National Conference with International Participation
ENGINEERING MECHANICS 2008
Svratka, Czech Republic, May 12 – 15, 2008

SOFTCOMPUTING METHODS IN NONLINEAR MODEL PARAMETERS IDENTIFICATION: A REVIEW

A. Kučerová* **and M. Lepš***

Summary: *The problem of inverse analysis occurs in many engineering tasks and, as such, attains several different forms and can be solved by a variety of very distinct methods. In this contribution, we present an overview of two basic philosophies of the inverse analysis aimed, in particular at parameters estimation with utilization of soft-computing methods.*

1 Introduction

A variety of engineering tasks nowadays lead to an inverse analysis problem. Generally, the aim of an inverse analysis is to rediscover unknown inputs from the known outputs. In common engineering applications, a goal is to determine the initial conditions and properties from physical experiments or, equivalently, to find a set of parameters for a numerical model describing the experiment.

When new numerical model is developed, the identification process is necessary for validating of proposed model to fit the experimental data. This process become a challenge especially in cases of complex nonlinear numerical models applied to simulate an experiment on structures undergoing the heterogeneous stress field such as three-point bending test, tensile test (in case of presence of localized failure) or nano-indentation.

Once the numerical model is validated, another use of identification method is on demand when new values of model parameters should be found to fit experimental measurements on new material. Such identification process is supposed to be performed repeatedly for any new measurement and therefore, the emphasis is in this case put on the efficiency of chosen identification method.

The numerical model able to correctly simulate the experiment together with a robust and effective identification method are essential tools for a structural modelling and reliability assessment. A description of a complex methodology for statistical and reliability analysis of concrete structures using nonlinear mechanical models and artificial intelligence based identifi-

*Ing. Anna Kučerová, Ph.D. and Ing. Matěj Lepš, Ph.D., Department of Mechanics, Faculty of Civil Engineering, Czech Technical University in Prague, Thákurova 7, 166 29 Prague 6, tel: (+420)-2-2435-4606, fax: (+420)-2-2431-077, e-mail: anicka@cml.fsv.cvut.cz, leps@cml.sv.cvut.cz

cation tools is presented in (Novák et al., 2007) and one particular application to fiber-reinforced concrete facade panels is presented in (Keršner et al., 2007).

In overall, there are two main philosophies to solution of identification problems. A *forward* (classical) mode/direction is based on the definition of an error function of the difference between outputs of the model and experimental measurements. A solution comes with the minimum of this function. This mode of identification could be considered as more general and robust and therefore, it is usually applied in numerical model validation.

The second philosophy, an *inverse* mode, assumes the existence of an inverse relationship between outputs and inputs. If such relationship is established, then the retrieval of desired inputs is a matter of seconds and could be easily executed repeatedly.

2 Preliminary definitions

More formally, the problem of an inverse analysis can be formulated based on the existence of an experiment E, which, physically or virtually, connects the known inputs (parameters) $\mathbf{x}^E$ to the desired outputs (measurements) $\mathbf{y}^E$. Formally, this can be written as

$$\mathbf{y}^E = E(\mathbf{x}^E). \tag{1}$$

Then, the problem of an inverse analysis is defined as a search for unknown inputs $\mathbf{x}^E$ from the known outputs $\mathbf{y}^E$, i.e. inversely to the experiment E. In common engineering applications, the experiment E is usually simulated by some virtual model M. Often, the model is a program based on numerical methods such as the finite element method. Such a model M usually does not describe a real experiment E exactly, but in our work it is considered as a "good" approximation and therefore we can write

$$M \approx E; \tag{2}$$

$$\mathbf{y}^M = M(\mathbf{x}^M). \tag{3}$$

This step is important from the economy point of view, where the cost of the evaluation of the model M is assumed to be by an order of magnitude smaller than the cost of the physical experiment E.

Input parameters $\mathbf{x}^M$ of a theoretical model should not necessarily correspond to physical parameters $\mathbf{x}^E$. Phenomenological models use often some parameters without physical interpretation. Usually, an experimentalist does not know the physical parameters accurately. Let us also note, that the number of theoretical model input parameter is usually smaller that ten, i.e.

$$\left\|\mathbf{x}^M\right\| < 10. \tag{4}$$

Theoretical models are usually constructed to describe some real experiment in order to obtain equivalent outputs (measurements). Therefore the *output parameters* $\mathbf{y}^M$ of a theoretical model usually correspond to that one from the experiment $\mathbf{y}^E$. The identification process should be completed by the validation based on comparing modelled outputs $\mathbf{y}^M$ with the experimental ones $\mathbf{y}^E$ in order to judge, whether the model parameters were found correctly and accurately enough. To comment the number of output parameters let us note that the measurements could vary in time τ, could be performed in a number of measuring points P and could be stored for different experiments, considering different boundary conditions C. The outputs are usually measured with respect to time in discrete points T and including different measuring points P

and different experiments C. Therefore, the number of outputs could be quite large and could reach tens or hundreds components, i.e.

$$\left\|\mathbf{y}^M\right\| = \left\|\mathbf{y}^E\right\| = T \times P \times C \approx 100. \tag{5}$$

It could be interesting to introduce here following two basic definitions in accordance with (Babuska and Oden, 2004):

> *Verification*: The process of determining if a computational model obtained by discretizing a mathematical model of a physical event and the code implementing the computational model can be used to represent the mathematical model of the event with sufficient accuracy.
>
> *Validation*: The process of determining if a mathematical model of a physical event represents the actual physical event with sufficient accuracy.

The authors in (Babuska and Oden, 2004) accepted that philosophically *absolute* validation and verification may be impossible, but validation and verification relative to a specific series of tests and preset tolerances may be perfectly legitimate as a basis for making decisions.

In the field of model parameters identification, the meaning of verification and validation is a little bit different and could be stated as follows:

> *Verification*: The process of determining whether the identification method is able to refind the model parameters $\mathbf{x}^M$ from the outputs $\mathbf{y}^{ref}$ of the reference simulation done for any choice of original inputs $\mathbf{x}^{ref}$.
>
> *Validation*: The process of determining whether the identification method is able to find the model parameters $\mathbf{x}^M$ corresponding to the experimental outputs $\mathbf{y}^E$.

