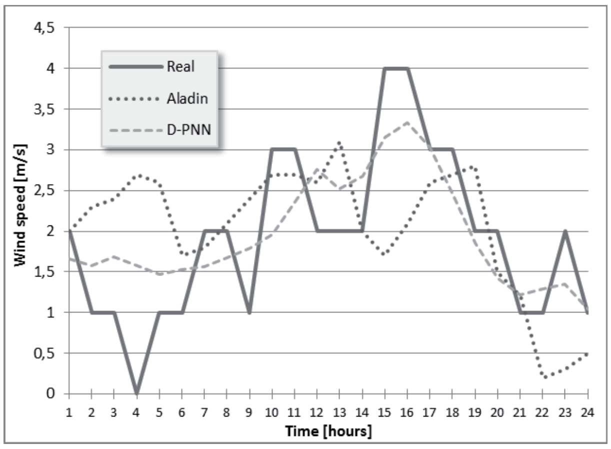

Fig. 7 *8. 6. 2013: Aladin (NR = 0.287, NM = 0.598), D-PNN (NR = 0.161, NM = 0.337).*

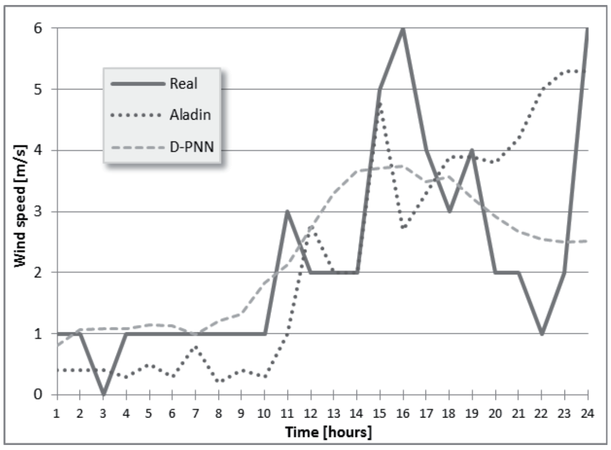

Fig. 8 *9. 6. 2013: Aladin (NR = 0.255, NM = 0.692), D-PNN (NR = 0.192, NM = 0.522).*

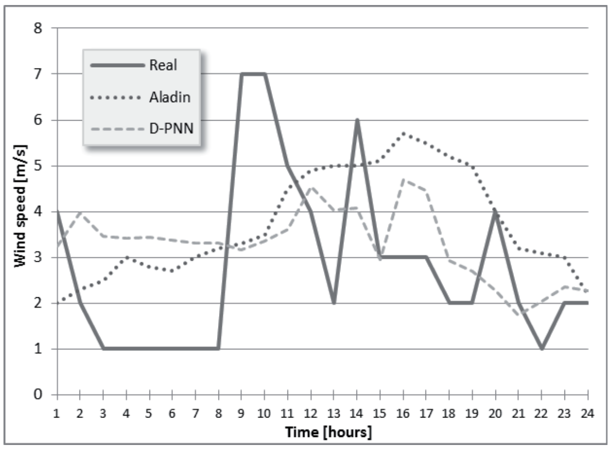

Fig. 9 *10. 6. 2013: Aladin (NR = 0.351, NM = 0.754), D-PNN (NR = 0.318, NM = 0.684).*

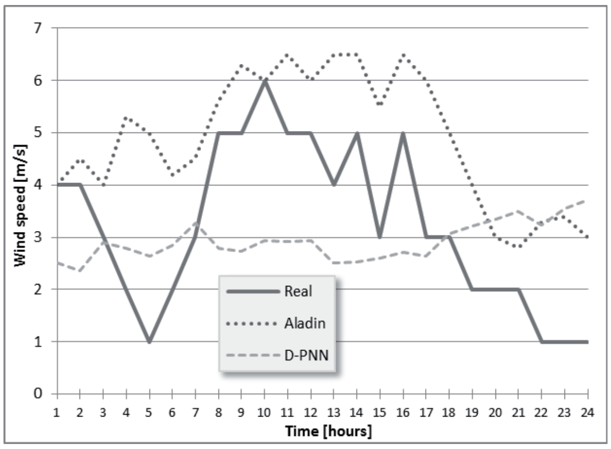

Fig. 10 *11. 6. 2013: Aladin (NR = 0.390, NM = 0.608), D-PNN (NR = 0.354, NM = 0.552).*

less successful in this day. The current wind speed quantity mainly induces a wind charger power output, much less affected by other weather conditions, e.g. unstable wind direction or temperature [16].

