# USING AI METHODS TO FIND A NON-LINEAR REGRESSION MODEL WITH A COUPLING CONDITION

Radomil Matoušek\*

In the real-life engineering practice, non-linear regression models have to be designed rather often. To ensure their technical or physical feasibility, such models may, in addition, require another coupling condition. This paper describes two procedures for designing a specific non-linear model using AI methods. A Radial Basis Functions (RBF) based optimization is presented of the model using Genetic Algorithms (GA). The problem solved was based on practical measurements and experiments. The results presented in the paper can be applied to many technical problems in mechanical and civil engineering and other engineering fields.

Keywords: non-linear regression, approximation, genetic algorithms, radial basis function, optimization

#### 1. Introduction

One of major problems encountered in processing the results of various measurements is to determine and mathematically describe the dependence of one observed quantity y on another observed quantity x, that is, to establish the mutual dependence of the problems observed. In the sequel, the quantity y, which is usually determined with a smaller relative precision (such as the star magnitude or forming speed in a ramming test) will be called a dependent (explained) variable while the other variable observed, which is usually determined more precisely (such as tome or degree of reshaping) will be taken for an independent (explaining) quantity. The actual relationship between these two variables is given by the dependence sought, that is, by a function y(x).

In technical applications, often a suitable mathematical regression model needs to be found on the basis of data measured in an experiment. As the regression model is designed to represent physical and technical processes, certain physical and technical limitations must be considered. Thus such a regression model should :

- well approximate the values measured,
- meet some physical and technical requirements,
- be of as low a degree as possible.

The second requirement refers to physical and technical limiting conditions such as the temperature being kept above a fixed limit, the forming speed being non-negative, and others.

The regression dependence presented relates to the design and optimization of a suitable regression model for describing the dependences between data measured in a material ramming test. Technically, this area is part of material forming. During the testing, multiple

<sup>\*</sup> Ing. R. Matoušek, Ph.D., Department of Applied Computer Science, Brno University of Technology, Technicka 2, Brno, Czech Republic

technological dependences were investigated. Being part of this paper, the solution proposed focuses on a specific part of the data measured with the independent variable being the degree of deformation (logarithmic degree of deformation) and the dependent variable being the deformation speed. Because of its nature, this project may also be applied to other dependence types with a similar type of limitation or type of the data measured, which abound in technical disciplines.

## 2. Material forming and input data

Material forming is a highly efficient technology. By forming, the material size and shape are gradually changed by a forming tool. During forming, each material is characterized by its specific properties. In addition, these properties are being influenced by the specific condition of the forming. For a particular forming process, it should of course be known, under what circumstances, the forming will be optimal. For this purpose, various characteristics are being set up based on what is called material forming tests [10] depending on the temperature and deformation. As the results of the forming tests may be influenced by random errors, they are approximated by a regression function.

b17 (	$arphi, \dot{arphi})$	a48 (	$(arphi,\dot{arphi})$
0.000	0.000	0.000	0.000
0.790	0.835	0.582	0.401
1.035	2.740	0.741	0.390
1.648	1.708	1.115	0.273
1.692	2.077	1.284	0.060
2.562	2.300	1.817	0.073
3.133	2.392	2.344	0.073
4.249	2.201	2.867	0.075
5.355	2.783	3.285	0.075
6.245	2.239	4.121	0.078
7.068	1.537	5.020	0.082
7.640	1.252	5.827	0.085
8.461	0.242	6.845	0.091
9.564	0.200	7.767	0.103
10.569	0.209	8.754	0.107
11.767	0.244	9.424	0.117
12.631	0.219	11.515	0.136
13.481	0.238	12.774	0.162
14.595	0.295	14.060	0.175
15.492	0.285	15.285	0.217
17.825	0.314		

Data characteristic : b17 (data set b17r.txt)			
Material: 423018.2		Sample	size [mm]
Temperature $[^{\circ}C]:$ 21		$D_0 = 9.96$	$H_0 = 16.06$
Characteristic: Nonferrous metal material containing the following			
elements and their percentages: Sn 7.53; Pb 0.013 and Cu up to 100			
percent. Alternative material denotation is CuSn8.			
Data characteristic : a48 (data set a48r.txt)			
Material: 16231.3		Sample	size [mm]
Temperature [°C]: 200 $D_0 = 15.10  H_0 = 24.07$			
Characteristic: Cementation chrome-nickel steel, known as 'Nital'			
also suitable for hardening treatment. Even with the low carbon con-			
tent, the steel has a high core strength.			

