Mathematical Modelling and Analysis
Volume 22 Number 1, January 2017, 78–94
https://doi.org/10.3846/13926292.2017.1269025

Publisher: Taylor&Francis and VGTU
http://www.tandfonline.com/TMMA
ISSN: 1392-6292
eISSN: 1648-3510

Modelling the Photovoltaic Output Power using the Differential Polynomial Network and Evolutionary Fuzzy Rules*

Ladislav Zjavka[b], Pavel Krömer[a,b], Stanislav Mišák[c] and Václav Snášel[a,b]

[a] *Dep. of Computer Science, FEECS, VSB - Technical University of Ostrava*
[b] *IT4innovations National Supercomputing Center, VSB - Technical University of Ostrava*
[c] *ENET Centre, VSB - Technical University of Ostrava*
17. listopadu 15/2172 Ostrava, Czech Republic
E-mail(*corresp.*): ladislav.zjavka@vsb.cz
E-mail: pavel.kromer@vsb.cz
E-mail: stanislav.misak@vsb.cz
E-mail: vaclav.snasel@vsb.cz

Received December 1, 2016; revised December 23, 2016; published online January 5, 2017

Abstract. The unstable production of renewable energy sources, which is difficult to model using conventional computational techniques, may be predicted to advantage by means of biologically inspired soft-computing methods. The photovoltaic output power is primarily dependent on the solar direct or global radiation, which short-term numerical forecasts are possible to apply for daily power predictions. The study compares two methods, which can successfully model dynamic fluctuant variances of the solar irradiance and corresponding output power time-series. Differential polynomial network is a new neural network class, which defines and substitutes for the general partial differential equation to model an unknown system function. Its total output is composed from selected neurons, i.e. relative polynomial substitution terms, formed in all network layers of a multi-layer structure. The proposed derivative polynomial regression using relative dimensionless fraction units, formed according to the Similarity analysis, can describe and generalize data relations on a wider range of values than defined by the training interval when using standard soft-computing composing techniques that apply only absolute data. 1-variable time-series observations are possible to model by time derivatives of a converted ordinary differential equation, solved analogously with partial derivative substitution terms of several time-point variables.

* This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science - LQ1602", by project LO1404 "Sustainable Development of Centre ENET", VSB-Technical University of Ostrava, student's projects SGS SP2016/177 and SP2016/175, and Czech Science Foundation project no. GJ16-25694Y.

Keywords: photovoltaic output power, solar irradiation, differential polynomial neural network, evolution fuzzy rules.

AMS Subject Classification: 68T05; 92B20.

1 Introduction

The electrical energy produced by energy sources within the network must be at the same time consumed by customers as the accumulation of its reasonable quantities is too demanding, even though experimental systems are installed at prototype. In order to maintain a reliable and efficient operation of the electrical network, the operator must be able to estimate the volume of the electrical energy produced by unstable energy sources (e.g. wind or photovoltaic power plants). In a power grid is needed a reliable prediction of the renewable power generation in order to ensure that the stable sources of electrical energy (such as coal, gas, and nuclear power plants) will be able to balance the energy production and satisfy a demand for electricity by all customers. Photovoltaic power plants offer stochastic supply of electricity, which deterministic prediction methods often do not provide sufficient accuracy. Applied algorithms usually model the electricity production within short time interval periods according to given sunshine forecasts. The photovoltaic systems, used at present, apply mostly technologies with specific parameters. The photovoltaic plant (PVP) with a mono-crystalline (PVP1) technology has higher total efficiency (about 25%) in comparison with a poly-crystalline (PVP2) technology (about 15-20%).

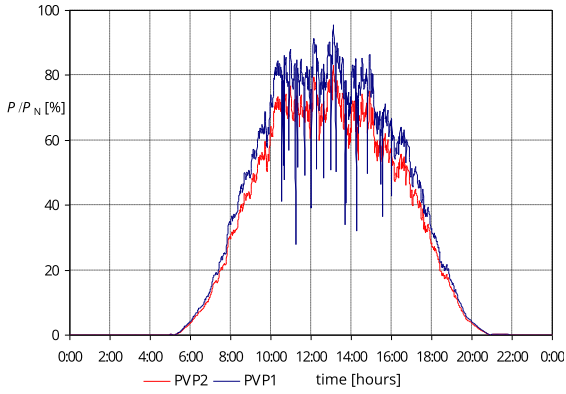

Figure 1. PVP output active power for the mono-crystalline (PVP1) and poly-crystalline technology (PVP2).

However the PVP1 can transfer only direct solar radiation by contrast to the PVP2, which can convert all components of the solar direct and diffusion radiation [18]. The influence of the direct solar radiation results in a dynamic course of the PVP1 output active power. Values of the PVP1 output active power can fluctuate in an interval 30-95% of the nominal power, whereas dynamic changes of the output active power (caused by cloud passing) are in

units of minutes in this specific case. The maximal value of the output active power of the PVP2 is not so high (about 85% of a nominal value) however the dynamic changes are smaller and this system is more stable. The characteristics are necessary to respect in prediction models. The analysis (Figure 1) characterizes a typical day during July 2011 with changeable cloudiness for the nominal power 1MWp.

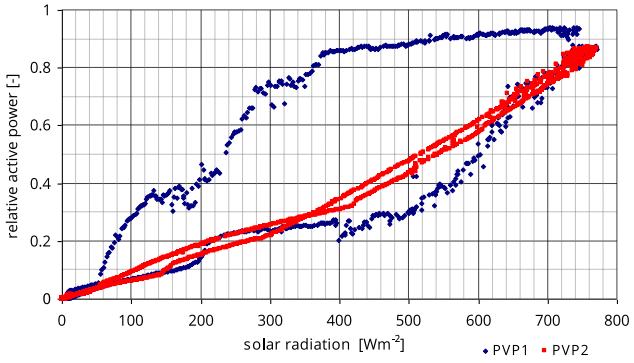

Figure 2. The power curve of PVP with monocrystalline technology (PVP1) and polycrystalline technology (PVP2).

