Identification des paramètres des modèles mécaniques non-linéaires en utilisant des méthodes basées sur intelligence artificielle

Anna Kucerova

Identification of nonlinear mechanical model parameters based on softcomputing methods

by

Anna Kučerová

A treatise submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

Ecole Normale Supérieure de Cachan

&

České Vysoké Učení Technické v Praze
Fakulta stavební

Committee:

Prof. Drahomír Novák	Vysoké Učení Technické v Brně	President
Prof. Tomaž Rodič	Univerza v Ljubljani	Opponent
Prof. Pierre Villon	Université de Technologie de Compiègne	Opponent
Delphine Brancherie, Ph.D.	Université de Technologie de Compiègne	Examiner
Jan Zeman, Ph.D.	České Vysoké Učení Technické v Praze	Examiner
Prof. Hermann Matthies	Technische Universität Braunschweig	Examiner
Prof. Djordje Peric	University of Wales, Swansea	Examiner
Prof. Zdeněk Bittnar	České Vysoké Učení Technické v Praze	Supervisor
Prof. Adnan Ibrahimbegović	Ecole Normale Supérieure de Cachan	Supervisor

Laboratoire de Mécanique et Technologie (ENS CACHAN/CNRS/UMR 8095)

27 Noverber 2007

To my brother

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to Prof. Ing. Zdeněk Bittnar, DrSc. for his support and patience not only during the entire course of the work on this thesis but also during my whole studies.

I would like also to express my deep and sincere gratitude to my supervisor in France, Prof. Adnan Ibrahimbegović from Laboratoire de mécanique et technologie de Ecole Normale Supérieure de Cachan, for his personal guidance, understanding, stimulating suggestions and encouragement throughout all my Ph.D. studies.

I would like to give my thanks to Ing. Jan Zeman, Ph.D. and Ing. Matěj Lepš, Ph.D. for their inspiration during my research as well as for very thorough proof-reading of my manuscripts.

I wish also to thank other colleagues who have given me valuable comments and advice. Particularly, I would like to thank Delphine Brancherie, Ph.D., Ing. Zuzana Vitingerová, Ing. Sergey Melnyk and Ing. Jan Skoček for close cooperation on many issues mentioned in this research work.

Most importantly, I would like to thank my parents, my boyfriend Jan and friends for their never ending encouragement and support that help me to attain the goal I have set for myself and for the opportunity to fully concentrate on my study.

This work was supported by the research project CEZ MSM 6840770003 and by a grant GAČR 103/05/H506. The financial support provided by France, particularly, Le Centre National des Oeuvres Universitaires et Scolaires (CNOUS) within the frame of bilateral agreement of Ph.D. studies under co-tutelle is gratefully acknowledged.

Last but not least, parts of this work were produced under the support of CTU grants CTU 0501511 and CTU 0613511.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

> There are many methods for predicting the future. For example, you can read horoscopes, tea leaves, tarot cards, or crystal balls. Collectively, these methods are known as 'nutty methods'. Or you can put well-researched facts into sophisticated computer models, more commonly referred to as 'a complete waste of time'.
>
> Scott Adams

Preface

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 thesis, we present an overview of two basic philosophies of the inverse analysis aimed, in particular at parameters estimation with utilization of soft-computing methods. Practical aspects will be shown in detail on several identification tasks, where parameters of highly non-linear material models are searched for.

1.1 Motivation and objectives

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 identification 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.

Nowadays, the most often method used for identification of model parameters in engineering practice is, however, the trial-and-error method. One reason is a lack of the literature summarizing the identification methodologies suitable for model parameters identification.

The main goals of the present work could be written as follows:

i) suggest a basic classification and notation of methods suitable for model parameters identification;

ii) provide a guide for the best choice of the type of algorithm most suitable for a particular application;

iii) develop new methods suitable for identification;

iv) enhance understanding of several nonlinear mechanical constitutive models both in terms of their domain of application and in terms of sensitivity of their parameters;

v) test the proposed identification methods in the framework of several constitutive models with inelastic behavior.

1.2 Preliminary notes and definitions

As mentioned previously, 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

$$\mathbf{y}^E = E(\mathbf{x}^E). \tag{1.1}$$

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

$$M \approx E; \tag{1.2}$$

$$\mathbf{y}^M = M(\mathbf{x}^M). \tag{1.3}$$

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.

$$\left\|\mathbf{x}^M\right\| < 10. \tag{1.4}$$

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.

$$\left\|\mathbf{y}^M\right\| = \left\|\mathbf{y}^E\right\| = T \times P \times C \approx 100. \tag{1.5}$$

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 continuum mechanics, the goal of verification processes to assess the difference between results produced by the computational model and the mathematical model. These types of errors arise in two basic ways and needs two corresponding categories of verification:

i) The code may not be a reliable and accurate implementation of the discretized model – *code verification*, a province of software engineering, is needed.

ii) The discretized model may not be an accurate representation of the mathematical model – *solution verification* is needed, which involves a posteriori error estimation.

If a code is free of error, the verification processes are by no means done: the error in the numerical values of the event or events of interest due to discretization remains to be quantified. If this error due to discretization can be quantified, estimated, or bounded in some way and judged to be acceptable by the analyst, then what remains to be assessed is the validity of the theory (i.e. the mathematical model), a goal of the validation process. "Thus, quantifying discretization error, a principal goal of verification processes, is in general, a necessary prelude to the validation processes, as to do otherwise could lead to an entanglement of modelling and discretization error and obviate the entire validation exercise" [Babuska and Oden, 2004].

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;

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. On the one hand, there are the apparatusis needed to make measurements and supply input to a program of physical tests, while on the other hand, there are devices and possibly technicians that can record and interpret the observed output. In analogy to the verification processes, the apparatusis must be calibrated to

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

and lead to accuracies regarded as acceptable by the experimentalist. In the analogy to validation, confidence that the experiment itself measures accurately the event of interest must be established.

Nevertheless, 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]. Therefore, this aspect of identification procedure is omitted in the sequel.

As a conclusion let us repeat the main source of error in the identification procedure:

i) noise in experimental measurements;

ii) inaccuracy of mathematical model or its numerical implementation;

iii) incompetence of an identification method.

In this work, two different modes in identification procedure are distinguished: a forward mode of an inverse analysis leading to an optimization problem is described in following chapter; an inverse mode of an inverse analysis based on determination of an inverse model is discussed in Chapter 3. Chapters 4 and 5 contain detailed description of several proposed methods applicable to forward and inverse mode of an inverse analysis, respectively, supplemented with results on representative mathematical tests. Chapters 6 to 8 presents the applications of proposed identification methodologies to optimal design, optimal control and parameters identification of two non-linear material models. The conclusion and final remarks are given in Chapter 9.

Chapter 2

FORWARD MODE OF AN INVERSE ANALYSIS

> I do not fear computers. I fear the lack of them.
>
> Isaac Asimov

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.

$$\min F(\mathbf{x}) = \min \|\mathbf{y}^E - M(\mathbf{x})\|. \tag{2.1}$$

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 (2.1) 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, cf. Section 4.1.3.

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, Coello, 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 (1.3) 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 2.3 and 2.4.

2.1 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.

$$\tilde{M} \approx M \tag{2.2}$$

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 2.1 and the consecutive steps of this methodology are described below.

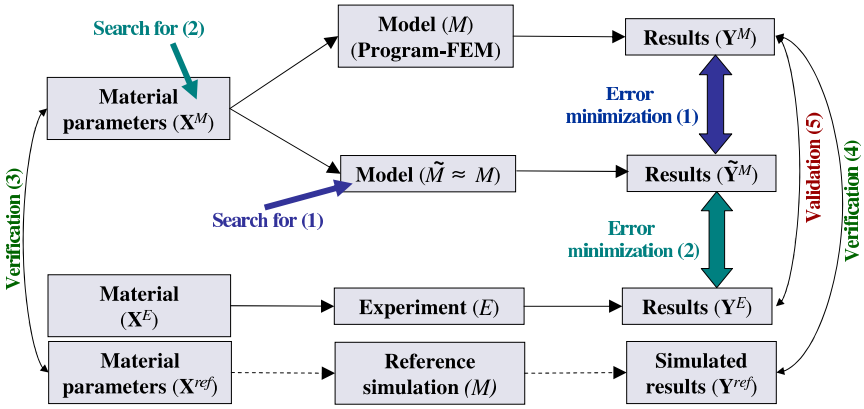


Figure 2.1: Forward mode of identification process using model approximation

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 and training process must be performed in order to minimize the error between response of model $\tilde{M}$ and the original model M. More details about such process could be found in Chapter 3, since approximative model estimation is very close to inverse model estimation.

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.

2.2 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

$$F = \left\| \mathbf{y}^E - M(\mathbf{x}) \right\| . \tag{2.3}$$

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.

$$\tilde{F} \approx F. \tag{2.4}$$

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 2.2. The consecutive steps of this implementation are almost the same

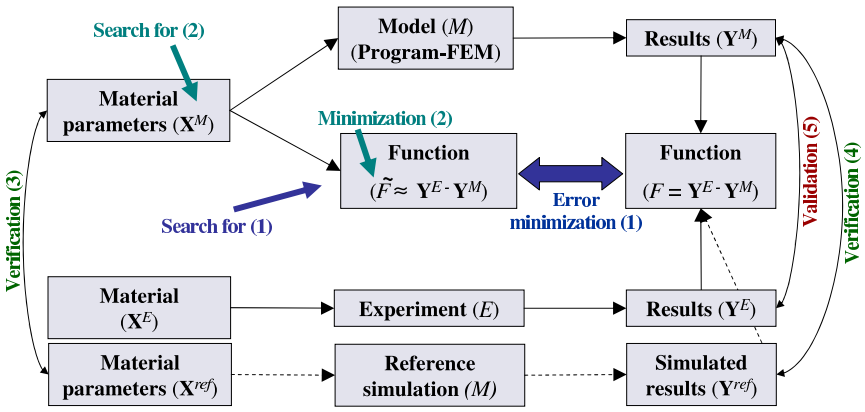


Figure 2.2: Forward mode of identification process using error function approximation

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.

Design of experiments	**Model choice**	**Model fitting**	**Sample techniques**
(Fractional) Factorial	Polynomial (Linear, quadratic)	Least squares regression	Response surface methodology
Central composite	Radial basis function network	Weighted least squares regression	**Forward mode Chapter 4**
D-optimal	Realization of stochastic process	Best linear predictor	Kriging
Random selection	Functions and terminals	Genetic algorithm	Genetic programming
Latin Hypercube	Splines (Linear, cubic)		**Inverse mode Chapter 5**
Selected by hand	Multi-layer perceptron	Back Propagation	Neural networks
Orthogonal array	Decision tree	Entropy	Inductive learning

Table 2.1: Review of some meta-model techniques

Some meta-model techniques, which could be found in literature, are listed in Table 2.1. 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 3.3.

2.3 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[1], 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

$$f_p(\mathbf{x}) = \mu(\mathbf{x}) + \varepsilon(\mathbf{x}), \tag{2.5}$$

   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 3.1 and one particular implementation with some "mathematical" objective function are described in Section 4.2. Some engineering applications could be also found in [Nakayama et al., 2004] and [Karakasis and Giannakoglou, 2004]. An application in identification domain is published in [Kucerova et al., 2007].

[1] a South African mining engineer

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].

2.4 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].

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 3.3), 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 3.1. 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].

Chapter 3

INVERSE MODE OF AN INVERSE ANALYSIS

> Everything should be made as simple as possible, but not one bit simpler
>
> Albert Einstein

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:

$$\mathbf{x} = M^{INV}(\mathbf{y}) \tag{3.1}$$

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 (3.1) should also fulfill the Equation (**??**). 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 (3.1).

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.

3.1 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. The field goes by many names, such as connectionism, parallel distributed processing, neurocomputing, natural intelligent systems, machine learning algorithms, and artificial neural networks.

In most cases, an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning (training) phase.

The first neuron model was proposed already in 1940s and since that there were a lot of developments and paper published in the field (see e.g. [Haykin, 1998] or [Hertz et al., 1991]). A brief history of the ANNs is following:

1942 McCulloch and Pitts proposed the McCulloch-Pitts neuron model, a crude approximation to real neuron that performs a simple summation and thresholding function on activation levels.

1949 Hebb published his book *The Organization of Behavior*, in which the Hebbian learning rule was proposed.

1958 Rosenblatt introduced the simple single layer networks now called Perceptrons.

1969 Minsky and Papert's book *Perceptrons* demonstrated the limitation of single layer perceptrons, and almost the whole field went into hibernation.

1982 Hopfield published a series of papers on Hopfield networks.

1982 Kohonen developed the Self-Organizing Maps that now bear his name.

1986 The back-propagation learning algorithm for multi-layer perceptrons was rediscovered and the whole field took off again.

1990s The sub-field of Radial Basis Function Networks was developed.

2000s The power of Ensembles of Neural Networks and Support Vector Machines becomes apparent.

ANN are a powerful technique to solve many real world problems. They have the ability to learn from experience in order to improve their performance and to adapt themselves to changes in the environment. In addition to that they are able to deal with incomplete information or noisy data and can be very effective especially in situations where it is not possible to define the rules or steps that lead to the solution of a problem. Their simple implementation and massive parallelism makes them very easy and efficient to use.

Existing papers suggest different categorization for Neural Networks. Following list represents one possible view on that subject and may vary from other publications.

Clustering A clustering algorithm explores the similarity between patterns and places similar patterns in a cluster. Best known applications include data compression and data mining.

Classification/Pattern recognition The task of pattern recognition is to assign an input pattern (like handwritten symbol) to one of many classes. This category includes algorithmic implementations such as associative memory. Typical application are speech recognition, hand-writing recognition, sonar signals.

Function approximation The tasks of function approximation is to find an estimate of the unknown function f subject to a noise. Various engineering and scientific disciplines require different function approximation.

Prediction/Dynamical Systems The task is to forecast some future values of a time-sequenced data. Prediction has a significant impact on decision support systems. Prediction differs from function approximation by considering a time factor. Here the system is dynamic and may produce different results for the same input data based on system state (time). Famous applications are predicting stocks, shares, currency exchange rates, climate, weather, airline marketing tactician.

Perhaps the greatest advantage of ANNs is their ability to be used as an arbitrary function approximation mechanism which 'learns' from observed data. However, using them is not so straightforward and a relatively good understanding of the underlying theory is essential. Once the model, cost function and learning algorithm are selected appropriately the resulting ANN can be extremely robust.

A basic unit of any ANN is an artificial neuron, firstly proposed by McCulloch and Pitts. The schema of the McCulloch-Pitts neuron (also known as a threshold logic unit) is shown in Figure 3.1. A set of synapses (i.e. connections) brings in activations from other neurons.

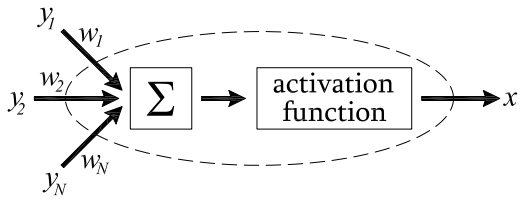


Figure 3.1: Schema of McCulloch-Pitts neuron

A processing unit sums the inputs ($y_1 ... y_N$), and then applies a non-linear activation function (i.e. squashing/transfer/threshold function). An output line ($x_1 ... x_N$) transmits the result to other neurons.

We can connect any number of McCulloch-Pitts neurons together in any way we like. An arrangement of one input layer of McCulloch-Pitts neurons feeding forward to one output layer of McCulloch-Pitts neurons is known as a *perceptron*. In this way it can be considered the simplest kind of a *feedforward network*.

Perceptrons can be trained by a simple learning algorithm that is usually called the delta rule. It calculates the errors between calculated output and sample output data, and uses this to create an adjustment to the weights, thus implementing a form of gradient descent.

Single-unit perceptrons are only capable of learning linearly separable patterns; the proof that it was impossible for a single-layer perceptron network to learn an XOR function is shown in [Minsky and Papert, 1969]. The authors also conjectured (incorrectly) that a similar result would hold for a multi-layer perceptron network. Although a single threshold unit is quite limited in its computational power, it has been shown that networks of parallel threshold units can approximate any continuous function from a compact interval of the real numbers into the interval $[-1, 1]$. This very recent result can be found in [Auer et al., 2005].

Many more powerful neural network variations are possible – we can vary the architecture and/or the activation function and/or learning algorithm. Some representative and often used types of ANNs are listed bellow:

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 $\mathbf{y} = \{y_i \in [0,1] : i = 1,\dots,I\}$ and $\varepsilon > 0$, there exists an integer J and real constants $\{\alpha_j, b_j, w_{jk} : j = 1,\dots,J, k = 1,\dots,I\}$ such that

$$F(y_1,\dots,y_I) = \sum_{j=1}^{J} \alpha_j \varphi\left(\sum_{k=1}^{I} w_{jk} y_k - b_j\right) \tag{3.2}$$

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

$$\|F(y_1,\dots,y_I) - f(y_1,\dots,y_I)\| < \varepsilon \tag{3.3}$$

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 of multi-layer perceptron could be found in [Weigend and Gershenfeld, 1994] concerning time series prediction; in [le Cun et al., 1989] for written zip code recognition. 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.

Intuitively, we can easily understand why linear superpositions of localized basis functions are capable of universal approximation. More formally:

> Hartman, Keeler & Kowalski in [Hartman et al., 1990] provided a formal proof of this property for networks with Gaussian basis functions in which the widths $\{\sigma_j\}$ are treated as adjustable parameters.
>
> Park & Sandberg in [Park and Sandberg, 1991, Park and Sandberg, 1993] showed that with only mild restrictions on the basis functions, the universal function approximation property still holds.

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.

Associating each input datum with a RBF leads naturally to kernel methods such as Support Vector Machines and Gaussian Processes (the RBF is the kernel function). All three approaches use a non-linear kernel function to project the input data into a space where the learning problem can be solved using a linear model. Like Gaussian Processes, and unlike SVMs, RBF networks are typically trained in a Maximum Likelihood framework by maximizing the probability (minimizing the error) of the data under the model. SVMs take a different approach to avoiding overfitting by maximizing instead a margin. RBF networks are outperformed in most classification applications by SVMs. In regression applications they can be competitive when the dimensionality of the input space is relatively small.

One successful real-world application of RBFN detects epileptiform artifacts in EEG recordings, for full details see [Saastamoinen et al., 1998]. Some applications to engineering problems could be found in [Nakayama et al., 2004] or [Karakasis and Giannakoglou, 2004].

Kohonen self-organizing network

This network invented by Teuvo Kohonen [Kohonen, 1982] uses a form of unsupervised learning. A set of artificial neurons learns to map points in an input space to coordinates in an output space. The input space can have different dimensions and topology from the output space, and the SOM will attempt to preserve these.

Recurrent network

While a feedforward network propagates data linearly from input to output, recurrent networks also propagate data from later processing stages to earlier stages, i.e. they contain at least one feed-back connection.

Hopfield network

It is a recurrent neural network in which all connections are symmetric. Invented by John Hopfield in 1982 (see [Hopfield, 1982]), this network guarantees that its dynamics will converge. If the connections are trained using Hebbian learning then the Hopfield network can perform as robust content-addressable memory, resistant to connection alteration.

Stochastic neural networks

They differ from a typical neural network in the fact that it introduces random variations into the network. In a probabilistic view of neural networks, such random variations can be viewed as a form of statistical sampling, such as Monte Carlo sampling.

Fuzzy neural networks

Fuzzy methods are used to enhance the learning capabilities or the performance of a neural network. This can be done either by creating a network that works with fuzzy inputs [Narazaki and Ralescu, 1991] or by using fuzzy rules [Halgamuge et al., 1994] to change the learning rate. Some engineering applications were published in [Rajasekaran et al., 1996] or [Waszczyszyn and Ziemianski, 2005]. These approaches are not to be confused with *neuro-fuzzy approaches*, where neural network is usually used to determine the parameters of a fuzzy system.

A variety of other types of ANNs could be found in literature.

3.2 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.

$$E(w_{ij}) = \frac{1}{2} \sum_p \sum_j \left(M^{INV}(\mathbf{y})_j - x_j \right)^2, \tag{3.4}$$

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 3.2 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 3.2. There is an important tradeoff between learning and generalization or under-fitting and over-fitting that arises quite generally. In Figure 3.2, an example is shown

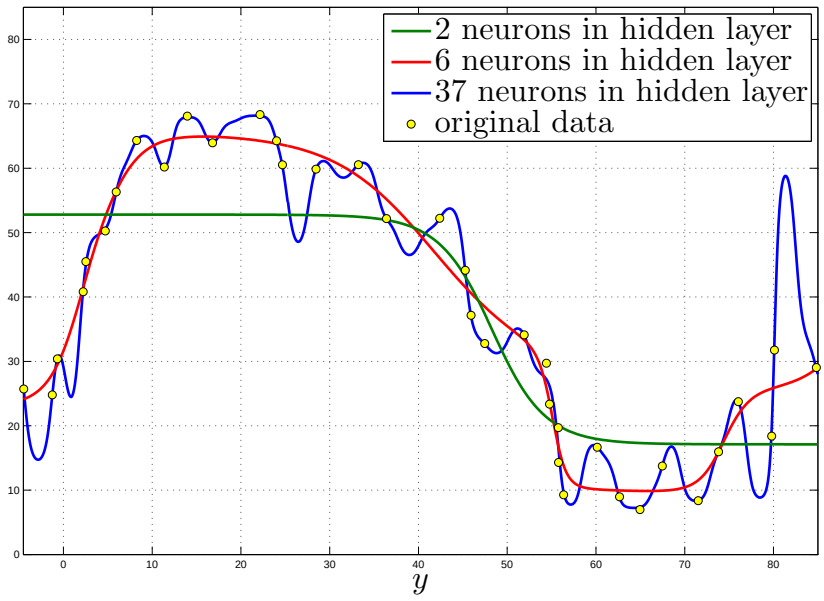


Figure 3.2: Approximation of data by multi-layer perceptron with different topology.

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 3.2 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.[1]

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.

We usually want to optimize our network's training procedures to result in the best generalization, but using the testing data to do this would clearly be cheating. What we can do is assume that the training data and testing data are drawn randomly from the same data set, and then any sub-set of the training data that we do not train the network on can be used to estimate what the performance on the testing set will be, i.e. what the generalization will be. The portion of the data we have available for training that is withheld from the network training is called the *validation data* set, and the remainder of the data is called the *training data* set. This approach is called the *hold out method*.

Often the availability of training data is limited, and using part of it as a validation set is not practical. An alternative is to use the procedure of *cross-validation*. In K*-fold cross-validation* we divide all the training data at random into K distinct subsets, train the network using $K-1$ subsets, and test the network on the remaining subset. The process of training and testing is then repeated for each of the K possible choices of the subset omitted from the training. The average performance on the K omitted subsets is then our estimate of the generalization performance. This procedure has the advantage that it allows us to use a high proportion of the available training data (a fraction $1-1/K$) for training, while making use of all the data points in estimating the generalization error. The disadvantage is that we need to train the network K-times. Typically $K \sim 10$ is considered reasonable.