Tab.1: Sample data to be used for the regression calculation



Fig.1: Graphical representation of the data samples to be used for the regression calculation

Data for the non-ferrous material (ČSN 423018) and steel (ČSN 16231) were chosen as representative samples. The testing samples of the above materials were made by turning with the  $H_0: D_0 = 1.6$  ratio.

$$\varphi = \ln \frac{H_0}{H} \,. \tag{1}$$

By changing the height H of the object deformed and the initial height  $H_0$  of the sample, the logarithmic deformation (reshaping) degree was determined at each measurement point.

#### 3. Regression with a functional limitation

The regression function (of n variables) is required to be closed on a set  $M \subset \Re^n$ . Under the physical and technical constraint, the regression function proposed should be non-negative as seen in Fig. 1. Thus the resulting optimization problem is an approximation with functional constraint, which can be formulated as follows:

$$\min E = \min \sum_{t=1}^{k} f_{\rm w}^{(t)} \left( d^{(t)} - f_{\rm a}^{(t)} \right)^2 , \qquad (2)$$

$$f_{\rm a} = f_{\rm a}(\mathbf{x}, \mathbf{p}) \ge 0 , \qquad (3)$$

where  $\{(\mathbf{x}^{(t)}, \mathbf{d}^{(t)}); t = 1, ..., k\}$  ... k measurements;  $\mathbf{x}$  ... vector of independent variables,  $\mathbf{x} \in M \subset \Re^n, d$  ... dependent value measured,  $f_{\mathbf{w}}$  ... weight function,  $f_{\mathbf{w}} = f_{\mathbf{w}}(\mathbf{x}) \ge 0$ ,  $f_{\mathbf{a}}$  ... approximation (nonlinear) function  $f_{\mathbf{a}}^{(t)} = f_{\mathbf{a}}(\mathbf{x}^{(t)}, \mathbf{p}^{(t)}), \mathbf{p}$  ... vector of model parameters (coefficients), E ... approximation error.

A general solution to (2) and (3) is rather complex. Because of the nature of the data, the function  $f_{\rm a}$  will be proposed as nonlinear. Thus the first problem considered may be to minimize (2) with respect to a general approximation function  $f_{\rm a}$ . The next problem to be solved is the constraint (3) on the values of  $f_{\rm a}$ . A system defined in this way has infinitely many solutions due to the unknown character of the function  $f_{\rm a}$  and an optimum solution requires a suitable compromise to be chosen.

To estimate the parameters of the regression function, mostly the *least-squares method* is employed minimizing the expression :

$$\sum_{t=1}^{k} \left( d^{(t)} - f_{\rm a}^{(t)} \right)^2 \tag{4}$$

for all the values of the parameters  $\mathbf{p}$  determining the regression function, which corresponds to (2) for a unit weight function. By the dependence type of the coefficients  $\mathbf{p}$ , with respect to the regression function, regression models are divided into:

- linear models linear with respect to the model coefficients,
- nonlinear models nonlinear with respect to the model coefficients.

In many cases, determining the adequacy of a model may be rather tricky and controversial even if many applicable statistical criteria exist [12, 14].

# 3.1. Polynomial regression

A polynomial regression function is among the most frequently used regression models. Such a model is based on a regression function written as:

$$f_{\rm a} = \sum_{j=0}^{m} p_j \, x^j \, . \tag{5}$$

To determine the degree m of the polynomial, the following approaches based on statistical methods may be used [1, 8]:

- 1. a model with m = 0 is used as a starting one, in each step testing whether another degree needs to be added. If such a next term is found to be statistically insignificant, the algorithm stops,
- 2. a model with  $m = \max m$  is used as a starting one based on theoretical grounds or practical restrictions. Then we proceed as in the previous case but, this time, reducing m step by step,
- 3. finally, a penalizing function may be introduced providing a relative price to be paid for adding another parameter.