Power curves can characterize the two above mentioned PVPs (Figure 2), where the PVP1 is a photovoltaic system placed on a tracker and the PVP2 is a system with a stabile construction. The power curve shows a dependence of the output active power on the solar radiation (Figure 2). The PVP1 power curve (placed on the tracker) has higher values of the hysteresis, which is caused by the different output power for eastern and western sun. Therefore it is possible to obtain the different output power for the same values of the solar radiation whereas output power values are higher for azimuth corresponding of eastern sun. The PVP2 hysteresis power curve is flatter by a virtue of the stabile placing and poly-crystalline technology. The power curves were created for same day as in the previous case (Figure 1).

Prediction models must respect all the above mentioned specifications of the PVP parameters. Developed prediction techniques for the PVP production, based on the optimization for specific areas, can use fuzzy sets and logic [4], neural networks [23], statistical short-term forecasting [11], non-linear function relations [25] or Gaussian equations [26]. The study compares two different types of daily prediction models: the general differential equation solutions of polynomial networks and symbolic tree expressions of evolutional fuzzy rules (of a searched function), tested at a 14-day period. The applied time-series data contain only information about the solar radiation intensity related to corresponding power output values in a location of the facility. No other additional relevant data (e.g. relative humidity, pressure, visibility) were considered, which might significantly impact the presented models results. Each of the two methods applies a different approach to the training data, either daily updated or long-term invariant models. The paper is organized as follows:

Section 2 and Section 3 present the principles of both the compared methods, Section 4 contains experimental results first using real solar irradiation data (for testing both model types) and then numerical forecast data (for real daily predictions), and Section 5 and Section 6 resume evaluations and implications.

2 General partial differential equation decomposition and substitution

2.1 General differential equation composition models

A lot of physical or natural systems, which is not possible to model using a unique explicit function, can be described to advantage by means of differential equations solved using neural networks [27], genetic programming [6, 10, 12] based on trajectory methods [22], wave series [3, 8] and other. The general partial differential equation (DE) (2.1), which selective exact form is not known in advance (for a specific system model), can generally describe a system of related variables with sum derivative terms. The searched function u, which is possible to calculate as a sum of the derivative terms (+ bias) (2.1) may be expressed in the form of sum series, consisting of convergent series arisen from partial derivative terms (2.2) of 2 input variables.

$$a + bu + \sum_{i=1}^{n} c_i \frac{\partial u}{\partial x_i} + \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} \frac{\partial^2 u}{\partial x_i \partial x_j} + \ldots = 0, \tag{2.1}$$

a, b, $\vec{c}(d_{11}, dc_{12}, \ldots)$ – parameters, $\vec{x}(x_1, x_2, \ldots, x_n)$ – vector of n input variables, $u(\vec{x})$ – searched function $u = \sum_{k=1}^{\infty} u_k$, u_k – partial separable functions of u:

$$\left(\sum \frac{\partial u_k}{\partial x_1}, \ \sum \frac{\partial u_k}{\partial x_2}, \ \sum \frac{\partial^2 u_k}{\partial x_1^2}, \ \sum \frac{\partial^2 u_k}{\partial x_1 \partial x_2}, \ \sum \frac{\partial^2 u_k}{\partial x_2^2} \right). \tag{2.2}$$

Volterra functional series, a discrete analogue of which is the Kolmogorov-Gabor polynomial (2.3), express general connections between input and output variables. This polynomial can approximate any stationary random sequence of observations and can be computed by either adaptive methods or a system of Gaussian normal equations. Group Method of Data Handling (GMDH), created by a Ukrainian scientist Aleksey Ivakhnenko in 1968, when the back-propagation technique was not known yet [13], decomposes the complexity of a system or process (2.3) into many simpler relationships each described by a low order 2-variable polynomial processing function (2.4) of a single neuron (node). The GMDH evolves progressively a polynomial neural network (PNN) multi-layer structure towards the output layer, adding layers one after one in successive steps calculating the actual last layer coefficients and selecting its optimal neurons. It defines an optimal structure of a complex system model identifying non-linear relations between input and output variables [20].

$$Y = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} x_i x_j + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} a_{ijk} x_i x_j x_k + \ldots, \tag{2.3}$$

$A(a_1, a_2, \ldots, a_n)$ – vector of parameters, $X(x_1, x_2, \ldots, x_n)$ – vector of n input variables.

Differential polynomial neural network (D-PNN) is a new neural network type, which extends the basic complete GMDH PNN structure to form sum series of relative combination derivation terms, which together define and substitute for the selective general partial differential equation (DE) of a multivariable function searched model based on data observations. This derivative series model is differs from simple upright computational techniques, which can compose a searched function from a collection of operators and terminals of a defined set to form symbolic tree-like structural expressions.

$$y = a_0 + a_1 x_1 + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2. \tag{2.4}$$

The similarity theory is based on the hypothesis functional relationships exist among the non-dimensional parameters, which can describe a physical system. The Buckingham π theorem removes extraneous information from a problem by forming dimensionless groups of variables and is the fundamental of dimensional analysis. It states if the Equation (2.5) is the only relationship among the q_i's and if it holds for any arbitrary choice of the units in which $q_1, q_2, \ldots, q_n$ are measured, then it can be written in the form using $\pi_1, \pi_2, \ldots, \pi_m$ as independent dimensionless products of the q_i's (7). If k is the minimal number of principal quantities necessary to express the dimensions of the q's, then $m = n - k$ [24].