Perhaps the most obvious way to prevent over-fitting in our models (i.e. neural networks) is to restrict the number of free parameters they have. The simplest way we can do this is to restrict the number of hidden units, as this will automatically reduce the number of weights. We can use some form of validation or cross-validation scheme to find the best number for each given problem. An alternative is to have many weights in the network, but constrain certain groups of them to be equal. If there are symmetries in the problem, we can enforce *hard weight sharing* by building them into the network in advance. In other problems we can use *soft weight sharing* where sets of weights are encouraged to have similar values by the learning algorithm.

Neural networks are often set up with more than enough parameters for over-fitting to occur, and so other procedures have to be employed to prevent it. During the training process,

[1] 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*.

the error on the unseen validation and testing data sets, however, will start off decreasing as the under-fitting is reduced, but then it will eventually begin to increase again as over-fitting occurs. The natural solution to get the best generalization, i.e. the lowest error on the test set, is to use the procedure of *early stopping*. One simply trains the network on the training set until the error on the validation set starts rising again, and then stops. That is the point at which we expect the generalization error to start rising as well. One potential problem with the idea of stopping early is that the validation error may go up and down numerous times during training. The safest approach is generally to train to convergence (or at least until it is clear that the validation error is unlikely to fall again), saving the weights at each epoch, and then go back to weights at the epoch with the lowest validation error.

Adding *noise* or *jitter* to the inputs during training is also found empirically to improve network generalization. This is because the noise will "smear out" each data point and make it difficult for the network to fit the individual data points precisely, and consequently reduce over-fitting. The gradient descent weight updates can then be performed with an extended back-propagation algorithm based on a standard *Tikhonov regularizer minimizing curvature*.

The approaches all work well, and which we choose will ultimately depend on which is most convenient for the particular problem in hand. Unfortunately, there is no overall best approach!

3.3 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 [Montgomery, 2005] 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.

3.3.1 Training data preparation

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

Monte Carlo methods

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.

Latin hypercube sampling

This method was first described in [McKay et al., 1979] and [Iman and Conover, 1980] and till now it 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.

When sampling a function of N variables, the range of each variable is divided into M equally probable intervals. M sample points are then placed to satisfy the Latin hypercube requirements; note that this forces the number of divisions, M, to be equal for each variable. Also note that this sampling scheme does not require more samples for more dimensions (variables); this independence is one of the main advantages of this sampling scheme. Another advantage is that random samples can be taken one at a time, remembering which samples were taken so far.

It is possible to distinguish two main LHS methods: the random LHS method and the optimal LHS designs. The random LHS method uses random sampling to get each point, whereas the optimal LHS methods use more structured approaches with the aim of optimizing the uniformity of the distribution of the points. An example of random LHS design is shown in Figure 3.3. Several different criteria and/or optimization methods were proposed for optimal LHS, such as:

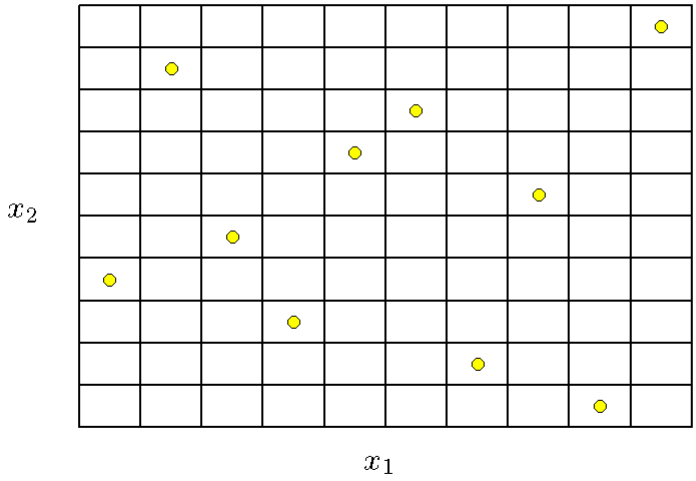


Figure 3.3: Distribution of 10 points for two variables by latin hypercube sampling.

i) maximizing entropy [Shewry and Wynn, 1987];

ii) integrated mean-squared error [Sacks et al., 1989];

iii) maximization of the minimum distance between points [Johnson et al., 1990];

iv) criterium based on potential energy of the points proposed in [Audze and Eglais, 1977] and developed in [Toropov et al., 2007];

v) minimizing correlation by the simulated annealing proposed in [Novák et al., 2003].

Factorial design

Such 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. An example of 2^3 is shown in Figure 3.4.

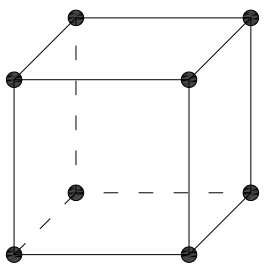


Figure 3.4: Samples distribution for 2^3 full factorial design.

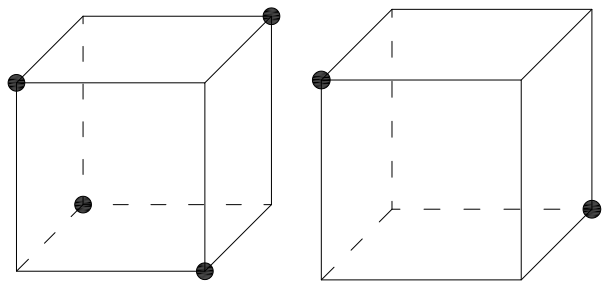
This design is usually applied in cases considering only two levels, where the number of samples N is equal to 2^k with k defining the number of factors. Also for higher numbers of factors, the number of samples becomes too high to be logistically feasible, e.g. for $k = 10$: $2^{10} = 1024$.

Fractional factorial design

Fractional factorial design consists of a carefully chosen subset (fraction) of the experimental runs of the full factorial design. The subset is chosen so as to exploit the sparsity-of-effects principle to expose information about the most important features of the problem studied, while using a fraction of the effort of the full factorial design in terms of experimental runs and resources.

Fractional designs are expressed using the notation l^{k-p}, where l is the number of levels of each factor investigated, k is the number of factors investigated, and p describes the size of the fraction of the full factorial used. Formally, p is the number of generators, assignments as to which effects or interactions are confounded, i.e., cannot be estimated independently of each other (see below). A design with p such generators is a l^{-p} fraction of the full factorial design.

For example, a 2^{5-2} design is $1/4$ of a two level, five factor factorial design. Rather than the 32 runs that would be required for the full 2^5 factorial experiment, this experiment requires only eight runs. The samples distributions for 2^{3-1} and for 2^{3-2} fractional factorial design are shown in Figures 3.5a and 3.5b, respectively.



(a) (b)

Figure 3.5: Samples distribution for (a) 2^{3-1} and for (b) 2^{3-2} fractional factorial designs.

In practice, one rarely encounters $l > 2$ levels in fractional factorial designs, the methodology to generate such designs for more than two levels is much more cumbersome.

More details about factorial designs could be found e.g. in [Montgomery, 2005].

3.3.2 Selection of input data

Principal component analysis

Principal components analysis (PCA) is a technique 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).

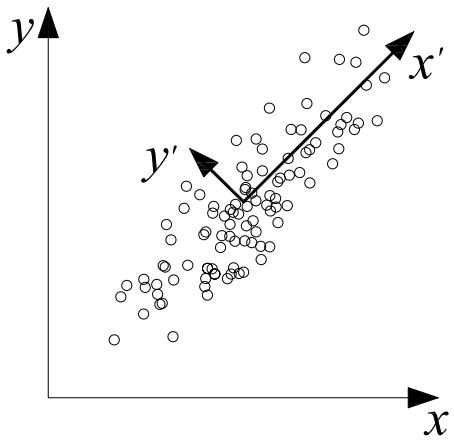


Figure 3.6: Example of data with two variables original variables x, y and new variables x', y' obtained by PCA.

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 3.6.

The mathematical technique used in PCA is eigen analysis: we solve for the eigenvalues and eigenvectors of a square symmetric matrix with sums of squares and cross products. The eigenvector associated with the largest eigenvalue has the same direction as the first principal component. The eigenvector associated with the second largest eigenvalue determines the direction of the second principal component. The sum of the eigenvalues equals the trace of the square matrix and the maximum number of eigenvectors equals the number of rows (or columns) of this matrix. For more details about this method, see [Jolliffe, 2002].

Correlation coefficients

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 some correlation coefficient could be used. It is a number between -1 and 1 which measures the degree to which two variables are linearly related. If there is perfect linear relationship with positive slope between the two variables, we have a correlation coefficient of 1; if there is positive correlation, whenever one variable has a high (low) value, so does the other. If there is a perfect linear relationship with negative slope between the two variables, we have a correlation coefficient of -1; if there is negative correlation, whenever one variable has a high (low) value, the other has a low (high) value. If the variables are independent then the correlation is 0, but the converse is not true because the correlation coefficient detects only linear dependencies between two variables.

Some examples of correlation and corresponding graphical visualization is shown in Figure 3.7. The data are graphed on the lower left and their correlation coefficients listed on the upper

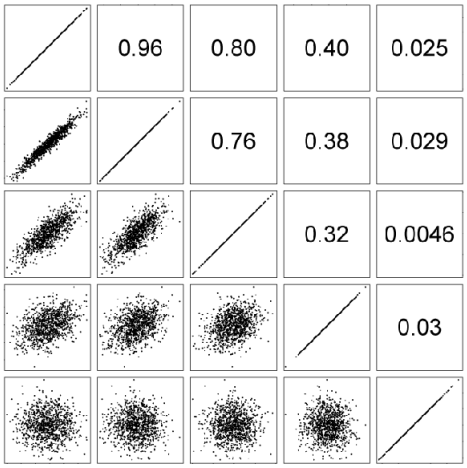


Figure 3.7: Positive linear correlations between 1000 pairs of numbers.

right. Each square in the upper right corresponds to its mirror-image square in the lower left, the "mirror" being the diagonal of the whole array. Each set of points correlates maximally with itself, as shown on the diagonal (all correlations = $+1$).

A number of different coefficients are 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. Despite its name, it was first introduced by Francis Galton.

If we have a series of n measurements of $\mathbf{x}$ and $\mathbf{y}$ written as x_i and y_i where $i = 1, 2, \ldots, n$, then the Pearson product-moment correlation coefficient can be used to estimate the correlation of $\mathbf{x}$ and $\mathbf{y}$. The Pearson coefficient is also known as the "sample correlation coefficient". It is especially important if $\mathbf{x}$ and $\mathbf{y}$ are both normally distributed. The Pearson correlation

coefficient is then the best estimate of the correlation of $\mathbf{x}$ and $\mathbf{y}$. The Pearson correlation coefficient is written as:

$$cor = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}, \tag{3.5}$$

where $\bar{x}$ and $\bar{y}$ are the sample means of $\mathbf{x}$ and $\mathbf{y}$.

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, named after Charles Spearman, 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.

Part I

Description of proposed identification methods

Chapter 4

FORWARD MODE METHODS

> Statistics, you can prove anything with statistics.
>
> Sir Humphrey Appleby

Recall that the *forward* mode of an inverse analysis is defined as a minimization of an error function $F(X)$ defined by the equation (2.1). 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. Sometimes it is also important to find a solution with a given or high accuracy (e.g. in cases of sequential identification, where the errors in starting steps accumulate in the following steps). This could be also easily done by this approach.

This chapter consists of two sections concerning optimization algorithms successfully applied at several engineering tasks. The first section deals with genetic algorithms, which are usually applied as very robust optimization methods. In Section 4.1.3 a novel niching strategy is presented, which enables in combination with a genetic algorithm to solve with almost 100% reliability a lot of very complex multi-modal and high-dimensional problems. Nevertheless, such algorithms seems to be very expensive when solving a rather smooth functions with small number of local extremes.

The second section presents a method aimed to decrease a number of objective function evaluations based on formation of cheap interpolation functions. In particular, the proposed method combines the radial basis function network (RBFN) as an interpolation function with the optimization by the genetic algorithm GRADE describe in Section 4.1.2. This methodology is not able to solve generic multi-modal, high-dimensional problems, but is very efficient for optimization of almost smooth functions with small number of local extremes.

As a principal disadvantage of both proposed methodologies remains the 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 in Chapter 3.

4.1 Genetic algorithms

At present, genetic algorithms belong to the most modern and most popular optimization methods available. They follow an analogy of processes that occur in living nature within the evolution of live organisms during a period of many millions of years. The principles of genetic algorithms were first proposed by J. H. Holland [Holland, 1975]; the books of D. E. Goldberg [Goldberg, 1989] and Z. Michalewicz [Michalewicz, 1999] are the most popular publications that deal with this topic. Genetic algorithms have been successfully used to solve optimization problems in combinatorics (see [Grefenstette, 1987]) as well as in different engineering tasks, see for example [Ibrahimbegović et al., 2004, Lepš and Šejnoha, 2003, Lepš, 2005] or [Rafiq and Southcombe, 1998].

Unlike the traditional gradient optimization methods, genetic algorithms operate on a set of possible solutions ("chromosomes"), called a "population". In the basic scheme, chromosomes are represented as binary strings. This kind of representation seems to be very convenient for optimization problems in combinatoric area (e.g., the travelling salesman problem). Nevertheless, we usually deal with the real-valued parameters in engineering and scientific problems. The mapping of real values onto binary strings usually used within standard genetic algorithms may cause serious difficulties. As a result, this concept of optimization leads to an unsatisfactory behavior, characterized by a slow convergence and an insufficient precision, even in cases where the precision is especially in focus. Of course, the development of genetic algorithms has brought several proposals to solve these difficulties to optimize problems on real domains using binary algorithms.

Another possibility is to develop a genetic algorithm (or other evolutionary algorithm) that operates directly on real values [Michalewicz, 1999]. In this case, the crucial problem is how to construct genetic operators. One of them is to use so-called differential operators that are based on determining mutual distances of chromosomes – which are real vectors instead of binary strings in this approach.

In the first Section of this Chapter, a differential genetic algorithm SADE developed at CTU in Prague several years ago [Hrstka and Kučerová, 2000] is described. A detailed comparison of this algorithm with the differential evolution proposed in [Storn and Price, 1995, Storn, 1996, Storn, WWW], a standard binary genetic algorithm and an extended binary genetic algorithm is presented in the [Hrstka and Kučerová, 2004]. Another comparison of the SADE algorithm with a real-valued augmented simulated annealing (RASA), an integer augmented simulated annealing (IASA) and also differential evolution on two mathematical and two engineering problems can be found in [Hrstka et al., 2003].

The results of test computations on twenty mathematical functions show definitely that for the optimization of multi-modal but still continuous problems on real domains the evolutionary methods based on real encoding and differential operators approve themselves much better than traditional binary genetic algorithms, even when extended by sophisticated improvements. The real encoded algorithms produced better results both in simple cases, where they have reached much better (several times) convergence rates as well as in the complicated cases,

where the obtained results were very satisfactory from the reliability point of view, even for functions where binary algorithms have failed completely.

The next interesting result is that the SADE algorithm has approximately the same reliability as the binary algorithm extended by several, rather sophisticated, improvements. The SADE algorithms reached more then 95% of successful runs on 17 functions, extended binary algorithm on 15 functions. The reliability of a differential evolution is somehow fluctuating (95% reliability only for 11 functions) and the standard binary algorithm does not show satisfactory behavior except the most simple cases. Nevertheless, the differential evolution seems to be the most effective (the fastest optimization method). For other cases, the SADE method was the fastest one. The binary algorithm has never reached the best convergence rate with this test computations.

During last few years some modifications and simplifications were proposed to the SADE algorithm and the new version called GRADE algorithm is described in very details in Section 4.1.2 and the comparison with SADE algorithm is presented in Section 4.1.4.

Although the outstanding ability of genetic algorithms to find global optima of multi-modal functions (functions which have several local extremes) is usually cited in the GA literature, it seems that both the binary genetic algorithms and the real coded ones tend to premature converge and to fall into local extremes, mainly in high dimensional cases. To overcome this difficulty, the so-called CERAF strategy was proposed and is described in Section 4.1.3.

4.1.1 SADE algorithm

This method was proposed as an adaption of the differential evolution after relatively long time of development. Its aim was to formulate a method which is able to solve optimization problems on real domains with a high number of variables (it was tested on problems with up to 200 variables). This algorithm combines the features of the differential evolution with those of the traditional genetic algorithms. It uses the simplified differential operator, but contrary to the differential evolution, the SADE method uses the algorithmic scheme very similar to the standard genetic algorithm:

1. As the first step, the initial population is generated randomly and the objective function value is assigned to all chromosomes in the population. The size of the population is defined as the number of variables of objective function multiplied by parameter pop_rate.

2. Several new chromosomes are created using the mutation operators - the mutation and the local mutation (their total number depends on the value of a parameter called $radioactivity$ – it gives the mutation probability).

3. Another new chromosomes are created using the simplified differential operator; the whole amount of chromosomes in the population is now doubled.

4. The objective function values are assigned to all newly created chromosomes.

5. The selection operator is applied to the double-sized population. Hence, the amount of individuals is decreased to its original value.

6. Steps 2-5 are repeated until a stopping criterion is reached.

Next, we describe the introduced operators in more detail. Let $\mathbf{x}_i(g)$ be the i-th chromosome in a generation g,

$$\mathbf{x}_i(g) = (x_{i1}(g), x_{i2}(g), \ldots, x_{in}(g)), \tag{4.1}$$

where n is the number of variables of the objective function. Now, the genetic operators can be written as follows:

mutation – If a certain chromosome $\mathbf{x}_i(g)$ was chosen to be mutated, a random chromosome $\mathbf{x}_{RP}$ is generated and the new chromosome $\mathbf{x}_k(g+1)$ is computed using the following relation:

$$\mathbf{x}_k(g+1) = \mathbf{x}_i(g) + MR(\mathbf{x}_{RP} - \mathbf{x}_i(g)), \tag{4.2}$$

where MR is a parameter called $mutation\_rate$,

local mutation – If a certain chromosome was chosen to be locally mutated, all its coordinates are altered by a random value from a given (usually very small) range,

crossing-over – Instead of traditional cross-over, the SADE method uses the simplified differential operator taken from the differential evolution[1], which can be written as

$$\mathbf{x}_k(g+1) = \mathbf{x}_p(g) + CR(\mathbf{x}_q(g) - \mathbf{x}_r(g)), \tag{4.3}$$

where $\mathbf{x}_p(g)$, $\mathbf{x}_q(g)$ and $\mathbf{x}_r(g)$ are three randomly chosen chromosomes and CR is parameter called $cross\_rate$. Figure 4.1 shows the geometrical meaning of this operator.

selection – this method uses modified tournament strategy to reduce the population size: two chromosomes are randomly chosen, compared and the worse is rejected. Therefore, the population size is decreased by one. This step is repeated until the population reaches its original size[2].

The detailed description of the SADE algorithm and the tests documentation for high-dimensional problems can be found in the article [Hrstka and Kučerová, 2000] and the source codes in C/C++ can be downloaded from the web-page [Hrstka, WWW].

A parameter setting of the SADE algorithm remains unchanged for all our computations (Section 4.1.4 results as well as [Hrstka and Kučerová, 2004] comparison) and it is shown in Table 4.1.

[1] Contrary to the binary genetic algorithm the real encoded method may generate chromosomes outside the given domain. In our implementation, this problem is solved by returning these individuals to the feasible domain boundary.

[2] Contrary to the traditional tournament strategy, this approach can ensures that the best chromosome will not be lost even if it was not chosen to any tournament.

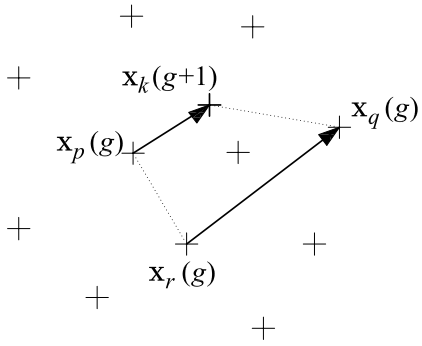


Figure 4.1: Geometrical meaning of simplified differential operator in SADE algorithm

Parameter	Value
pop_rate	10
CR	0.2
MR	0.5
local mutation range	0.25%
$radioactivity$	0.2

Table 4.1: Parameter setting for SADE algorithm

4.1.2 GRADE algorithm

The modifications of the SADE algorithm have two principal motivations:

- To increase the convergence rate (i.e. the speed of convergence) of the algorithm for smooth objective functions with just one optimum;
- To reduce the number of external parameters of the algorithm and their influence on the algorithm's behavior, because their values are usually set up simply by trial-and-error method.

The GRADE algorithm has the same scheme as the SADE algorithm except the following modifications:

- Elimination of the "local mutation" operator;
- The parameter MR is no more constant, but for each new chromosome created by mutation is randomly chosen from interval $\langle 0, 1 \rangle$;
- The relation defining the crossing-over operator is modified in the following way:

$$\mathbf{x}_k(g+1) = max(\mathbf{x}_q(g); \mathbf{x}_r(g)) + CR(\mathbf{x}_q(g) - \mathbf{x}_r(g)). \tag{4.4}$$

Only two chromosomes $\mathbf{x}_q(g)$ and $\mathbf{x}_r(g)$ are randomly chosen from the current population. Vector of their difference is reduced by CR parameter and contrarily to SADE algorithm added to the better one of the two chromosomes.

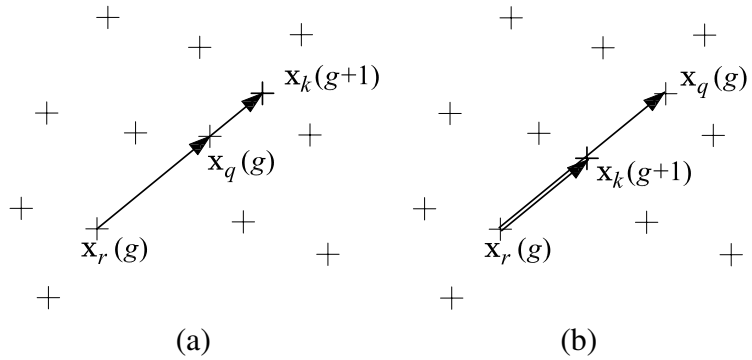


Figure 4.2: Geometrical meaning of simplified differential operator in GRADE algorithm

Figure 4.2 shows two possible geometrical meanings of this operator: Figure 4.2a for the case, that chromosome $\mathbf{x}_q(g)$ is better than $\mathbf{x}_r(g)$ and Figure 4.2b for the case, that chromosome $\mathbf{x}_r(g)$ is better than $\mathbf{x}_q(g)$.[3] It could be remarkable that by this operator, a new chromosome is created either somewhere between its parents (see Figure 4.2a) or out of them but in the direction of the better one (i.e. in the sense of a numerical gradient, see 4.2b). We have tried also the possibility of cross-over operator, where the new chromosome was created only between its parents or only in the sense of gradient. Both these variants achieve much worse results when solving the set of twenty test problems reported below.