Fig.2: Example of a polynomial regression for data b17

For the data class given (see, for example, Table 1, Fig. 1) also this class of regression functions was tested for comparison. The result proved to be unsuitable, which can, for example, be seen in Fig. 2. Moreover, the chances of affecting the negative values of the regression model by (3) were relatively small.

#### 3.2. Radial basis regression

Based on the facts described in the previous paragraph, a regression model with different and 'more suitable' properties with respect to the physical and technical constraint of the problem in question (1.4) and to the expected shape of the regression was used. A Radial Basis Function (RBF) serves as the basis of this function class. The following (Gauss-curvebased) function is a typical representative of this class:

$$\varphi_j(\mathbf{x}, \mathbf{x}_j, s_j) = \exp\left[-\left(\frac{|\mathbf{x} - \mathbf{c}_j|}{s_j}\right)^2\right] , \qquad s_j > 0 .$$
(6)

Each RB function is generally determined by n parameters. The regression considered is formed by h linear combinations of RBF's  $\varphi_i$ :

$$f_{\mathbf{a}} = \sum_{j=1}^{h} w_j \varphi_j(\mathbf{x}, \mathbf{c}_j, s_j) \qquad \text{or} \qquad f_{\mathbf{a}} = \sum_{j=1}^{h} w_j \varphi_j(\mathbf{x}, \mathbf{c}_j, s_j) + w_0 \tag{7}$$

with  $w_j$ ,  $\mathbf{c}_j$  and  $s_j$  being unknown coefficients (parameters  $\mathbf{p}$ ).

Thus, in a general case, the searched for regression function has the following form:

$$f_{a} = \sum_{j=1}^{h} p_{j,1} \varphi_{j}(\mathbf{x}, \mathbf{p}_{j,2}) \quad \text{or} \quad f_{a} = \sum_{j=1}^{h} p_{j,1} \varphi_{j}(\mathbf{x}, \mathbf{p}_{j,2}) + p_{0,1} , \quad (8)$$

where  $\varphi_j \dots$  general RBF,  $p_{j,1} \dots$  RBF linear combination coefficients,  $\mathbf{p}_{j,2} \dots$  RBF parameters (the second index indicating parameter count),  $p_{0,1} \dots$  additional RBF coefficient.

In the regression function considered, a 'suitable behaviour' can be found, that is a property, with respect to the problem in question and the GA methods further used:

- each RB function may be designed as positive and bounded,
- 'control' of the model as defined by (3) can be implemented well because of its parameters **p** and the nature of (6) and (8),
- if conditions (12), (13) and (14) are met, a sufficient condition for the regression model being non-negative, that is, an approximation of (3) may be set as follows:

$$p_{j,1} \ge 0 (9)$$

 according to [3,4] an RBF model will be less sensitive to the coefficient than a polynomial model.

#### 3.3. RBF neural networks

As a model, also a new and, in many respects, very 'strong' artificial-neural-network type should be mentioned, *Radial Basis Function Network* – *RBFN*. This is a model of a forward neural network with a layer containing what is referred to as local units. With respect to the perceptrons, such units have a behaviour which is dual in a certain sense:

- perceptron divides the input space into two subspaces with markedly different output values,
- local units the relevant neuron output is localised in a neighbourhood of the point given by its parameters.

Models of such artificial neural networks were first discovered and investigated in the early 1980's. This new model was designed and used in (1988, 1989) by Broomhead and Low [2]. The basic results on RBF-network approximation come from Powell [13] and Micchelli [9].



Fig.3: RBF neural network, basic diagram

Next the development of the RBF networks was strongly influenced by the results of Moody and Darken as well as Poggio and Girosi [11]. A very well researched and comprehensive description can be found in [4, 15], and an algorithmization of the learning rules in [5].