$$\Phi(q_1, q_2, \ldots, q_n) = 0, \quad \Phi(\pi_1, \pi_2, \ldots, \pi_n) = 0, \tag{2.5}$$

where $\pi_1, \pi_2, \ldots, \pi_m$ are independent dimensionless products of the q's. The method of integral analogues provides an essential principle for the partial derivative terms (2.2) polynomial substitution. It is a part of the similarity dimensional analysis, which applies various formal adaptations of a DE (or data observations) to form dimensionless characteristic groups of variables [7], thus the original mathematical operators and symbols in a DE are replaced by the ratio of corresponding variables. The derivatives are replaced by their integral analogues, i.e. derivative operators are removed and replaced by analogue or proportion signs in equations. The complete input polynomials (of a uniform degree) replace the partial searched functions u_k of derivative terms (2.2), while the reduced combination polynomials of denominators represent the derivative versatile parts (2.9). If the $u(x_1, x_2, \ldots, x_n)$ function of the partial DE (2.1) is defined around a point $A(a_1, a_2, \ldots, a_n)$, its partial derivation in respect of x_i may be considered 1-variable function $g(x_i)$ (2.6).

$$\begin{aligned}
\left[\frac{\partial u}{\partial x_i}\right]_A &= \frac{\partial u(A)}{\partial x_i} \\
&= \lim_{x_i \to a_i} \frac{u(a_1, \ldots, a_{i-1}, x_i, a_{i+1}, \ldots, a_n) - u(a_1, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_n)}{x_i - a_i} \\
&= \lim_{x_i \to a_i} \frac{g(x_i) - g(a_i)}{x_i - a_i}.
\end{aligned} \tag{2.6}$$

If the partial DE involves $u(\vec{x})$ function derivatives with respect only to time t variable, which the independent variables $x_1, x_2, \ldots, x_n$ depends on, the equation (2.6) is possible to express in (2.7).

$$\left[\frac{du(t, x_1, \ldots, x_n)}{dt}\right]_{t=t_a} = \lim_{t \to t_a} \frac{u(t, a_1, \ldots, a_n) - u(t_a, a_1, \ldots, a_n)}{t - t_a},$$
$$dx = x_i - a_i = t - t_a = dt. \tag{2.7}$$

If only one independent variable x occurs in the function u, the equation (2.7) becomes the form (2.8).

$$\left[\frac{du(t, x)}{dt}\right]_{t=t_a} = \lim_{t \to t_a} \frac{u(t, a) - u(t_a, a)}{t - t_a}. \tag{2.8}$$

On this assumption the general partial DE (2.1), which describes 1-variable time-series, might be converted into an ordinary DE with only time derivatives. The variables $x_1, x_2, \ldots, x_n$ of the u function are observations in various time t, so partial derivatives (2.1) turns into derivatives in respect of the only time t variable. The input vector $\vec{x}$ variables depend only on time t so the searched function u is replaced by $f(\vec{x})$ time observations. Combinations of the partial DE term denominators (2.1) project themselves into partial numerator $f(\vec{x})$ functions in the ordinary DE

$$a + bf + \sum_{i=1}^{n} c_i \frac{df(x_i)}{dt} + \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} \frac{df(x_i, x_j)}{dt} + \ldots + \sum_{i=1}^{n} cc_i \frac{df^2(x_i)}{dt^2} + \ldots = 0,$$

where $f(\vec{x})$ – function of independent time observations $x(x_1, x_2, \ldots, x_n)$. This equation is formed and solved analogous to the general partial DE (2.1) by means of partial derivative terms substitutions using combinations of several time-point variables.

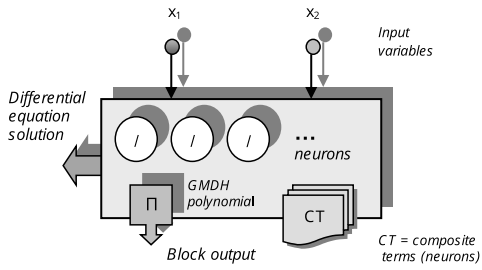

Figure 3. 2-variable combination block consists of basic (/) and composite terms (CT) - neurons.

Blocks (nodes) of the D-PNN (Figure 3) form simple neurons, one for each fractional polynomial derivative combination, so each neuron represents a sub-

stitution DE term (2.9) (2.10) (2.11)

$$y_1 = \frac{\partial f(x_1, x_2)}{\partial x_1} = w_1 \frac{(a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2)^{\frac{1}{2}}}{1.5(b_0 + b_1 x_1)}, \quad (2.9)$$

$$y_3 = \frac{\partial^2 f(x_1, x_2)}{\partial^2 x_2} = w_3 \frac{a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2}{2.7(b_0 + b_1 x_2 + b_2 x_2^2)}, \quad (2.10)$$

$$y_5 = \frac{\partial^2 f(x_1, x_2)}{\partial x_1 \partial x_2} = w_5 \frac{a_0 + a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 + a_4 x_1^2 + a_5 x_2^2}{2.3(b_0 + b_1 x_{11} + b_2 x_{12} + b_3 x_{11} x_{12})}. \quad (2.11)$$

Each block contains a single output polynomial (2.4), without derivative part. Neurons do not affect the block output but can be directly included in the total network output sum of a DE solution. Each block has 1 and neuron 2 vectors of adjustable parameters $\vec{a}$ and $\vec{a}, \vec{b}$ respectively. The numerator root function decreases a combination degree of the input polynomial (2.9), in order to get the dimensionless values [7].

