Strojniški vestnik - Journal of Mechanical Engineering 55(2009)1, 64-75
UDC 621.73
Paper received: 04.01.2008
Paper accepted: 09.02.2009

Application of Artificial Neural Network for Modeling the Flash Land Dimensions in the Forging Dies

Velibor Marinković*
Faculty of Mechanical Engineering, Niš, Serbia

In designing technological process and forging dies, the determination of the flash land dimensions represents one of the most important and most complex problems. Namely, it is on the flash land dimensions that the appropriate ("laminar") flow of metal and the complete filling of the die cavity largely depend on. The literature abounds with purely theoretical or semi-empirical formulae for calculating the flash land dimensions which are either too complex to be applied in engineering practice or insufficiently accurate. On the other hand, an important quantity of data as well as recommendations for selecting the flash land dimensions are available in the handbooks of metal forming, which do not take into consideration all the conditions and complexity of the forging process.

This paper proposes an approach to modeling the flash land by applying artificial neural network (ANN). A three-layer feed-forward ANN, with backpropagation algorithm for supervised learning is created. A sigmoid type of non-linearity is applied to neurons. In the reference literature there are many examples showing that the prediction model developed by means of ANN is more accurate than the one developed by the regression analysis. The trained ANN has shown a high level of prediction so that it can be used for designing and optimizing the conventional forging process.

Keywords: forging, forging dies, modeling, neural networks

0 INTRODUCTION

In a die forging processes, an excessive material flows over the flash land into the gutter (Fig. 1). In this way the flash of the forged part is formed which is subsequently eliminated in the special trimming tools [1] to [3].

The flash of the forged part, together with its slug (if there is one) represents a primary scrap in forging process [4].

With finished parts having a central opening, the slug of the forged part is necessary to separate the protruding parts of the upper and lower dies forming central cavities (most often double) in the forged part. An important part of the excessive material is in the flash of the forged part so that its mass should be reduced as much as possible.

In doing so, two facts should be respected. Firstly, the excessive material in the flash of the forged part is a consequence of inaccuracy while determining volume (mass) of the billet with respect to volume (mass) of the pure forged part as well as the tendency to do complete filling of the final die cavity. Secondly, the filling of the die cavity is ensured not earlier than the final phase of forging when the excessive material starts to flow over the flash land into the gutter.

Namely, during the material flow from the die cavity, the width of the forged part's flash constantly increases unlike its decreasing thickness. With respect to the given dimensions of the forged part's flash, the material flow resistances over the flash land increase. When the excessive material gets to the gutter, further resistance increase is obtained only by reducing the forged part's flash thickness (Fig. 2). This represents the final phase of forging process where the material (in accordance with the law of the least resistance) changes the direction of the flow.

Mostly radial material flow is replaced by mostly axial flow and in this way the remaining hollows in the die are filled.

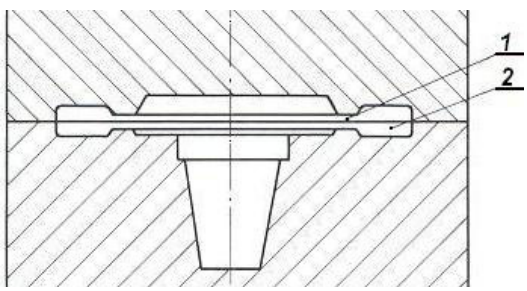
The final phase of forging process represents a critical moment of forging technology due to a sudden increase of load upon the forging dies. Accordingly, the quantity of the excessive material in the gutter can serve as a measure of quality of the designed technological procedure.

Generally speaking, with increasing b/h ratio, the resistance of the material flow over the flash also increases thereby ensuring the filling of

*Corr. Author's Address: Faculty of Mechanical Engineering, A. Medvedeva 14, Niš, Serbia, velmar@masfak.ni.ac.yu

the die cavity. This is especially important for forging parts of higher complexity and forged parts with high ribs. At the same time, however, the forging force rapidly increases and so does expenditure of deformation work (energy) [2]. At the same time, it is assumed that the gutter dimensions are so selected that there is no additional deformation of material in it.

This is especially important for forging parts of higher complexity and forged parts with high ribs. At the same time, however, the forging force rapidly increases and so does expenditure of deformation work (energy) [2]. On the other side, it is assumed that the gutter dimensions are so selected that there is no additional deformation of material in it.



1. Flash land; 2. Gutter

Fig. 1. *Forging die*

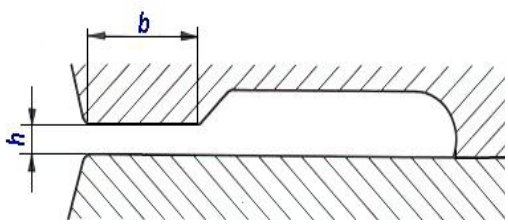


Fig. 2. *Dimensions of flash land of forging dies*

1 DETERMINING FLASH LAND DIMENSIONS

From all that has previously been said, it becomes quite clear how important for the forging process it is to select properly the flash land dimensions. If it is not possible to ensure, by the flash land form and dimensions, complete filling of the final die cavity, the technological process of forging must also comprise the operations of the preparatory forging by which such redistribution of the material is done that, at the finished forging process, not only the complete filling of the cavity is ensured but also the appropriate material flow, with no overlapping or other defects in the forged part.

For determining the flash land dimensions it is possible to find, in the reference literature, many formulae based on purely theoretical approach [1], [3] and [5] to [7].