RBF unit (RBF neuron), denoted by  $\varphi$ , has *n* real input values  $\mathbf{x} = (x_1, \ldots, x_n)$  and one real output value *y*. A parameter **c** is assigned to each of these input values. The output value is calculated by a *transfer function* applied to the inner potential of the unit, see below. The unit has a parameter *s*, which is the *spread* of the RBF unit. The inner potential of an *RBF unit*  $\xi$  is calculated as the distance of the input vector **x** from the centre of the RBF unit **c** using a Euclidean metric  $\varrho_{\rm E}$ .

$$\xi = \frac{\|\mathbf{x} - \mathbf{c}\|}{s} \ . \tag{10}$$

The output value y is obtained by applying the transfer function  $\varphi$  to the potential  $\xi$ .

$$y = \varphi(\xi) \ . \tag{11}$$

A whole range of transfer functions may be designed. Because of their specific properties, such functions are called *radial*. A radial function may be thought of as one determined by a significant point, a centre  $\mathbf{c}$ , assigning the same values to arguments equally distant from  $\mathbf{c}$ .



Fig.4: An RBF unit in 3D space

Their further properties include:

- locality (see above), e.g. for a function of one variable by:

$$\int_{-\infty}^{\infty} f(x) \doteq \int_{a}^{b} f(x) , \qquad (12)$$

with f(x) being a transfer function (RBF) and a, b being the boundary values of the interval, for which, with respect to the centre of the RB function, we already have:

$$\int_{-\infty}^{a} f(x) \, \mathrm{d}x \approx 0 \,, \qquad \int_{b}^{\infty} f(x) \, \mathrm{d}x \approx 0 \,, \tag{13}$$

This property is important because of the approximation properties of the RBF networks.

- functional properties of D(f) ... domain and H(f) ... — range, by:

$$D(f) \subseteq \Re$$
,  $H(f) \subseteq (0,1]$ . (14)

This property is advantageous because of the functional constraint of the approximation (3), which is required in the technical application in question.

The following is a typical RB function used in *RBF networks*:

$$\varphi_j(\mathbf{x}) = \exp\left[-\left(\frac{\|\mathbf{x} - \mathbf{c}_j\|}{s_j}\right)^2\right] , \quad \text{for} \quad s_j > 0 , \quad i = 1, 2, \dots, h .$$
 (15)

The input layer of an RBF network contains n units corresponding to the *n*-dimensional input values. To each connection between input value *i* and RBF unit *j* a weight  $c_{ij}$  (j = 1, ..., h) is assigned. Such a weight represents coordinate *i* of the centre  $\mathbf{c}_j$  of RBF unit *j*. The output value of RBF unit *j* is connected to the output layer by a synopsis with a weight  $w_{js}$ . The output units or neurons are linear and calculate a weighted sum of its input values. An RBF network performs two different transformations. The first one, being implemented by RBF units, is a linear transformation from  $\Re^n$  to  $\Re^h$ . The second transformation takes this space of hidden units to the output space  $\Re^m$ , is linear and implemented by the output linear neurons of the network, Fig. 3.

As a rule, the learning process of an RBF network consists of three stages. Each stage of the learning algorithm determines the values of another parameter group of the RBF network. The following is a summary of an RBF network algorithm :

> INPUT: training set of k patterns  $- \{ (\mathbf{x}^{(t)}, \mathbf{d}^{(t)}), t = 1, ..., k \}$ OUTPUT: three algorithm stages calculating: 1. positions of h centres – centres of RBF units  $\mathbf{c}_j$ 2. parameters RBF – spread s and other parameters if any 3. weights  $w_{js}$

**Stage 1:** Determines the positions h of the centres of RBF units, in figures 3,4 represented by the weights between the input values and the RBF layer, that is,  $\{c_{ij}; i = 1, ..., n; j = 1, ..., h\}$ . Basically, at this stage, the densities of the pattern occurrences are appro-

ximated. There may be several approaches: uniform distribution – regular arrangement of the h centres of the RBF units all over the input space, random patterns – h random training patterns are chosen and the centres of the RBF units are placed on their input parts, optimum patterns – the method [15] selects h training patterns serving as the centres of the RBF units, self-organization – k-means clustering [5] etc.