The GMDH polynomial (2.4) is applied in the block outputs and neuron fractions as it proves generally to yield best results besides being easy to use. Each block includes 5 simple substitution neurons of the 2nd order partial DE (2.12) of a searched 2-variable partial u function model, formed with respect to 2 single x_1, x_2 (2.9), 2 squared x_1^2, x_2^2 (2.10) and 1 combination $x_1 x_2$ (2.11) derivative variables, which correspond exactly to the GMDH polynomial variables. This DE type is preferably used to describe natural and physical systems non-linearities.

$$F\left(x_1, x_2, u, \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \frac{\partial^2 u}{\partial x_1^2}, \frac{\partial^2 u}{\partial x_1 \partial x_2}, \frac{\partial^2 u}{\partial x_2^2}\right) = 0, \quad (2.12)$$

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

2.2 Differential polynomial neural network

Multi-layer D-PNN forms composite functions (Figure 4). The previous layer blocks produce internal functions, which substitute for the next hidden layer input variables of neuron and block polynomials in external functions. Composite DE terms, i.e. composite function derivatives in respect of the variables of previous layers, are calculated according to the partial derivation rules (2.13), (2.14) by products of the external and internal function partial derivatives.

$$F(x_1, x_2, \ldots, x_n) = f(y_1, y_2, \ldots, y_m) = f(\Phi_1(X), \Phi_2(X), \ldots, \Phi_m(X)), \quad (2.13)$$

$$\frac{\partial F}{\partial x_k} = \sum_{i=1}^{m} \frac{\partial f(y_1, y_2, \ldots, y_m)}{\partial y_i} \cdot \frac{\partial \Phi_i(X)}{\partial x_k}, \quad k = 1, \ldots, n. \quad (2.14)$$

The blocks of the 2nd and following hidden layers produce extended neurons - composite terms (CT), which substitute for the composite function derivatives in respect of the output and input variables of the back connected previous layer blocks, e.g. the 1st block of the 3rd layer (Figure 4) can form linear CT

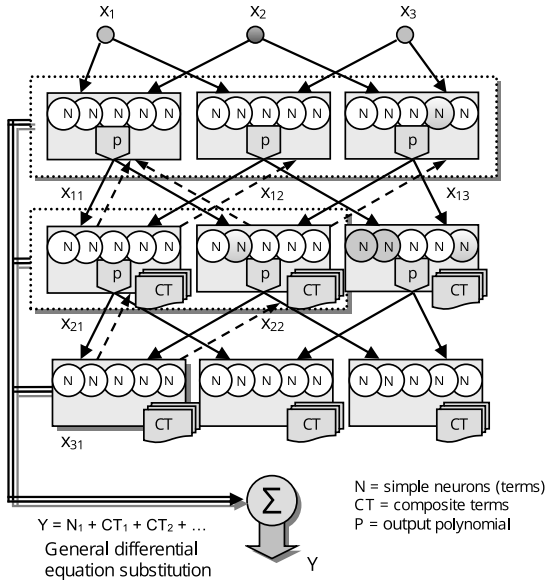

Figure 4. Backward connections of the D-PNN 3rd layer 1st block (dashed lines).

with respect to 2nd (2.15) and 1st layer (2.16) derivative input variables. As the couples of input variables of the internal functions $\Phi_1(x_1, x_2)$ and $\Phi_2(x_3, x_4)$ (2.11) can differ from each other, the partial derivatives are calculated separately in respect of each variable. This way the sums (2.12) consist of only 1 derivative term, which a single neuron represents again. The number of the neurons in blocks, which form composite polynomials, doubles with each previous back-connected layer. A backward recursive algorithm can easy form and calculate the CT from composite functions in the network tree-like structure. The squared and combination derivative terms are calculated analogously according to the composite function derivation rules however this study does not apply the complete composed terms from the formulas [29].

$$y_2 = \frac{\partial f(x_{21}, x_{22})}{\partial x_{11}} = w_2 \frac{(a_0 + a_1 x_{21} + a_2 x_{22} + a_3 x_{21} x_{22} + a_4 x_{21}^2 + a_5 x_{22}^2)^{\frac{1}{2}}}{1.5 x_{22}} \times \frac{(x_{21})^{\frac{1}{2}}}{1.5(b_0 + b_1 x_{11})}, \tag{2.15}$$

$$y_3 = \frac{\partial f(x_{21}, x_{22})}{\partial x_1} = w_3 \frac{(a_0 + a_1 x_{21} + a_2 x_{22} + a_3 x_{21} x_{22} + a_4 x_{21}^2 + a_5 x_{22}^2)^{\frac{1}{2}}}{1.5 x_{22}} \times \frac{(x_{21})^{\frac{1}{2}}}{1.5 x_{12}} \frac{(x_{11})^{\frac{1}{2}}}{1.5(b_0 + b_1 x_1)}. \tag{2.16}$$

Only some of all the potential substitution derivative terms (neurons) may form a DE solution, even though they have an adjustable term weight w_i. A proper neuron combination, which can substitute for a general DE solution, is not able to accept a disturbing effect of the rest of the neurons (which may compose

other solutions) in the parameter optimization. The D-PNN total output Y is calculated as the arithmetic mean of all active neuron outputs $Y = \frac{1}{k}\sum_{i=1}^{k} y_i$, so as to prevent a changeable number of selected active neurons (of a combination) from influencing its value. Here k is the number of active neurons (DE terms). The selection of a fit neuron combination is the principal part of the DE composition and may apply the binary particle swarm optimization in the initial composing phase, rather than a standard genetic algorithm as the chain lengths (number of neurons) may differ in each individual solution. Parameters of polynomials are adjusted by means of the gradient steepest descent (GSD) method, supplied with sufficient random mutations to prevent from trapping to a local error minima [29]. The GSD parameter updates result from the partial derivatives of the polynomial substitution DE terms with respect to the single parameters $\vec{a}, \vec{b}$ [20] analogous to the standard neural network error calculation.

3 Evolutionary fuzzy rules

Fuzzy classification is an umbrella term for different methods capable of efficient soft classification of data. In contrast to its crisp counterpart, fuzzy classification provides a more sensitive tools for data analysis [4]. Fuzzy decision trees and fuzzy if-then rules are prime examples of efficient, transparent, and intelligible fuzzy classifiers and value estimators [4].