In general, there could be two steps of identification method verification:

1. *Verification I*: comparing the reference model inputs $\mathbf{x}^{ref}$ with the identified ones $\mathbf{x}^M$;
2. *Verification II*: comparing the reference model outputs $\mathbf{y}^{ref}$ with the identified ones $\mathbf{y}^M$

and one step of validation: comparing the experimental outputs $\mathbf{y}^E$ with the identified ones $\mathbf{y}^M$.[1]

In engineering practice, nevertheless, the model parameters identification often takes part in the process of mathematical model verification and validation. In these cases it is quite difficult to judge whether the errors are caused by the incorrectness of the mathematical model or by the incorrectness of identification procedure. We propose the following order of verification and validation:

1. *code verification*, a province of software engineering;
2. *solution verification* is needed, which involves a posteriori error estimation;
3. *identification method verification I + II*, once the computational model is verified, the identification method should be theoretically able to refind parameter's values corresponding to reference simulations exactly;

[1] Recall, that physical experimental inputs $\mathbf{x}^E$ are practically always unknown.

4. *model validation*, comparing the outputs from the model simulation with the experimental outputs. Such a step typically involves a certain identification procedure used to fit the experimental data. To suppress this aspect, the optimization procedure needs to be extremely robust and therefore very computationally expensive;

5. *identification method validation*, we consider this method to be computationally efficient and not producing significant additional errors when the outputs $\mathbf{y}^{ref}$ from reference simulation are replaced with experimental ones $\mathbf{y}^{E}$ in comparison with the identified outputs $\mathbf{y}^{M}$.

This work focuses on identification methods which are verified and validated together with models proposed by other authors. All the employed models are a priory supposed to be verified and validated.

Other source of the error are, unfortunately, almost always the experimental data itself. The problem of estimating, controlling, and quantifying experimental error goes also together with model validation and identification procedure validation. In practice, the engineers usually work with modelling some measurements affected by noise. There are many regularization procedures to overcome the noise in measured data during identification procedure, such as the Tikhonov regularization etc. Some examples of regularization based techniques applied in parameters identification could be found e.g. in (Iacono et al., 2006; Mahnken and Stein, 1996) or (Maier et al., 2006).

3 Forward mode of an inverse analysis

Based on the above-mentioned statements, the *forward* (classical) mode/direction of an inverse analysis is defined as a minimization of an error function $F(\mathbf{x})$ defined as the difference between the outputs of the model $\mathbf{y}^{M}$ and the output of the experiment $\mathbf{y}^{E}$, i.e.

$$\min F(\mathbf{x}) = \min \|\mathbf{y}^{E} - M(\mathbf{x})\|. \tag{6}$$

A solution $\mathbf{x}^{M}$ comes with the minimum of this function and if $F(\mathbf{x}^{M}) > 0$, the remaining error is caused by inaccuracy of a model or by some noise in measured data.

The problem (6) has been classically solved by *gradient-based optimization methods*. Nowadays, the model M is usually hidden in a program which is limited by license conditions, compact code etc. and therefore, the knowledge of derivatives is missing even if the function is differentiable. Hence, the soft-computing methods can be successfully applied here. Methods in the spirit of *the simulated annealing method* (Ingber, 1993; Vidal, 1993) with one solution in time or *evolutionary algorithms* (Goldberg, 1989; Michalewicz, 1999) with a 'population' of solutions are usually used.

The main advantage of this approach is that the forward mode is general in all possible aspects and is able to find an appropriate solution if such exists. This statement is confirmed with special cases like

a) A problem of a same value of outputs $\mathbf{y}$ for different inputs $\mathbf{x}$, i.e. existence of several global optima. This case leads to a multi-modal optimization (Mahfoud, 1995b) but is solvable by an appropriate modification of an optimization algorithm (Hrstka and Kučerová, 2004).

b) There are different outputs $\mathbf{y}$ for one input $\mathbf{x}$. This is the case of stochastic and probabilistic calculations as well as experiments polluted with a noise or an experimental error. This obstacle can be tackled e.g. by introduction of stochastic parameters for outputs or by a regularization of the objective function, see e.g. (Iacono et al., 2006; Mahnken and Stein, 1996) or (Maier et al., 2006).

c) There is more than one experiment for one material. This task can be handled as a multi-objective optimization problem, see e.g. (Coello, 2004, 2000; Miettinen, 1999).

One disadvantage of the forward mode, following the definition, is a fact that the computationally expensive search should be repeated for any change in data, e.g. even for small change in an experimental setup. This feature handicaps the forward mode from an automatic and frequent usage. The opposite is true for the second mode of an inverse analysis presented later.

The other disadvantage of the forward mode is the need for a huge number of error function evaluations. This problem can be managed by two approaches, which are based on:

a) parallel decomposition and parallel implementation;

b) computationally inexpensive approximation or interpolation method.

The parallel decomposition is based on an idea of the so-called implicit parallelism, i.e. the independence of any two solutions $\mathbf{x}$. This can be utilized by a global parallel model (Cantú-Paz, 2001), where the main (master, root) processor/computer controls the optimization process while the slave processors compute the expensive evaluations of the model M. Thanks to the independency of solutions, nearly linear speed-up can be reached until a high number of processors.

The second methodology is based on reducing the number of simulations of a complex model M. A similar idea used already in Equation (3) is employed here in two different possible implementations: meta-modelling (or so called surrogate modelling) of a computational model or meta-modelling of an error function, described in following two sections. The most often tools applied here could be categorized into three groups described in Sections 3.3 and 3.4.

3.1 Meta-model of computation model

One possibility to reduce the number of simulations of a complex mechanical model M is to estimate a model $\tilde{M}$ similar to the model M, i.e.

$$\tilde{M} \approx M \tag{7}$$

whereas $\tilde{M}$ should be computationally much cheaper than M. Then the optimization process could be started using the cheap model $\tilde{M}$ instead of M. Moreover, since the approximative model $\tilde{M}$ is determined, it could be used again when identifying parameters corresponding to new measurements. A scheme of such form of forward identification is shown in Figure 1 and the consecutive steps of this methodology are described below.

Step 1 *Approximative model $\tilde{M}$ estimation*: This step is computationally very demanding. Usually an artificial neural network is applied and trained to approximate original computational model. For neural network training, a certain number of simulations by original model are needed, appropriate topology of neural network should be determined

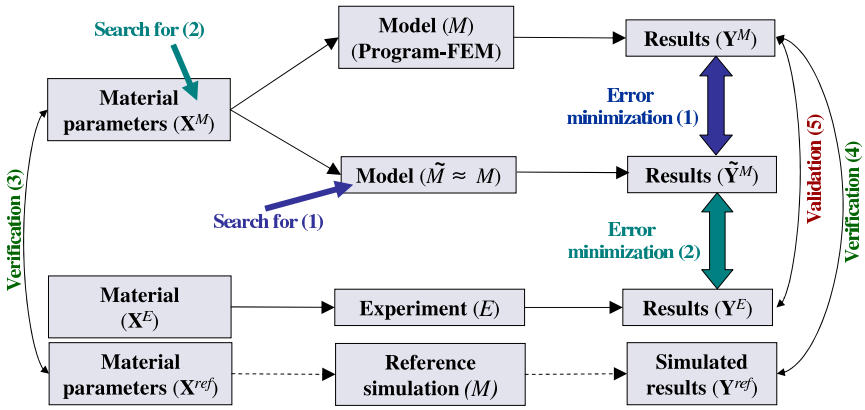

Figure 1: Forward mode of identification process using model approximation

and training process must be performed in order to minimize the error between response of model $\tilde{M}$ and the original model M.