**Stage 2:** This is to set further parameters of the RBF units if they exist or are adapted. By (15), parameter s represents the spread of an RB function, that is, the size of the controlled area and influences the generalization abilities of an RBF network. Parameter s is mostly set in proportion to average Euclidean distances q of the nearest neighbours of a unit. **Stage 3:** Is referred to as *learning with a teacher*, which means that a training set  $\{(\mathbf{x}^{(t)}, \mathbf{d}^{(t)})\}$  is used to set the parameters. At this stage, the weights  $w_{js}$ , or  $w_j$  are adapted as the coefficients of a linear combination of RBF units. Optimum weights  $w_j$  are found

using an error function  $E(\mathbf{w})$ 

$$E(\mathbf{w}) = \frac{1}{2} \sum_{t=1}^{k} \left\| \mathbf{d}^{(t)} - \mathbf{y}^{(t)} \right\|$$
(16)

using a direct method

$$\mathbf{w} = \left(\mathbf{\Phi}^{\mathrm{T}} \,\mathbf{\Phi}\right)^{-1} \,\mathbf{\Phi}^{\mathrm{T}} \,\mathbf{d} \tag{17}$$

with  $\mathbf{\Phi} = (\varphi_{ij})_{i,j} = \varphi(\|\mathbf{x}_i - \mathbf{c}_j\|), i, j = 1, \dots, h$ , or using an iterative procedure described by the theory of linear neural networks, denoted by LMS or Widrow-Hoff algorithm [15]

$$\mathbf{w}(i+1) = \mathbf{w}(i) + lr \,\mathbf{e}(i) \,\mathbf{x}^{\mathrm{T}}(i) \,, \tag{18}$$

where lr – learning parameter, step;  $\mathbf{e}$  – RBF network error; i – iteration number.

#### 4. Design of a regression model and GA optimization

Based on the type of the actual data, and the resulting shape of the regression function, the number of the RBF units was set at h = 3. This also determines the maximum number of parameters **p** of the model (degrees of freedom) for the neural network in question, which, by (7) and (8) for an RBF network, is:

$$f_{\text{approx.RBFN}} = \sum_{j=1}^{h} p_{j,1} \varphi(\mathbf{x}, \mathbf{p}_{j,2}) + p_{0,1} = \sum_{j=1}^{h} w_j \varphi(\mathbf{x}, c_j, s) + w_0 .$$
(19)

The results and previous experience from implementing classical RBF neural networks were used to design a more 'vivid' RFB-based model. A change as compared with model (19) consists in introducing a variable basis spread parameter for each RBF unit and removing the shift parameter  $f_{\rm a}$ . The model parameterization corresponds to (20).

$$f_{\text{approx.RBFGA}} = \sum_{j=1}^{h} p_{j,1} \varphi(\mathbf{x}, \mathbf{p}_{j,2}) = \sum_{j=1}^{h} w_j \varphi(\mathbf{x}, c_j, s_j) .$$
(20)

Description	Denoted in graphs as	by $(19)$	number of parameters
RBF centre coordinates	$RB_{ ext{center}}$	$c_1,\ldots,c_3$	3
RBF spread	$RB_{\rm spread}$	s	1
RBF weight (size, amplitude)	$RB_{ m weight}$	$w_1,\ldots,w_3$	3
$f_{\rm a}$ shift along $y$ axis	$RB_{ m yshift}$	$w_0$	1

Tab.2: Description of the symbols used in the parameterized RBFN model

Description	Denoted in graphs as	by (20)	number of parameters
RBF centre coordinates	$RB_{center}$	$c_1,\ldots,c_3$	3
RBF spread	$RB_{\rm spread}$	$s_1, \ldots, s_3$	3
RBF weight (size, amplitude)	$RB_{weight}$	$w_1,\ldots,w_3$	3

Tab.3: Description of the symbols used in the parameterized RBFGA model

The GA optimization mechanism application was tested with 2 alternative settings:

- 1. With the  $RB_{center}$  parameter range within the domain of the data to be approximated. This solution proved to be rather unsatisfactory, Fig. 5. Even if the MSE values or the regression coefficients were better, this variant was rejected already by visual inspection.
- 2. With the  $RB_{center}$  parameter range within an extended interval.