However, more often applied are semi-empirical formulae based on the statistical treatment of the experimental results. Teterin [6] gives the best-known formula of this kind. This author has statistically analyzed the influence of the billet dimensions, of the dimensions of the mere forged part and a degree of complexity of the forged part on the dimensions of the flash land on a great number of forging dies used in practice. Using the regression analysis, with elimination of non-significant influential factors, Teterin has come to the following semi-empirical formula:

a) For forging on hammers,

$$h = -0.09 + 2\sqrt[3]{m_{\mathrm{fp}}} - 0.01 m_{\mathrm{fp}} \tag{1.a}$$

$$\frac{b}{h} = -0.02 + 0.0038\,\frac{D_0}{h} S + \frac{4.93}{m_{\mathrm{fp}}^{0.2}} \tag{1.b}$$

b) For forging on presses,

$$h = 2.17 + 1.39\; m_{\mathrm{fp}}^{0.2} \tag{2.a}$$

$$\frac{b}{h} = \text{-}1.985 + 0.00256\,\frac{D_0}{h} + \frac{5.258}{m_{\mathrm{fp}}^{0.1}} \tag{2.b}$$

where: m_{fp} (kg) mass of forged part (in the die), h (mm) height of flash land, b (mm) width of flash land, D_0 (mm) billet diameter, S (-) relative degree of complexity.

A relative degree of complexity is defined as a ratio between a degree of complexity of the forged part and a degree of complexity of the billet form [6]:

$$S = S_{\mathrm{fp}} / S_{\mathrm{b}} \tag{3}$$

where: S_{fp} degree of complexity of the forged part form, S_{b} degree of complexity of the billet form.

For rotational forged parts is $S_{\mathrm{b}} = 1$, so that $S = S_{\mathrm{fp}}$. The following relation defines degree of complexity of the forged part form [6]:

$$S_{\mathrm{fp}} = \frac{(O^2/A)_{\mathrm{fp}}}{(O^2/A)_{\mathrm{fp,c}}} \frac{2R_{\mathrm{t,fp}}}{R_{\mathrm{fp}}} \quad (4)$$

where: $(O^2/A)_{\mathrm{fp}}$ (mm) is ratio of volume squared and area of longitudinal section of the forged part, $(O^2/A)_{\mathrm{fp,c}}$ (mm) ratio of volume squared and area of longitudinal section of the cylinder around the forged part, $R_{\mathrm{t,fp}}$, R_{fp} (mm) abscissa of center of half the longitudinal section of the forged part and radius of the mere forged part, respectively.

From all said above it can be seen that the determination of a relative degree of freedom is a time-consuming and demanding work; that is why it is possible to use examples from the literature for approximate determination of its value which is permissible regarding its smaller impact upon the flash land dimensions.

2 ARTIFICIAL NEURAL NETWORKS

The appearance of artificial neural networks (ANNs) is related to the attempt to form an artificial system based on mathematical models which will be, by its structure, function and information (signal) processing, similar to biological nervous systems, and thus able to process "intelligently" information (signals), that is, simulate biological intelligence. The basic elements of the biological nervous systems are neurons. The model of artificial neuron is similar to some sorts of biological neurons. However, regarding the degree of complexity of some neurons (especially their number and organization), the models of neurons and ANN are far simpler than biological neurons and neural networks [8].

Contemporary ANNs are used for solving many complex (technical and the like) problems, especially physical processes and objects (systems) which are not sufficiently known or for which there are no theoretical solutions.

It should be stressed that ANN models are statistical ones, that is, they are not based on any physical theory.

The ANNs are capable of solving the problems that can hardly be solved by the traditional methods such as identification of forms and speech. The ANNs have been in use for a relatively long time for both identification and management of dynamic systems as well as solving of a wide spectrum of algebraic and optimization problems.

It is typical for the ANN to manipulate data. If the data are presented in a graphic or any other format, they must be translated into their numerical equivalents.

An important feature of the ANNs is that they have the capability of "learning" on examples as well as of generalizing problems after training. The ANNs are capable of incremental learning. In other words, the performances of an ANN can be improved by using an additional set of data that have become available subsequently.

On the other hand, the ANN, as, in essence, an advanced method of statistical analysis, exhibits a higher elasticity to disturbances in the input data.

The basic drawback of the ANNs lies in the use of processing of a relatively great number of data (samples) in order to provide sufficient prediction accuracy as well as the desired degree of generalization.

The application of the ANNs for solving concrete tasks is realized in the following phases:

1. Collection, analysis and preparation of data,
2. Designing ANN,
3. Training ANN,
4. Testing of the trained ANN,
5. Exploiting ANN.

2.1 Designing ANNs

In designing an ANN it is necessary to define:

- architecture,
- learning rule (algorithm),
- transfer neuron function.

2.1.1 Architecture of ANNs

Modern ANNs have a parallel-distributed architecture. They consist of a larger number of neurons distributed in several layers and inter-connected. Each ANN must have at least three layers:

- input layer,
- hidden layer,
- output layer.

The greatest number of ANNs has one hidden layer though there may be more of them (though rarely more than two).

Neurons of one layer are connected by specific connections (synapses) to neurons of

their neighboring layers. Between the neurons belonging to the same layer there are no connections.

Generally speaking, neurons are the basic processing units. The neurons of the current layer receive data from the neurons of the previous layer, process them and forward them to the neurons of the next layer.

However, it should be stressed that the neurons of the input layer do not process information (data); instead, they only forward them to the neurons of the next layer. The outputs from the neurons of the hidden layers are not known; that is how these layers came to be called after them.

The connections between particular neurons by the layers are characterized by weights (connection weights), which change during the ANN training.

The number of neurons in the input layer (usually several) and the output layer (often only one) are defined by the nature of the problem and the goal of the research, that is, by the number of the selected influential factors and the target functions (system responses).