It is worth mentioning that the outputs $\mathbf{y}$ often represent some diagram or curve (load-deflection diagram etc.) which could be defined as a vector of discrete points coordinates with tens to hundreds of components. The model inputs $\mathbf{x}$ usually represents just several model parameters. Therefore the approximative model $\tilde{M}$ should describe a mapping from several inputs to tens or hundreds outputs, what could be quite complicated task for e.g. a artificial neural network.

Step 2 *Optimization of model* $\tilde{M}$ for experimental or reference simulated data. For given outputs either from experiment $\mathbf{y}^E$ or from reference simulation $\mathbf{y}^{ref}$, an optimization process is started in order to find corresponding inputs $\mathbf{x}^M$ into the model $\tilde{M}$.

Step 3 *Verification I* consists of execution of optimization process in Step 2 for some reference couple of data $[\mathbf{x}^{ref}, \mathbf{y}^{ref}]$. Then the identified inputs $\mathbf{x}^M$ should be compared with the original input data $\mathbf{x}^{ref}$ as the first step of identification procedure verification.

Step 4 *Verification II*: For identified input data $\mathbf{x}^M$ a simulation by computational model should be performed and the outputs $\mathbf{y}^M$ should be then compared with the reference outputs $\mathbf{x}^{ref}$ as the second step of verification.

Step 5 *Validation*: consists again of execution of optimization process in Step 2, but here for experimental data $\mathbf{y}^E$. Then the identified inputs $\mathbf{x}^M$ should be used for a simulation by computational model M and the obtained outputs $\mathbf{y}^M$ should be compared with the experimental outputs $\mathbf{y}^E$.

To conclude this variant of forward approach implementation, some of its typical features are listed bellow:

i) the largest inconvenience is the complicated estimation of an approximative model $\tilde{M}$, especially considering the complexity of mapping from several inputs to tens or hundreds of outputs;

ii) the biggest advantage is the establishment of the approximative model $\tilde{M}$, that could be used for parameter identification for any new measurements;

iii) the optimization process necessary for parameter estimation should be started again for any new measurements.

3.2 Meta-model of error function

As it was already mentioned at the beginning of this chapter, the forward approach leads to an optimization process, where some error function is defined as

$$F = \left\| \mathbf{y}^E - M(\mathbf{x}) \right\|. \tag{8}$$

In other words the error is the difference between the outputs from experiment $\mathbf{y}^E$ and the outputs $\mathbf{y}^M$ from a model M. The second possibility to reduce the number of simulations of a complex mechanical model M is to estimate an approximative error function $\tilde{F}$ similar to the error function F, i.e.

$$\tilde{F} \approx F. \tag{9}$$

It is again assumed that $\tilde{F}$ is cheaper to evaluate than F, since its evaluation will not include an expensive simulation by model M. A scheme of such kind of forward approach implementation is shown in Figure 2. The consecutive steps of this implementation are almost the same

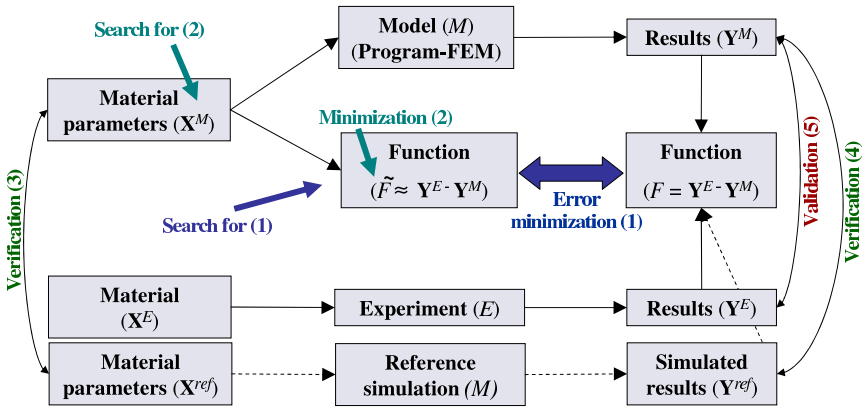

Figure 2: Forward mode of identification process using error function approximation

as in the previous case, excepts several details, which are mentioned bellow:

i) Step 1 consist of determination of an approximative function $\tilde{F}$ of the error function F. This could be done in the same way as a determination of an approximative model $\tilde{M}$. The difference is in the mapping, since the inputs remains the same, but the number of outputs here decreases usually to one value of error function. In particular cases, several objectives could be included while defining the error function, hence, several criteria could lead to multi-objective formulation of the error function. Finally, the determination of approximative error function $\tilde{F}$ is much more easier than determination of approximative model $\tilde{M}$;

ii) the biggest disadvantage is that this formulation usually leads to a multi-modal optimization problem, especially in cases, where several criteria are accumulated in a single-objective function using weighting approaches;

iii) other disadvantage is that the approximative error function needs to be established again for any new measurements, nevertheless, some expensive simulations by computation model M once performed could be used again for the determination of a new approximation .

Since the difference between the usage of meta-model $\tilde{M}$ of a computational model M and meta-model $\tilde{F}$ of an error function F is only in details, the terms meta-model $\tilde{M}$ and model M will cover both cases in following sections for the sake of simplicity.