This approach proved to be very efficient even when tested with another two subsettings 2A a 2B.

The basic characteristics and parameters of the GA employed were established in an expert way in view of the problem in question. A GATE toolbox in the Matlab environment was used for the solution. The average number of generations used to achieve the results presented was 200.

	GA data o	characteristic		GA operator characteristic
funName	faprox6	nParam	9	select(GA,'tournamentE',4)
funOpt	min	nBitParam	10	cross(GA, 'pcross', [0.8 3])
iType	[-]	mInit	random	<pre>mutation(GA,'bitmut',0.02)</pre>
mCode	GC	nIndi	100	<pre>mod(GA.nGener,20)==0</pre>
iParam by settings 2A, 2B			<pre>mutationHC(GA,'HC12',5,'rand',5,)</pre>	
2B	2A [-0.5 3];			<pre>elite(GA,'clone',3,'refine')</pre>
[-1.0 0.0; -0.5 0.0; 0.01 1.0; 0.0 3.0; 0.0 1.0; 0.01 4.0; 0.0 3.0; 1.0 3.0; 0.01 6.0]				





Fig.5: RBFGA data approximation for setting 1

The following is a simplified description of the goal function used with the domain of the parameters  $\mathbf{p}$  being expressed symbolically with respect to Table 4:

$$f_{\rm objSSE,RBFGA} = (\mathbf{d} - f_{\rm approx,RBFGA}(\mathbf{x}, \mathbf{p}))^2 + f_{\rm penalization} ,$$
  
min  $f_{\rm objSSE,RBFGA} = \min\{f_{\rm objSSE,RBFGA}(\mathbf{p}) | \mathbf{p} \in [\text{data set } 2A, 2B]\} .$  (21)

The coding of the bit string parameters (individuals) was done sequentially for each RBF unit as shown below.



Fig.6: System of the actual parameter bit string encoding and the meaning of the parameters for the RBF units

A common basis for setting 2 as described below was the normalization of the approximated data by a linear transformation  $T_{D\to U}$  on the universe  $U \in [0, 1]^2$  with the meaning of the *x*-axis and *y*-axis coordinates remaining the same. In this way, a uniform setting of the parameter intervals with respect to any data approximated could be achieved. An inverse transformation  $T_{U\to D}$  of the model parameters calculated was performed after the optimization algorithm was finished. The difference between settings 2A and 2B is as follows:

- 2A: a single interval  $\mathbf{p} \in [-0.5, 3]$  used of the optimized parameter values.
- 2B: a more logical interval structure is used related to the nature of the data, for each parameter p, a particular interval is determined by Table 4. In this case, the RBF unit position space is divided as shown by Fig. 7. Thanks to this method of encoding of the RBF unit centre intervals, the initial point penalization could be removed.



Fig.7: Positions of the RBF centre intervals in setting 2B

As compared with 2A, the setting 2B had a greater frequency during the tests made, which was due to the more precise specification of the RBF unit parameter intervals. On the other hand, setting 2A is designed more generally, that is, outside the technological data considered. The non-negativity test of the entire approximation function for setting 2B was simpler because of the non-negative  $RB_{weight}$  parameters consisting of only one condition (22). With this condition, the technical nature of the problem was of course taken into consideration, that is, the interval domain in question beginning at zero.

$$f_{\text{approx.RBFGA}}(0) \ge 0$$
 . (22)

In both settings (2A, 2B), due to the nature of the RBF, it was not necessary to include the non-negativity in the goal function (21) with the non-negativity check being made only after terminating the GA optimization process.

The following figures show the part of the space in which the  $(RB_{center})$  positions and other parameters  $(RB_{weight}, RB_{spread})$  of the units were searched for. The small circles stand for the approximated data transformed into the universe  $U \in [0, 1]^2$ .