The number of hidden layers and the number of neurons in each one of them are not defined in advance; instead, these numbers can change during the ANN training until the optimum architecture is defined, namely, the one that produces the best performances of ANN. The researchers usually rely on the derived examples and their own experience ("trial and error approach").

It can be assumed that a greater number of hidden layers together with a greater number of neurons in them will produce better prediction and generalization [9]. However, the modeling of many complex processes and systems shows that this is not always the case or improvements are negligibly small [10] to [14]. On the other hand, such a procedure causes a drastic increase of the needed samples (input-output data) for training ANN. If this is not provided for, it may happen that the number of connections between nodes (neurons) of the neural network and the weights that should be set is greater than the number of the available pairs of data for training ANN. Then ANN can likewise be trained though the case is mathematically indefinite [15].

When the size of the available sample is small compared with the number of weights of the ANN the resulting neural model is considered "overparametrized" [16]

The number of neurons in the hidden layers of ANNs with several outputs can be determined on the basis of simple relations:

a) ANN with one hidden layer,

$$N_{\text{tot}} \geq (N_{\text{i}} + N_0) \cdot N_{\text{h}} \tag{5a}$$

b) ANN with two hidden layers,

$$N_{\text{tot}} \geq (N_{\text{i}} + N_{\text{h2}}) \cdot N_{\text{h1}} + N_0 \cdot N_{\text{h2}} \tag{5b}$$

where: N_{tot} total number of series of input - output data, designed for training ANN (without biases), N_{i} number of input neurons, N_{h1}, N_{h2} number of neurons in hidden layers, N_0 number of neurons in the output layer.

Hecht - Neilson (in [17]) propose the following relation:

$$N_{\text{h}} \leq 2\,N_{\text{i}} + 1 \tag{6}$$

More details and explanations can be found in many books and papers related to ANN [8], [18] and [20] to [21].

The number of hidden neurons can be greater ("stretched" ANN) or smaller ("compressed" ANN) than the number of input neurons.

Each neuron in the input layer represents in ANN the value of independent variable while, analogously, each neuron in the output layer represents the value of the particular dependent variable. At the same time, the distinction should be made between the output data given by the trained network and the output data, which are presented to the ANN for training and which are called target or desired values. Otherwise, the output data are expressed in a numerical or quasi-logical format (1→ "yes"; 0→ "no").

The hidden layers enable ANN to solve complex nonlinear problems that makes them a more powerful tools comparing to perceptrons.

Viewed schematically, the ANN represents an oriented graph in which the nodes are processor units while the arrows on the lines point to the direction/sense of the signal (information) flow.

Regarding the way of connection ANNs are divided into:

- layered,
- fully connected,
- cellular.

A more detailed classification of ANNs can be found in [8], [17] to [19] and [21].

Regarding the direction of transfer and data processing (information, signals) ANNs are divided into two groups:

- direct (Feed - forward),
- indirect (Feedback - recurrent).

With the former information is transferred in only one direction, from the input to the output layer. Therefore, there is no feedback toward any of the processing elements. With the latter, information from the current layer returns to the previous layers of ANN.

Regarding the way of training ANNs are classified into three groups:

- ANN with supervised training,
- ANN with unsupervised training,
- ANN with partially supervised training.

In the first case, algorithm (supervisor) supervises and directs the process of ANN training. In the second case, ANN is free in processing and analyzing input data. The ANNs of the third group are different from the previous two since the already trained ANN is occasionally assessed.

The widest application in practice is of ANNs with supervised training where two types are differentiated, namely:

- ANN with backpropagation (BP),
- ANN with propagation in time.

2.1.2 Activation Functions

The neuron in the current layer of the ANN, as the basic processing unit, receives data (signals) from neurons from the previous layers that, "loaded" by multiplication with weight coefficients make, in sum, the neuron potential. When this potential exceeds a certain threshold, the artificial neuron produces a respective response (output signal) that is further forwarded to neurons of the subsequent layer.

The value of the signal at the output of the artificial neuron is obtained by applying an activation function for which the neuron potential is argument.

The following relations can be mathematically formulating all said above:

$$a_{\mathrm{i}} = \sum_{j=1}^{N_{\mathrm{x}}} w_{\mathrm{ij}} x_{\mathrm{j}} - \theta_{\mathrm{i}} \tag{7a}$$

$$y_{\mathrm{i}} = f(a_{\mathrm{i}}) \tag{7b}$$

where: a_{i} potential of i-th neuron, w_{ij} adjustable connection weights, between j-th neuron in the previous layer and i-th neuron in the current layer, x_{j} output from j-th neuron serving as input into i-th neuron, θ_{i} threshold of activation of i-th neuron, N_{x} total number of inputs into i-th neuron, y_{i} output from i-th neuron, f activation function (transfer function).

Adding scalar parameter-bias can expand argument of the activation function. Bias is not treated as an input parameter.

In defining the state of artificial neuron and activation function the threshold of activation and bias can be left out.

Activation functions are needed for the sake of introducing nonlinearity into the ANN that makes multilayer ANNs so powerful.

The most important activation functions are:[1]

- hard limit transfer function/symmetric hard limit transfer function,
- linear transfer function,
- log-sigmoid transfer function,
- tan-sigmoid transfer function.

The last three of the above-listed functions are continuous, non-falling and differentiable.

The choice of transfer function depends on the kind of problem that is being modeled while, in the case of normalizing input data, it also depends on the interval (range) of their normalization.

Most frequently, a linear function is applied at the output layer of the ANN and so are some of the sigmoid functions (s-shaped functions) in the hidden layers (because of their nonlinearity and differentiability).