The cloudiness can comprise several layers: low, middle and high cloud amount levels and also their thickness; the study considers the global mean measure. Following day (24-hour) accurate forecasts are necessary just like wind speed predictions to estimate a photovoltaic power plant output. Cloud amount processes mainly affect the solar radiance intensity, less influenced by visibility conditions and relative humidity [15]. The cloudiness local correction models were trained and tested analogously to the wind speed local revisions (Fig. 11–Fig. 13). The D-

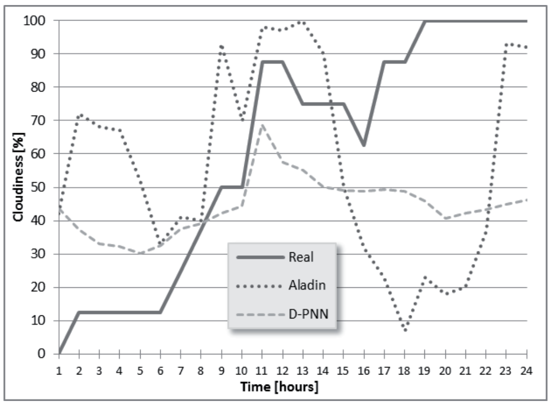

Fig. 11 *14. 6. 2013: 48-hour prognoses – Aladin (NR = 0.466, NM = 0.765), D-PNN (NR = 0.349, NM = 0.573).*

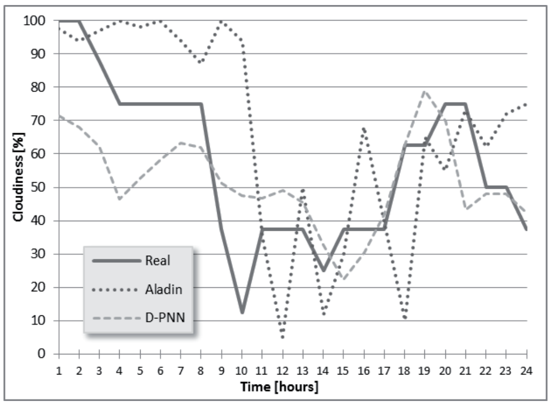

Fig. 12 *15. 6. 2013: Aladin (NR = 0.334, NM = 0.511), D-PNN (NR = 0.206, NM = 0.314).*

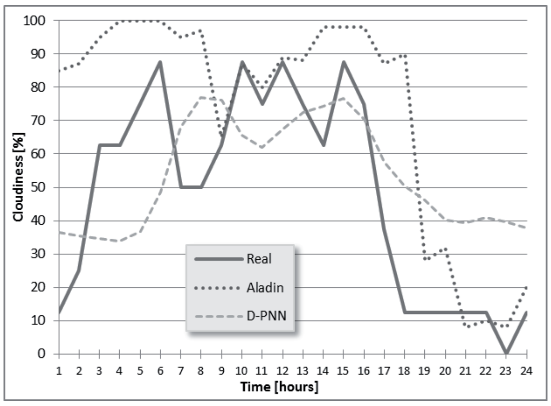

Fig. 13 *16. 6. 2013: Aladin (NR = 0.390, NM = 0.712), D-PNN (NR = 0.288, NM = 0.526).*

PNN model of Fig. 11 was formed with respect to the 48-hour "Aladin" prognosis, i.e. it applied 24-hour forecast variables of the 2nd day. The training data samples were measured at one station locality of Ostrava-Mošnov airport [19]. Each of the 3 presented D-PNN correction models were trained with hourly data series for the period of 1 to 3 days previously (24-62 hours, i.e. data samples).

5. Conclusions

D-PNN is a new neural network type, which non-linear regression is based on a derivative generalization of polynomial data relations. It forms and resolves an unknown DE of a searched function description from data observations. A general DE is substituted, producing sum series of selected fractional polynomial derivative terms (neurons). In contrast with the ANN functionality, each neuron can take part directly in the total network output calculation, generated by the sum of all active neuron output values. Its relative data processing is different from the common soft-computing method approach (e.g. ANN, GP, fuzzy), applications which are subjected to a fixed interval of absolute values. The D-PNN's operating principle differs by far from other common neural network techniques. PNN model complexity in general increases proportionally along with raising amount of input variables, which is contrary to the application of an artificial neural network common flat 1 or 2-layer structure.