Fig.8: The approximated data b17 (left) and a48 (right) in the normalized universe U and the RBFGA approximation found



Fig.9: RBFGA approximation of data b17, setting 2



Fig.10: RBFGA approximation of data a48, setting 2

#### 5. Conclusion

The graphs in Fig. 9, 10 show examples of the results obtained using the method described. Using Table 1, the reader can test the design and optimization of the regression model by an easy experiment comparing the results achieved. In [6] the (GA) optimization method used was compared with the standard three-stage learning method and with direct optimization of all the parameters by non-linear solvers (Gauss-Newton, Levenberg-Marquardt algorithm).

In this paper, a solution was presented to a non-linear regression problem on a given class of technological data. Determining the dependency of the deformation rate on the deformation degree is an important characteristic for the design of a technological process of material forming. For such a process, a constraint is explicitly given on the non-negativity of the model function value, which may complicate the model design in a considerable way. The solution presented provides a suitable non-linear model in the form of RBF functions and a method suitable for its optimization, in particular, using direct optimization of all the RBF network parameters rather than a standard three-stage learning method. The GA method given along with the model designed proved to be very efficient. Further use of the procedure design is of course not restricted to the particular data class used.

#### Acknowledgements

The access to the MetaCentrum supercomputing facilities provided under the research intent MSM6383917201 is highly appreciated. This work was supported by the Czech Ministry of Education in the frame of MSM 0021630529 'Intelligent Systems in Automation' and by the GACR No. 102/091668 'Control Algorithm Design by Means of Evolutionary Approach' and by the grant project GD103/09/H085 'Modern Composite Construction'.

## References

- [1] Anděl J.: Statistické metody (in Czech), MATFYZPRESS, Praha, 1993
- [2] Broomhead D.S., Lowe D.: Multivariable functional interpolation and adaptive networks, Complex Systems, 2:321–355, 1988
- [3] Himmelblau D.M., Process Analysis by Statistical Methods, John Wiley & Sons, New York, 1969
- [4] Light W.: Advances in Numerical Analysys, volume II Wavelets, Subdivision Algorithms and Radial Basisi Functions. Clarendon Press, Oxford, 1992, ISBN 0-19-853439-6
- [5] Looney C.G.: Pattern Recognition Using Neural Networks, Oxford University Press, New York, Oxford, 1997
- [6] Matoušek R.: Selected methods of Artificial Intelligence (in Czech), PhD thesis, Brno University of Technology, 2004, Brno
- [7] User's Guide: Fuzzy Logic Toolbox for Use with MatLab, The MathWorks, Inc., 1995
- [8] Meloun M., Militký J.: Kompendium statistického zpracování dat (in Czech), Academia, Praha, 2002, ISBN 80-200-1008-4
- [9] Micchelli C.A.: Interpolation of scattered data: distance matrices and conditionally positive definite functions, Construct. Approx., 2:11–22, 1986
- [10] Maroš B. Janíček L., Beneš M., Marošová M.: Katalog přetvárných odporů (in Czech), VUT FS Brno, 1997
- [11] Poggio T., Girosi F.: Networks for approximation and learning, In Proceedings of the IEEE, volume 78, 1990
- [12] Popela P., Dupačová J.: Melt Control: Charge Optimization via Stochastic Programming, W. Zieamba, S. W. Wallace (eds.): Applications of Stochastic Programming, Chapter 15, pp. 277–289, ISBN 0-89871-555-5, 2005, SIAM
- [13] Powell M.J.D.: The theory of radial basis function approximation, In W. Light, editor, Advances in Numerical Analysis, pages 105–210, Oxford Science Publications, 1992
- [14] Roupec J., Popela P.: Genetic Algorithms for Scenario Generation in Stochastic Programming: Motivation and General Framework, LNEE, book series: Advances in Computational Algorithms and Data Analysis, Vol. 14, Ao S.L., Rieger B., Chen S.S. (Eds.)., pp. 527–536, ISBN 978-1-4020-8918-3, 2008, Springer
- [15] Šíma J., Neruda R.: Teoretické otázky neuronových sítí (in Czech), MatFyzPress, UK Praha, 1996

Received in editor's office: March 1, 2010 Approved for publishing: January 5, 2011

*Note*: This paper is an extended version of the contribution presented at the national colloquium with international participation *Dynamics of Machines 2010* in Prague.