The need for interpretable and linguistically comprehensible classification and regression tools is widely recognized [9, 28]. It is also well-known that bio-inspired and evolutionary methods possess the ability to learn and optimize various types of fuzzy systems [9] and data mining models [2]. Evolutionary fuzzy rules (FR) [16, 17] are simple yet powerful classification and regression instruments based on the merger of fuzzy information retrieval (IR) and genetic programming.

Fuzzy information retrieval uses extended Boolean queries that consist of search terms, operators, and weights, and evaluates them against an internal representation (index) of a collection of documents. It is based on the fuzzy set theory and fuzzy logic that facilitate flexible and accurate search [21]. Evolutionary fuzzy rules use similar basic concepts, data structures, and operations, and apply them to general data processing tasks such as classification, prediction, and so forth. Here, the concepts of information retrieval are employed to interpret data and to define the classification or regression models. Symbolic rules of such models are then evolved using genetic programming [1, 15], a generic, problem-independent meta-heuristic machine learning algorithm.

The data processed by a fuzzy rule is a real valued matrix. Each row of the matrix corresponds to a single data record interpreted as a fuzzy set of features. Such a general, real valued matrix $\boldsymbol{D}$ with m rows (data records) and n columns (data attributes, features) can be mapped to an IR index that describes a collection of objects.

A fuzzy rule takes the form of a weighted symbolic expression roughly corresponding to an extended Boolean query in the fuzzy IR analogy. The rule consists of weighted feature names and weighted aggregation operators. The evaluation of such an expression assigns to each data record a real value from

the range $[0, 1]$. Such valuation can be interpreted as an ordering, labeling, or a fuzzy set induced on the data records.

The fuzzy rule is a symbolic expression that can be parsed into a tree structure consisting of nodes and leafs. There are three types of leafs (a.k.a. terminal nodes)

a) *Feature* node which represents the name of a feature (a search term in the IR analogy). It specifies a requirement for a particular feature in the currently processed data record.

b) *Past feature* node which defines a requirement on certain feature in a previous data record. The index of the previous data record (current - 1, current - 2, etc.) is a parameter of the node.

c) *Past output* node which puts the requirement on a previous output of the predictor. The index of the previous output (current - 1, current - 2, etc.) is a parameter of the node.

A fuzzy rule can be expressed using a simple infix notation

feature1:0.5 and:0.4 (feature2[1]:0.3 or:0.1 ([1]:0.1 and:0.2 [2]:0.3)),

where *feature1:0.5* is a feature node, *feature2[1]:0.3* is a past feature node, and *[1]:0.5* is a past output node. Different node types can be used when dealing with different data sets. For example, the past feature node and past output node are useful for the analysis of time series and data sets where the ordering of the records matters. But their usage is pointless for the analysis of regular data sets. The feature node is the basic building block of classifiers and predictors developed for arbitrary data. An example is given in Figure 5.

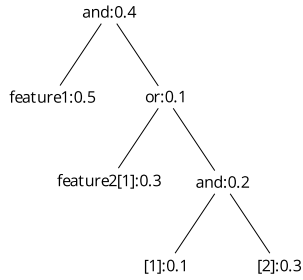

Figure 5. An example of a fuzzy rule.

The evaluation of a node, *f:a*, with value $f \in [0, 1]$ and weight $a \in [0, 1]$, is performed using a threshold interpretation of the retrieval status value (RSV) concept known from fuzzy information retrieval [5]

$$g(f, a) = \begin{cases} P(a)f/a, & f < a, \\ P(a) + Q(a)(f - a)/(1 - a), & f \geq a, \end{cases}$$

where $P(a) = (1 + a)/2$ and $Q(a) = (1 - a^2)/4$ are coefficients used to fine-tune the threshold curve.

Typical evolutionary fuzzy rules support *and*, *or*, *not*, *prod*, and *sum* operator nodes. However, more general or domain specific operators can be used as well. Both nodes and leafs are weighted to soften the criteria they represent. The operators *and*, *or*, *not*, *prod*, and *sum* are evaluated using fuzzy set operations. In this study, the standard t-norm and s-norm are used to implement *and* and *or* operators, respectively

$$t(x, y) = \min(x, y), \quad s(x, y) = \max(x, y),$$

operator *not* is evaluated using the standard fuzzy complement $c(x) = 1 - x$ and *prod* and *sum* operators, respectively, using the product t-norm and its dual s-norm, bounded sum

$$t_{\text{prod}}(x, y) = xy, \quad s_{\text{sum}}(x, y) = a + b - ab.$$

However, other classes of complement, intersection, and union models [14, 19] can be used as well.

A fuzzy rule is a simple version of a general fuzzy rule-based system that consists of a single expression describing soft requirements on data records in terms of their features. In evolutionary fuzzy rules, this expression is evolved using genetic programming [1]. The tree structures, corresponding to the parsed fuzzy rules, are developed by an iterative application of crossover, mutation, and selection operators in order to find an accurate model of the training data.

The general process of rule evolution is used for data-driven search for custom classifiers or predictors. Different data sets may by characterized by different properties and different hidden structure, and the adaptability of genetic programming is essential for the evolution of problem-specific fuzzy rules. On the other hand, the stochastic nature of genetic programming introduces probabilistic elements into the process of rule evolution.

The evolutionary fuzzy rules, although machine-generated, retain the understandable structure and ease of interpretation inherited from the extended Boolean search expressions, and allow a soft classification/regression without the complexity and computational costs of full-featured fuzzy rule-based systems [16].

4 Real data experiments

The above two methods prediction models were verified on the database of measured values (output active power, solar radiation) for PVP1 with monocrystalline technology and nominal power 1MWp. The volumes of electrical energy produced by the PVP and values of the solar radiation in the same location were recorded in 10 minute intervals between November 2010 and April 2011. The complete data set for the evolutionary fuzzy rules application was divided into two halves. The first part (i.e. for about 3 months) was used as the training data and the second part as the testing set.