– The CR parameter is no more constant, but randomly chosen from interval $\langle 0, CL\rangle$, where CL is new parameter of the algorithm, but its influence is supposed to be lower than that one of the CR parameter.

Finally, the GRADE algorithm has only three parameters: $pool\_rate$, $radioactivity$ and CL parameter. The $pool\_rate$ parameter was in all previous applications always set to 10, because this value was already approved either by differential evolution or by SADE algorithm. Therefore, we were especially interested in an influence of $radioactivity$ and CL parameter values. We have tested different settings of these parameters for solving twenty mathematical functions taken from [Andre et al., 2000] and used also in the article [Hrstka and Kučerová, 2004] for comparison of the SADE algorithm with differential evolution and two other binary genetic algorithms. Definitions of these functions are presented in Appendix A. The algorithm was started 100 times for each objective function and the number of objective function calls were counted till the optimum was found with given accuracy.

[3] It is worth mentioning that both presented algorithms are implemented as algorithms for maximization, therefore better chromosome is the one with higher value of objective function.

The first comparison stored in Table 4.2 shows the number of objective function, where the algorithm with the given setting was fastest (i.e. had the highest convergence rate or the lowest number of objective functions calls).

Convergence rate						
		$radioactivity$				
		0.1	0.2	0.3	0.4	0.5
CL	0.1	0	0	1	1	1
	0.3	0	0	0	0	0
	0.5	1	2	2	2	2
	1.0	1	1	2	1	1
	2.0	2	0	0	0	0

Table 4.2: Comparison of number of objective functions, where GRADE algorithm was fastest for given values of CL parameter and $radioactivity$

Another comparisons are presented in Tables 4.3a and 4.3b. This one is focused on the reliability of the algorithm, because sometimes and only for some objective functions the algorithm did not find the optimum before 2000000 function calls. Such case was considered as a failure. The number of objective functions where the reliability of the algorithm was higher than 95% or 99%, respectively (i.e. the algorithm failed in less than 5% or 1% runs, respectively) is shown in Tables 4.3a and 4.3b, respectively.

Reliability $>$ 95%						
		$radioactivity$				
		0.1	0.2	0.3	0.4	0.5
CL	0.1	16	15	17	17	17
	0.3	17	17	17	17	17
	0.5	17	17	17	17	17
	1.0	18	18	18	18	17
	2.0	18	18	18	18	18

(a)

Reliability $>$ 99%						
		$radioactivity$				
		0.1	0.2	0.3	0.4	0.5
CL	0.1	14	15	15	15	15
	0.3	14	13	15	16	16
	0.5	14	17	17	16	17
	1.0	17	18	17	17	16
	2.0	16	18	18	17	17

(b)

Table 4.3: Comparison of number of objective functions, where GRADE algorithm found optimum in more than (a) 95% or (b) 99% cases for given values of CL parameter and $radioactivity$

These studies are presented to give the reader an idea about the meaning and influence of the $radioactivity$ and CL parameter and it should be helpful for the choice of appropriate values of these parameters for a given objective function.

Table 4.2 shows, that a better convergence rate was obtained especially for $CL = 0.5$ or for higher values of CL parameter with smaller values of $radioactivity$. Contrarily, Tables

4.3a and 4.3b show that better reliability of the algorithm is achieved for higher values of CL parameter and $radioactivity$ in the range from 0.2 to 0.3.

The GRADE algorithm tends to create a cluster of individuals at a limited sub-area that moves through the domain. As a consequence, if the cluster is deadlocked in a local extreme, it is necessary to wait until the mutation gives a chance to escape to another sub-area with better values. Of course, the probability of this effect is not high and it highly depends on the $radioactivity$. Also the speed of convergence to such a cluster depends on CL parameter.

As a simple conclusion we can say that for highly multi-modal functions (i.e. with high number of local extremes) it is better to set higher value of CL parameter and $radioactivity$ around 0.2 or 0.3, otherwise $CL = 0.5$ could be used for functions with lower number of local extremes to increase the speed of convergence of the algorithm.

A parameter setting of GRADE algorithm for our computations presented in Chapter 4.1.4 is shown in Table 4.4.

Parameter	Value
pop_rate	10
CL	1.0
$radioactivity$	0.2

Table 4.4: Parameter settings for GRADE algorithm

4.1.3 CERAF strategy

As already shown in the previous section, the GRADE algorithm tends to create clusters of chromosomes. This behavior somehow resembles gradient optimization methods, however, with several differences: First, it operates with more than one possible solution at a time, therefore it is able to better locate the sub-area with the desired solution. Secondly, since the changes of individuals are determined from their mutual distances, this method is able to adapt the step size to reach an optimal solution.

However, each time this method is caught in a local extreme, it has no chance to escape unless a mutation randomly finds a sub-area with better values. But the probability of this effect is small, especially for the high-dimensional problems. If the gradient optimization methods are applied, this case is usually resolved by so-called *multi-start* principle. It consists of restarting the algorithm many times with different starting points. Similarly, any type of a genetic algorithm could be restarted many times. Nevertheless, the experience shows that there are functions with so-called deceptive behavior, characterized by a high probability that the restarted algorithm would fall again into the same local extreme rather than focus on another sub-area.

Generally speaking, there are several solutions to this obstacle. All of them are based on the leading idea of preventing the algorithm from being trapped in the local extreme that has

been already found and to force the algorithm to avoid all of these. As the most natural way, we tried some penalization that deteriorates the objective function value in the neighborhood of all discovered local extremes. When the shape of a penalization function is not determined appropriately, new local extremes appear at the boundary of a penalization function activity area. As an alternative, so called niching strategies are proposed to overcome the multi-modality of the objective functions, see [Mahfoud, 1995a] or [Mahfoud, 1995b].

In [Hrstka and Kučerová, 2004], one particular niching strategy, the CERAF[4] method was proposed. It produces areas of higher level of "radioactivity" in the neighborhood of all previously found local extremes by increasing the mutation probability (i.e. $ceraf\_radioacitivity$) in these areas many times (usually we set this probability directly to 100%). The radius of the radioactivity area (an *n*-dimensional ellipsoid) is set to a certain percentage of the domain – we denote it as RAD. The time of stagnation that precedes the markup of a local extreme and the initiation of a radioactive zone are another parameters of the method. The $quiet$ parameter define the number of generations, while the best found value has not changed more than half of a $precision$ parameter defined for a given objective function.

Similarly to the living nature, the radioactivity in the CERAF method is not constant in time but decreases in an appropriate way: each time some individual is caught in that zone and mutated, the radioactivity zone range is decreased by a small value (in our implementation, the radius of the radioactive zone is multiplied by the parameter called $deact\_rate$); this recalls the principle of disintegration of a radioactive matter. During the numerical experiments it turned up that the chromosomes created by the mutation parameter should not affect the radioactivity zone range. The radioactive area never disappears completely, so the chromosomes can never find the marked local extreme again.

Hereafter, the algorithmic scheme of the GRADE method is supplied with several steps of the CERAF method. It determines whether some individuals got into any of the discovered "radioactive zones" and if so, mutates them with a high level of probability. Moreover, when the algorithm stagnates too long, it declares a new radioactivity area:

1. As the first step, the initial population is generated randomly and the objective function value is assigned to all chromosomes in the population.

2. Several new chromosomes are created using the mutation operator (their total number depends on the value of a parameter called $radioactivity$ – it gives the mutation probability).

3. Another new chromosomes are created using the modified differential operator; the whole amount of chromosomes in the population is now doubled.

4. If any radioactive zone already exists, each chromosome caught in a radioactive area is, with a high probability depending on parameter called $ceraf\_radioactivity$, subjected to the mutation operation.

[4] Abbreviation of the French expression *CEntre RAdioactiF* - the radioactivity center.

5. Depending on the number of chromosomes created by cross-over operator and simultaneously determined in the previous step, the ranges of radioactive zones are appropriately decreased.

6. The objective function values are assigned to all newly created chromosomes.

7. The selection operator is applied to the double-sized population. Hence, the amount of individuals is decreased to its original value.

8. The number of stagnating generations is determined and if it exceeds a given limit, the actual best solution is declared as the center of the new radioactive area.

9. Steps 2-8 are repeated until a stopping criterion is reached.

Extensive test computations have shown that this methodology can be considered as a universal technique capable of solving any multi-modal optimization problem provided that the method that is running underneath (i.e. the algorithm that generates new chromosomes) has a sufficient ability to find new possible solutions. In our case, the GRADE algorithm works as the "exploration" method.

For the purpose of the algorithm performance testing presented in the following section, the CERAF method parameters were set to values stored in Table 4.5.

Parameter	Value
$ceraf\_radioacitivity$	1.0
RAD	0.25
$deact\_rate$	0.995
$quiet$	100

Table 4.5: Parameter setting for GRADE algorithm

4.1.4 Comparison of proposed genetic algorithms

In this section we present a comparison of the proposed methods. All of them were started 1000 times on 20 mathematical functions and the number of successful runs[5] and the average number of function calls necessary to find the optimum with a given precision are listed in Table 4.6.

The overall reliability-based comparison is given in Table 4.7 (values represent the number of objective functions, where the algorithm achieved more than 95% or 99% successful runs).

Table 4.8 shows the comparison of all methods from the convergence rate point of view ($\times$ marks the cases, where the method reached the optimum at the shortest time).[6]

[5] Runs, where the optimum was found before $2,000,000$ function calls

[6] If the difference among results of algorithms were smaller than 5%, $\times$ mark was distributed to all of them.

Test function	N	SADE		GRADE		GRADE+CERAF	
		SR %	ANFC	SR %	ANFC	SR %	ANFC
F1	1	100.0	61	100.0	61	100.0	60
F3	1	100.0	87	100.0	97	100.0	94
Branin	2	100.0	668	100.0	371	100.0	368
Camelback	2	100.0	306	100.0	223	100.0	222
Goldprice	2	100.0	634	100.0	360	100.0	358
PShubert1	2	100.0	1518	100.0	5501	100.0	1844
PShubert2	2	100.0	1043	100.0	1403	100.0	970
Quartic	2	100.0	534	100.0	341	100.0	339
Shubert	2	100.0	682	100.0	649	100.0	654
Hartman1	3	100.0	478	100.0	319	100.0	320
Shekel1	4	100.0	7719	100.0	33776	100.0	3434
Shekel2	4	100.0	4595	100.0	13522	100.0	2638
Shekel3	4	100.0	4127	100.0	10857	100.0	2650
Hartman2	6	71.2	57935	60.8	165622	100.0	10284
Hosc45	10	100.0	7759	100.0	2265	100.0	2274
Brown1	20	91.1	160515	100.0	209214	100.0	195250
Brown3	20	100.0	60554	100.0	36339	100.0	36429
F5n	20	94.4	26786	99.8	7197	100.0	7259
F10n	20	66.4	227577	70.3	90687	98.2	289702
F15n	20	97.5	48533	99.4	23358	100.0	24894

Table 4.6: Comparison of results of investigated methods. SR = success rate, ANFC = average number of function calls, N = number of variables

Reliability	SADE	GRADE	GRADE+CERAF
$> 95\%$	16	18	20
$> 99\%$	15	18	19

Table 4.7: Overall reliability-based comparison of investigated methods.

Several interesting facts are immediately evident when comparing these results:

i) Comparing the results in Table 4.8 with the graphical illustration of test functions presented in Appendix A.2 we can conclude that for smooth objective functions with one or just several local extremes the GRADE algorithm achieved better convergence rate than the SADE algorithm. Number of GRADE algorithm parameters is smaller than of the SADE algorithm, so we can note, that all motivations for creating the GRADE algorithm were fulfilled.

ii) The single GRADE algorithm achieved success rate more than 99% for 18 test functions, while the SADE algorithm achieved such success only for 15 test functions. Hence, it is

Test function	SADE	GRADE	GRADE+CERAF
F1	×	×	×
F3	×		
Branin		×	×
Camelback		×	×
Goldprice		×	×
PShubert1	×		
PShubert2			×
Quartic		×	×
Shubert	×	×	×
Hartman1		×	×
Shekel1			×
Shekel2			×
Shekel3			×
Hartman2			×
Hosc45		×	×
Brown1	×		
Brown3		×	×
F5n		×	×
F10n		×	
F15n		×	
Summary	5	12	15

Table 4.8: Comparison of convergence rate

possible to designate the GRADE algorithm as a better algorithm also from the reliability point of view.

iii) The GRADE algorithm extended by the CERAF strategy has achieved the 100% success for all function except for F10n function, where it has achieved a success in 98.2% of runs.

iv) In 10 cases the number of function calls is approximatively the same for the single GRADE algorithm as for the GRADE algorithm extended by the CERAF strategy; in those cases the CERAF technology was not even activated because the simple algorithm found the global extreme itself.

v) For the *F10n* function the success has been significantly improved from 70.3% to 98.2%, however, at the cost of slowing down the convergence. Indeed, we consider the reliability of the method of greater value than the speed. These are the cases where the algorithm extended by the CERAF method was able to continue searching even after the previous simple method has been caught in a local extreme hopelessly.

vi) In 5 cases the computation of the GRADE algorithm was even accelerated by the CERAF method, while the reliability was not decreased; in two particular cases (the *Hartman 2*

and $F5n$ function) the reliability was even increased from 60.8% and 99.8% to 100%, respectively. This may appear as a paradox, because the CERAF method needs long periods of stagnation and repeated optimum searching. The acceleration comes from the fact that the method does not have to wait until the random mutation hits an area with better values, but it is forced to start searching in a different location.

Note: According to [Wineberg and Christensen, 2007] we have tried more reliable statistical comparison than only that one based on mean values of function calls. The authors propose to take at least the variance into account or better to use e.g. the Student t test of hypotheses of equal means of two normal distributions. Nevertheless, Figure 4.3 shows histograms of number of function calls obtained from 1000 runs of GRADE algorithm for several test functions and it is clearly visible that number of function calls are far from being normally distributed. For such case, the authors propose a non-parametric test based on ranked data[7] *under the condition that the ranks are already normally distributed. We have tried to calculate the ranks for first objective function and we applied Jarque-Bera test of hypothesis, if a sample comes from a normal distribution with unknown mean and variance. Already for this first objective function the test has shown than even ranks are not normally distributed. Therefore even non-parametric test cannot be applied for our comparison of the number of function calls.*

Moreover it seems that for different test functions or different optimization algorithms we can obtain different distributions of the number of function calls and therefore neither confidence interval nor variance can be applied to our results.

4.2 Radial Basis Function Network

In Section 2 we presented several methods for forward mode of an inverse analysis. As examples of one type of these methods, two genetic algorithms were described in previous Section with an extension by niching strategy to increase their reliability and robustness. As a big disadvantage of proposed genetic algorithms, nevertheless, remains high number of objective functions evaluations needed to find a solution. Especially for rather smooth functions with only a limited number of extremes, genetic algorithms will be too expensive in comparison with gradient methods. In this section we propose one interpolation method to reduce number of objective function evaluations based on interpolation of error function as described in Section 2.2. A particular implementation applies radial basis function network (RBFN) as proposed e.g. in [Nakayama et al., 2004, Karakasis and Giannakoglou, 2004].

The need for interpolation is crucial here, because new model $\tilde{M}$, except for already computed values $\tilde{Y}^M(\tilde{X}^M) \equiv Y^M(X^M)$, does not correspond to the model M and therefore the model $\tilde{M}$ is assumed to be unreliable. To obtain a sufficiently accurate interpolation of the model M, an iterative process is usually used: starting with $\tilde{M}_k$ from known values (pairs X_k^M,Y_k^M), the minimum $\tilde{X}^M$ of the error function $f(X) = \|Y^E - \tilde{M}_k(X)\|$ is found using

[7] Two other techniques with similar results are commonly seen: Wilcoxon's Rank-Sum test or Mann-Whitney U test. All are nearly equivalent and the test is often called the "Mann-Whitney-Wilcoxon test" by statisticians.

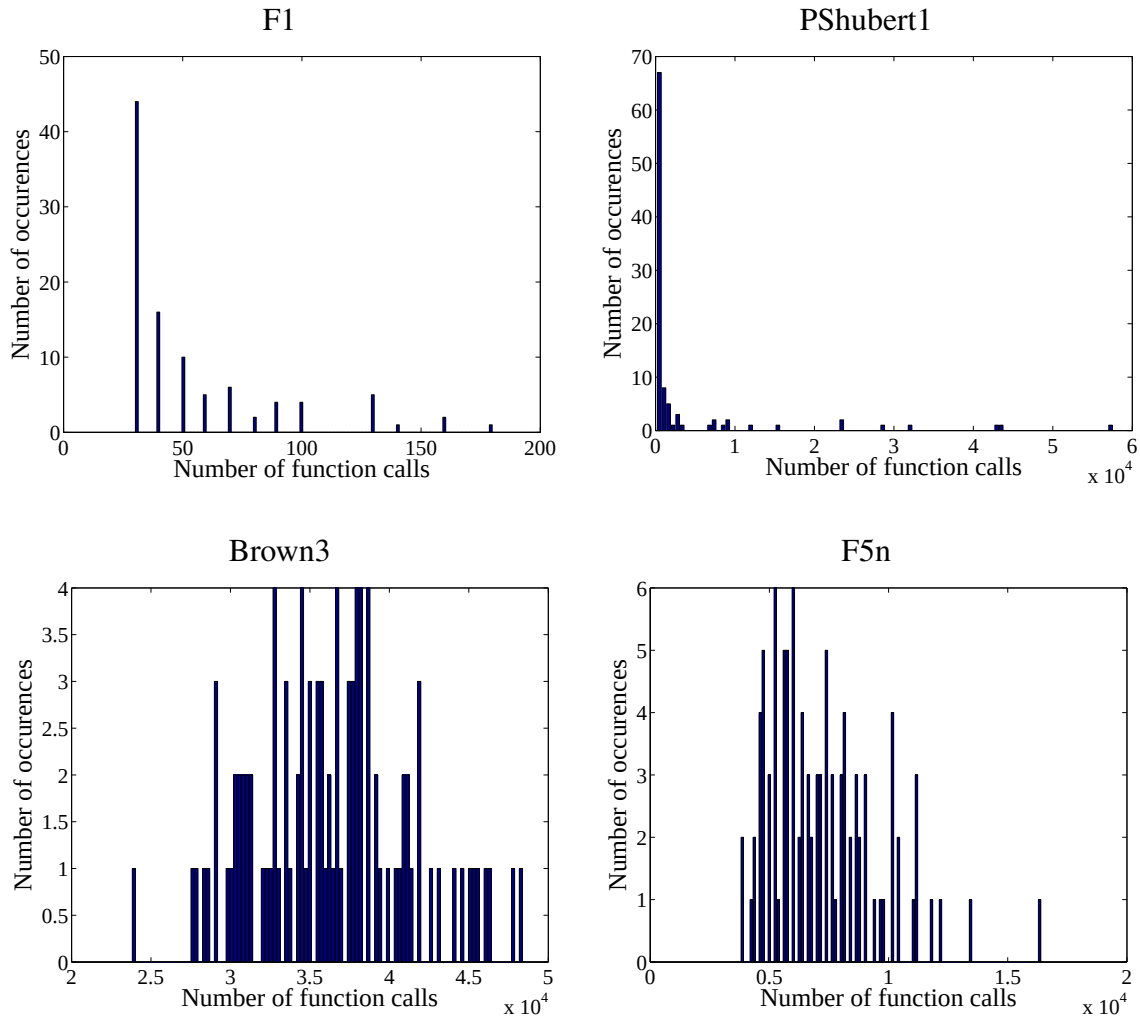


Figure 4.3: Histograms of number of function calls obtained from 1000 runs of GRADE algorithm

multi-modal optimization (this step should by computationally inexpensive thanks to the cheap model $\tilde{M}$), correct values of the model M are computed by $Y_{k+1}^M = M(\tilde{X}^M)$ and these new values are added to the set of already computed pairs. This procedure is repeated until the minimum of the error function $F(X) = \|Y^E - M(X)\|$ is reached. A subscript k is used to describe a number of iterations. Finally, the problem is to minimize a number of evaluations of the expensive model M. Individual iterative procedures proposed by different authors, see e.g. [Nakayama et al., 2004, Kucerova et al., 2007], differ in the details of how the algorithm is implemented.

Our implementation of RBFN is based on an analogy with artificial neural networks, but differs in several points: RBFN is created only with one layer of neurons, it has a specific type of a transfer function and the training of this network leads to the solution of a linear system of equations. The novelty in our approach is the use of the evolutionary algorithm GRADE described in previous Section to find the global maximum of the RBFN interpolation.

RBFN replace the objective function $F(\mathbf{x})$ by an interpolation $\tilde{F}(\mathbf{x})$ defined as a sum of radial basis functions multiplied by synaptic weights, see Figure 4.4, or

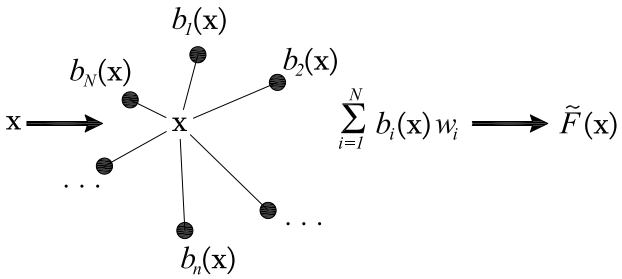


Figure 4.4: An interpolation using RBFN

$$F(\mathbf{x}) \approx \tilde{F}(\mathbf{x}) = \sum_{i=1}^{N} b_i(\mathbf{x}) w_i \ , \tag{4.5}$$

where $\mathbf{x}$ is a vector of unknowns, $b_i(\mathbf{x})$ is a basis function associated with the i-th neuron, w_i is a weight of the i-th neuron and N is the total number of neurons creating the network. The basis function b_i has the most often used "Gaussian" shape given by

$$b_i(\mathbf{x}) = e^{-\|\mathbf{x}-\mathbf{c}_i\|^2/r} \ , \tag{4.6}$$

where $\mathbf{c}_i$ is a vector of the center coordinates for the i-th basis function and r is a normalizing factor set to

$$r = \frac{d_{max}}{\sqrt[D]{DN}} \ , \tag{4.7}$$

where d_{max} is the maximal distance within the domain and D is the number of dimensions.

Synaptic weights are computed from the condition

$$F(\mathbf{c}_i) = \tilde{F}(\mathbf{c}_i), \tag{4.8}$$

imposing the equality of the interpolation $\tilde{F}$ and objective function F values $\overline{y_i}$ in all neurons. This leads to a minimization problem in the form:

$$\min \sum_{i=1}^{N} [(\overline{y_i} - \sum_{j=1}^{N} b_j(\mathbf{c}_i) w_j)^2 + \lambda_i w_i^2] \ . \tag{4.9}$$

where λ_i is a regularization factor set to 10^{-7}. The solution of (4.9) leads to a system of linear equations determining the values of synaptic weights $\mathbf{w}$.