Neural networks trained with past hourly local data observations, entered only a few variables of current 24 or 48-hour prognoses can improve a numerical meteorological forecast model in the majority of cases and thus replace the role of standard MOS techniques. Time-series observations of some local weather conditions over several days can describe data relations of a neural network trained model, for a certain time interval exactly. After that D-PNN can revise 24 (or 48) hour prognosis of a specific variable, applying several previous days trained real model of the general DE solution and input variables of the "Aladin" forecast. The predicted values, which enter the input vector, are only partly valid and the correction model, trained with real historical data, is also more or less correct. In the case of an unexpected overnight change in the weather from one day to another the trained model is not equally accurate and able to improve the original numerical model. The neural network local revisions of the numerical meteorological forecast models could probably be further improved using several surrounding localities of observations and an increased number of meteorological quantities.

Acknowledgement

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] Affenzeller M., Winkler S., Wagner S., Beham A.: Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications. Chapman & Hall/CRC, 2009.

[2] Balser W.: Summability of formal power-series solutions of partial differential equations with constant coefficients. Journal of Mathematical Sciences, 124, December 2004, pp. 5085–5097.

[3] Bertsimas D., Tsitsiklis J.: Simulated annealing. Statistical science, 8(1), 1993, pp. 10–15.

[4] Cao H., Kang L., Chen Y., Yu J.: Evolutionary modeling of systems of ordinary differential equations with genetic programming. Genetic Programming and Evolvable Machines, 1, October 2000, pp. 309–337.

[5] Chan K. L., Chau W. Y.: Mathematical theory of reduction of physical parameters and similarity analysis. International Journal of Theoretical Physics, 18, November 1979, pp. 835–844.

[6] Chaquet J. M., Carmona E. J.: Solving differential equations with fourier series and evolution strategies. Applied Soft Computing, 12, September 2012, pp. 3051–3062.

[7] Cornforth T. W., Lipson H.: Inference of hidden variables in systems of differential equations with genetic programming. In: Genetic Programming and Evolvable Machines. Springer, 2012.

[8] Ivakhnenko A. G.: Polynomial theory of complex systems. IEEE Transactions on systems, 1(4), 1971.

[9] Kilbas A. A., Srivastava H. M., Trujillo J. J.: Theory and applications of fractional differential equations. Mathematics Studies. Elsevier, North-Holland, 2006.

[10] Nikolaev N. Y., Iba H.: Polynomial harmonic GMDH learning networks for time series modeling. Neural Networks, 16, 2003, pp. 1527–1540.

[11] I. Podlubny. Fractional Differential Equations. Academic, New York, 1999.

[12] Tsoulos I., Gavrilis D., Glavas E.: Solving differential equations with constructed neural networks. Neurocomputing, 72, 2009, pp. 2385–2391.

[13] Zjavka L.: Combined differential polynomial neural network. Journal of Electrical and Control Engineering, 2, 2012, pp. 15–19.

[14] Zjavka L.: Recognition of generalized patterns by a differential polynomial neural network. Engineering, Technology & Applied Science Research, 2(1), 2012, pp. 167–172.

[15] Zjavka L.: Forecast models of partial differential equations using polynomial networks. In Advances in Intelligent Systems and Computing, Berlin, Springer-Verlag, volume 238, 2013, pp. 1–11.

[16] Zjavka L., Snášel V.: Power output models of ordinary differential equations by polynomial and recurrent neural networks. In Advances in Intelligent Systems and Computing, Berlin, Springer-Verlag, volume 237, pp. 1–11, 2013.

[17] Czech hydro-meteorological institute (CHMI) "Aladin" forecast http://www.chmi.cz/files/portal/docs/meteo/ov/aladin/results/public/meteogramy/meteogram_page_portal/m.html (in Czech language only)

[18] CHMI actual data observations http://vvv.chmi.cz/pocasi-na-stanici_ostrava.html (in Czech language only)

[19] CHMI actual data charts http://portal.chmi.cz/files/portal/docs/poboc/OS/KW/Captor/tmp/DMULTI-O1MOSN01.gif (in Czech language only)