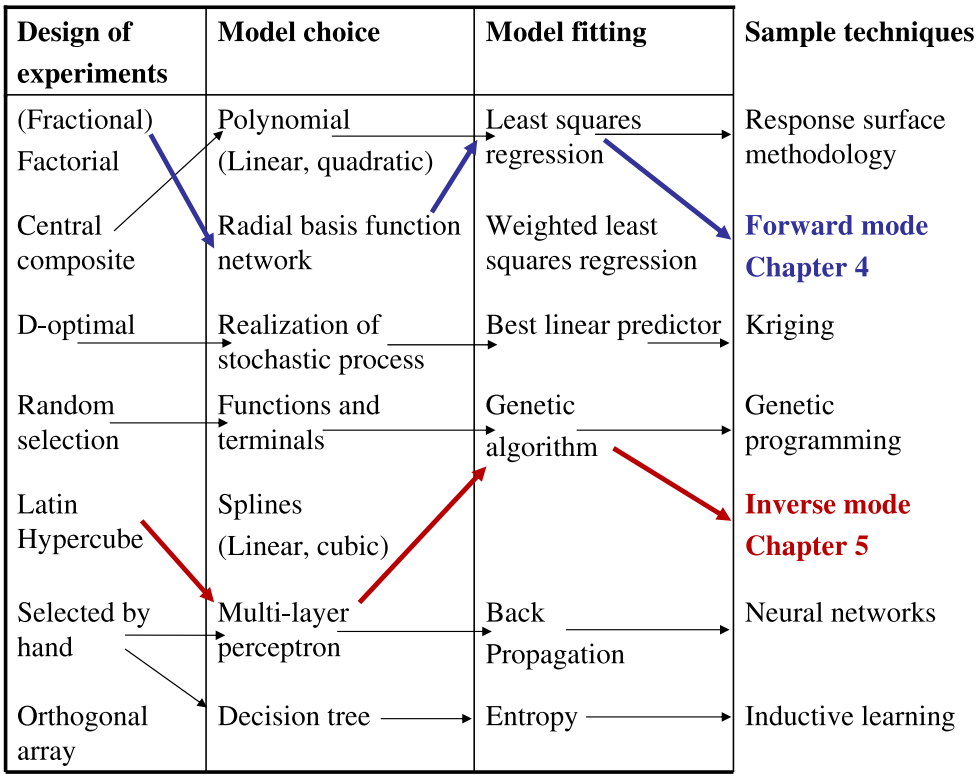

Table 1: Review of some meta-model techniques

Some meta-model techniques, which could be found in literature, are listed in Table 1. A brief description of some meta-model tools are described in following sections. More details with some examples and applications of meta-modelling can be found in (Jin, 2003), (Simpson et al., 2001) or (Queipo et al., 2005). Some comments about design of experiments, which should precede any meta-model establishment, are gathered in Section 4.3.

3.3 Interpolation tools

The first group of tools, let us call it *interpolation tools*, are used to interpolate the model M using sampled design points carefully chosen by some type of design of experiments, where the values of model $\tilde{M}$ are equal to values of model M, i.e. $\tilde{\mathbf{y}}^M(\tilde{\mathbf{x}}^M) \equiv \mathbf{y}^M(\mathbf{x}^M)$. The advantage here is, that in general, near any design point the interpolation is supposed to be more precise than some general approximation. Therefore, some iterative techniques are usually applied in order to add more design points in the area where the global optimum is supposed to be located.

Other typical feature of interpolation methods listed bellow is the fact, that the interpolation is established without any knowledge of an inner structure of the model M.

i) *Kriging* is named after the pioneering work of D. G. Krige[2], and was formally developed by Matheron (Matheron, 1963). More recent publication with the theoretical details could be found in Jin (2003). Some engineering applications of Kriging modelling are presented e.g. in (Varcol and Emmerich, 2005). The Kriging method in its basic formulation estimates the value of a function (response of a model) at some unsampled location as the sum of two components: the polynomial model and a systematic departure representing low (large scale) and high frequency (small scale) variation components, respectively.

   Hence, these models (Ordinary Kriging) suggest estimating deterministic functions as

$$f_p(\mathbf{x}) = \mu(\mathbf{x}) + \varepsilon(\mathbf{x}), \tag{10}$$

   where $f_p(\mathbf{x})$ is the unknown function of interest, $\mu(\mathbf{x})$ is a known polynomial function and $\varepsilon(\mathbf{x})$ is the realization of a normally distributed Gaussian random process with mean zero, variance σ^2 and non-zero covariance.

ii) *Radial Basis Function Network* (RBFN) have been developed for the interpolation of scattered multivariate data. The method uses linear combinations of radially symmetric functions based on the Euclidean distance or other metric, to approximate given function (or response of a given model). More details about this model are written in Section 4.1. Some engineering applications could be also found in (Nakayama et al., 2004) and (Karakasis and Giannakoglou, 2004).

iii) *Genetic programming* could be possibly used as an interpolation tool, if the equality of the meta-model $\tilde{M}$ and the computational model M in the design points is imposed. The theory of genetic programming could be found in (Koza, 1992); an application in parameters identification is published in (Toropov and Yoshida, 2005).

3.4 Approximation tools

The approximation tools includes, in general, also the interpolation tools. Nevertheless, we distinguish these groups of tools since for approximation tools there is no implicit condition, that the value of meta-model should be equal to the value of the original model in all design points as it is defined for interpolation tools. The optimum of the meta-model will have with high probability different value from the original model. Moreover, it is not clear how to include this discrepancy into identification procedure (contrary to the interpolation approach).

The approximation tools could be divided into two groups according to the knowledge about the original model M utilized during the choice of meta-model $\tilde{M}$:

a) *High and low fidelity models* assumes that, for a physical model M, there is a less accurate physical model $\tilde{M}$, which is computationally less expensive than the model M. This situation occurs in cases where, for one physical phenomenon, there are two or more describing theories, e.g. wave vs. particle theories. More often, there are cases, where different topologies, geometries, a different number of finite elements, a simple or a difficult model, a 2D or a spatial model etc. for a studied problem can be used. Some engineering applications of this method could be found in (González et al., 2004) or (Wang et al., 2002).

[2] a South African mining engineer

b) Meta-models determined without any insight into the physical model applies approximation tools like:

   i) *Response surface methods* (RSM) is described differently by different authors. Myers and Montgomery (Myers and Montgomery, 1995) state that RSM "is a collection of statistical and mathematical techniques useful for developing, improving, and optimizing process. It also has important application in the design, development, and formulation of new products, as well as in the improvement of existing product designs". The 'collection of statistical and mathematical techniques' of which these authors speak refers to the design of experiments (Section 4.3), least squares regression analysis, response surface model building and model exploitation.

   Response surfaces are typically second-order polynomial models; therefore, they have limited capability to model accurately nonlinear functions of arbitrary shape. Obviously, higher-order response surfaces can be used to model a nonlinear design space; however, instabilities may arise or too much sample points will be necessary in order to estimate all of the coefficients in the polynomial equation, particularly in high dimensions. Hence, many researchers advocate the use of a sequential response surface modelling approach using move limits or a trust region approach. An application of RSM in engineering design is presented in (Lee and Hajela, 2001) and an application in parameter identification is published in (Toropov and Yoshida, 2005).

   ii) *Multi-layer perceptron* (MLP) is a variant of artificial neural network. It is composed of neurons (single-unit perceptrons) which are multiple linear regression models with a nonlinear (typically sigmoidal) transformation on their output. These neurons are in this case organized in several layer, where each neuron is connected with all neurons in previous and following layer. More details about this procedure could be found in Section 4.1. An application of forward identification using MLP is presented in (Pichler et al., 2003).

   iii) Also the methods included in previous section could be used as approximative tools, but some modifications to their typical implementations are needed.