For the most often used sigmoid activation functions (*logsig, tansig*) the following relations are valid [8]:

$$y_{\mathrm{i}} = \frac{1}{1+\exp(-\lambda\, a_{\mathrm{i}})} \tag{8a}$$

$$y_{\mathrm{i}} = \frac{2}{1+\exp(-\lambda\, a_{\mathrm{i}})} - 1 \tag{8b}$$

[1] In the software package MATLAB ("Neural Network Toolbox") these functions are denoted as: "hardlim/hardlims", "purelin", "logsig", "tansig".

where λ is a positive number (scaling factor) used to control of the slope of the semi-linear region.

A typical architecture of one three-layered ANN with backpropagation is shown in Fig. 3[2].

2.2 Training of ANNs

Every ANN is designed for modeling a concrete problem. For an ANN to react appropriately when unknown input data are added to it, it must be trained. ANN training is done on examples that make up a set of input-output data obtained by experiment or some other way.

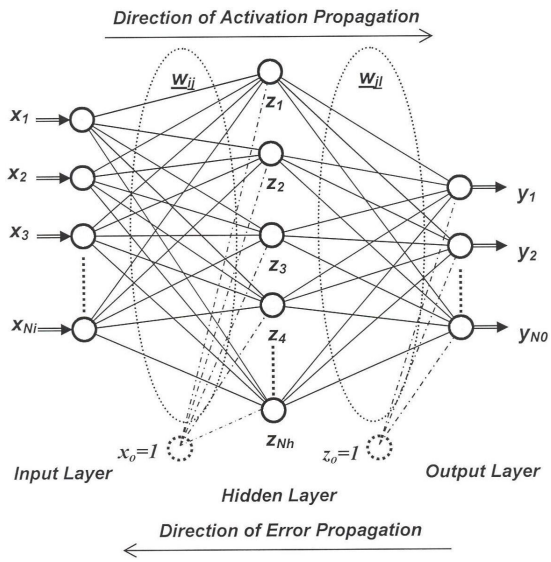


Fig. 3. *Three-layer feedforward BP ANN*

Just like a biological nervous system, ANN will behave more reliably and precisely in exploitation if the number of samples for its training is greater.

Training belongs to the most important parts of ANN designing and procedure.

The ANN training represents a process of adjusting weights of connections and bias that are adjoined to every synapsis between neurons on the basis of comparing the output values with the desired (target) ones for the same input ones. The process is repeated with incremental changes of the connection weights until the network becomes stabilized or the total error between the target and the output values is reduced to the given level (according to the selected criterion), that is, until ANN is able to simulate correctly the behavior of the studied process (system).

In using the commercial software the training process can be interrupted even when the desired error threshold is not achieved. Namely, one ANN is considered to have adequate performances if error of training and error of generalization do not exceed the acceptable levels [11] and [18].

With increasing number of iterations in the process of training ANN, the training error falls but, as a rule, the generalization error rises. Such an ANN is said to be "overfitting". That is why the error size must be selected in accordance with the target and nature of the studied problem. For overcoming the given problem various techniques are used (such as, for instance, early stopping approach [18]).

To solve the greatest number of practical problems direct ANNs with supervised training and backpropagation are used. The backpropagation algorithm is based on the method of gradient descent by which the error function is minimized.

There are a great number of algorithms for training ANN of which several already standard algorithms are listed here:

- algorithms with conjugated gradient,
- quasi-Newton algorithms,
- algorithms with backpropagation,
- method of linear search,
- Levenberg-Marquardt algorithm and others.

An essential difference between the above-listed algorithms is reflected in the quantity of the taken memory on the computer and the convergence rate.

It is difficult to claim beforehand which algorithm is the most suitable for modeling a certain process (system). For example, for research of the technological processes, the widest application is that of the Levenberg-Marquardt training algorithm.

As for error functions, the most frequent are two of them, namely:

- mean - squared error,
- sum - squared error.

Otherwise, the set of input-output data can be presented to ANN for training in two ways:

- by incremental method,
- by batch method.

[2] Feedforward multilayer network with sigmoid non-linearities is often termed multilayer perceptron (MLP).

For the standard gradient descent algorithm, the sum-squared error is determined from the relation:

$$E_{\mathrm{T}} = \sum_{k=1}^{N} E(k) = \sum_{k=1}^{N} \frac{1}{2} \sum_{j=1}^{N_0} \left[y_{\mathrm{j}}(k) - y_{\mathrm{j}}^{\mathrm{t}}(k) \right]^2 \qquad (9)$$

where: $E(k)$ sum squared error for k-th sample from a set of samples (input-output data) for training, N total number of samples for training, $y_{\mathrm{j}}(k)$ current output of j-th neuron for k-th sample, $y_{\mathrm{j}}^{\mathrm{t}}(k)$ desired (target) output j-th neuron for k-th sample.

In the general case for ANN with backpropagation, regardless of the training procedure, the following equation is valid (in vector notation) for adjusting connection weights [8] and [20]:

$$w(k+1) = w(k) - \eta \frac{\partial E}{\partial w} + \alpha \, \Delta w(k-1) \qquad (10)$$

where: $w(k)$ vector of connection weights in current (k-th) iteration, $w(k+1)$ vector of connection weights in subsequent iteration, $\Delta w(k-1)$ increment of vectors of connection weights in the previous operation, η learning rate, α momentum term.

Learning rate η and momentum term α ($(0 \le \eta \le 1)$, $(0 \le \alpha \le 1)$) are constant [9], [20] and [22]. Parameter η directly affects the rate of changing connection weights. Member with parameter α in the previous relation is not applied in all the training procedures. Otherwise, parameter α determines the effect of the previous weight changes on the current weight change. This parameter is used to level high-frequency variations of the error surface in the connection weight space, thus reducing oscillations around minimum error. In this way the convergence of the ANN training process convergence is quickened.