At this point, the RBFN interpolation of the objective function is created and the above-mentioned evolutionary algorithm GRADE is used to locate the 'approximate' global optima. In the next step, to improve the quality of interpolation, three new points are added into the neural network:

1. The optimum of previous interpolation,
2. A random point
3. Another point in the descent direction defined by optima found in two previous steps.

Some other methods of the choice of new points were tested in [Kučerová et al., 2005]. The detailed description of individual steps of proposed procedure follows:

1. Create initial neurons[8]
2. Compute parameters of basis functions d_{max} and r,
3. Repeat next steps until stopping criteria are met
4. Compute objective function values in new neurons
5. Compute values of basis functions $b_i(\mathbf{c}_i)$ and output weights w_i,
6. Find a maximum using the evolutionary algorithm GRADE,
7. Add three new points using the differential method.

A more detailed description of the method is available in [Kučerová et al., 2005].

To show the abilities of the proposed method, another comparison is presented in Table 4.9. The same set of twenty multi-modal functions was used again as in previous section. The complete list of these functions can be found in Appendix A. The proposed algorithm is compared with previously described genetic algorithm GRADE and its combination with CERAF strategy.

Again, the statistics over one thousand runs was computed to tackle random circumstances. The optimization process was stopped after 300 cycles (equals to 900 evaluations + evaluation of initial neurons) or if the maximum was found with the given precision. Otherwise, the optimization run was marked as unsuccessful.

To conclude this section we present several remarks:

i) From the Table 4.9 it is obvious that the proposed combination of RBFN and evolutionary algorithm cannot be aimed at optimization of multi-modal functions (i.e. F3, PShubert1–2, Shubert, Shekel1–3), because the reliability for these cases is very low.

[8] In our implementation we choose initial neurons in all corners of given domain for objective function plus one neuron is chosen in the center of the domain.

Test function	N	GRADE		GRADE+CERAF		GRADE+RBFN	
		SR %	ANFC	SR %	ANFC	SR %	ANFC
F1	1	100.0	61	100.0	60	100.0	23
F3	1	100.0	97	100.0	94	96.7	159
Branin	2	100.0	371	100.0	368	100.0	43
Camelback	2	100.0	223	100.0	222	100.0	61
Goldprice	2	100.0	360	100.0	358	11.6	472
PShubert1	2	100.0	5501	100.0	1844	2.1	466
PShubert2	2	100.0	1403	100.0	970	2.5	530
Quartic	2	100.0	341	100.0	339	100.0	77
Shubert	2	100.0	649	100.0	654	18.0	506
Hartman1	3	100.0	319	100.0	320	99.9	63
Shekel1	4	100.0	33776	100.0	3434	0.0	-
Shekel2	4	100.0	13522	100.0	2638	0.0	-
Shekel3	4	100.0	10857	100.0	2650	0.0	-
Hartman2	6	60.8	165622	100.0	10284	97.7	163

Table 4.9: Comparison of results of investigated methods. SR = success rate, ANFC = average number of function calls, N = dimension of the problem

ii) On the other hand, the ability of the proposed algorithm to solve problems with a smaller number of local minima and rather smooth shape (i.e. F1, Branin, Camelback, Quartic, Hartman1–2) is remarkable.

iii) The proposed methodology in our particular implementation is also not well-suited to optimization of high-dimensional objective function, because the number of initial neurons is defined as

$$N = 2^D + 1, \tag{4.10}$$

where N is the total number of neurons and D is the number of dimensions. Then, for e.g. $D = 10$ the initial number of neurons N is equal to 1024 and solving corresponding system of 1024 linear equation should not be already negligible comparing to computational time necessary to evaluate the objective function. Therefore we calculated the comparative statistics only for first 13 objective functions with number of dimensions smaller or equal to six.

iv) Other computations has shown that the CERAF strategy brings no effect to the combination of RBFN and GRADE algorithm.

Chapter 5

INVERSE MODE METHODS

> Simplicity is the most deceitful
> mistress that ever betrayed man.
>
> Henry Adams

The philosophy of an inverse mode assumes existence of an inverse relation between outputs and inputs, i.e. there is an "inverse" model M^{INV} associated to the model M, which fulfils the equation (3.1). The main advantage is the possibility to find desired inputs for new measured outputs repeatedly by a simple and cheap evaluation of M^{INV}. On the contrary, the main disadvantage is an exhausting search for the inverse relationship. Nowadays, artificial neural networks have became the most frequently used tools applicable in inverse model determination.

The Chapter 3 contains a brief description of several methods appropriate for inverse model determination including artificial neural networks, design of experiments and also some notes concerning the ANN's training.

In this chapter we will describe the implementation of a multi-layer perceptron in parameters identification.

Individual steps of the identification procedure involve:

Step 1 *Setup of* a virtual and/or real experimental *test* used for the identification procedure and saving the measurements $\mathbf{y}^E$.

Step 2 Formulation of an appropriate *computational model* M. Input data to the model coincide with the parameters to be identified.

Step 3 *Randomization* of input parameters. Input data are typically assumed to be random variables uniformly distributed on a given interval. A representative set of input vectors $\mathbf{X}^M_{Train} = \left(\mathbf{x}^M_1, \mathbf{x}^M_2, \ldots, \mathbf{x}^M_{ntr}\right)$ is carefully chosen for ANN training following design of experiments methodology. Another set of input vectors $\mathbf{X}^M_{Test} = \left(\mathbf{x}^M_1, \mathbf{x}^M_2, \ldots, \mathbf{x}^M_{nte}\right)$ is randomly chosen for ANN testing. ntr and nte denote the number of training and testing samples, respectively.

Step 4 *Training and testing data* sets preparation. The computation model M is applied to simulate the experiment E for all training and testing input vectors in order to obtain corresponding output vectors $\mathbf{Y}^M_{Train} = \left(\mathbf{y}^M_1, \mathbf{y}^M_2, \ldots, \mathbf{y}^M_{ntr}\right)$ and $\mathbf{Y}^M_{Test} = \left(\mathbf{y}^M_1, \mathbf{y}^M_2, \ldots, \mathbf{y}^M_{nte}\right)$, respectively, where $\mathbf{y}^M_i = M(\mathbf{x}^M_i)$.

Step 5 Stochastic *sensitivity analysis* using the Monte Carlo-based simulation. This provides us with *relevant model parameters* which can be reliably identified from the computational simulation. Usually this step is performed by calculation the correlation between inputs $\mathbf{X}^M_{Train}$ to the computation model and corresponding outputs $\mathbf{Y}^M_{Train}$.

Step 6 Definition of *topology* of an ANN used for the identification procedure.

Step 7 *Training* of the ANN, i.e. developing of M^{INV}. Some optimization algorithm is applied to appropriately setup values of synaptic weights of ANN by minimizing the error function (3.4) for training pairs $(\mathbf{x}, \mathbf{y})^M \in (\mathbf{X}, \mathbf{Y})^M_{Train}$.

Step 8 *Verification I* of the ANN with respect to the computational model. This step is usually performed by comparing the ANN's prediction of model input parameters $\tilde{\mathbf{X}}^M_{Test} = \left(\tilde{\mathbf{x}}^M_1, \tilde{\mathbf{x}}^M_2, \ldots, \tilde{\mathbf{x}}^M_{nte}\right)$, where $\tilde{\mathbf{x}}^M_i = M^{INV}(\mathbf{y}^M_i)$, with the original one $\mathbf{X}^M_{Test}$ for unseen testing (or reference) data.

Step 9 *Verification II* of the ANN with respect to the computational model. In this step, a computation model should be evaluated for predicted values $\tilde{\mathbf{X}}^M_{Test}$ in order to obtain corresponding model outputs $\tilde{\mathbf{Y}}^M_{Test}$. Then the outputs $\tilde{\mathbf{Y}}^M_{Test}$ could be compared with the original one $\mathbf{Y}^M_{Test}$. This step is not necessary, but is utmost recommended.

Step 10 *Validation* of the ANN with respect to the experiment. Trained ANN M^{INV} is evaluated for experimental data $\mathbf{y}^E$ in order to obtain corresponding input values $\tilde{\mathbf{x}}^E = M^{INV}(\mathbf{y}^E)$ to the computation model M. The model M is then evaluated for obtained inputs $\tilde{\mathbf{x}}^E$ and results $\tilde{\mathbf{y}}^E = M(\tilde{\mathbf{x}}^E)$ are compared with original measured data $\mathbf{y}^E$.

For clarity, the scheme of such identification procedure is displayed in Figure 5.1, which could be easily compared with schemes 2.1 and 2.2.

The following sections contain a detailed description of several methods suitable for inverse mode of identification.

5.1 Multi-layer perceptron

Multi-layer perceptron is a particular type of artificial neural network, which is usually applied for the parameter identification of non-linear mechanical models [Tsoukalas and Uhrig, 1997]. More precisely, we use a fully connected two-layer perceptron with bias neurons, see Figure 5.2. In accordance with the universal approximation theorem, such neural network can approximate any continuous function.

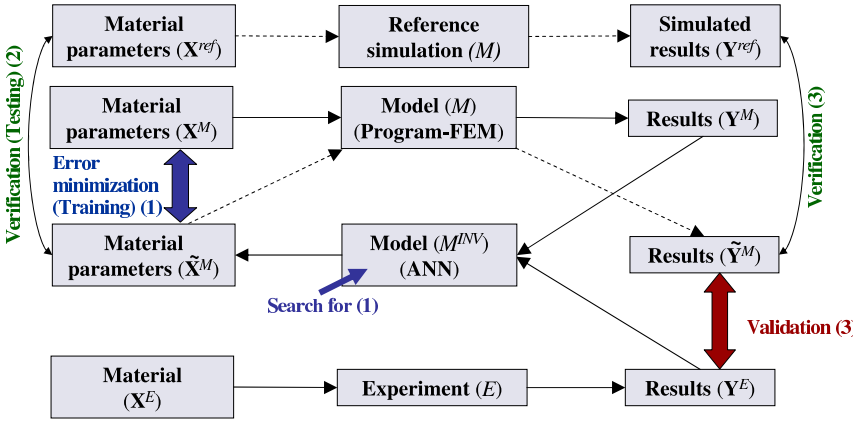


Figure 5.1: Scheme of inverse analysis procedure

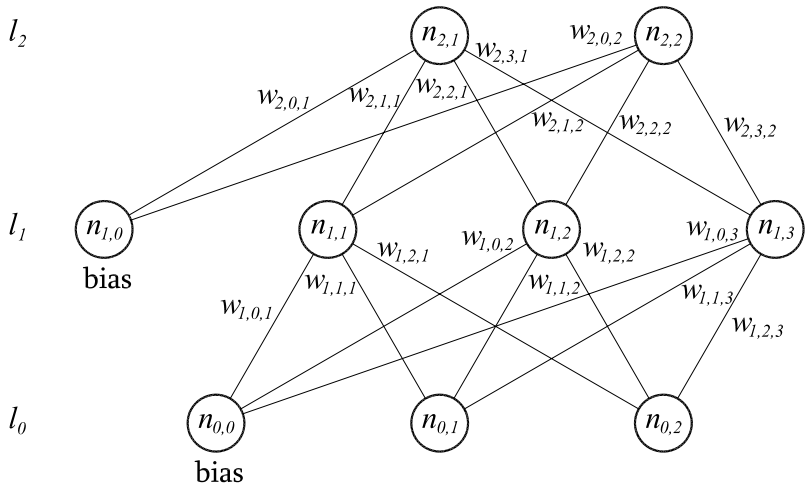


Figure 5.2: Neural network architecture

The neural network is created to map the input vector, i.e. model output parameters $\mathbf{y}^M = (y_1, y_2, \dots y_{N_0})$ on a target vector, i.e. model input parameters $\mathbf{x}^M = (x_1, x_2, \dots x_{N_L})$. There are $L = 2$ layers denoted as l_1, l_L, where l_1 is one hidden layer and l_L is the output layer. l_0 represent the ANN's inputs and it is not counted in the number of ANN's layers. The i-th layer l_i has N_i neurons denoted as $n_{i,1}, n_{i,2}, \dots n_{i,N_i}$. Each layer except the output layer has the bias neuron $n_{i,0}$. The connections are described by the weights $w_{l,i,j}$, where $l = 1, 2 \dots L$ denotes a layer, $i = 0, 1 \dots N_{l-1}$ is the index number of a neuron in the preceding layer $l - 1$ ($i = 0$ for bias neurons) and $j = 1, 2 \dots N_l$ is the index number of a neuron in the layer l. The output $O_{l,j}$ of the neuron $n_{l,j}$ is then defined as

$$O_{l,j} = f_{\text{act}} \left(\sum_{i=0}^{N_{l-1}} O_{l-1,i} \cdot w_{l,i,j} \right), \; l = 1, 2 \dots L, \; j = 1, 2 \dots N_l \,, \tag{5.1}$$

$$O_{0,j} = y_j^M, \; j = 1, 2 \dots N_0 \,, \tag{5.2}$$

$$O_{l,0} = 1, \; l = 0, 1 \dots L - 1 \,, \tag{5.3}$$

where f_{act} is an activation function. In our implementations we use the log-sigmoid activation function, which has the following form:

$$f_{\text{act}}(\Sigma) = \frac{1}{(1 + e^{-\alpha\Sigma})}, \tag{5.4}$$

where α is the gain of the f_{act}. The value $\alpha = 0.5$ is used in all reported calculations. The impact of α parameter is visible in Figure 5.3. Finally, the neural network is propagated as follows:

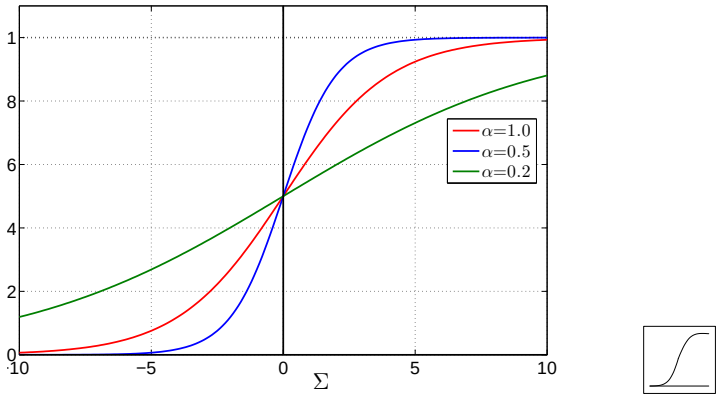


Figure 5.3: Log-sigmoid activation function

1. Let $l = 1$.
2. Calculate $O_{l,i}$ for $i = 1, 2 \ldots N_l$.
3. $l = l + 1$.
4. If $l < L$ go to 2, else $\mathbf{O}_L = \tilde{\mathbf{x}}^M$ is the network's approximation of target vector $\mathbf{x}^M$.

5.2 Training algorithms

Behavior of a neural network is determined by a preceding training process. It consists of finding the synaptic weights, which have influence on the response of a neural network, depending on the different components of an input signal. The training of a neural network itself could be considered as an optimization process, because it can be seen as a minimization of neural network output error defined as

$$E(w_{l,i,j}) = \frac{1}{2} \sum_{i=1}^{N_L} (x_i - \tilde{x}_i)^2 . \tag{5.5}$$

The most often used training algorithm for feed-forward layered neural networks is back-propagation algorithm. Other possibility is e.g. to use a genetic algorithm. One comparison of back-propagation training algorithm and training by genetic algorithm SADE described in Section 4.1.1 was presented in [Drchal et al., 2003].

5.2.1 Back-propagation training

More specifically, the momentum [Tsoukalas and Uhrig, 1997] version of back-propagation was used to speed up the convergence. A short description of the method is following. Note, that the error connected with the output of the neuron $n_{l,i}$ is denoted as $e_{l,i}$. The algorithm could be written in following form:

1. for $i = 1, 2 \ldots N_{L-1}$ calculate $e_{L-1,i} = \alpha \,.\, O_{L-1,i} \,.\, (1 - O_{L-1,i}) \,.\, (T_i - O_{L-1,i})$. Set $l = L - 1$.
2. $e_{l-1,i} = \alpha \,.\, O_{l-1,i} \,.\, (1 - O_{l-1,i}) \,.\, \left(\sum_{j=1}^{N_l} w_{l,i,j} \,.\, e_{l,j} \right)$ for each neuron i in $(l-1)$th layer.
3. $l = l - 1$.
4. While $l > 1$ go to 2.
5. All the weights are adjusted: $w_{l,i,j} = w_{l,i,j} + \Delta w_{l,i,j} + \mu \,.\, \Delta w'_{l,i,j}$, where $\Delta w_{l,i,j} = \eta \,.\, e_{l,j} \,.\, O_{l-1,i}$. The term $\Delta w'_{l,i,j}$ stands for the value from the previous training iteration, η is the learning constant, and μ is the momentum coefficient. The typical values were $\eta = 0.5$ and $\mu = 0.9$.

5.2.2 Comparison of back-propagation training and SADE algorithm training

The performance of two proposed algorithms was tested on a following simple problem. The function

$$f(x) = ax \, \sin(bx) + c \tag{5.6}$$

was used to generate a sequence of points $x_1, x_2, \ldots x_n$:

$$\begin{aligned} x_1 &= \text{a random number from training interval}\,, \\ x_i &= x_{i-1} + d\,, \;\; i = 2, 3, \ldots n\,, \;\; d \text{ is a constant}\,, \end{aligned} \tag{5.7}$$

i.e., all the points are equidistant and the sequence starts in a random point. The network input vector was created as $I = (f(x_1), f(x_2), \ldots f(x_{n-1}))$, the next sequential point $f(x_n)$ was expected on the output. Typically, two input points ($n = 3$) was used. The situation is depicted in Figure 5.4.

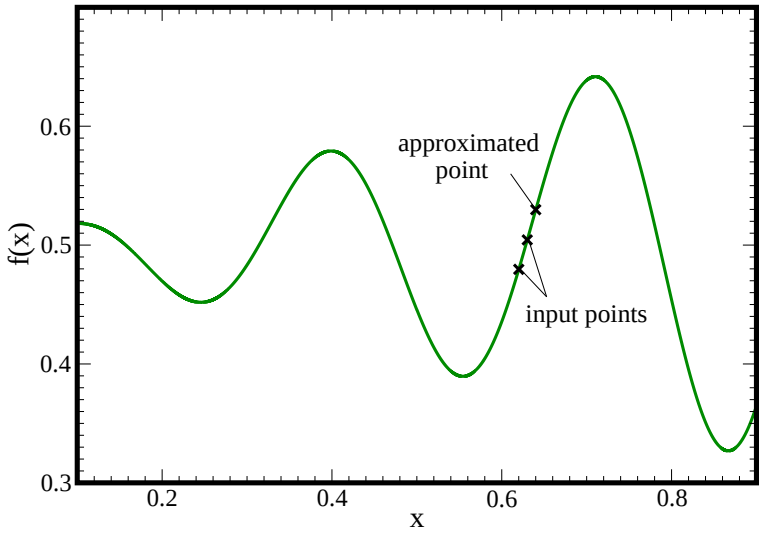


Figure 5.4: A function used for testing: $f(x) = 0.2x\ \sin(20x) + 0.5$.

The two–layer neural network had two inputs, three neurons in the hidden layer and one neuron in the output layer, and, in addition, bias neurons. The output error according to Equation (5.5) is

$$E(w_{l,i,j}) = \frac{1}{2}\left(x_n - \tilde{x}_n\right)^2. \tag{5.8}$$

The constants were set $a = 0.2$, $b = 20$, $c = 0.5$ in order to avoid the necessity of normalization (the network output ranges from 0 to 1). Other parameter values were also tested and the results were similar.

We compared both optimization methods in 100 runs of testing computations, each starting from a new random generator seed. Each run comprised 300,000 iterations. The value of an error function was saved every 1000th iteration. The minimum, maximum, and average values of the error function (calculated from 100 testing runs) are shown in the Figure 5.5 for both the SADE and the backpropagation method. The graph shows that the SADE training clearly outperforms the backpropagation. The Figure 5.6 shows the distribution of the error in $f(x)$ approximation on the interval $\langle 0.1; 0.9\rangle$.

The test results have shown that the SADE training is a fully capable method of training a neural network. The number of iterations needed to achieve the same output error is significantly lower than with the backpropagation. Also the minimal output error is by about three orders lower, which could be explained by the genetic optimization's higher resistance to fall into local extremes.

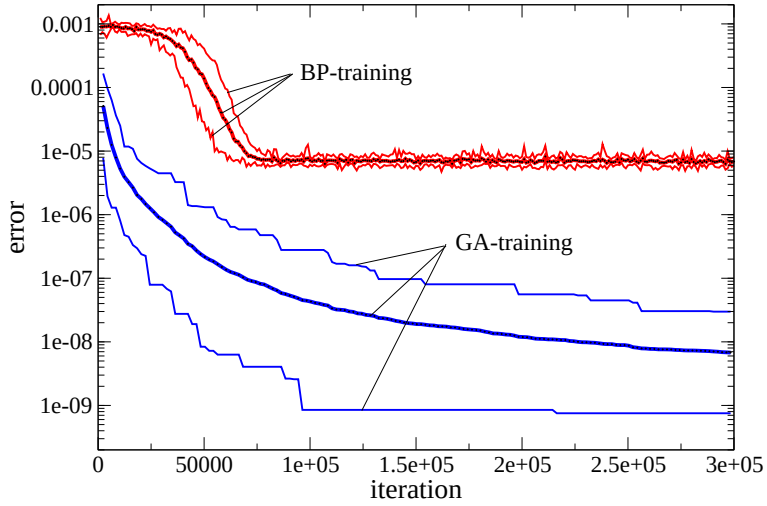


Figure 5.5: An error function in the estimation of neural network weights during an optimization process.

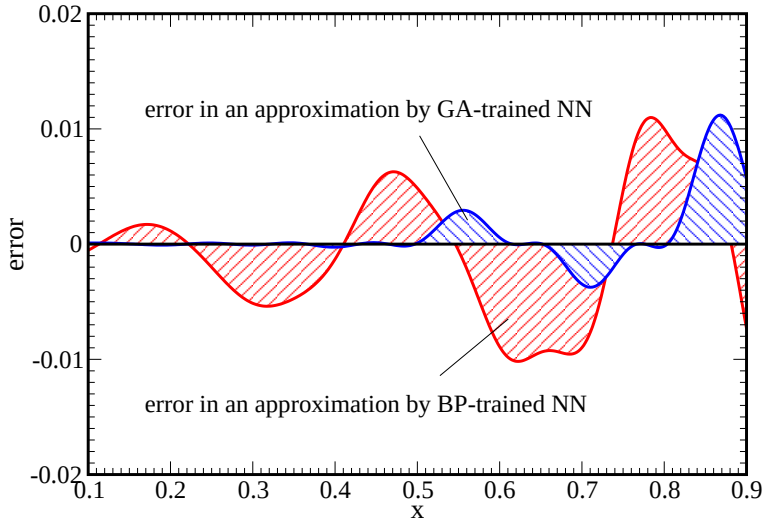


Figure 5.6: An error distribution in the approximation of $f(x) = 0.2x\ \sin(20x) + 0.5$.