One of the possible ways to solve inconsistency among models and their meta-models can be a multi-objective formulation. For instance, (Quagliarella, 2003) uses an error between model and meta-model and error between meta-model and experiments as a two independent objectives.

Some combinations of forward and inverse mode of an inverse analysis are also possible, one example is published e.g. in (Most et al., 2007).

4 Inverse mode of an inverse analysis

The second philosophy, an inverse mode, assumes an existence of an inverse relationship between outputs and inputs, i.e. there is an inverse model M^{INV} associated to the model M, which fulfils the following equation:

$$\mathbf{x} = M^{INV}(\mathbf{y}) \tag{11}$$

for all possible $\mathbf{y}$. Generally, this inverse model does not need to exist. Nevertheless, we assume that the inverse model can be found sufficiently precise on some closed subset of the definition

domain. Next, we will limit our attention to an approximation of the inverse relationship, not its exact description. A quality of this approximation is easy to measure since a pair $\mathbf{x}$, $\mathbf{y}$ obtained using Equation (11) should also fulfill the Equation (3). Final usage of this methodology is trivial because a desired value $\mathbf{x}^M$ can be obtained by simple insertion $\mathbf{y}^E$ into Equation (11).

The main advantage is clear. If an inverse relationship is established, then the retrieval of desired inputs is a matter of seconds even if executed repeatedly. This can be utilized for frequent identification of one model. On the contrary, the main disadvantage is an exhausting search for the inverse relationship.

Further obstacles are the existence problems for the whole search domain and inability to solve the problem of a same value of outputs $\mathbf{y}$ for different inputs $\mathbf{x}$, i.e. existence of several global optima.

The case of different outputs $\mathbf{y}$ corresponding to one input $\mathbf{x}$ introduced by stochastic and probability calculations or by experiments polluted with a noise or an experimental error can be tackled e.g. by introduction of stochastic parameters for outputs (Lehký and Novák, 2005; Fairbairn et al., 2000).

Another case, when there is more than one experiment for one material, can be handled by sequential, cascade or iterative processes. As a solution, different approximation tools are applied. Nowadays, artificial neural networks have became the most frequently used methods.

Since the inverse mode is based on an approximation of the inverse model, the other problem concern the accuracy of inverse model predictions. It could be solved in following ways:

i) Taking into account an *expert guess*. This brings the possibility to reduce the inputs domain when preparing design points for the inverse model development. That leads to better accuracy of the inverse model in the vicinity of the expert guess and probably also near the desired inputs $\mathbf{x}^M$ corresponding to measurements $\mathbf{y}^E$. Nevertheless this approach suppose the existence of a competent expert with wide experience with the model as well as the experiment. This approach is published e.g. in (Novák and Lehký, 2006).

ii) *Cascade neural networks* suppose the possibility to identify the individual inputs x_i in a sequential way, where the predictions of some inputs identified in the first step could be used as known during the development of inverse model in next steps in order to reduce the complexity of the approximated relationship. Particular applications of this methodology to parameters identification are presented e.g. in Waszczyszyn and Ziemianski (2005) or in Kučerová et al. (2007).

iii) *Sequential refining* consists also of several sequential steps as the previous case. Nevertheless, in this case in all steps all inputs $\mathbf{x}$ remains to be identified. The predictions from previous steps are used only to reduce the design space and the inverse model is determined using new design points from narrower design space around the supposed solution. An application of such procedure to the parameters identification is published e.g. in Most et al. (2007).

All these methodologies lead to better accuracy of inverse model predictions. Nevertheless, the inverse relation becomes accurate only near the inputs $\mathbf{x}^M$ corresponding to the particular measurements $\mathbf{y}^E$. Therefore such inverse model M^{INV} could not be applied again for new measurements. Hence, the principal advantage of the inverse approach is more or less lost.

4.1 Artificial neural networks

Artificial neural networks (ANN) are powerful computational systems consisting of many simple processing elements connected together to perform tasks analogously to biological brains. There are two types of neural networks, which are successfully applied as model (or function) approximators.

The feedforward neural network

This network consist of one input layer, one output layer, and one or more hidden layers of processing units. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes and to the output nodes. There are no cycles or loops in the network, no feed-back connection. Mostly used example is a *multi-layer perceptron* (MLP) with a sigmoid transfer function and gradient descent method of training called Back-Propagation Learning Algorithm.

The *universal approximation theorem* can be stated as:

Let $\varphi(\cdot)$ be a non-constant, bounded, and monotone-increasing continuous function. Then for any continuous function $f(\mathbf{y})$ with $\mathbf{y} = \{y_i \in [0,1] : i = 1,\ldots,I\}$ and $\varepsilon > 0$, there exists an integer J and real constants $\{\alpha_j, b_j, w_{jk} : j = 1,\ldots,J, k = 1,\ldots,I\}$ such that

$$F(y_1,\ldots,y_I) = \sum_{j=1}^{J} \alpha_j \varphi \left(\sum_{k=1}^{I} w_{jk} y_k - b_j \right) \tag{12}$$

is an approximate realization of $f(\cdot)$, that is

$$\|F(y_1,\ldots,y_I) - f(y_1,\ldots,y_I)\| < \varepsilon \tag{13}$$

for all $\mathbf{x}$ that lie in the input space.

Clearly this applies to an multi-layer perceptron with J hidden units, since $\varphi(\cdot)$ can be a sigmoid, w_{jk}, b_j can be hidden layer weights and biases, and α_j can be output weights. It follows that, given enough hidden units, ***a two layer multi-layer perceptron can approximate any continuous function***.

Applications to computational mechanics could be found in (Yagawa and Okuda, 1996) or in more recent papers (Novák and Lehký, 2006) and (Waszczyszyn and Ziemiański, 2006).

Radial basis function network (RBFN)

RBFN are powerful techniques for interpolation in multidimensional space. A radial basis function (RBF) is a function which has built into a distance criterion with respect to a center. RBFN have two layers of processing: In the first, input is mapped onto each RBF in the "hidden" layer. The RBF chosen is usually a Gaussian. In regression problems the output layer is then a linear combination of hidden layer values representing mean predicted output.

RBF networks have the advantage of not suffering from local minima in the same way as multi-layer perceptrons. This is because the only parameters that are adjusted in the learning process are the linear mapping from hidden layer to output layer. Linearity ensures that the error surface is quadratic and therefore has a single easily localizable minimum. Solution to the regression problem can be found in one matrix operation.

As with the corresponding proofs for MLPs, these are existence proofs which rely on the availability of an arbitrarily large number of hidden units (i.e. basis functions). However, they do provide a theoretical foundation on which practical applications can be based with confidence.