It follows from the above-said that the ANN training comprises three phases:

- initialization,
- forward processing;
- backward processing.

The training should be preceded by ANN initialization that implies the selection of: structure of network, transfer function, training algorithm and parameters, initial matrix of weights and bias. The initial values of weights and bias are chosen randomly and with them the training algorithm begins.

2.3 Testing ANNs

The trained ANN should be tested in order to assess its ability to predict and generalization on the basis of the acquired "knowledge" on the selected set of input-output data (examples).

For testing ANN, a set of input-output data is used otherwise not used for the network training. If the testing shows that error does not exceed the desired threshold, the ANN can be considered ready for exploitation.

It should be noted that ANN is not suitable for extrapolation so that the use of input data which are outside the space covered by a set of input data for the ANN training will produce a non-real prediction.

Some authors apply, between the sequences of training and testing, the procedure of validation.

Validation is a procedure for testing the trained ANN to correctly generalize information (data). For this procedure the ANN is presented with a set of data that is not used for training the network. As has already been said, poor generalization points to "overfitting" which requires a change of the network parameters. However, many authors do the adjustment of structure and/or parameters of ANN during the training since validation reduces the number of samples for the network training.

2.4 Exploitation of ANNs

The trained and tested ANN can be used for modeling and prediction when it is presented with new (original) input data.

As has already been said, during the exploitation, the ANN performances can be improved if adjoined by a set of data that was not available during the training.

2.5 Data Preparation

ANN performances can, to a considerable degree, depend on quality and quantity of the collected data.

Data can be collected in various ways, namely, from literature, or by experimental research, or from the current production.

In any case (especially, if the data are taken from various sources), it is necessary to do an analysis as well as data preparation for training ANN.

As each process (system) is affected by a multitude of factors, it is necessary to reduce their number only to significant ones [18]. Namely, by increasing the number of influential factors (input neurons), the size of the set of input-output data, necessary for successful ANN training, also increases. Unifying influential factors can sometimes successfully solve this task. It is implied that, among the selected influential factors, no correlation is allowed.

If individual value of some parameter essentially deviates from its mean value, such data should be removed from the set of input-output data designed for ANN training. With relatively small deviations mean values should be operated with.

If an extensive set of data is available, a representative sample can be selected for ANN training. This strategy is very useful when the dimension of the input space is quite large [16].

In the cases when the interval of variations of input-output variables is very different (in order to make balance of significance for each input factor during the training process), it is recommended to carry out the procedure of normalization[3], before their presentation to the ANN that should be trained.

When it comes to linear normalization of a set of data, the general relation is valid:

$$x_{\mathrm{N}} = (x_{\mathrm{N}\max} - x_{\mathrm{N}\min}) \frac{(x - x_{\min})}{(x_{\max} - x_{\min})} + x_{\mathrm{N}\min} \qquad (11)$$

where: x real value of variable, $x_{\min}$, $x_{\max}$ minimum and maximum value of real variable, respectively, $x_{\mathrm{N}\min}$, $x_{\mathrm{N}\max}$ minimum and maximum value of normalized variable, respectively.

For normalized value of variables, various constants are chosen, most often:
a) $x_{\mathrm{N}\max} = 1$; $x_{\mathrm{N}\min} = -1$ *(for function "tansig")*,
b) $x_{\mathrm{N}\max} = 1$; $x_{\mathrm{N}\min} = 0$ *(for function "logsig")*.

Selection of data for training that is, testing of ANN, should be done by random method (for instance, by using the program for generating random numbers).

The total set of available data (N) should be distributed into two subsets: for training ANN ($N_1 \approx 2/3\ N$), for testing ANN ($N_2 \approx 1/3\ N$).

3 MODEL ANN

Data, which Teterin [6] has collected and used for analyzing the factors affecting the dimensions of the flash land in the forging dies, can be used for designing ANN (Table 1).

A detailed technological analysis has shown that the number of influential factors can be reduced.

In a concrete case, by uniting and eliminating insignificant influential factors, their number is reduced from 5 to 3.

Likewise, one series of input-output data can be excluded from the set of available data for training and testing ANN since its range essentially differs from the usual one in practice.

In view of all that has been said before about ANN, for this concrete task the optimum architecture was selected as a three-layer ANN (3-5-2), with three neurons (independent variables) ($x_1 = D_{\mathrm{fp}}/H_{\mathrm{fp}}$; $x_2 = m_{\mathrm{fp}}$; $x_3 = SD_0/h$) in the input layer, five neurons in the hidden layer and two neurons (dependent variables) ($y_1 = h$; $y_2 = b/h$) in the output layer. Such an ANN architecture was "extorted" regarding the size of the sample and relations (5a) and (6). Truly, in the literature there are the cases of application of unnecessarily large and complex ANNs in terms of their structure [15], [22] to [24]. In the above quoted papers, as a consequence, the ANNs are trained on insufficiently large samples. Though, in all this, the ANNs trained in this way still have satisfactory performances, such an approach cannot be recommended for analyses of complex nonlinear problems.

The selected ANN is with forward data processing and back propagation. As an ANN training algorithm, the selected one is the Levenberg - Marquardt algorithm ("trainlm"), which is considered as the fastest for the ANNs of moderate size.

For the given algorithm it is common to select, for neurons of the hidden layers, a sigmoid transfer function.

[3] In the software package MATLAB ("Neural Network Toolbox") normalization before training and denormalization after training can be done by using the functions: "Premnmx", "Postmnmx", "Tramnmx".