The D-PNN was trained in other way, updating its DE models per diem (Figure 6). It was trained with time-series of last 1 up to 4 days to form the following day power model, which is possible to test a day before the training interval in real-time.

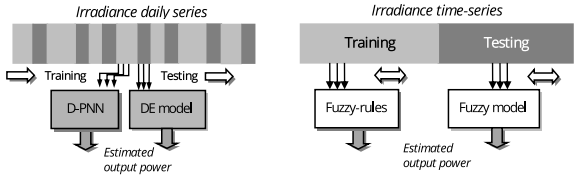

Figure 6. The daily updated (D-PNN) and completely trained (Fuzzy-rules) models.

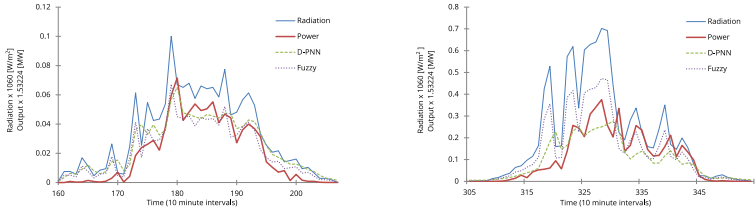

Figure 7. RMSE: **Day 0.** D-PNN = 0.00833, Fuzzy = 0.00987 - back-time check D-PNN model computed from the Day 1. **Day 3.** D-PNN = 0.05433, Fuzzy = 0.09243.

Figure 7 shows this back-computed model of day 0, trained with the following day 1. The average number of the applied derivative neurons (substitution DE terms) in the total network sum output was around 60 and it can a few vary as per a model solution.

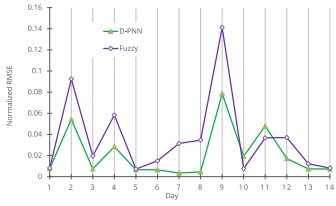

Figure 8. 14-day PVP power average testing RMSE of normalized data D-PNN = 0.0212, Fuzzy = 0.0364.

Figure 8 shows the root mean squared error (RMSE), calculated on the testing normalized data, in each of 14 modelled successive days. Both methods managed to estimate the PVP active power quite well for some days and less accurately for several other days (day 3. and day 10.).

Each Figure 7 – 10 show a day time-window with non-zero output power (i.e. solar radiation). The average testing error is around 1-2% of a daily output active power peak, which is a satisfactory result, considering that only 1-variable time-series of the solar radiation intensity were available. Both the applied methods provide mostly similar results but the responses to some days input patterns differ fairly as the computing paradigms do. However the D-PNN models, based on the up-dated time ordinary DE solutions, appear any better. The presented models applied only 3 input time-series of the solar

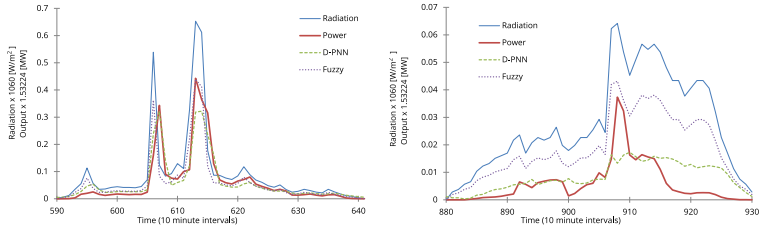

Figure 9. RMSE: **Day 5.** D-PNN = 0.02837, Fuzzy = 0.05832. **Day 7.** D-PNN = 0.00650, Fuzzy = 0.014978.

irradiance to estimate the PVP active power, corresponding to the end-time (3rd) variable.

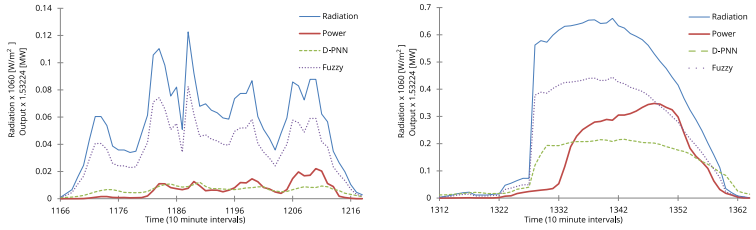

Figure 10. RMSE: **Day 9.** D-PNN = 0.00450, Fuzzy = 0.03458. **Day 10.** D-PNN = 0.07857, Fuzzy = 0.14127.

The D-PNN (more successful method) was chosen to predict the daily PVP2 output power using 24-hour "Aladin" forecasts of the solar irradiance, which accuracy primarily influences the power model estimations (Figure 11), tested on real actual data only in the previous experiments.

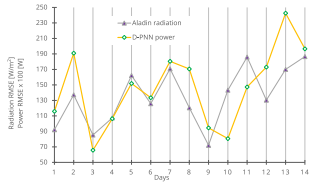

Figure 11. 14-day average prediction RMSE of the PVP power: D-PNN = 134.98 and solar irradiance: Aladin = 146.43 from 13.6.2011 to 26.6.2011 in Starojicska Lhota (Czech rep.).

The short-term "Aladin" numerical forecast model refines the global French ARPEGE model on a middle-scale target area using a more detailed time and spatial resolution of the interpolation. The forecast model, calculated for following 48-hours at 0:00 CET in the nodal point 17.9132/49.5860 (the closed to Starojicska Lhota, Czech rep.), can predict the direct and diffusion (together

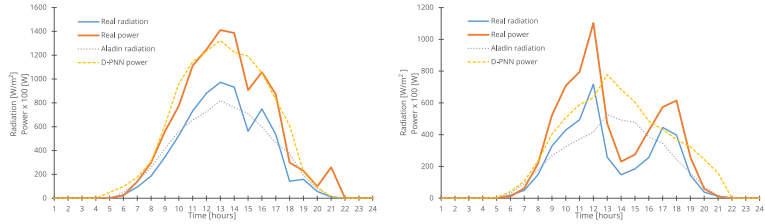

Figure 12. RMSE: **Day 1.** D-PNN = 99.09, Aladin = 115.91. **Day 2.** D-PNN = 137.21, Aladin = 190.99.