RBF networks have the disadvantage of requiring good coverage of the input space by radial basis functions. RBF centers are determined with reference to the distribution of the input data, but without reference to the prediction task. As a result, representational resources may be wasted on areas of the input space that are irrelevant to the learning task. A common solution is to associate each data point with its own center, although this can make the linear system to be solved in the final layer rather large, and requires shrinkage techniques to avoid overfitting.

Some applications to engineering problems could be found in Nakayama et al. (2004) or Karakasis and Giannakoglou (2004).

4.2 Artificial neural network training

There are numerous tradeoffs between learning algorithms. Almost any algorithm will work well with the correct hyperparameters for training on a particular fixed dataset. However selecting and tuning an algorithm for training on unseen data requires a significant amount of experimental investigations.

There are three major learning paradigms, each corresponding to a particular abstract learning task. These are supervised learning, unsupervised learning and reinforcement learning. Usually any given type of network architecture can be employed in any of those tasks. We will focus here on the supervised learning which is used for training feedforward neural networks or radial basis function networks usually applied in an inverse analysis.

In the supervised learning, given a set of example pairs $(\mathbf{x}, \mathbf{y}), \mathbf{x} \in \mathbf{X}, \mathbf{y} \in \mathbf{Y}$, the goal is to find a model M^{INV} in the allowed class of functions that matches the examples. In other words, to infer the mapping implied by the data; the cost function is related to the mismatch between our mapping and the data and it implicitly contains prior knowledge of the problem domain.

In all but the simplest cases, however, the direct computation of the weights is intractable. Instead, we usually start with random initial weights and adjust them in small steps until the required outputs are produced.

A commonly used cost function $E(w_{ij})$ is the mean-squared error which tries to minimize the average error between all the network's output units, $M^{INV}(\mathbf{y})_j$ and all the target values x_j over all the example pairs p, i.e.

$$E(w_{ij}) = \frac{1}{2} \sum_p \sum_j \left(M^{INV}(\mathbf{y})_j - x_j \right)^2, \tag{14}$$

where i coincide with the number of neurons in the last hidden layer adjacent to the output layer. The minimization of this cost function using the gradient descent for the feedforward neural network leads to the well-known backpropagation algorithm. Many other minimization algorithms could be applied. A variety of methods based on mathematical programming are implemented in Matlab Neural Network Toolbox including the backpropagation algorithm, conjugate gradient algorithms, quasi-Newton algorithms, Levenberg-Marquardt algorithm and line search routines. Other interesting algorithms for ANN training are evolutionary algorithms, which have the ability to deal with the multi-modality of cost function appearing in feedforward neural networks.

There are two important aspects of the network's operation to consider:

Learning The network must learn relation between inputs and outputs from a set of training pairs so that these training pairs are fitted correctly.

Generalization After training, the network must also be able to generalize, i.e. correctly fit test pairs it has never seen before.

Usually we want our neural networks to learn well, and also to generalize well. If an ANN is not trained well even on training data, it is called as *under-fitting* or *under-learning* of ANN. The red line in Figure 3 is an example of such case. Sometimes, the training data may contain errors (e.g. noise in the experimental determination of the input values, or incorrect classifications). In this case, learning the training data perfectly may make the generalization worse, this case is called as a *over-fitting* or *over-learning* of neural network. This case is represented by the black line in Figure 3. There is an important tradeoff between learning and generalization or under-fitting and over-fitting that arises quite generally. In Figure 3, an example is shown

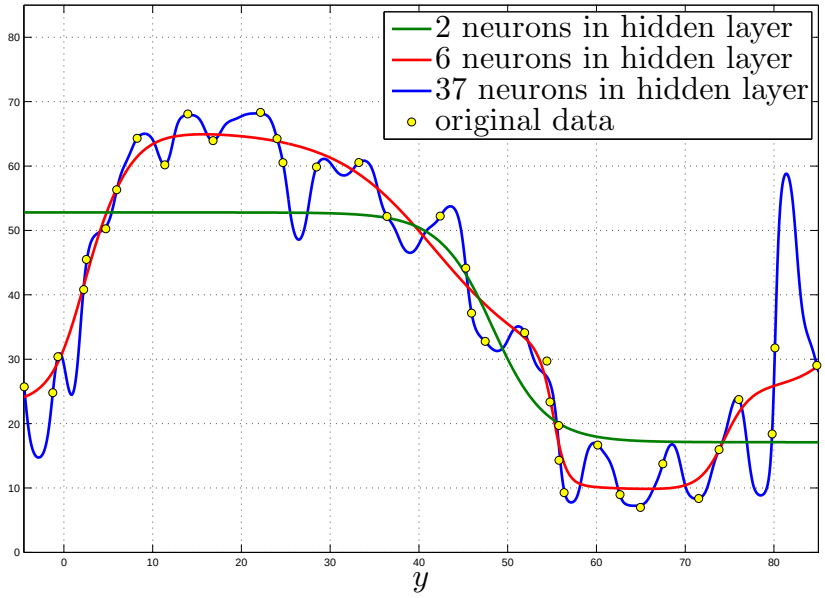

Figure 3: Approximation of data by multi-layer perceptron with different topology.

of two-layer perceptron applied on one relatively simple task of approximation the relation between one input and output. Three different topologies were examined with two, six and 37 neurons in hidden layer. Conjugate-gradient method were used as a training algorithm. From the Figure 3 it is clearly visible that too few hidden units leave high training and generalization errors due to under-fitting. Too many hidden units result in low training errors, but make the training unnecessarily slow, and result in poor generalization unless some other technique (such as regularization) is used to prevent over-fitting. Virtually all "rules of thumb" you hear about are actually nonsense. A sensible strategy is to try a range of numbers of hidden units and see which works best.

To prevent under-fitting we need to make sure that:

i) the network has enough hidden units to represent to required relationship;

ii) we train the network for long enough so that the sum squared error cost function is sufficiently minimized.

To prevent over-fitting we can:

i) stop the training early – before it has had time to learn the training data too well;

ii) restrict the number of adjustable parameters the network has – e.g. by reducing the number of hidden units, or by forcing connections to share the same weight values.[3]

iii) add some form of regularization term to the error function to encourage smoother network mappings;

iv) add noise to the training patterns to smear out the data points.

4.3 Design of experiments

In principle, we can just use any raw input-output data to train our networks. However, in practice, it often helps the network to learn appropriately if we carry out some preprocessing of the training data before feeding it to the network.

We should make sure that the training data is representative – it should not contain too many examples of one type at the expense of another. On the other hand, if one part of pairs is easy to learn, having large numbers of pairs from that part in the training set will only slow down the over-all learning process.