Table 1. *Empirical data for flash land dimensions from literature*

n_0	D_{fp}	H_{fp}	(D_{fp}/H_{fp})	m_{fp}	S	$S \cdot D_0/h$	h	b	(b/h)
1	174.8	51.0	(3.427)	6.7	2.32	90.0	4	15.45	(3.8625)
2	**379.5**	**60.0**	**(6.325)**	**23.5**	**2.0**	**138.0**	**5**	**14.3**	**(2.86)**
3	**515.2**	**110.0**	**(4.686)**	**109.1**	**2.1**	**94.5**	**10**	**23.7**	**(2.37)**
4	265.4	56.0	(4.739)	10.3	3.09	176.34	4	15.45	(3.8625)
5	243.7	80.0	(3.046)	16.0	3.37	148.1	5	16.3	(3.26)
6	283.6	70.0	(4.051)	22.0	2.5	132.0	5	16.3	(3.26)
7	420.9	56.0	(7.516)	33.0	3.76	231.5	6	20.2	(3.3667)
8	661.2	130.0	(5.086)	192.1	4.5	330.0	8	21.9	(2.7375)
9	503.8	153.0	(3.293)	117.0	4.23	235.4	8	21.9	(2.7375)
10	370.7	87.0	(4.261)	43.0	2.92	119.0	8	21.9	(2.7375)
11	**686.3**	**142.0**	**(4.833)**	**195.1**	**5.0**	**239.4**	**12**	**25.35**	**(2.1125)**
12	778.4	150.0	(5.189)	379.7	3.21	155.2	12	24.0	(2.0)
13	176.8	56.0	(3.157)	4.0	1.3	65.2	3	12.6	(4.2)
14	108.0	43.0	(2.512)	1.6	1.53	68.4	2	9.74	(4.87)
15	**120.5**	**81.0**	**(1.487)**	**5.3**	**0.89**	**23.2**	**3**	**10.6**	**(3.5333)**
16	**123.2**	**50.0**	**(2.464)**	**2.7**	**1.4**	**51.3**	**3**	**12.6**	**(4.2)**
17	131.0	119.0	(1.101)	9.0	0.94	11.2	8	26.9	(3.3625)
18	124.0	41.5	(2.988)	3.2	1.38	50.9	3	12.6	(4.2)
19	212.4	144.0	(1.475)	13.5	1.36	52.5	4	12.0	(3.0)
20	171.0	37.0	(4.622)	4.9	1.36	71.4	3	12.0	(4.0)
21	**441.4**	**156.0**	**(2.829)**	**64.0**	**3.3**	**153.5**	**8**	**22.0**	**(2.75)**
22	450.0	106.0	(4.245)	84.0	4.0	213.6	8	22.0	(2.75)
23	372.0	91.0	(4.088)	49.5	3.21	168.2	6	17.7	(2.95)
24	287.2	64.0	(4.487)	21.4	2.79	139.4	5	16.3	(3.26)
25	**344.1**	**54.0**	**(6.372)**	**19.0**	**4.56**	**286.1**	**5**	**19.3**	**(3.86)**
26	241.1	66.0	(3.653)	17.0	2.72	116.3	5	16.3	(3.26)
27	122.2	54.0	(2.263)	2.1	1.4	67.9	2	9.7	(4.85)
28	114.7	27.0	(4.248)	0.9	1.59	88.9	1.5	9.9	(6.6)
29	116.7	44.0	(2.652)	3.0	1.85	61.1	3	12.6	(4.2)
30	102.3	49.0	(2.088)	1.87	0.93	42.7	2	9.7	(4.85)
31	109.0	50.0	(2.180)	2.1	2.22	88.2	2	9.7	(4.85)
32	**624.9**	**196.5**	**(3.180)**	**264.9**	**2.35**	**146.2**	**8**	**16.9**	**(2.1125)**
33	655.2	290.0	(2.259)	449.9	1.88	67.74	10	18.7	(1.87)
34	297.3	134.0	(2.219)	42.2	7.92	484.9	4	19.44	(4.86)

*Note: Data given in the brackets are not given in the original Table [6]. By random method, 8 series of input-out data are chosen (**2,3,11,15,16,21,25,32**) for ANN testing while the other data are used for its training. Experiment numbered as **17** is not taken into consideration during the ANN training and testing (explanation in the text).*
By randomization the researcher's subjective influence upon the experiment results is eliminated. For this purpose, the following approach is applied [10]. After a certain number of trainings of the ANN with different random dividing of data base (with the other parameters remaining the same), the case is chosen with the most suitable dividing of the data for the ANN training and testing. The re-distribution of data base into a subset for ANN training and a subset for ANN testing, as given in this Table, has guaranteed the best network performances.

In the concrete case, for the transfer function of the hidden layer, hyperbolic tangent ("tansig") is selected and so is the linear transfer function ("purelin") for neurons of the output layer.

Further, the given value of the mean squared error ("mse") that should be achieved by the ANN training represents a criterion for optimization of the network synapses weights (Fig. 4).