5.3 Input parameter randomization and stochastic sensitivity analysis

The novelty of the identification approach proposed in [Novák and Lehký, 2006] is a systematic use of small-sample Monte Carlo simulation method for generation of neural network training sets as well as stochastic sensitivity analysis.

The Latin Hypercube Sampling (LHS) method [Iman and Conover, 1980] is used to generate particular realization of input variables as it enables to minimize the amount of simulations needed to reliably train a neural network. Moreover, the Simulated Annealing optimization

method available in the software package FREET [Novák et al., 2003] is used to maximize the statistical independence among individual samples.

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}. In civil engineering, the measurements are usually represented by some diagrams (e.g. load-deflection curves). The question arise how to transform a diagram into the input vector for an ANN. A diagram is usually defined in a discrete points, hence, one possibility is to include the coordinates of all known points. In such case, however, the topology of ANN will be very huge and the training process will become quite complicated. Moreover, there is usually a lot of neighboring points which are highly correlated between each other and therefore a group of them bring the same information as just one of them. Hence, it is quite useful to choose only a set of "interesting" points.

One possibility to reduce number of ANN's inputs is a so-called principal component analysis described in Section 3.3. In our applications, nevertheless, we have not got so good results as by a "intuitive choice by hand" driven by results from stochastic sensitivity analysis. To investigate the influence of individual parameters to a structural response we use a Pearson product moment correlation coefficient defined by equation (3.5). Note that the correlation coefficient is normalized as $-1 \leqslant cor \leqslant 1$, where higher absolute values indicate statistical dependence of the random output variable x on the random input variable y.

Part II

Applications of parameters identification methodologies

Chapter 6

OPTIMAL DESIGN AND OPTIMAL CONTROL

> The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' (I found it!) but 'That's funny...'
>
> Isaac Asimov

Modern structures should often be designed to withstand very large displacements and rotations and remain fully operational. Moreover, the construction phase has also to be mastered and placed under control, with trying to precisely guide the large motion of a particular component of the total structural assembly. Ever increasing demands to achieve more economical design and construction thus require that the problems of this kind be placed on a sound theoretical and computational basis, such as the one explored in this work. Namely, optimization methods can be called upon to guide the design procedure and achieve desired reduction of mechanical and/or geometric properties. Similarly, the control methods are employed to provide an estimate of the loads and the minimal effort in placing the structure, or its component, directly into an optimal (desired) shape. Either of these tasks, optimal design or optimal control, can formally be presented as the minimization of the chosen objective function specifying precisely the desired goal. The main difference between two procedures concerns the choice of the variables defining the objective function: the design variables are typically related to the mechanical properties (e.g. Young's modulus) or geometry of the structure (e.g. particular coordinates in the initial configuration), whereas the control variables are related to the actions (e.g. forces) applied on the structure in order to place it into desired position. Rather then insisting upon this difference and treating the optimal design and optimal control in quite different manners (as done in a number of traditional expositions on the subject), the work presented in this Chapter focus on their common features which allow a unified presentation of these two problems and the development of a novel solution procedure applicable to both problems. The latter implies that the nonlinear mechanics model under consideration of geometrically exact beam has to be placed on the central stage and one should show how to fully master the variation in chosen system properties or loads in order to achieve the optimal goal.

Although this thesis deals mostly with identification of nonlinear material models, the algorithms proposed in Chapter 4 were also successfully applied in optimal design and optimal control of structures undergoing large rotations as will presented in this Chapter.

The main task is to show how to find the corresponding initial configuration and the corre-

sponding set of multiple load parameters in order to recover a desired deformed configuration or some desirable features of the deformed configuration as specified more precisely by the objective function. The model problem chosen to illustrate the proposed optimal design and optimal control methodologies is the one of geometrically exact beam. The 2D version is employed in following computations. The 3D version together with the optimal design and optimal control formulations could be found in [Ibrahimbegović et al., 2004].

First, a non-standard formulation of the optimal design and optimal control problems is presented, relying on the method of Lagrange multipliers in order to make the mechanics state variables independent from either design or control variables and thus provide the most general basis for developing the best possible solution procedure. Three different solution procedures are then explored, one based on the diffuse approximation of response function and gradient method, the second one based on genetic algorithm and the third one consists of an iterative approximation of the objective function based on radial basis function network. A number of numerical examples are given in order to illustrate both the advantages and potential drawbacks of each of the presented procedures.

6.1 Model problem: geometrically exact 2D beam

This Chapter concerns the formulation of a model problem: an initially deformed geometrically exact 2D beam with a non-linear kinematics (see [Ibrahimbegović and Frey, 1993]).

The Marquerr's hypothesis is applied in order to describe an initially deformed beam and Timoshenko's hypothesis is incorporated to take into account the shear deformation. The presented formulation is obtained by a linearization of Reissner's beam theory. According to [Ibrahimbegovic et al., 1991], the initial deformation is deduced from a formulation for a straight by an isometric transformation. When $(\mathbf{g_1}, \mathbf{g_2})$ are supposed to be a basis vectors of a reference orthogonal coordinate system, then basis vectors of a local coordinate system of a deformed beam $(\hat{\mathbf{g}}_1, \hat{\mathbf{g}}_2)$ can be defined as:

$$\begin{bmatrix} \hat{\mathbf{g}}_1^{\mathbf{T}} \\ \hat{\mathbf{g}}_2^{\mathbf{T}} \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} \mathbf{g}_1^{\mathbf{T}} \\ \mathbf{g}_2^{\mathbf{T}} \end{bmatrix}, \tag{6.1}$$

where α is an initial rotation of a cross-section of a deformed unloaded beam.

Generalized deformation is described according to the Reissner theory. In the local rotated coordinate system with the rotation of the system $(\hat{\mathbf{g}}_1, \hat{\mathbf{g}}_2)$ one obtain an axis $\mathbf{n}$ orthogonal to the cross-section and the other axis $\mathbf{t}$ in the plane of the cross-section, see Figure 6.1. Then

$$\begin{bmatrix} \mathbf{n}^{\mathbf{T}} \\ \mathbf{t}^{\mathbf{T}} \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} \hat{\mathbf{g}}_1^{\mathbf{T}} \\ \hat{\mathbf{g}}_2^{\mathbf{T}} \end{bmatrix} = \begin{bmatrix} \cos(\alpha+\psi) & \sin(\alpha+\psi) \\ -\sin(\alpha+\psi) & \cos(\alpha+\psi) \end{bmatrix} \begin{bmatrix} \mathbf{g}_1^{\mathbf{T}} \\ \mathbf{g}_2^{\mathbf{T}} \end{bmatrix}, \tag{6.2}$$

where ψ is a rotation of cross-section resulting by the loading.

Taking into account large displacements, in the position vector in the deformed configura-

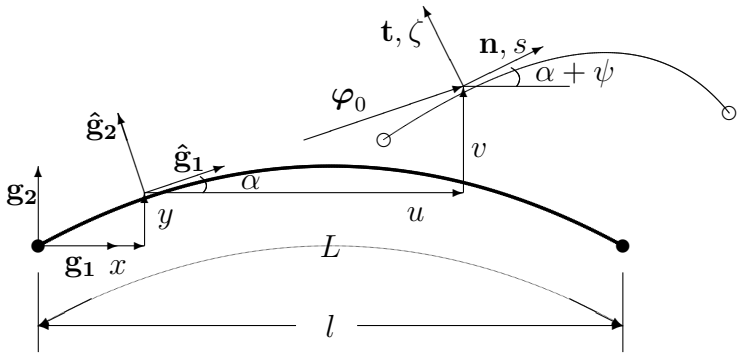


Figure 6.1: Initial and deformed configuration of the 3D geometrically exact beam.

tion can be defined as

$$\boldsymbol{\varphi} = \boldsymbol{\varphi}_0 + \zeta \mathbf{t} = \begin{pmatrix} x+u \\ y+v \end{pmatrix} + \zeta \begin{pmatrix} -\sin(\alpha+\psi) \\ \cos(\alpha+\psi) \end{pmatrix}, \tag{6.3}$$

where x and y are the coordinates of the initial configuration of the beam, u and v are the components of the displacement in the global coordinate system and ζ is a coordinate along the axis orthogonal to the beam's cross-section.

The gradient of deformation could be defined as

$$\mathbf{F} = \begin{bmatrix} \frac{\mathrm{d}x}{\mathrm{d}s} + \frac{\mathrm{d}u}{\mathrm{d}s} - \zeta \frac{\mathrm{d}\psi}{\mathrm{d}s}\cos(\alpha+\psi) & -\sin(\alpha+\psi) \\ \frac{\mathrm{d}y}{\mathrm{d}s} + \frac{\mathrm{d}v}{\mathrm{d}s} - \zeta \frac{\mathrm{d}\psi}{\mathrm{d}s}\sin(\alpha+\psi) & \cos(\alpha+\psi) \end{bmatrix}, \tag{6.4}$$

where s is a coordinate along the axis of the deformed beam.

Then the gradient of deformation could be decomposed in

$$\mathbf{F} = \mathbf{R}\mathbf{U}; \quad \mathbf{R} = \begin{bmatrix} \cos(\alpha+\psi) & -\sin(\alpha+\psi) \\ \sin(\alpha+\psi) & \cos(\alpha+\psi) \end{bmatrix} \tag{6.5}$$

and using the measure of deformation $\mathbf{H} = \mathbf{U} - \mathbf{I}$, where $\mathbf{U} = \mathbf{R}^T\mathbf{F}$, its non-zeros components are obtained

$$H_{11} = \Sigma - \zeta K\,; \quad H_{21} = \Gamma \quad , \tag{6.6}$$

where Σ, K, Γ are measures of a generalized deformation defined by Reissner in the form

$$\begin{aligned} \Sigma &= \cos(\alpha+\psi)\left(\frac{\mathrm{d}x}{\mathrm{d}s} + \frac{\mathrm{d}u}{\mathrm{d}s}\right) + \sin(\alpha+\psi)\left(\frac{\mathrm{d}y}{\mathrm{d}s} + \frac{\mathrm{d}v}{\mathrm{d}s}\right) - 1 \quad , \\ \Gamma &= -\sin(\alpha+\psi)\left(\frac{\mathrm{d}x}{\mathrm{d}s} + \frac{\mathrm{d}u}{\mathrm{d}s}\right) + \cos(\alpha+\psi)\left(\frac{\mathrm{d}y}{\mathrm{d}s} + \frac{\mathrm{d}v}{\mathrm{d}s}\right) \quad , \\ K &= \frac{\mathrm{d}\psi}{\mathrm{d}s} \quad . \end{aligned} \tag{6.7}$$

The matrix notation of the Equation (6.7) leads to

$$\boldsymbol{\Sigma} = \boldsymbol{\Lambda}^T(\mathbf{h}(\mathbf{u}) - \mathbf{n}) = \boldsymbol{\Lambda}^T\mathbf{h}(\mathbf{u}) - \mathbf{e}_1 \quad , \tag{6.8}$$

where

$$\mathbf{\Sigma} = \begin{pmatrix} \Sigma \\ \Gamma \\ K \end{pmatrix}, \quad \mathbf{\Lambda} = \begin{bmatrix} \cos(\alpha+\psi) & -\sin(\alpha+\psi) & 0 \\ \sin(\alpha+\psi) & \cos(\alpha+\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{h}(\mathbf{u}) = \begin{pmatrix} \frac{\mathrm{d}x}{\mathrm{d}s} + \frac{\mathrm{d}u}{\mathrm{d}s} \\ \frac{\mathrm{d}y}{\mathrm{d}s} + \frac{\mathrm{d}v}{\mathrm{d}s} \\ \frac{\mathrm{d}\psi}{\mathrm{d}s} \end{pmatrix}, \quad \mathbf{n} = \mathbf{\Lambda}\mathbf{e}_1, \quad \mathbf{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

Using the following constitutive laws

$$N = (EA)\Sigma, \quad V = (GA)\Gamma, \quad M = (EI)K \tag{6.9}$$

the vector of internal forces $\mathbf{N}$ becomes

$$\mathbf{N} = \mathbf{C\Sigma} = \mathbf{C\Lambda}^T(\mathbf{h}(\mathbf{u}) - \mathbf{n}), \tag{6.10}$$

where

$$\mathbf{N}^T = (N, V, M)^T, \quad \mathbf{C} = \mathrm{diag}(EA, GA, EI),$$

and section area A and moment of inertia I are supposed to be constant during the loading.

To define a weak form of equilibrium equation, the expression of a virtual deformation is needed

$$\begin{aligned} \delta\mathbf{\Sigma} &= \delta\left[\mathbf{\Lambda}^T\mathbf{h}(\mathbf{u}) - \mathbf{e}_1\right] \\ &= \delta\mathbf{\Lambda}^T\mathbf{h}(\mathbf{u}) + \mathbf{\Lambda}^T\delta\mathbf{h}(\mathbf{u}) \\ &= \mathbf{\Lambda}^T\left(\mathbf{W}\mathbf{h}(\mathbf{u})\delta\psi + \mathbf{d}(\delta\mathbf{u})\right), \end{aligned} \tag{6.11}$$

where

$$\mathbf{W} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{d}(\delta\mathbf{u}) = \begin{pmatrix} \frac{\mathrm{d}\delta u}{\mathrm{d}s} \\ \frac{\mathrm{d}\delta v}{\mathrm{d}s} \\ \frac{\mathrm{d}\delta\psi}{\mathrm{d}s} \end{pmatrix}, \quad \delta\mathbf{u} = \begin{pmatrix} \delta u \\ \delta v \\ \delta\psi \end{pmatrix}.$$

The resulting equilibrium equation could be stated as follows

$$G(\mathbf{u}, \delta\mathbf{u}) = \underbrace{\int_L \left(\delta\mathbf{\Sigma}^T\mathbf{N}\right)\mathrm{d}s}_{\mathbf{G}_{int}} - \underbrace{\int_L \delta\mathbf{u}^T\mathbf{f}^{ext}\mathrm{d}s}_{\mathbf{G}_{ext}} = 0 \quad , \tag{6.12}$$

where $\mathbf{f}^{ext}$ is a vector of extern forces applied to the structure.

Then the internal virtual work becomes

$$G_{int}(\mathbf{u}, \delta\mathbf{u}) = \int_L \left((\mathbf{d}(\delta\mathbf{u}) + \mathbf{W}\mathbf{h}(\mathbf{u})\delta\psi)^T\mathbf{\Lambda}\mathbf{C}\mathbf{\Lambda}^T(\mathbf{h}(\mathbf{u}) - \mathbf{n})\right)\mathrm{d}s \quad . \tag{6.13}$$

The finite element approximation at the previous relation is developed in detail e.g. in [Kučerová, 2003].

6.2 Optimal design

The optimal design suppose the choice of mechanical properties of structure and optimization of a geometrical properties such as thickness of beams or initial configuration of a structure. The loading is supposed to be given. Then the optimal design leads to the minimization of the objective function $J(\cdot)$, which defines desired properties of a structure. Such objective function is not a function only of design variables $\mathbf{d}$ defining the geometry of a structure, but it is a function also of displacements and rotations of a structure $\mathbf{u}$.

The traditional approach defines the optimization procedure $\hat{J}(\cdot)$ of such a problem as

$$\hat{J}(\mathbf{d}) = \min J(\mathbf{u}(\mathbf{d}),\ \mathbf{d})\ ; \quad \mathbf{u}(\mathbf{d}) :\ G(\mathbf{u}(\mathbf{d}),\ \ \delta\mathbf{u}) = 0 \quad . \tag{6.14}$$

The main advantage of such approach is a small number of optimized variables including only design variables. The components of deformation are obtained for any set of design variables as a result of an iterative solution of a weak form of equilibrium equation. Once calculated components of deformation are then used together with design variables for evaluation of objective function. Disadvantage of this approach is the constraint imposing the fulfilment of the equilibrium equation of any admissible solution, i.e. a time-consuming iterative solution of equilibrium equations for each set of design variables.

The simultaneous solution of the presented optimization task is based on application of the Lagrange multipliers inserted into the weak form of equilibrium equations instead of virtual displacements and rotations:

$$\mathbf{G}(\mathbf{u}, \boldsymbol{\lambda}) = \int_L \left((\mathbf{d}(\boldsymbol{\lambda}) + \mathbf{W}\mathbf{h}(\mathbf{u})\lambda_\psi)^T \boldsymbol{\Lambda}\mathbf{C}\boldsymbol{\Lambda}^T(\mathbf{h}(\mathbf{u}) - \mathbf{n})\right) \mathrm{d}s \quad , \tag{6.15}$$

where $\boldsymbol{\lambda} = (\lambda_u, \lambda_v, \lambda_\psi)^T$.

Optimal design then leads to an unconstrained minimization problem as

$$\max_{\forall\lambda} \min_{\forall(\mathbf{u},\mathbf{d})} L(\mathbf{u}, \mathbf{d};\ \boldsymbol{\lambda}), \tag{6.16}$$

where Lagrangian $L(\cdot)$ is defined as

$$L(\mathbf{u}, \mathbf{d}; \boldsymbol{\lambda}) = J(\mathbf{u}, \mathbf{d}) + G(\mathbf{u}, \mathbf{d}; \boldsymbol{\lambda}). \tag{6.17}$$

The main difference of (6.16) with respect to constrained minimization problem in (6.14) pertains to the fact that state variables $\mathbf{u}$ and design variables $\mathbf{d}$ are now considered independent and they can be iterated upon (and solved for) simultaneously. Nevertheless, the number of optimized variables is significantly increased, since they include not only design and state variables but also all Lagrange multipliers. On the other hand, no time-consuming iterative solution of equilibrium equation is anymore needed.

Karush-Kuhn-Tucker optimality condition (see e.g. [Luenberger, 1984]) associated with the minimization problem in (6.16) can be written as

$$0 = \mathbf{r}_u^T \delta\mathbf{u} = \left(\frac{\partial L(\cdot)}{\partial \mathbf{u}}\right)^T \delta\mathbf{u} = \left(\frac{\partial J(\mathbf{u},\mathbf{d})}{\partial \mathbf{u}}\right)^T \delta\mathbf{u} + \boldsymbol{\lambda}^T \mathbf{K} \delta\mathbf{u} \quad , \tag{6.18}$$

$$0 = \mathbf{r}_d^T \delta\mathbf{d} = \left(\frac{\partial L(\cdot)}{\partial \mathbf{d}}\right)^T \delta\mathbf{d} = \left(\frac{\partial J(\mathbf{u},\mathbf{d})}{\partial \mathbf{d}}\right)^T \delta\mathbf{d} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{f}^{int}(\mathbf{u},\mathbf{d})}{\partial \mathbf{d}} \delta\mathbf{d} \,, \tag{6.19}$$

$$0 = \mathbf{r}_\lambda^T \delta\boldsymbol{\lambda} = \left(\frac{\partial L(\cdot)}{\partial \boldsymbol{\lambda}}\right)^T \delta\boldsymbol{\lambda} = \left[\mathbf{f}^{int}(\mathbf{u},\mathbf{d}) - \mathbf{f}^{ext}\right]^T \delta\boldsymbol{\lambda} \quad , \tag{6.20}$$

where $\mathbf{K}$ is a tangent stiffness matrix.

In Equations (6.18), (6.19) and (6.20), $\mathbf{r}_u, \mathbf{r}_d, \mathbf{r}_\lambda$ are residual vectors of optimization problem. Then the optimization procedure can be defined as:

$$\min_{\forall(\mathbf{u},\mathbf{d},\lambda)} \mathbf{r}^T\mathbf{r} \,; \quad \mathbf{r}(\mathbf{u},\mathbf{d};\ \boldsymbol{\lambda}) = (\mathbf{r}_u, \mathbf{r}_d, \mathbf{r}_\lambda). \tag{6.21}$$

6.3 Optimal control

The optimal control problem studied herein concerns the quasi-static external loading sequence which is chosen to bring the structure directly towards an optimal or desired final state, which may involve large displacements and rotations. More precisely, we study the mechanics problems where introducing the pseudo-time parameter 't' to describe a particular loading program is not enough and one also needs to employ the control variables $\mathbf{c}$. The latter contributes towards the work of external forces, which can be written as

$$G^{ext}(\mathbf{c};\delta\mathbf{u}) := \int_l \delta\mathbf{u}^T \mathbf{F}_0 \mathbf{c} \mathrm{d}s \quad , \tag{6.22}$$

where $\mathbf{F_0}$ contains the (fixed) distribution of the external loading to be scaled by the chosen control.

The traditional approach to optimal control again suppose the iterative solution of equilibrium equation for any chosen set of control variables $\mathbf{c}$ to obtain corresponding state variables $\mathbf{u}$. The optimization procedure $\hat{J}(\cdot)$ then leads to following formulation

$$\hat{J}(\mathbf{c}) = \min\, J(\mathbf{u}(\mathbf{c}),\ \mathbf{c}) \,; \quad \mathbf{u}(\mathbf{c}) :\ G(\mathbf{u}(\mathbf{c}),\ \ \delta\mathbf{c}) = 0 \quad , \tag{6.23}$$

which is an equivalent to the formulation of optimal design in (6.14).

Also here, the application of Lagrange multipliers leads to the simultaneous formulation of optimal control

$$\max_{\forall\lambda} \min_{\forall(\mathbf{u},\mathbf{c})} L(\mathbf{u},\mathbf{c};\ \boldsymbol{\lambda}), \tag{6.24}$$

$$L(\mathbf{u},\mathbf{c};\boldsymbol{\lambda}) = J(\mathbf{u},\mathbf{c}) + G(\mathbf{u},\mathbf{c};\boldsymbol{\lambda}), \tag{6.25}$$

which could be easily compared with the simultaneous formulation of optimal design in Equation (6.16).