Beside the decision concerning a choice of training pairs for an ANN, we should pay attention also to a proper choice of ANN's input vector. Once we deal with developing an inverse model M^{INV} to a computational mechanical model M, the model outputs representing some experimental measurements become the inputs into the inverse model M^{INV}. When we transform these measurements into an input vector, the size of this vector could be very huge and then also the topology of ANN become very huge and the training process become quite complicated. In the input vector, nevertheless, one can usually find a lot of highly correlated values, which do not bring any new information.

The methodologies used for stratified choice of representative data in order to determine the relationship between input factors $\mathbf{x}$ affecting a process and the output of that process $\mathbf{y}$ are collectively known as ***design of experiments*** (DOE). Some methods are described in **?** and are mostly based upon the mathematical model of the process.

For the purposes of the inverse analysis, there are two main tasks to be solved by the DOE:

1) choice of representative data (pairs) for ANN's training;

2) choice of important inputs to ANN.

The choice of representative data for ANN's training could be governed by a particular group of methods of DOE called *sampling methods*. Several of them are listed bellow:

- Monte Carlo methods are sampling methods based on generating random vectors (points) with the defined statistical distribution for each variable. For solution of most of problems a lot of simulations are needed, e.g. thousands or millions in order to represent desired statistical distributions.

[3]Forcing connections to share the same weight values could be done by adding an appropriate term to the error/cost function. This method can be seen as a particular form of *regularization*.

- Latin hypercube sampling is probably the most popular example of sampling method, which is independent of the mathematical model of a problem. Comparing to the Monte Carlo methods, LHS needs much less simulations to represent correctly desired statistical distribution of each variable.

- Factorial design and fractional factorial design consists of two or more factors, each with discrete possible values or "levels". All factors should have the same number of levels. Factorial design (called also as full factorial design) choose the samples combining all levels for all factors. Each combination of a single level selected from every factor is present once. Fractional factorial design consists of a carefully chosen subset (fraction) of the experimental runs of the full factorial design.

The first three listed methods take part in a group of quasi-random numbers generators, whereas last two methods are deterministic approaches.

Two other techniques can be useful for selection of important input data. The first one is the principal components analysis (PCA). This technique is used to reduce multidimensional data sets to lower dimensions for analysis. Depending on the field of application, it is also named the discrete Karhunen-Loève transform (or KLT, named after Kari Karhunen and Michel Loève), the Hotelling transform (in honor of Harold Hotelling), or proper orthogonal decomposition (POD).

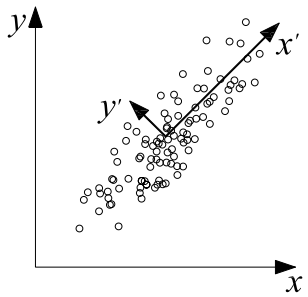

Figure 4: Example of data with two variables original variables x, y and new variables x', y' obtained by PCA.

PCA involves a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. An example of data with two variables original variables x, y and new variables x', y' obtained by PCA is shown in Figure 4.

Other possibility is to keep original data and calculate the correlation between each variable from input vector and each component from output vector. For that purpose, a number of different coefficients can be used for different situations. The best known is the *Pearson product-moment correlation coefficient*, which is obtained by dividing the covariance of the two variables by the product of their standard deviations.

Pearson's correlation coefficient is a parametric statistic, and it may be less useful if the underlying assumption of normality is violated. Non-parametric correlation methods, such as *Chi-square*, *Point biserial correlation*, *Spearman's* ρ and *Kendall's* τ may be useful when distributions are not normal; they are a little less powerful than parametric methods if the assumptions

underlying the latter are met, but are less likely to give distorted results when the assumptions fail.

In statistics, *Spearman's rank correlation coefficient* ρ is a non-parametric measure of correlation – that is, it assesses how well an arbitrary monotonic function could describe the relationship between two variables, without making any assumptions about the frequency distribution of the variables. Unlike the Pearson product-moment correlation coefficient, it does not require the assumption that the relationship between the variables is linear, nor does it require the variables to be measured on interval scales; it can be used for variables measured at the ordinal level. In principle, ρ is simply a special case of the Pearson product-moment coefficient in which the data are converted to rankings before calculating the coefficient.

5 Conclusion

The proposed paper brings an insight into procedures of inverse analysis based on soft-computing methods suitable for parameters identification. To describe problems, which are usually encountered in engineering practice as well as science, basic notation and classification is introduced. Namely, two basic modes of an inverse analysis are described: a forward mode leading to an optimization of an error function and an inverse mode leading to an inverse model development. Many applications of soft-computing methods applied to parameters identification in literature are mentioned.

An overview of softcomputing optimization methods suitable for forward mode of identification is presented in Section 3. A group of the most effective optimization methods is represented by deterministic gradient-based methods. Nevertheless, these methods are very limited in application. They are suitable for smooth objective functions with no local extremes. There are a lot of applications in literature, where some regularization technique is applied on objective function originally non-smooth in order to enable the application of gradient-based method. The problem of local extremes is usually solved by incorporation of an initial guess of an expert in order to choose the starting point in the vicinity of a global optimum. Several examples of gradient-based optimization applied to material models identification are presented e.g. in Iacono et al. (2006); Mahnken and Stein (1996); Maier et al. (2006).

All previously mentioned methods are optimization methods suitable for forward mode of an inverse analysis. The main features common for these methods are the certain possibility to control the precision of the optimum and the necessity to carry out the whole identification process for any new measurements. Only meta-modelling of a computational model leads to the identification methodology, where only a cheap optimization needs to be executed for new measurements and the time-consuming development of the meta-model is performed only once. An example of this identification methodology is presented e.g. in Pichler et al. (2003).

The inverse mode of an inverse analysis leads to the development of an inverse model to the mechanical model and it has similar properties as a forward mode based on metamodelling of the computational model. An overview of the methodologies suitable for such a mode of identification is presented in Section 4. Once the inverse model is established, it could be used repeatedly for any new measurement by a simple evaluation of the inverse model. From the implementation point of view, this approach is very simple to use, because only a limited number of simulations by the mechanical model is needed as a first step of a inverse model development and there is no need to link any optimization algorithm to the code of the mechanical model. Nevertheless, the precision of the inverse model is fixed and usually not very high.

6 Acknowledgments

The financial support of this work by the research project MSM6840770003 is gratefully acknowledged.

7 References

Novák, D., Vořechovský, M., Lehký, D., Bergmeister, K., Pukl, R., and Červenka, V. (2007). Stochastic nonlinear analysis of concrete structures - Part I: From simulation of experiment and parameter identification to reliability assessment. In Kanda, Takada, and Furuta, editors, *Applications of Statistics and Probability in Civil Engineering: Proceedings of the 10th International Conference*, London. Taylor & Francis Group.