Table 2. *Comparative results of calculations by different methods*

n_0	Empirical data [6]		Regression Method [6]		Neural Network	
	h	b	h_1	b_1	h_2	b_2
1	4	15.45	3.61	14.98	4.22	15.06
2	5	14.3	5.40	15.88	5.15	17.85
3	10	23.7	8.38	23.16	9.59	18.49
4	4	15.45	4.16	15.17	4.37	15.55
5	5	16.3	4.79	17.13	4.96	16.43
6	5	16.3	5.29	15.94	4.94	16.39
7	6	20.2	5.99	20.17	5.17	20.22
8	8	21.9	9.53	24.08	7.85	21.94
9	8	21.9	8.52	22.62	8.28	21.91
10	8	21.9	6.49	22.44	8.07	21.88
11	12	25.35	9.56	31.86	10.05	20.94
12	12	24.0	10.60	25.44	12.12	23.89
13	3	12.6	3.05	12.04	3.00	12.76
14	2	9.74	2.23	9.55	1.85	9.79
15	3	10.6	3.34	10.92	3.17	11.21
16	3	12.6	2.67	12.78	2.45	10.30
17	8	26.9	3.98	25.97	-	-
18	3	12.6	2.83	12.43	3.12	12.49
19	4	12.0	4.54	12.60	3.72	12.19
20	3	12.0	3.26	11.67	2.78	12.40
21	8	22.0	7.27	22.07	8.93	17.41
22	8	22.0	7.83	23.00	7.85	21.94
23	6	17.7	6.76	17.52	5.62	17.72
24	5	16.3	5.25	16.17	4.51	15.60
25	5	19.3	5.06	19.29	5.18	20.82
26	5	16.3	4.88	16.34	5.05	16.59
27	2	9.7	2.45	9.08	2.11	9.78
28	1.5	9.9	1.83	8.03	1.43	9.90
29	3	12.6	2.76	12.63	3.10	12.08
30	2	9.7	2.36	9.08	2.24	9.62
31	2	9.7	2.45	9.23	1.85	9.74
32	8	16.9	10.11	17.58	10.79	20.82
33	10	18.7	10.74	17.40	9.93	18.86
34	4	19.44	6.45	17.24	5.21	19.45

A subset of data for training, as well as a subset of data for testing is presented to the ANN by the batch method.

The data obtained after training and testing of the ANN are given in sum in Table 2. For comparison, in the same Table, the target values of the given parameters are also given.

A much clearer insight into errors while calculating the given parameters by different methods is obtained in graphic version (Fig. 5).

The comparison of the two methods can also be done by means of statistic parameters, for instance, by correlation coefficient (r) and standard deviation (σ), that is, its estimate (S). For the problem in question, the following results are obtained:

- for regression model,
  ($r_h = 0.946$, $S_h = 2.67$; $r_b = 0.974$, $S_b = 5.61$)
- for ANN model,
  ($r_h = 0.971$, $S_h = 2.98$; $r_b = 0.924$, $S_b = 4.20$).

4 CONCLUSIONS

This paper has shown that the ANN can be a very suitable tool for empirical modeling of the observed data.

ANN designed for generating geometric parameters of the flash land of the forging die possesses satisfying properties of prediction and generalization.

It is common for ANN that the accuracy of the data obtained at the output of the network during testing is slightly less than that of the data obtained at the end of the training of the same network.

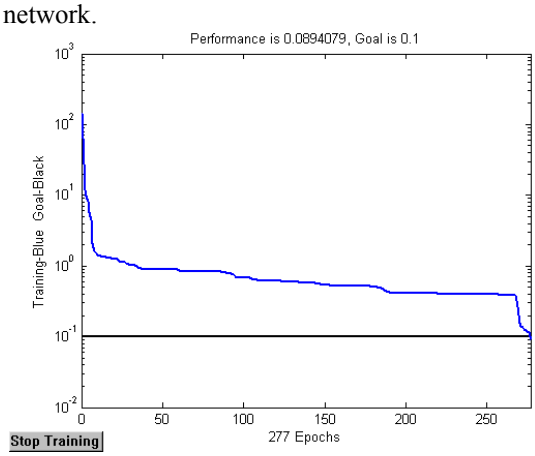


Fig. 4. *The mean sum of squared error during the training phase versus the number of epochs*

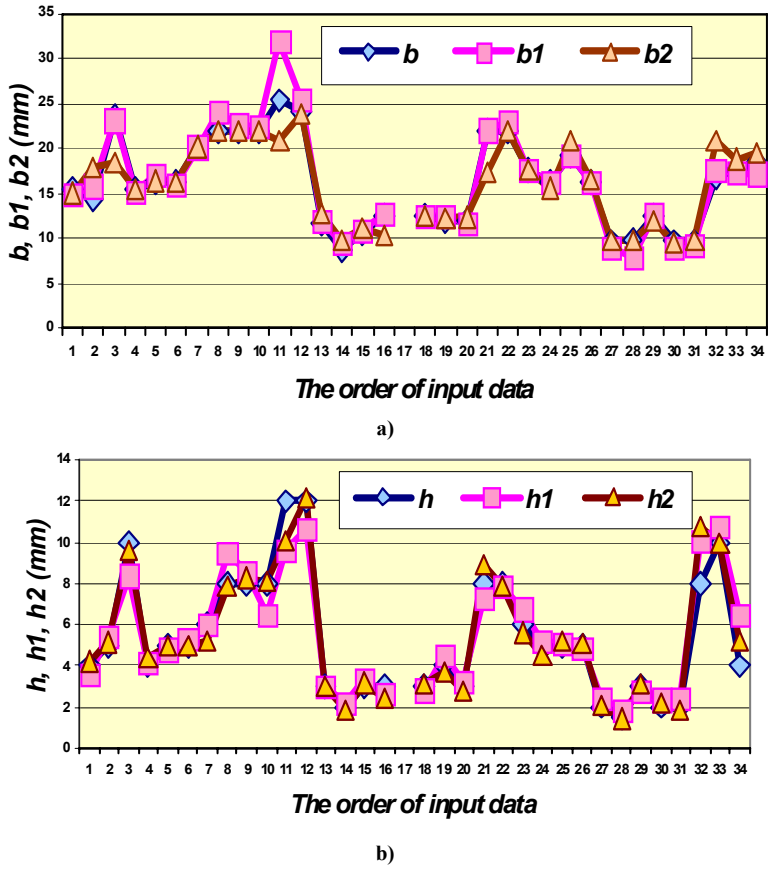


Fig. 5. *Comparative results: a) for width of flash land, b) for height of flash land*

In the light of the results given above, it is fair to conclude that the ANN model appears to be at least as good as the regression one. However, it should also be noted that the ANN model, in the analyzed example, provides for the choice of the geometric parameters of the flash land on the basis of three important technological factors (D_{fp}/H_{fp}, m_{fp}, SD_0/h), while the regression model comprises the impact of only one factor (m_{fp}, for h, that is, two factors (m_{fp}, SD_0/h, for b). The practical advantage of the ANN in this respect is obvious.