Some differences could be found in expressions of Karush-Kuhn-Tucker optimality condition:

$$0 = \mathbf{r}_\lambda^T \delta\boldsymbol{\lambda} = \left(\frac{\partial L(\cdot)}{\partial \boldsymbol{\lambda}}\right)^T \delta\boldsymbol{\lambda} := \left[\mathbf{f}^{int}(\mathbf{u},\mathbf{c}) - \mathbf{F}_0\mathbf{c}\right]^T \delta\boldsymbol{\lambda} \quad , \tag{6.26}$$

$$0 = \mathbf{r}_u^T \delta\mathbf{u} = \left(\frac{\partial L(\cdot)}{\partial \mathbf{u}}\right)^T \delta\mathbf{u} := \left(\frac{\partial J(\mathbf{u},\mathbf{c})}{\partial \mathbf{u}}\right)^T \delta\mathbf{u} + \boldsymbol{\lambda}^T\mathbf{K}\delta\mathbf{u} \quad , \tag{6.27}$$

$$0 = \mathbf{r}_c^T \delta\mathbf{c} = \left(\frac{\partial L(\cdot)}{\partial \mathbf{c}}\right)^T \delta\mathbf{c} := \left(\frac{\partial J(\mathbf{u},\mathbf{c})}{\partial \mathbf{c}}\right)^T \delta\mathbf{c} - \boldsymbol{\lambda}^T\mathbf{F}_0\delta\mathbf{c} \quad . \tag{6.28}$$

The resulting optimization procedure could be described in an equivalent way as in the case of optimal design:

$$\min_{\forall(\mathbf{u},\mathbf{c},\lambda)} \mathbf{r}^T\mathbf{r} \; ; \quad \mathbf{r}(\mathbf{u},\mathbf{c};\; \boldsymbol{\lambda}) = (\mathbf{r}_u, \mathbf{r}_d, \mathbf{r}_\lambda). \tag{6.29}$$

6.4 Solution procedure

Three solution procedures are employed for solving this class of problems of optimal design and optimal control. One optimization method is represented by the proposed GRADE genetic algorithm described in Section 4.1.2. Second proposed procedure is based on a metamodelling of the objective function by radial basis function network presented in Section 4.2. Third employed methodology is based on diffuse approximation of the objective function and gradient based optimizer presented e.g. in [Villon, 1991] and briefly summarized in the next paragraph.

6.4.1 Diffuse approximation based gradient methods

The first solution procedure is a sequential one, where one first computes grid values of the objective function and then carry out the optimization procedure by employing the approximate values interpolated from the grid. It is important to note that all the grid values provide the design or control variables along with the corresponding mechanical state variables of displacements and rotations which must satisfy the weak form of equilibrium equation. In other to ensure this requirement, for any grid value of design or control variables one also has to solve associated nonlinear problem in structural mechanics.

The main goal of the subsequent procedure is to avoid solving these nonlinear mechanics problems for other but grid values, and simply assume that the interpolated values of the objective function will be "sufficiently" admissible with respect to satisfying the equilibrium equations. Having relaxed the equilibrium admissibility requirements we can pick any convenient approximation of the objective function, which will simplify the subsequent computation

of the optimal value and thus make it much more efficient. These interpolated values of the objective function can be visualized as a surface (yet referred to as the *response surface*) trying to approximate sufficiently well the true objective function. The particular method which is used to construct the response surface of this kind is the method of diffuse approximations (see [Villon, 1991] or [Breitkopf et al., 2002]). By employing the diffuse approximations the approximate value of the objective function $\hat{J}_{appr}$ is constructed as the following quadratic form

$$\hat{J}_{appr}(\mathbf{x}) = c + \mathbf{x}^T\mathbf{b} + \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} \tag{6.30}$$

where c is a constant reference value, $\mathbf{b} = (b_i)\ ;\ b_i = \frac{\partial \hat{J}_{appr}}{\partial x_i}$ is the gradient and $\mathbf{H} = [H_{ij}]\ ;\ H_{ij} = \frac{\partial^2 \hat{J}_{appr}}{\partial x_i \partial x_j}$ is the Hessian of the approximate objective function. In (6.30) above variables $\mathbf{x}$ should be replaced by either design variables $\mathbf{d}$ for the case of an optimal design problem or by control variables $\boldsymbol{\nu}$ in the case when we deal with an optimal control problem.

We further elaborate on this idea for a simple case where only 2 design or control variables are used, such that $\mathbf{x} = (x_1, x_2)^T$. For computational proposes in such a case one uses the polynomial approximation typical of diffuse approximation (see [Breitkopf et al., 2002]) employing the chosen quadratic polynomial basis $\mathbf{p}(\mathbf{x})$ and a particular point dependent coefficient values $\mathbf{a}(\mathbf{x})$

$$J_{appr}(\mathbf{x}) = \mathbf{p}^T(\mathbf{x})\mathbf{a}(\mathbf{x})\ ;\ \mathbf{p}^T(\mathbf{x}) = [1, x_1, x_2, x_1^2, x_1x_2, x_2^2]\ ;\ \mathbf{a}(\mathbf{x}) = [a_1(\mathbf{x}), a_2(\mathbf{x}), \ldots, a_6(\mathbf{x})] \tag{6.31}$$

By comparing the last two expressions one can easily conclude that

$$c = a_1\ ;\ \mathbf{b} = \begin{pmatrix} a_2 \\ a_3 \end{pmatrix}\ ;\ \mathbf{H} = \begin{bmatrix} a_4 a_5 \\ a_5 a_6 \end{bmatrix} \tag{6.32}$$

The approximation of this kind is further fitted to the known, grid values of the objective function; $J(\mathbf{x}_i), i = 1, 2, \ldots, n$, trying to achieve that the point-dependent coefficients remain smooth when passing from one sub-domain to another. This can be stated as the following minimization problem:

$$\mathbf{a}(\mathbf{x}) = arg \min_{\forall a^*} f(\mathbf{a}^*, \mathbf{x})\ ;\ f(\mathbf{a}^*, \mathbf{x}) := \frac{1}{2}\sum_{i=1}^{n} W(\mathbf{x}, \mathbf{x}_i)\left[J(\mathbf{x}_i) - \mathbf{p}^T(\mathbf{x}_i)\mathbf{a}^*\right]^2 \tag{6.33}$$

where $W(\mathbf{x}, \mathbf{x}_i)$ are the weighting functions associated with a particular data point $\mathbf{x}$, which are constructed by using a window function $\rho(\cdot)$ based on cubic splines according to

$$W(\mathbf{x}, \mathbf{x}_i) = \rho\left(\frac{\parallel \mathbf{x} - \mathbf{x}_i \parallel}{r(\mathbf{x})}\right)\ ; \rho(s) := 1 - 3s^2 + 2s^3\ ;\ r(\mathbf{x}) = \max_k [dist(\mathbf{x}, \mathbf{x}_k)] \tag{6.34}$$

with $\mathbf{x}_k(\mathbf{x}), k = 1, \ldots n+1$ ($= 7$ for the present 2-component case) as the closest grid nodes of the given point $\mathbf{x}$. We can see that the weighting functions $W(\cdot)$ in (6.33) and (6.34) above

take a unit value at any of the closest grid nodes $\mathbf{x}_i$ and vanish outside the given domain of influence. While the former assures the continuity of the coefficients $\mathbf{a}(\mathbf{x})$, the latter ensures that the approximation remains local in character. Similar construction can be carried out for higher order problems, which requires an increased number of closest neighbors in the list.

By keeping the chosen point $\mathbf{x}$ fixed and considering the coefficients $\mathbf{a}$ of diffuse approximation as constants, the minimization of $f(\cdot)$ amounts to using the pseudo-derivative of diffuse approximation (see [Villon, 1991]) in order to compute $\mathbf{x}$ yielding the minimum of $J_{app}(\mathbf{x})$ according to

$$0 = \sum_{i=1}^{n} \mathbf{p}(\mathbf{x}_i)\, W(\mathbf{x}, \mathbf{x}_i)(J_i - \mathbf{p}^T(\mathbf{x}_i)\mathbf{a}) \tag{6.35}$$

which allows us to write a set of linear equations

$$\begin{aligned} \mathbf{a}(\mathbf{x}) &= (\mathbf{P}\mathbf{W}\mathbf{P}^T)^{-1}\mathbf{P}\mathbf{W}\mathbf{j} \\ \mathbf{P} = [\mathbf{p}(\mathbf{x}_1), \mathbf{p}(\mathbf{x}_2) \ldots \mathbf{p}(\mathbf{x}_n)]\,;\, \mathbf{j} &= (J(\mathbf{x}_1), J(\mathbf{x}_2), \ldots, J(\mathbf{x}_n)) \\ \mathbf{W} &= diag\,(W(\mathbf{x}_1, \mathbf{x}), W(\mathbf{x}_2, \mathbf{x}), \ldots, W(\mathbf{x}_n, \mathbf{x})) \end{aligned} \tag{6.36}$$

We note in passing that the computed minimum value of $\hat{J}_{appr}$ does not necessarily provide the minimum of the true objective function, which also has to satisfy the equilibrium equations; however, for a number of applications this solution can be quite acceptable. If the latter is not sufficient, we ought to explore an alternative solution procedure capable of providing the rigorously admissible value of any computed minima of the objective function, by carrying out the simultaneous solution of the objective function minimization and the equilibrium equations. The proposed procedure is based on genetic algorithm as described next.

6.5 Numerical examples

This section several illustrative examples dealing with the coupled problems of mechanics and either optimal control or optimal design are presented. The computations are carried out by using a mechanics model of 2D geometrically exact beam developed either within the *MATLAB* environment for diffuse approximation based solution procedure or within the C^{++} computer code for GRADE algorithm and RBFN based meta-model.

6.5.1 Optimal control of a cantilever structure in the form of letter T

This example is concerned with the optimal control problem of deploying initially curved cantilever beam in the final configuration which takes the form of letter T. See Figure 6.2 for initial and final configurations indicated by thick lines and a number of intermediate deformed configurations indicated by thin lines. The chosen geometric and material properties are as follows: The diameter of the circular curved part and the length of the flat part of the cantilever are both

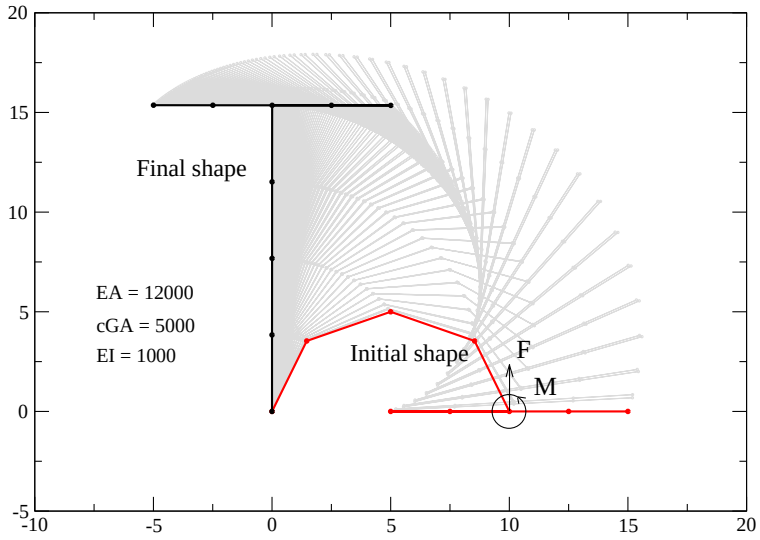


Figure 6.2: T letter cantilever: Initial, final and intermediate configurations

equal to 10; the beam cross-section is a unit square; the chosen values of Young's and shear moduli are 12000 and 6000, respectively.

The deployment is carried out by applying a vertical force F and a moment M at the end of the curved part of the cantilever. In other words the chosen control is represented by a vector $\mathbf{c} = (F, M)^T$. The desired shape of the cantilever which takes the form of letter T corresponds to the values of force $F = 40$ and moment $M = 205$. The optimal control problem is then defined as follows. The objective function is defined by imposing the desired shape only on displacement degrees of freedom with

$$J(\mathbf{c}) = \frac{1}{2}\int_L \|\mathbf{u}(\mathbf{c}) - \mathbf{u}^d\|^2 \mathrm{d}s \tag{6.37}$$

which is further recast in the discrete approximation setting as

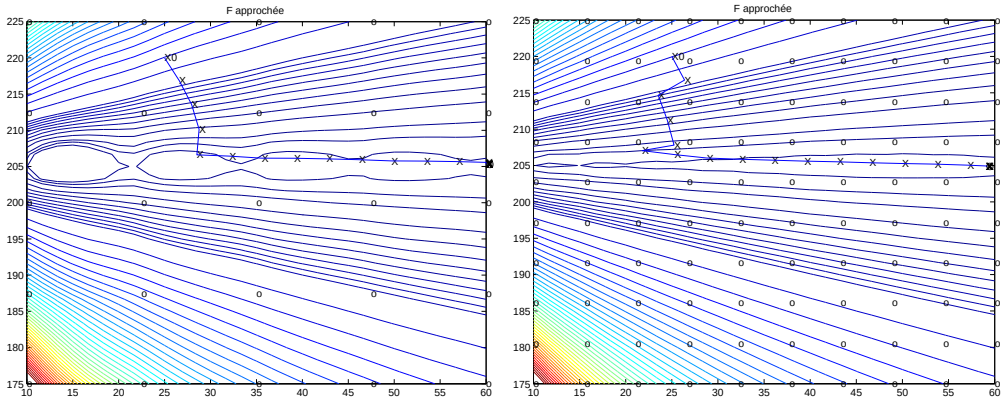
$$J^h(\mathbf{c}) = \frac{1}{4}\sum_{e=0}^{n_{el}}\sum_{a=1}^{2} l^e \left(\mathbf{u}_a^e(\mathbf{c}) - \mathbf{u}_a^d\right)^T \left(\mathbf{u}_a^e(\mathbf{c}) - \mathbf{u}_a^d\right) \tag{6.38}$$

where $\mathbf{u}_a^e(\mathbf{c})$ are computed and $\mathbf{u}_a^d$ are desired nodal displacements. Note that no condition is imposed through the objective function on either rotational degrees of freedom or control vector, which nevertheless introduces no difficulties in solving this problem.

The first solution is obtained by the diffuse approximation based gradient method. The calculation of the objective function is first carried out for all the 'nodes' of the following grids: $5 \times 5; 10 \times 10; 15 \times 15$ and 20×20. The gradient type procedure is then started on the grid and, thanks to the smoothness of the diffuse approximation base representation of the approximate value of the objective function, converged with no difficulty in roughly 20-40 iterations. The

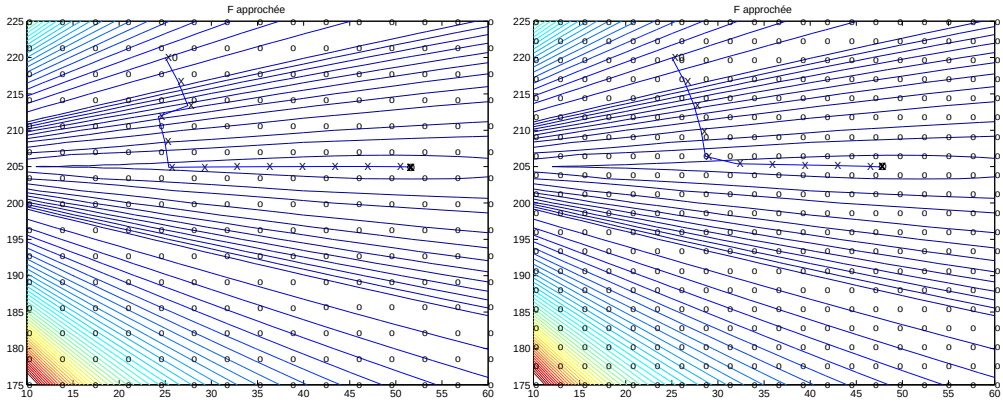
iterative values obtained in the gradient method computations are for different grids shown in Figure 6.3. Grid is constructed for the following interval of values of force and moment:

$$F \in \langle 10, 60\rangle \; ; \quad M \in \langle 175, 225\rangle.$$



Grid (5×5)
Solution : $F = 60.000$
$M = 205.26$
38 evaluations of AD

Grid (10×10)
Solution : $F = 59.073$
$M = 204.91$
38 evaluations of AD



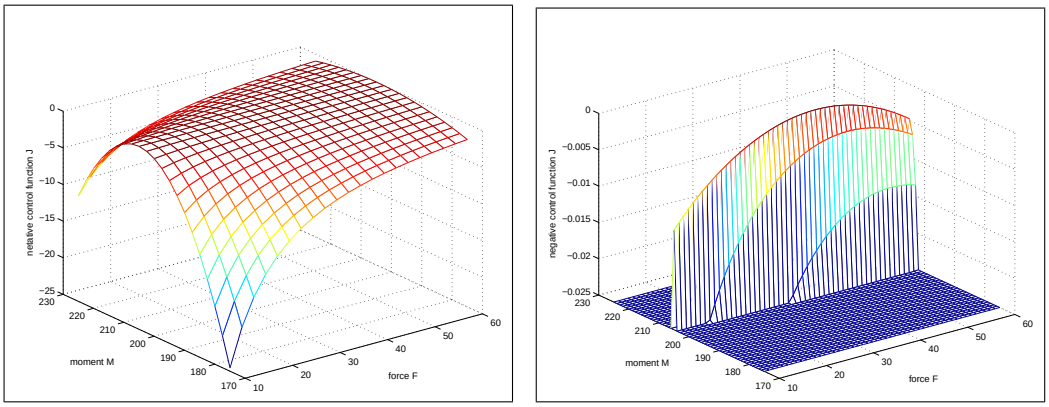
Grid (15×15)
Solution : $F = 51.218$
$M = 204.95$
19 evaluations of AD

Grid (20×20)
Solution : $F = 47.444$
$M = 204.97$
15 evaluations of AD

Figure 6.3: T letter cantilever: Gradient method iterative computation on a grid.

Note that all different choices of the grid can result in different solutions, since none of the solutions of this kind will satisfy the equilibrium equations.

Quite large difference between known optimal solution and solutions of response surface could be explained by different influence of each control variable to value of objective function, see Figure 6.4. All used grids weren't able to describe small influence of force F.



Whole scale of values. More detailed about value of optimum.

Figure 6.4: T letter cantilever: contour of the objective function.

The second solution method used for this problem employs the GRADE algorithm (see Section 4.1.2) and the third solution method applies the interpolation of the objective function based on radial basis function network iteratively improved by adding new points in the vicinity of predicted optimum (see Section 4.2 for more details). The same admissible intervals were used like in the previous case for the control variables, force and moment. The computations are stopped when the first value of the objective function $J \leqslant 10^{-7}$ is found.

In order to be able to look into statistics, one hundred runs are performed with each one converging to the exact solution. The results for both proposed methodologies are written in Table 6.1.

Algorithm	Average number of function calls
GRADE	512.4
RBFN + GRADE	104.2

Table 6.1: T letter cantilever: performance of GRADE algorithm and method based on RBFN interpolation

The second solution procedure applied to the same example is so-called simultaneous solution of mechanics equilibrium and optimal control equations, written explicitly in (6.26) to (6.28). The total number of unknowns in this case is equal to 44, with 2 control variables (the force and the moment), 21 components of nodal displacements and rotations and 21 Lagrange multiplier. The minimization problem is defined by Equation (6.29). The solution efficiency of the proposed simultaneous procedure depends on the chosen upper and lower bounds of the admissible interval and the initial guess for the solution. For example, the mechanics state variables are chosen as these featuring in the desired beam shape, $\mathbf{u}^d$, and the bounds are controlled by the chosen parameter EP according to

$$\mathbf{u} \in \left[(1-EP)\mathbf{u}^d, (1+EP)\mathbf{u}^d\right] \tag{6.39}$$

The results of optimization for four different values of EP parameter are presented in Table 6.2. For each value of EP parameter, one hundred of parameters were executed and stopped after $2,000,000$ of function evaluations.

	EP	Minimum	Maximum	Average
F	0.05	36.982	40.648	39.761
	0.02	39.219	40.420	39.894
	0.005	39.536	40.327	39.986
	0.001	39.880	40.089	39.998
M	0.05	199.01	207.80	204.27
	0.02	201.86	205.82	204.54
	0.005	204.27	205.48	204.85
	0.001	204.73	205.16	204.98
$-\mathbf{r}^T\mathbf{r}$	0.05	-120.44	-3.10	-20.94
	0.02	-35.908	-1.698	-11.427
	0.005	-16.754	-0.565	-3.699
	0.001	-1.1357	-0.0686	-0.4233

Table 6.2: T letter cantilever: impact of EP parameter to simultaneous solution procedure.

Finally, the Lagrange multipliers could be solved from (6.27) where the adopted values for $\mathbf{u}$ and $\mathbf{c}$ are chosen. One hundred computations are performed with EP parameter equal to 0.0001. All computations are stopped when $-\mathbf{r}^T\mathbf{r} > -0.01$. EP parameter is set to 0.00001. Table 6.3 summarizes the statistics of this computation.

	Minimal	Maximal	Mean Value
F	39.973	40.034	40.000
M	204.96	205.05	205.00
number of evaluations of $J(\cdot)$	14720	201480	37701

Table 6.3: T Letter cantilever : solution statistics

We can see from Table 6.3 that the proper choice of the bounds can force the algorithm to always converge to the same solution. The latter is the consequence of using the simultaneous solution procedure which assures that the computed solution also satisfies the equilibrium equations.

6.5.2 Optimal control of a cantilever structure in form of letter I

In the second example we deal with a problem which has a multiple solution and its regularized form which should restore the solution uniqueness. To that end, a cantilever beam is used very much similar to the one studied in the previous example, except for a shorter straight bar with length equal 2. The cantilever is controlled by a moment M and a couple of follower forces

which follow the rotation of the cross-sections to which they are attached. The initial and final configuration, which is obtained for a zero couple and a moment $M = 205.4$ are shown in Figure 6.5, along with a number of intermediate configurations.

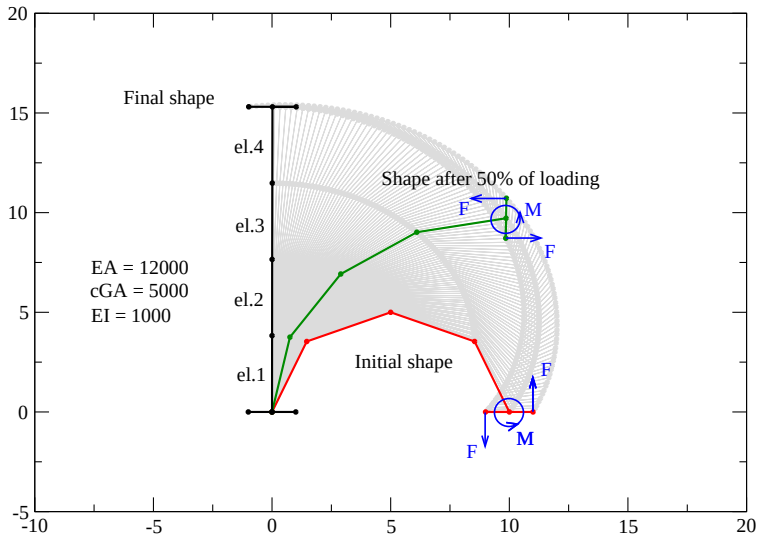


Figure 6.5: I letter cantilever: initial, final and intermediate configurations

The first computation is performed with the objective function identical to the one in (6.38), imposing only the minimum of difference between the desired and the computed deformed shape, with no restriction on control variables. The computation is carried out by using the GRADE genetic algorithm starting with random values within the selected admissible intervals for the force couple and moment according to

$$F \in \langle 0, 20 \rangle \; ; \quad M \in \langle 0, 230 \rangle$$

The algorithm performance is illustrated in Table 6.4.