Keršner, Z., Novák, D., Řoutil, L., and Podroužek, J. (2007). Stochastic nonlinear analysis of concrete structures - Part II: Application to fiber-reinforced concrete facade panels. In Kanda, Takada, and Furuta, editors, *Applications of Statistics and Probability in Civil Engineering: Proceedings of the 10th International Conference*, London. Taylor & Francis Group.

Babuska, I. and Oden, J. T. (2004). Verification and validation in computational engineering and science: basic concepts. *Comput. Methods Appl. Mech. Engrg.*, 193:4057–4066.

Iacono, C., Sluys, L. J., and van Mier, J. G. M. (2006). Estimation of model parameters in nonlocal damage theories by inverse analysis techniques. *Computer Methods in Applied Mechanics and Engineering*, 195(52):7211–7222.

Mahnken, R. and Stein, E. (1996). Parameter identification for viscoplastic models based on analytical derivatives of a least-squares functional and stability investigations. *International Journal of Plasticity*, 12(4):451–479.

Maier, G., Bocciarelli, M., Bolzon, G., and Fedele, R. (2006). Inverse analyses in fracture mechanics. *International Journal of Fracture*, 138:47–73.

Ingber, L. (1993). Simulated annealing: Practice versus theory. *Mathematical and Computer Modelling*, 18(11):29–57.

Vidal, R. V. V., editor (1993). *Applied Simulated Annealing*, volume 396 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Michalewicz, Z. (1999). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 3^{rd} edition.

Mahfoud, S. W. (1995b). *Niching methods for genetic algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA.

Hrstka, O. and Kučerová, A. (2004). Improvements of real coded genetic algorithms based on differential operators preventing the premature convergence. *Advances in Engineering Software*, 35(3–4):237–246.

Coello, C. A. C. (2000). Constraint-handling using an evolutionary multiobjective optimization technique. *Civil Engineering and Environmental Systems*, 17:319–346.

Coello, C. A. C. (2004). List of references on evolutionary multiobjective optimization. `http://www.lania.mx/~ccoello/EMOO/EMOObib.html`.

Miettinen, K. (1999). *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Dordrecht.

Cantú-Paz, E. (2001). *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers.

Jin, Y. (2003). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing Journal*, 9(1):3–12.

Simpson, T. W., Peplinski, J. D., Koch, P. N., and Allen, J. K. (2001). Metamodels for computer-based engineering design: survey and recommendations. *Engineering with Computers*, 17:129–150.

Queipo, N. V., Haftka, R. T., Shyy, W., Goel, T., Vaidyanathan, R., and Tucker, P. K. (2005). Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 41:1–28.

Matheron, G. (1963). Principles of geostatistics. *Econ Geol*, 58:1246–66.

Varcol, C. M. and Emmerich, T. M. (2005). Metamodel-assisted evolution strategies applied in electromagnetic compatibility design. In *Evolutionary and Eterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems EUROGEN 2005*. FLM, Munich.

Nakayama, H., Inoue, K., and Yoshimori, Y. (2004). Approximate optimization using computational intelligence and its application to reinforcement of cable-stayed bridges. In Neittaanmäki et al. (2004).

Karakasis, M. K. and Giannakoglou, K. C. (2004). On the use of surrogate evaluation models in multi-objective evolutionary algorithms. In Neittaanmäki et al. (2004).

Neittaanmäki, P., Rossi, T., Korotov, S., Oñate, E., Périaux, P., and Knörzer, D., editors (2004). *European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2004)*, Jyväskylä.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

Toropov, V. and Yoshida, F. (2005). *Parameter identification of materials and structures*, chapter Application of advanced optimization techniques to parameter and damage identification problems, pages 177–263. SpringerWienNewYork.

González, L. F., Whitney, E. J., Périaux, J., Sefrioui, M., and Srinivas, K. (2004). Multidisciplinary aircraft conceptual design and optimisation using a robust evolutionary technique. In Neittaanmäki et al. (2004).

Wang, J., Periaux, J., and Sefrioui, M. (2002). Parallel evolutionary algorithms for optimization problems in aerospace engineering. *Journal of Computational and Applied Mathematics*, 149:155–169.

Myers, R. H. and Montgomery, D. C. (1995). *Response surface methodology: Process and Product Optimization Using Designed Experiments*. John Wiley and Sons, New York, NY.

Lee, J. and Hajela, P. (2001). Application of classifier systems in improving response surface based approximations for design optimization. *Computers & Structures*, 79:333–344.

Pichler, B., Lackner, R., and Mang, H. (2003). Back analysis of model parameters in geotechnical engineering by means of soft computing. *International Journal for Numerical Methods in Engineering*, 57(14):1943–1978.

Quagliarella, D. (2003). Airfoil design using Navier-Stokes equations and an asymmetric multi-objective genetic algorithm. In Bugeda, G., Désidéri, J.-A., Périaux, J., Schoenauer, M., and Winter, G., editors, *Evolutionary Methods for Design, Optimization and Control: Applications to Industrial and Societal Problems*, Eurogen 2003. International Center for Numerical Methods in Engineering (CIMNE).

Most, T., Hofstetter, G., Hofmann, M., Novák, d., and Lehký, D. (2007). Approximation of constitutive parameters for material models using artificial neural networks. In Topping, B. H. V., editor, *Proceedings of the Ninth International Conference on the Application of Artificial Intelligence to Civil, Structural and Environmental Engineering*. Civil-Comp Press.

Lehký, D. and Novák, D. (2005). Probabilistic inverse analysis: Random material parameters of reinforced concrete frame. In *Ninth International Conference on Engineering Applications of Neural Networks, EAAN2005, Lille, France*, pages 147–154.

Fairbairn, E. M. R., Ebecken, N. F. F., Paz, C. N. M., and Ulm, F.-J. (2000). Determination of probabilistic parameters of concrete: solving the inverse problem by usign artificial neural networks. *Computers & Structures*, 78(1–3):497–503.

Novák, D. and Lehký, D. (2006). ANN inverse analysis based on stochastic small-sample training set simulation. *Engineering Applications of Artificial Intelligence*, 19(7):731–740.

Waszczyszyn, Z. and Ziemianski, L. (2005). *Parameter identification of materials and structures*, chapter Neural networks in the identification analysis of structural mechanics problems, pages 265–340. SpringerWienNewYork.

Kučerová, A., Lepš, M., and Zeman, J. (2007). Back analysis of microplane model parameters using soft computing methods. *CAMES: Computer Assisted Mechanics and Engineering Sciences*, 14(2):219–242.

Waszczyszyn, Z. and Ziemiański, L. (2006). Neurocomputing in the analysis of selected inverse problems of mechanics of structures and materials. *Computer Assisted Mechanics and Engineering Sciences*, 13(1):125–159.

Yagawa, G. and Okuda, H. (1996). *Neural networks in computational mechanics*. CIMNE.