For a smaller number of inputs and a larger set of available data for training, the advantages of the ANN with respect to the regression model would undoubtedly come to the fore.

This is especially valid for the cases when a set of input data is not subject to normal distribution.

For this purpose it may be better to use a separate ANN for each parameter of the flash land. Further research is being carried out to investigate this approach further.

Finally, it should be noted that the influential factors are dispersed in wide intervals (m_{fp} = 0.9 to 450, $SD_0/h \cong$ 23 to 485, $D_{fp}/H_{fp} \cong$ 1.5 to 7.5) thereby comprising practically all the cases that may be met in manufacturing.

The results presented in this paper show the potential of the ANN as an empirical modeling tool that can be used in the fields of design, analysis and optimization of the forging process and forging dies.

5 REFERENCES

[1] Spur, G., Stöferle, Th. (1984) Handbook of Production Engineering, Vol. 2/2 - Forming, *Carl Hanser Verlag,* München/ Wien (in German).

[2] Kalpakjian, S., Schmid, S. (2006) Manufacturing, Engineering and

Technology, Fifth Edition, *Pearson Education*, New York.

[3] Atroshenko, A.P. et al. (1986) Forging and Stamping, vol. 2 - Warm Bulk Stamping, *"Mashinostroenie",* Moscow (in Russian).

[4] Marinković, V. (2004) Degree of utilization in the metal forming processes, *IMK-14. Research and development* 10 (1-2) , p. 33 - 41 (in Serbian).

[5] Marinković, V. (2005) On the problem of determining the excessive material in open die forging, *IMK-14. Research and development* 10 (1-2) , p. 87 - 96 (in Serbian).

[6] Teterin, G. P., Polukhin, P. I. (1979) Basics of Optimization and Automatization of Design for Warm Bulk Stamping Processes, *"Mashinostroenie"* Moskow (in Russian).

[7] Tomov, B., Radev, R., Gagov, V. (2004) Influence of flash design upon process parameters of hot die forging, *Journal of Materials Processing Technology* 157-158, p. 620-623.

[8] Bose, N.K., Liang P. (1996) Neural Network Fundamentals With Graphs, Alogorithms, and Applications, *Mc Graw -Hill*, New York.

[9] Szecsi, T. (1999) Cutting force modeling using artificial neural networks, *Journal of Materials Processing Technology* 92-93, p. 344-349.

[10] Malinov, S., Sha, W. (2003) Software products for modelling and simulation in materials science, *Computational Materials Science* 28, p. 179-198.

[11] Mani, V., Omkar, S.N. (2002) Understanding weld modelling processes using a combination of trained neural networks, *International Journal of Production Research* 40 (3), p. 547-559.

[12] Calcaterra, S., Campana, G., Tomesani, L. (2000) Predicrion of machanical properties in spheroidal cast iron by naural networks, *Journal of Materials Processing Technology* 104, p. 74-80.

[13] Marinković, V. (2007) Determination of steel formability for warm forming by applying artificial naural network. *7th International Conference RaDMI 2007*, Belgrade, p. 217-224.

[14] El-Kassass, E.M.A., Mackie, R.I., El-Sheikh, A.I. (2001) Using neural networks in cold-formed steel design, *Computers and Structures* 79, p. 1687-1696.

[15] Sha, W. (2004) Comment on "Modeling of tribological properties of alumina fiber reinforced zinc-aluminum composites using artificial neural network" by K.Genel et al. *Materials Science and Engineeering A* 372, p. 334-335.

[16] Ingrassia, S., Morlini, I. (2005) Neural network modeling for small datasets, *Technometrics* 47, p. 297-311.

[17] Swingler, K. (1996) Applying Neural Networks, A practical Guide, *Academic Press*, London.

[18] Dreyfus, G. (2005) Neural Networks, *Springer Verlag*, Berlin/Heidelberg.

[19] Miljković, Z. (2003) Systems of atificial neural networks in production technologies, *Faculty of Mechanical Engineering*, Belgrade (in Serbian).

[20] Inamdar, M.V., Date, P.P., Desai, U.B. (2000) Studies on the prediction of springback in air vee bending of metallic sheets using an artificial neural network, *Journal of Materials Processing Technology* 108, p. 45-54.

[21] Haykin, S. (1998) Neural Networks: Comprehensive Foundation, *Prentice Hall,* New York.

[22] Altinkok, N., Koker, R. (2004) Neural network approach to prediction of bending strength and hardening behaviour of particulate reinforced (Al-Si-Mg)-aluminium matrix composites, *Materials and Design* 25, p. 595-602.

[23] Liu, J., Chang, H., Hsu, T.Y., Ruan, X. (2000) Prediction of the flow stress of high-speed steel during hot deformation using a BP artificial neural network, *Journal of Materials Processing Technology* 103, p. 200-205.

[24] Patrik, M.W., Rho, H.M., Park, B.T. (1996) Generation of Modified Cutting Condition Using Neural Network for an Operation Plannung System*, Annals of the CIRP* 45(1), p. 475-478.