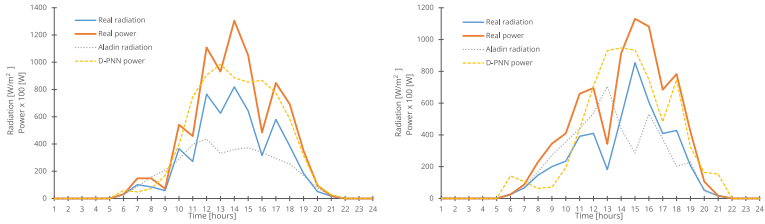

Figure 13. RMSE: **Day 5.** D-PNN = 162.16, Aladin = 151.87. **Day 7.** D-PNN = 170.82, Aladin = 180.41.

global) solar radiation, which sum enter the PVP2 power prediction models using 2nd data set (Figure 12 – Figure 14). The D-PNN model, trained with hourly averaged real time-series (in respect of hourly forecasts) of several previous days, applies the "Aladin" irradiance prognoses, which inaccuracies it may partly compensate in some cases (Figure 13). An extended D-PNN model, based on a partial DE solution, using weather surface observations of several previous days (in several surrounding locations) combined with complete 24-hour forecasts (of the relative humidity, static pressure, etc.), might improve the daily solar irradiance and consequent PVP power predictions (Figure 12, Day 2).

5 Discussion

The D-PNN models, based on ordinary differential equation solutions, proved to provide more accurate PVP power estimations than the presented fuzzy rule algorithm. The daily updated D-PNN model merits become evident in the real 24-hour PVP2 power prediction using global solar irradiance forecasts of the numerical weather prediction model "Aladin". D-PNN models are updated with only some few last day time-series, thus the latest derivative changes projected themselves into a DE solution, which utilizes less but up to date data samples. Analogous daily trained recurrent neural network (RNN) models provide very similar results [30]. On the other hand if the weather conditions change suddenly, overnight from the training to testing day(s), the D-PNN

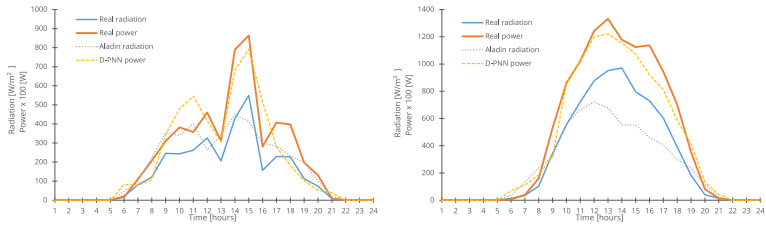

Figure 14. RMSE: **Day 9.** D-PNN = 72.11, Aladin = 94.55. **Day 10.** D-PNN = 143.28, Aladin = 80.68.

model can also show itself out of truth enough (day 10. in Figure 10) however these cases are less frequent. As a result if a sizable (fixed) testing error of the D-PNN daily power estimations (in the last not trained day verification) is exceeded, there should be applied the alternative fuzzy (or ANN) completely trained prediction model, which results from much varied long-term training conditions and accordingly comprises considerable features of far more data samples.

6 Conclusions

The PVP1 power supplies are very unstable as a result of the applied technology and dynamic changes in the solar radiation intensity, which influences also the more stable PVP2 power generation. To eliminate the rapid power changes a regulation system is used by the network operator to stable the operation. The key is mainly to estimate the power generation from these sources for certain short time (day ahead) future intervals. Two different types of the neuro and fuzzy-computing methods were tested at the 14-day data period. The evolution fuzzy rule algorithm applies genetic programming techniques with the principles of fuzzy information retrieval. Its training data set included one half of the complete data at disposal. D-PNN was trained only with time-series of several last days, updating its DE model per diem. Its relative data computing allow apply more varied training and testing data intervals of the daily time-series. The presented fuzzy model needs apply a larger training data set to eliminate this handicap, as the absolute data values may differ considerably from day to day (day 9. and 10. in Figure 10). For this reason some days testing errors may quite intensify (day 3. in Figure 7, day 7. in Figure 9 and day 9. in Figure 10), which applies also for completely trained ANN power models [23].

References

[1] M. Affenzeller, S. Wagner, S. Winkler and A. Beham. *Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications*. Chapman and Hall/CRC, 2009. ISBN 1584886293, 9781584886297.

[2] J. Bacardit and X. Llorà. Large-scale data mining using genetics-based machine learning. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, **3**(1):37–61, 2013. https://doi.org/10.1002/widm.1078.

[3] C. Bai. Exact solutions for nonlinear partial differential equation: a new approach. *Physics Letters A*, **288**(3-4):191–195, 2001. https://doi.org/10.1016/S0375-9601(01)00522-9.

[4] J.C. Bezdek, J. Keller, R. Krisnapuram and N.R. Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing (The Handbooks of Fuzzy Sets)*. Springer-Verlag New York, Inc., Secaucus, NY, USA, 2005.

[5] G. Bordogna and G. Pasi. Modeling vagueness in information retrieval. In *Lectures on information retrieval*, pp. 207–241. Springer-Verlag New York, Inc. New York, NY, USA, 2001.

[6] H. Cao, L. Kang, Y. Chen and J. Yu. Evolutionary modeling of systems of ordinary differential equations with genetic programming. *Genetic Programming and Evolvable Machines*, **1**(4):309–337, 2000. https://doi.org/10.1023/A:1010013106294.