	Minimum	Maximum	Mean Value
Number of fitness calls	180	640	359.8

Table 6.4: I letter cantilever: GRADE algorithm performance

A number of different solutions have been obtained with different computer runs which were performed (see Figure 6.6). However, all these solutions remain in fact clearly related considering that the applied moment and the force couple play an equivalent role in controlling the final deformed shape. It can be shown for this particular problem that any values of force and moment which satisfy a slightly perturbed version (because of the straight bar flexibility) of the force equilibrium

$$F \cdot h + M = \bar{M} = 205.4$$

will be admissible solution, thus we have infinitely many solutions for the case where only the final shape is controlled by the choice of the objective function.

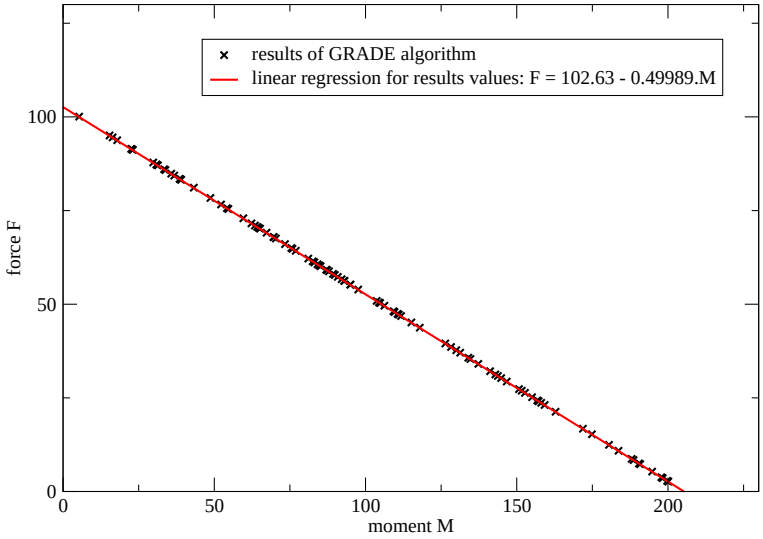


Figure 6.6: I letter cantilever: 100 different solutions

In order to eliminate this kind of problem we can perform a regularization of the objective function, by requiring not only that the difference between computed and final shape be minimized but also that control variables be as small as possible. Namely, with a modified form of the objective function

$$J^h(\hat{\mathbf{u}}_a(\mathbf{c}), \mathbf{c}) = \frac{1}{4} \sum_{e=0}^{n_{el}} \sum_{a=1}^{2} l^e \left(\hat{\mathbf{u}}_a(\mathbf{c}) - \mathbf{u}_a^d\right)^T \left(\hat{\mathbf{u}}_a(\mathbf{c}) - \mathbf{u}_a^d\right) + \alpha \mathbf{c}^T \mathbf{c} \tag{6.40}$$

where α is a chosen weighting parameter specifying the contribution of the control. We set a very small value $\alpha = 10^{-9}$ and choose the convergence tolerance to 10^{-12} and carry out the computation for yet a hundred times. Whereas a more stringent value of the tolerance requires somewhat larger number of objective function evaluation, the result in each of the runs remains always the same, given as

$$F = 68.466\,; \quad M = 68.526$$

and the found optimal value of the objective function is $J = 1.4078631.10^{-5}$.

	Minimum	Maximum	**Mean Value**
Number of fitness calls	820	16320	**1758.8**

Table 6.5: I letter cantilever: GRADE algorithm performance

6.5.3 Optimal control of deployment of a multibody system

The optimal control procedure of the deployment problem of a multibody system is studied in this example. In its initial configuration the multibody system consists of two flexible component (with $EA = 0.1$, $GA = 0.05$ and $EI = 10^3$) each 5 units long connected by the revolute joints (see [Ibrahimbegović and Mamouri, 2000]) to a single stiff component (with $EA = 1$, $GA = 0.5$ and $EI = 10^5$) of the length equal to 10, which are all placed parallel to horizontal axis. In the final deployed configuration the multibody system should take the form of a letter B with the stiff component being completely vertical and two flexible component considerably bent. The deployment of the system is controlled by five control variables, three moments M_1, M_2 and M_3, a vertical V and a horizontal force H. See Figure 6.7.

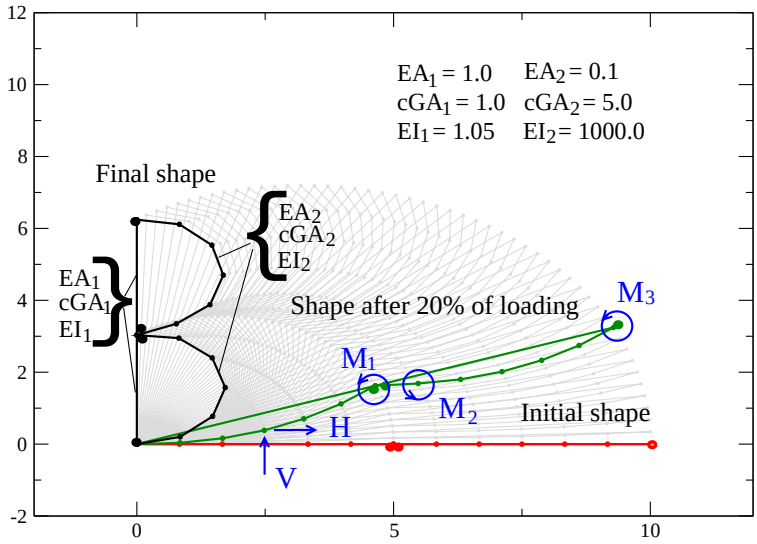


Figure 6.7: Multibody system deployment: initial, final and intermediate configurations.

The objective function in this problem is again chosen as the one in (6.38), which controls that the system would find the configuration as close as possible to the desired configuration. The desired configuration of the system corresponds to the values of forces $H = 0.04$, $V = -0.05$ and moments $M_1 = 0.782$, $M_2 = -0.792$ and $M_3 = 0.792$. The solution is computed by using the GRADE algorithm and starting with the random choice in the interval of interest defined as

$$H \in \langle 0.025; 0.05 \rangle, \quad V \in \langle -0.06; -0.035 \rangle, \quad M_1 \in \langle 0.6; 0.9 \rangle,$$
$$M_2 \in \langle -0.9; -0.65 \rangle, \quad M_3 \in \langle 0.6; 0.85 \rangle.$$

The solution of the problem is typically more difficult to obtain with an increase in the number of control variables, one of the reasons being a more irregular form of the objective function. In that sense, an illustrative representation of the objective function contours in different subspaces of control variables are given in Appendix B.

The convergence tolerance on objective function is chosen equal to 10^{-6}. The GRADE algorithm performance is presented in Table 6.6.

	Minimum	Maximum	Mean Value
Number of fitness calls	1900	117850	20632.0

Table 6.6: Results of GRADE algorithm for 5D task

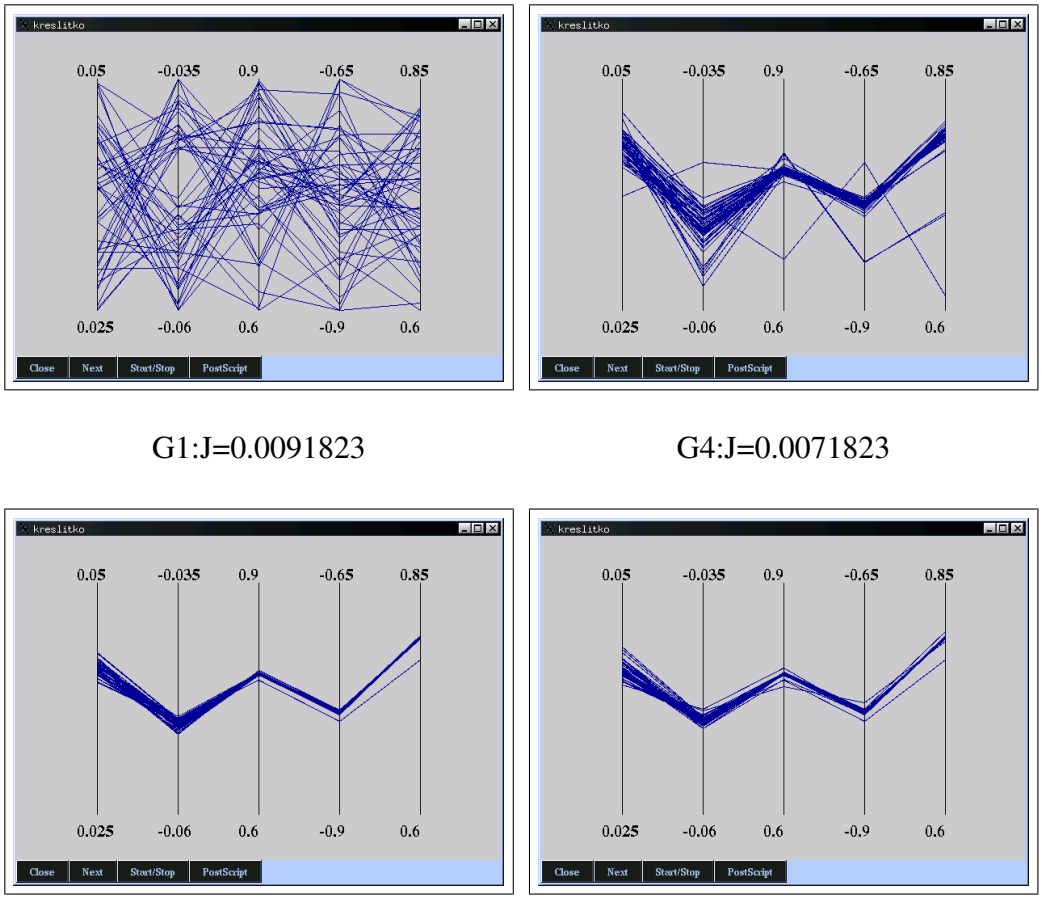


Figure 6.8: Multibody system deployment: convergence of iterative chromosome populations

One can notice the order of magnitude of increase in objective function evaluation, which is brought about by a more elaborate form of the objective function (see Figures B.1 and B.2). However, the latter is not the only reason. In this particular problem the role of moments in the list of control variables is much more important than the role of the horizontal and vertical forces in bringing the system in the desired shape. This affects the conditioning of the equations to be solved and the slow convergence rate of the complete system is in reality only the slow convergence of a single or a couple of control components. The latter is illustrated in Figure 6.8, where we provide the graphic representation of iterative values for computed chromosomes, where every chromosome is represented by a continuous line. We can note that the population

of optimal values of moments converges much more quickly than the force values which seek a large number of iteration in order to stabilize. Another point worthy of further exploration is the best way to accelerate the convergence rate in the final computational phase.

6.5.4 Optimal design of shear deformable cantilever

Last example is focused on an optimal design problem which considers the thickness optimization of a shear deformable cantilever beam, shown in Figure 6.9.

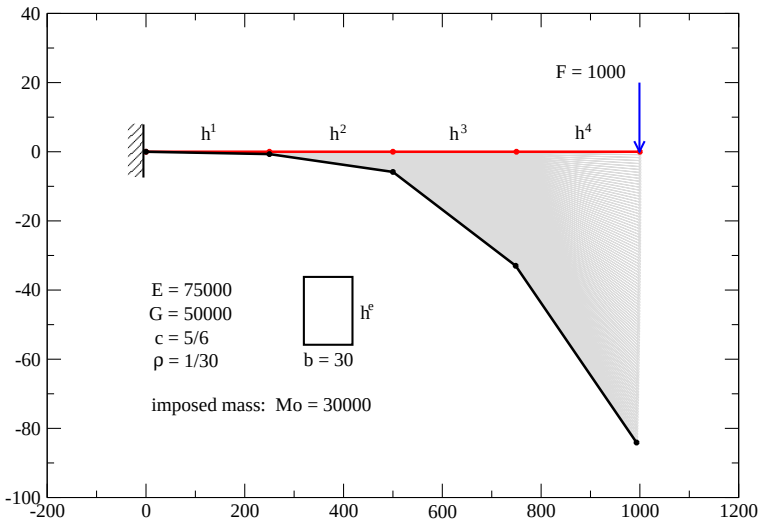


Figure 6.9: Shear deformable cantilever beam optimal design : initial and deformed shapes

The beam axis in the initial configuration of the cantilever and the thickness is considered as the variable to be chosen in order to assure the optimal design specified by a objective function. In the setting of discrete approximation, four beam elements are chosen each with a constant thickness h_i, which results with four design variables $\mathbf{d} \equiv \mathbf{h} = (h_1, h_2, h_3, h_4)$. The beam mechanical and geometric properties are: Young's modulus $E = 75000$, shear modulus $G = 50000$, rectangular cross section $b \times h_i$ with width $b = 30$ and mass density $\rho = 1/30$. The latter is needed for computing the total mass of the beam $M = \int_L \rho b h(s) \mathrm{d}s$ to be used as the corresponding limitation on the computed solution assuring reasonable values of the optimal thickness under the free-end vertical force $F = 1000$. In order to assure a meaningful result the computations are performed under chosen value of mass limitation is $M_0 = 30000$. Other limitations are also placed on the admissible values of the thickness for each element.

The first computation is performed by using the diffuse approximation based response function and the sequential solution procedure. The objective function is selected as the shear energy

of the beam and problem is cast as maximization of the shear energy, with

$$J(\mathbf{u}(\mathbf{d})) = \max_{G(\mathbf{u}^*(\mathbf{d}^*),\cdot)=0} J(\hat{\mathbf{u}}^*(\mathbf{d}^*)) \; ; \quad J(\mathbf{u}^*) = \int_L \frac{1}{2} GA\gamma^2 \mathrm{d}s \tag{6.41}$$

where γ is the shear strain component. The bounds on thickness values are chosen as shown in Table 6.7. The diffuse approximation computations on the grid are started from an initial

Thickness	h_1	h_2	h_3	h_4
Min	30	30	15	15
Max	60	60	35	35

Table 6.7: Shear deformable cantilever optimal design : thickness admissible values

guesses for thickness $\mathbf{h}_0 = (55, 50, 30, 20)$. It took 11 iterations to converge to solution given as

$$\mathbf{h} = (45.094, 40.074, 19.832, 15.000)$$

The corresponding value of shear energy for this solution is $J_{appr} = 16.3182$; we recall it is only an approximate solution, since the computed value does not correspond to any of the grid nodes.

The same solution is next sought by using GRADE algorithm. The genetic algorithm is executed 100 times, leading to the computational statistics reported in Table 6.8.

	Minimum	Maximum	Mean Value
nb. of comput. $J(\cdot)$	120	3400	674.8

Table 6.8: Shear deformable cantilever optimal design : computation statistics

The algorithm yielded two different solutions, both essentially imposed by the chosen bounds; Namely, out of 100 runs, 57 converged to $\mathbf{h} = (60, 30, 15, 15)$, with the corresponding value of $J = 17.974856$, whereas 43 settled for $\mathbf{h} = (30, 60, 15, 15)$ with a very close value of $J = 17.926995$. Hence, each of two solutions leads to an improved value of the objective function.

The second part of this example is a slightly modified version of the first one, in the sense that the mechanics part of the problem is kept the same and only a new objective function is defined seeking to minimize the Euclidean norm of the computed displacement vector, i.e.

$$J(\mathbf{u}(\mathbf{d})) = \max_{G(\mathbf{u}^*(\mathbf{d}^*),\cdot)=0} J(\hat{\mathbf{u}}^*(\mathbf{d}^*)) \; ; \quad J(\mathbf{u}^*) = \frac{1}{2} \int_L \|\mathbf{u} - \mathbf{x}\|^2 \mathrm{d}s \tag{6.42}$$

Such a choice of the objective function is made for being well known to result with a well-conditioned system expressing optimality conditions. Indeed, the same type of sequential solution procedure using diffuse approximation of objective function now needs only a few iterations to find the converged solution, starting from a number initial guesses. The final solution value is given as

$$\mathbf{h} = (42.579, 35.480, 26.941, 15.000)$$

.

In the final stage of this computation the solution of this problem is recomputed by using the genetic algorithm. The admissible value of the last element thickness is also slightly modified by reducing the lower bound to 5 (instead of 15) and higher bound to 25 (instead of 35) in order to avoid the optimal value which is restricted by a bound. The first solution to this problem is obtained by using again the sequential procedure, where the GRADE genetic algorithm is employed at the last stage. The computed value of the displacement vector norm for found solution is 623808 and mass M is 30062. The computations are carried out a hundred times starting from random initial values. The statistics of these computations are given in Table 6.9.

	Minimum	Maximum	Mean Value
h_1	43.772	43.807	43.790
h_2	35.914	35.949	35.932
h_3	26.313	26.346	26.328
h_4	14.184	14.210	14.197
nb. of comput. $J(\cdot)$	1440	9960	3497

Table 6.9: Shear deformable cantilever optimal design : computation statistics

The same kind of problem is now repeated by using the simultaneous solution procedure, where all the optimality condition are treated as equal and solved simultaneously resulting with four thickness variables, 15 displacement and rotation components and as many Lagrange multipliers as unknowns. The latter, in fact, is eliminated prior to solution by making use of optimality condition in (6.18). The chosen upper and lower bounds of the admissible interval are chosen as

$$\mathbf{u} \in [(1 - EP)\mathbf{u}^p, (1 + EP)\mathbf{u}^p] \tag{6.43}$$

where the guess for the displacement $\mathbf{u}^p$ is obtained by solving mechanics problem with the values of thickness parameters given in Table 6.9. The limitation on total mass is added to the objective function. The choice of GRADE algorithm parameters is given as $PR = 20, CL = 2$ and 'radioactivity' equal to 0.1. The computation is stopped with a fairly loose tolerance, which allows to accelerate the algorithm convergence but does not always lead a unique solution. Yet, the results in Table 6.10 show that standard deviation indeed remains small, or that the solution is practically unique.

	Minimum	Maximum	Mean Value
h_1	43.782	43.794	43.789
h_2	35.925	35.935	35.930
h_3	26.315	26.324	26.319
h_4	14.197	14.202	14.200
nb. of comput. $\mathbf{r}^T\mathbf{r}$	111340	968240	313006

Table 6.10: Shear deformable cantilever optimal design : simultaneous computation statistics

6.6 Summary

The approach advocated herein for dealing with a coupled problem of nonlinear structural mechanics and optimal design or optimal control, which implies bringing all the optimality conditions at the same level and treating all variables as independent rather than considering equilibrium equations as a mere constraint and state variables as dependent on design or control variables, is fairly unorthodox and rather unexplored. For a number of applications the proposed approach can have a great potential. In particular, the problems of interest to this work concern the large displacements and rotations of a structural systems. The key ingredient of such an approach pertains to geometrically exact formulation of a nonlinear structural mechanics problem, which makes dealing with nonlinearity description or devising the solution schemes much easier then for any other model of this kind. The model problem of the geometrically exact beam explored in detail herein is not the only one available in this class. We refer to work in [Ibrahimbegović, 1994] for shells or to [Ibrahimbegović, 1995] for 3D solids, with both models sharing the same configuration space for mechanics variables as 2D beam. The latter also allows to directly exploit the presented formulation and the solution procedures of a coupled problem of nonlinear mechanics, for either shells or 3D solids, and optimal control or optimal design.

Three different solution procedures are presented herein; the first one, which exploits the response surface representation of the true objective function followed by a gradient type solution step, leads to only an approximate solution. Although the quality of such a solution can always be improved by further refining the grid which serves to construct the response surface, the exact solution is never computed unless the minimum corresponds to one of the grid points. Moreover, the higher number of optimized variables increases extremely the number of grid points necessary to construct the approximation, which makes the application of such solution procedure impossible.

The second solution procedure employs the interpolation of the objective function based on radial basis function networks and significantly outperforms the approximation based methodology in the efficiency point of view as well as in the accuracy of found optimum. Nevertheless, also this procedure is limited in application to problems with only a several optimized variables.

The third solution procedure applies the GRADE algorithm and is able to solve optimality conditions and nonlinear mechanics equilibrium equations. Although the number of optimized variables is in this case very high, the methodology does deliver the exact solution, nevertheless often only after the appropriate care is taken to choose the sufficiently close initial guess and to select the admissible intervals of all variables accordingly. Probably the best method in that sense is the combination of sequential and simultaneous procedure, where the first serves to reduce as much as possible the admissible interval and provide the best initial guess, whereas the second furnishes the exact solution.

Chapter 7

PARAMETERS IDENTIFICATION OF CONTINUUM-DISCRETE DAMAGE MODEL CAPABLE OF REPRESENTING LOCALIZED FAILURE

> The more original discovery, the more obvious it seems afterwards.
>
> Arthur Koestler

This Chapter deal with the parameters identification of a relatively simple model capable of describing the behavior of a massive structure until the point of localized failure. The model contains all the ingredients for taking into account both the diffuse damage mechanism, which leads to the appearance of microcracks, as well as the failure process characterized by the propagation of macrocracks. Perhaps the most important advantage of the proposed model is the fact that all its parameters have a clear physical interpretation and can be straightforwardly visualized in terms of the shape of a stress-strain diagram. In addition, influence of each parameter is dominant only for a specific, easily recognizable, stages of material behavior. This kind of a priori knowledge has a potential to greatly simplify the model calibration and will be systematically used throughout the chapter.

The emphasis is put on the identification of the model parameters from experimental measurements made on a structural level. Generally speaking, the complexity of the identification procedure is determined by the choice of experimental setup. Solely from the identification point of view, the simplest experiment to execute is the uniaxial tensile test. In this case, the strain field stays mostly homogeneous during the whole procedure and the global response represented by the load-displacement diagram is very similar to the stress-strain curve for one material point; see Section 7.2 for more details. The model parameters can then be directly determined from the shape of the load-displacement curve. Such a uniform loading is, however, very difficult if not impossible to impose in a laboratory test, especially for quasi-brittle materials. Therefore, other testing techniques are often used in experimental practice.

The three-point bending test, in particular, is considered to be much simpler to perform and its results to be well-reproducible. Therefore, we focus on the identification procedure for the proposed model parameters directly from results of three-point bending test. Main difficulty is in this case imposed by heterogeneity of the stress and the strain fields, which is present since the very start of the experiment. The macro-scale measurements provide the load-deflection curve that integrates data from different parts of the specimen experiencing different regimes

of (in)elastic behavior. For that reason, the possibility of a simple determination of model parameters from load-deflection curve is lost and an advanced calibration procedure needs to be applied.

To take advantage of the model specific structure, already mentioned above, the identification procedure should be divided into three sequential stages discussed in detail in Section 7.3. From the algorithmic point of view, the material calibration can then be understood as a sequential optimization problem. Such approach has two main advantages: first, solving three simpler identification steps in a batch form is typically much more efficient then the full-scale problem; second, it allows to use only a subset of simulations for initial stages of an identification process.