[7] K.L. Chan and W. Y. Chau. Mathematical theory of reduction of physical parameters and similarity analysis. *International Journal of Theoretical Physics*, **18**(11):835–844, 1979. https://doi.org/10.1007/BF00670461.

[8] J.M. Chaquet and E.J. Carmona. Solving differential equations with Fourier series and evolution strategies. *Applied Soft Computing*, **12**(9):3051–3062, 2012. https://doi.org/10.1016/j.asoc.2012.05.014.

[9] O. Cordón. A historical review of evolutionary learning methods for Mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems. *International Journal of Approximate Reasoning*, **52**(6):894–913, 2011. https://doi.org/10.1016/j.ijar.2011.03.004.

[10] T.W. Cornforth and H. Lipson. Inference of hidden variables in systems of differential equations with genetic programming. In *Genetic Programming and Evolvable Machines*, pp. 155–190. Springer, 2013. https://doi.org/10.1007/s10710-012-9175-4.

[11] L.A. Fernandez-Jimenez, A. Munoz-Jimenez, A. Falces, M. Mendoza-Villena, E. Garcia-Garrido, P.M. Lara-Santillan, E. Zorzano-Alba and P.J. Zorzano-Santamaria. Short-term power forecasting system for photovoltaic plants. *Renewable Energy*, **44**(C):311–317, 2012.

[12] H. Iba. Inference of differential equation models by genetic programming. *Information Sciences: an International Journal*, **178**(23):4453–4468, 2008. https://doi.org/10.1016/j.ins.2008.07.029.

[13] A.G. Ivakhnenko. Polynomial theory of complex systems. *IEEE Transactions on Systems, Man and Cybernetics*, **1**(4):364–378, 1971. https://doi.org/10.1109/TSMC.1971.4308320.

[14] G.J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall, INc. Upper Saddle River, NY, USA, 1995.

[15] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[16] P. Krömer, S. J. Owais, J. Platoš and V. Snášel. Towards new directions of data mining by evolutionary fuzzy rules and symbolic regression. *Computers & Mathematics with Applications*, **66**(2):190–200, 2013. https://doi.org/10.1016/j.camwa.2013.02.017.

[17] P. Krömer, J. Platoš, V. Snášel and A. Abraham. Fuzzy classification by evolutionary algorithms. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pp. 313–318, 2011. https://doi.org/10.1109/ICSMC.2011.6083684.

[18] P. Mastný, J. Drápela, S. Mišák, J. Macháček, M. Ptáček, L. Radil, T. Bartošík and T. Pavelka. *Renewables sources of electric energy.* ČVUT,, Praha, 2011.

[19] P. Musilek, R. Guanlao and G. Barreiro. Genetic programming of fuzzy aggregation operations. *Journal of Intelligent & Fuzzy Systems*, **16**(2):107–118, 2005.

[20] N.Y. Nikolaev and H. Iba. Polynomial harmonic GMDH learning networks for time series modeling. *Neural Networks*, **16**(10):1527–1540, 2003. https://doi.org/10.1016/S0893-6080(03)00188-6.

[21] G. Pasi. Fuzzy sets in information retrieval: State of the art and research trends. In Humberto Bustince, Francisco Herrera and Javier Montero(Eds.), *Fuzzy Sets and Their Extensions: Representation, Aggregation and Models*, volume 220 of *Studies in Fuzziness and Soft Computing*, pp. 517–535. Springer Berlin Heidelberg, 2008. https://doi.org/10.1007/978-3-540-73723-0_26.

[22] P. Perona, A. Porporato and L. Ridolfi. On the trajectory method for the reconstruction of differential equations from time series. *Nonlinear Dynamics*, **23**:13–33, 2000. https://doi.org/10.1023/A:1008335507636.

[23] L. Prokop, S. Mišák, T. Novosád, P. Krömer, J. Platoš and V. Snášel. Photovoltaic power plant output estimation by neural networks and fuzzy inference. In *Intelligent Data Engineering and Automated Learning, Proceedings of the* 13*th International Conference, Natal, Brazil, August* 29 − 31*,* 2012., pp. 810–817, 2012.

[24] D. Randall. *Dimensional Analysis, Scale Analysis, and Similarity Theories.* 2012.

[25] C.R. Sanchez Reinoso, M. Cutrera, M. Battioni, D.H. Milone and R.H. Buitrago. Photovoltaic generation model as a function of weather variables using artificial intelligence techniques. *International Journal of Hydrogen Energy*, **37**(19):14781–14785, 2012.

[26] Y. Su, L.-Ch. Chan, L. Shu and K.-L. Tsui. Real-time prediction models for output power and efficiency of grid-connected solar photovoltaic systems. *Applied Energy*, **93**:319–326, 2012. https://doi.org/10.1016/j.apenergy.2011.12.052.

[27] I.G. Tsoulos, D. Gavrilis and E. Glavas. Solving differential equations with constructed neural networks. *Neurocomputing*, **72**(10-12):2385–2391, 2009. https://doi.org/10.1016/j.neucom.2008.12.004.

[28] L.-X. Wang and J.M. Mendel. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man and Cybernetics*, **22**(6):1414–1427, 1992. https://doi.org/10.1109/21.199466.

[29] L. Zjavka and V. Snášel. Constructing ordinary sum differential equations using polynomial networks. *Information Sciences*, **281**:462–477, 2014. https://doi.org/10.1016/j.ins.2014.05.036.

[30] L. Zjavka and V. Snášel. *Power output models of Ordinary Differential Equations by Polynomial and Recurrent Neural Networks*, volume 237 of *Advances in Intelligent Systems and Computing*, pp. 1–11. Springer International Publishing, Cham, 2014. https://doi.org/10.1007/978-3-319-01781-5_1.