A variety of techniques is available to identify material parameters via optimization methods, see e.g. [Mahnken, 2004, and reference therein]. The gradient-based methods are usually considered to be the most computationally efficient optimization algorithms available and as such have been successfully used in a variety of identification problems [Mahnken and Stein, 1996, Iacono et al., 2006, Maier et al., 2006]. For the current model, however, analytic determination of sensitivities is fairly difficult, mainly due to the history dependency of the model as well as complex interaction of individual parameters. The accuracy of numerical approximation to the 'exact' sensitivities, on the other hand, is driven by the choice of pseudo-time step used in numerical simulations. Clearly, to reduce the computational time, the pseudo-time step should be used as large as possible. Therefore, the response-based objective function will not be smooth and gradient-based methods are unlikely to be very successful.

As an alternative, techniques of soft-computing can be employed for optimization of complex objective functions. For example, evolutionary algorithms have been successfully used for solution of identification problems on a level of material point [Furukawa and Yagawa, 1997, Pyrz and Zairi, 2007] or on a level of simple structures [Ibrahimbegović et al., 2004, Lepš, 2005]. For the current case, however, complexity of the optimization can be attributed rather to its non-smooth character than to the appearance of multiple optima; the family of problems where evolutionary algorithms are the most successful methods. This opens the way to more specialized tools, which deliver higher efficiency when compared to usually time-consuming evolutionary algorithms, see Chapter 2.

The approach adopted in the present work is based on an adaptive smoothing of the objective function by artificial neural networks (see e.g. [Waszczyszyn and Ziemiański, 2006] or [Pichler et al., 2003] for alternative ANN-based solutions to identification problems). In particular, the approximated model is provided by the Radial Basis Function Network, described in Section 4.2, dynamically evolved by minima located by a real-encoded genetic algorithm, described in Section 4.1.2. The proposed sequential numerical strategy is systematically verified in Section 7.4 with attention paid to a detailed assessment of the proposed stochastic algorithm reliability.

7.1 A brief description of the identified model

In the present section, we give a brief description of the model on which the identification procedure is based. For the readers interested in more details, the complete description of the model is given in [Brancherie and Ibrahimbegovic, 2007].

As already mentioned, the proposed model is capable of taking into account two different types of dissipation (the main principles are given in [Ibrahimbegovic, 2006]):

- a bulk dissipation induced by the appearance of uniformly distributed microcracks. This bulk dissipation is taken into account by the use of a classical continuum damage model;
- a surface dissipation induced by the development of macrocracks responsible for the collapse of the structure. As presented in [Brancherie and Ibrahimbegovic, 2007], this phase is taken into account by the use of a strong discontinuity model. The surface dissipation is taken into account by the introduction of a traction/displacement jump relation.

Therefore, two different models are involved in the constitutive description: the one associated with the bulk material and the one associated with the displacement discontinuity. Both are built on the same scheme considering the thermodynamics of continuous media and interfaces.

The key points of the construction of each of the two models are summarized in Table 7.1 and Table 7.2.

Helmholtz energy	$\bar{\psi}(\bar{\boldsymbol{\varepsilon}}, \overline{\mathbf{D}}, \bar{\xi}) = \frac{1}{2}\bar{\boldsymbol{\varepsilon}} : \overline{\mathbf{D}}^{-1} : \bar{\boldsymbol{\varepsilon}} + \bar{\Xi}(\bar{\xi})$
Damage function	$\bar{\phi}(\boldsymbol{\sigma}, \bar{q}) = \underbrace{\sqrt{\boldsymbol{\sigma} : \mathbf{D}^e : \boldsymbol{\sigma}}}_{\|\boldsymbol{\sigma}\|_{\mathbf{D}^e}} - \frac{1}{\sqrt{E}}(\sigma_f - \bar{q})$
State equations	$\boldsymbol{\sigma} = \overline{\mathbf{D}}^{-1} : \bar{\boldsymbol{\varepsilon}} \quad \text{and} \quad \bar{q} = -\frac{d}{d\bar{\xi}}\bar{\Xi}(\bar{\xi})$
Evolution equations	$\dot{\overline{\mathbf{D}}} = \dot{\bar{\gamma}}\frac{\partial \bar{\phi}}{\partial \boldsymbol{\sigma}} \otimes \frac{\partial \bar{\phi}}{\partial \boldsymbol{\sigma}} \frac{1}{\|\boldsymbol{\sigma}\|_{\mathbf{D}^e}} \quad ; \quad \dot{\bar{\xi}} = \dot{\bar{\gamma}}\frac{\partial \bar{\phi}}{\partial \bar{q}}$
Dissipation	$0 \leqslant \bar{\mathcal{D}} = \frac{1}{2}\dot{\bar{\xi}}(\bar{\sigma}_f - \bar{K}\bar{\xi})$

Table 7.1: Main ingredients of the continuum damage model

For the discrete damage model, the isotropic softening law is chosen as:

$$\bar{\bar{q}} = \bar{\bar{\sigma}}_f \left[1 - exp\left(-\frac{\bar{\bar{\beta}}}{\bar{\bar{\sigma}}_f} \bar{\bar{\xi}} \right) \right] \tag{7.1}$$

In Tables 7.1 and 7.2 the variables $\dot{\bar{\gamma}}$, $\dot{\bar{\bar{\gamma}}}_1$ and $\dot{\bar{\bar{\gamma}}}_2$ denote Lagrange multipliers induced by the use of the maximum dissipation principle. Moreover, $\bar{\bar{\mathbf{u}}}$ denotes the displacement jump on the surface of discontinuity. Finally, $\overline{\mathbf{D}}$ and $\bar{\bar{\mathbf{Q}}}$ correspond to the damaged compliance of the continuum and discrete model, respectively.

Helmholtz energy	$\bar{\bar{\psi}}(\bar{\bar{\mathbf{u}}}, \bar{\bar{\mathbf{Q}}}, \bar{\bar{\xi}}) = \frac{1}{2}\bar{\bar{\mathbf{u}}} \cdot \bar{\bar{\mathbf{Q}}}^{-1} \cdot \bar{\bar{\mathbf{u}}} + \bar{\bar{\Xi}}(\bar{\bar{\xi}})$
Damage functions	$\bar{\bar{\phi}}_1(\mathbf{t}_{\Gamma_s}, \bar{\bar{q}}) = \mathbf{t}_{\Gamma_s} \cdot \mathbf{n} - (\bar{\bar{\sigma}}_f - \bar{\bar{q}})$ $\bar{\bar{\phi}}_2(\mathbf{t}_{\Gamma_s}, \bar{\bar{q}}) = \lvert\mathbf{t}_{\Gamma_s} \cdot \mathbf{m}\rvert - (\bar{\bar{\sigma}}_s - \frac{\bar{\bar{\sigma}}_s}{\bar{\bar{\sigma}}_f}\bar{\bar{q}})$
State equations	$\mathbf{t}_{\Gamma_s} = \bar{\bar{\mathbf{Q}}}^{-1} \cdot \bar{\bar{\mathbf{u}}}$ and $\bar{\bar{q}} = -\frac{\partial \bar{\bar{\Xi}}}{\partial \bar{\bar{\xi}}}$
Evolution equations	$\dot{\bar{\bar{Q}}} = \dot{\bar{\bar{\gamma}}}_1 \frac{1}{\mathbf{t}_{\Gamma_s} \cdot \mathbf{n}} + \dot{\bar{\bar{\gamma}}}_2 \frac{1}{\lvert\mathbf{t}_{\Gamma_s} \cdot \mathbf{m}\rvert}$; $\dot{\bar{\bar{\xi}}} = \dot{\bar{\bar{\gamma}}}_1 + \frac{\bar{\bar{\sigma}}_s}{\bar{\bar{\sigma}}_f}\dot{\bar{\bar{\gamma}}}_2$
Dissipation	$0 \leqslant \bar{\bar{\mathcal{D}}} = \frac{1}{2}\dot{\bar{\bar{\xi}}}(\bar{\bar{\sigma}}_f - \bar{\bar{K}}\bar{\bar{\xi}})$

Table 7.2: Main ingredients of the discrete damage model

E	$\in$	(25.0, 50.0) GPa
ν	$\in$	(0.1, 0.4)
$\bar{\sigma}_f$	$\in$	(1.0, 5.0) MPa
$\bar{K}$	$\in$	(10.0, 10000.0) MPa
$\bar{\bar{\sigma}}_f$	$\in$	($\bar{\sigma}_f$+0.1, $2\bar{\sigma}_f$) MPa
$\bar{\bar{\beta}}$	$\in$	($0.1\bar{\bar{\sigma}}_f$, $10.0\bar{\bar{\sigma}}_f$) MPa/mm

Table 7.3: Limits for the model parameters.

Note that in a three-point bending or a uniaxial tensile test, the simulated response is almost independent of the limit tangential traction $\bar{\bar{\sigma}}_s$. Therefore, its value was set to $0.1\bar{\bar{\sigma}}_f$. With such simplification, there are six independent material parameters to be identified:

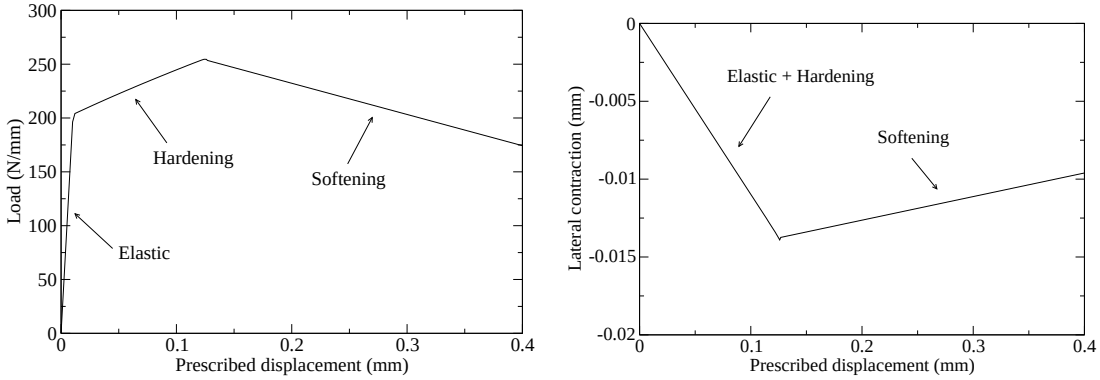
- the elastic parameters: the Young modulus E and the Poisson ratio ν;
- the continuum damage parameters : the limit stress $\bar{\sigma}_f$ and the hardening parameter $\bar{K}$;
- the discrete damage parameters : the limit normal traction $\bar{\bar{\sigma}}_f$, and the softening parameter $\bar{\bar{\beta}}$.

The limits of realistic values for each parameter are shown in the Table 7.3. Note that in our identification methodology we do not suppose to have an expert capable of giving the initial estimate of material parameters values, as in e.g. [Iacono et al., 2006, Novák and Lehký, 2006, Maier et al., 2006]. Therefore, the bounds on model parameters were kept rather wide.

7.2 Tensile test

The simplest possibility to identify material parameters for a particular concrete is to perform a uniaxial tensile test. In this case, the stress and strain fields within the specimen would remain homogeneous until the final localized failure phase, and the behavior on the structural level is very close to the response of a material point. The load-displacement diagram consist

of three easily recognizable parts, as shown in Figure 7.1a: the first one corresponding to the elastic response of the material, the second one describing the hardening and the third part the softening regime. The calibration of model parameters can be carried out to follow the same pattern: first, Young's modulus E and Poisson's ratio ν are determined from the elastic part of the load-displacement diagram, followed by the limit stress $\bar{\sigma}_f$ and the hardening parameter $\bar{K}$ identification from the part of the diagram corresponding to the hardening regime and, finally, the limit normal traction $\bar{\bar{\sigma}}_f$ and the softening parameter $\bar{\bar{\beta}}$ are estimated from the softening branch. Note that for Poisson's ratio identification, one additional local measurement is needed to complement the structural load-displacement curve, namely the measurement of lateral contraction of the specimen, see Figure 7.1b.



(a) (b)

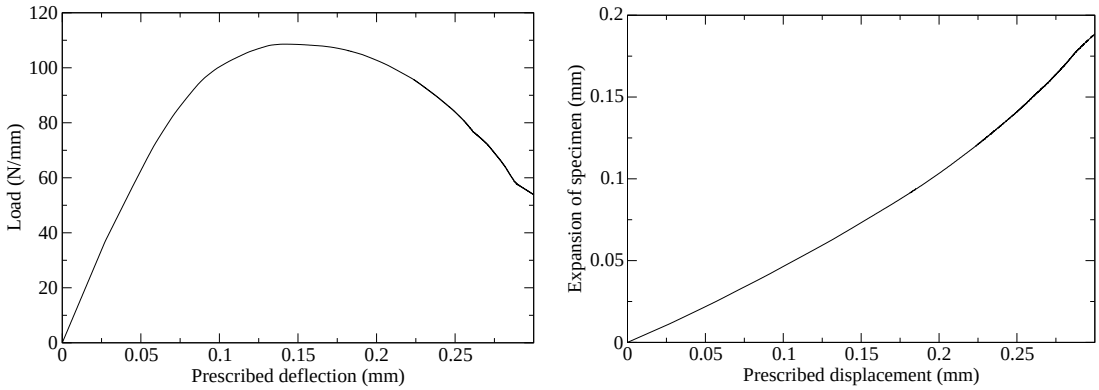
Figure 7.1: Tensile loading test: (a) Load-deflection diagram (b) Evolution of lateral contraction.

Although this kind of calibration procedure is robust and accurate [Kučerová et al., 2006], the experiment dealing with a simple tension test is rather difficult, if not impossible, to perform in a well-reproducible way. For that reason, we turn to study the possibility of parameter estimates by using three-point bending tests, which is much simpler to practically perform in a laboratory.

7.3 Three-point bending test

In the case of a three-point bending test the global response of a specimen represented by the load-deflection (L-u) diagram for the structure cannot be simply related to three-stage material response with elastic, hardening and softening part (see Figure 7.2a). Nevertheless, we assume that it will be still possible to employ the three-stage approach developed for the uniaxial tensile experiment. Similarly to the previous case, the solution process will be divided into the optimization of elastic, hardening and softening parameters in the sequential way. Each step is described in detail in the rest of this section.

Due to lack of experimental data, a reference simulation with parameters shown in Table 7.4 will be used to provide the target data. These are the same values as considered for simulation



(a) (b)

Figure 7.2: Three-point bending test: (a) Load-deflection diagram (b) Evolution of expansion of specimen.

E	=	38.0 GPa
ν	=	0.1
$\bar{\sigma}_f$	=	2.2 MPa
$\bar{K}$	=	1000.0 MPa
$\bar{\bar{\sigma}}_f$	=	2.35 MPa
$\bar{\bar{\beta}}$	=	23.5 MPa/mm

Table 7.4: Parameter's values for reference simulation.

presented in [Brancherie and Ibrahimbegovic, 2007].

7.3.1 Identification of elastic parameters

In the elastic range, Young's modulus and Poisson's ratio are determined using a short simulations describing only the elastic response of a specimen. Similarly to the uniaxial tensile test, the elastic behavior is represented by the linear part of load-deflection diagram. To identify both elastic parameters this information needs to be supplemented with an additional measurement. In particular, we propose to include the specimen expansion Δl defined as the relative horizontal displacements between the left and the right edge of the specimen (as indicated by arrows in the Figure 7.3), or in other words $\Delta l = v_2 - v_1$. The reference expansion-deflection curve is shown in Figure 7.2b.

The objective function F_1 applicable for the determination of elastic parameters can be defined as follows:

$$F_1 = (L_{ref}(u) - L(u))^2 w_1 + (\Delta l_{ref}(u) - \Delta l(u))^2 w_2 \quad ; \qquad u = 0.01\text{mm} \tag{7.2}$$

The load L and the expansion Δl are quantities depending not only on displacement u, but also on the values of material parameters. In particular, at the beginning of the loading regime (where $u = 0.01$mm), the important parameters are only Young's modulus and Poisson's ration,

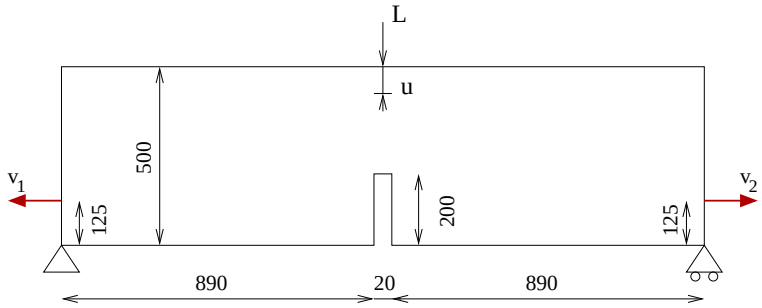
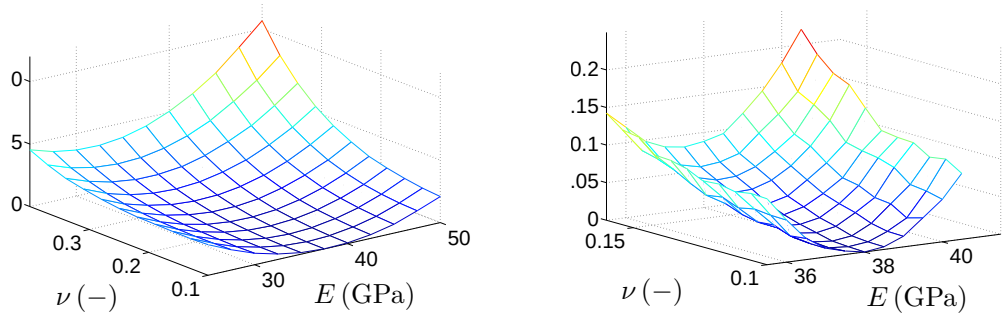


Figure 7.3: Displacements measured to evaluate the expansion $\Delta l = v_2 - v_1$ of the specimen.

because the other parameters are not yet activated. The quantities with index $_{ref}$ correspond to the values taken from the reference diagram. The corresponding value of weights w_1 and w_2 were calculated using 30 random simulations to normalize the average value of each of summation terms in (7.2) to one. It is worth noting that all the quantities in the objective function are evaluated for the prescribed deflection $u = 0.01$ mm, which allows the simulation to be stopped after reaching this value. Therefore, the first optimization stage is computationally very efficient.

For the sake of illustration, the shape of objective function F_1 is shown in the Figures 7.4a and 7.4b. As shown in this figure, the objective function remains rather wiggly in the neighborhood of the optimal value.



(a) (b)

Figure 7.4: Objective function F_1: (a) Whole domain (b) Detail close to optimal value.

7.3.2 Identification of hardening parameters

Once we have successfully determined Young's modulus and Poisson's ratio, we can continue towards the estimate of the elastic limit stress $\bar{\sigma}_f$ (representing a threshold of elastic behavior) and the hardening parameter $\bar{K}$. In the spirit of the uniaxial tensile test the limit stress will be related to the limit displacement $\bar{u}_f$ at the end of the linear part of the load-deflection diagram. The hardening parameter $\bar{K}$ will then govern the slope of the diagram in the hardening regime.

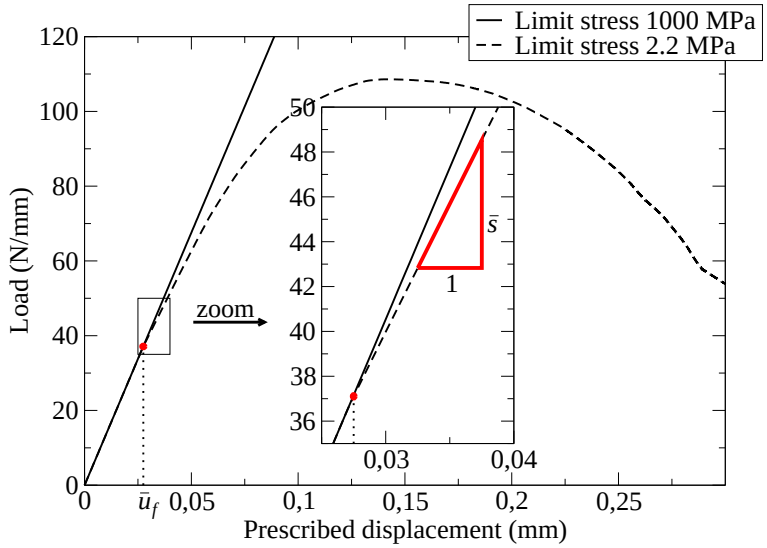


Figure 7.5: Measurements for objective function F_2 definition.

In our particular case, the slope $\bar{s}$ is approximated as a secant determined from two distinct points, see Figure 7.5. There are two contradictory requirements for that choice: first, the points should not be too close to $\bar{u}_f$ to ensure that numerical errors due to pseudo-time discretization do not exceed the impact of $\bar{K}$ parameter; second, they should be close enough to $\bar{u}_f$ to ensure that the specimen does not enter the softening regime and $\bar{\bar{\sigma}}_f$ will not be reached. If the second requirement is fulfilled the corresponding objective function depends only on values of the elastic limit stress $\bar{\sigma}_f$ and the hardening parameter $\bar{K}$, because Young's modulus and Poisson's ratio are fixed on the optimal values determined during the previous optimization stage. The particular choice of objective function adopted in this work is

$$\bar{s} = (L(\bar{u}_f + 0.01\text{mm}) - L(\bar{u}_f + 0.005\text{mm}))/0.005\text{mm} \tag{7.3}$$

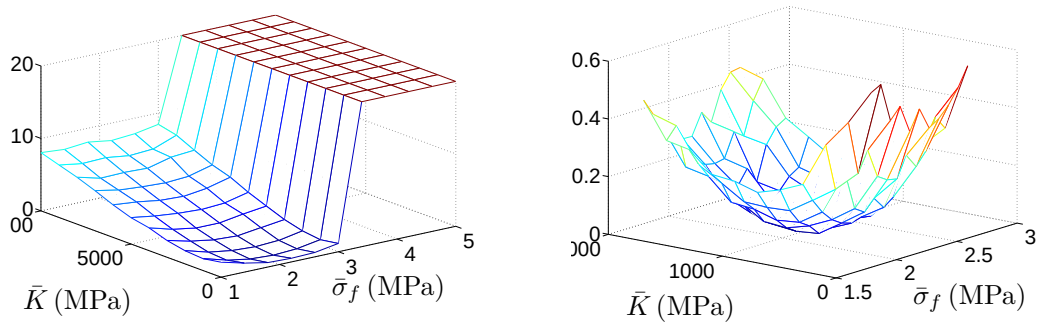
leading to the objective function in form

$$F_2 = (\bar{u}_{f,ref} - \bar{u}_f)^2 w_3 + (\bar{s}_{ref} - \bar{s})^2 w_4. \tag{7.4}$$

To keep this optimization step efficient, the simulations should again be restricted to a limited loading range, where the limit displacement can be related to the value of $\bar{u}_{f,ref}$ from the reference diagram. Note that during optimization process, there is no guarantee that $\bar{\sigma}_f$ will be exceeded when subjecting the specimen to the limit displacement. Such a solution is penalized by setting the objective function value to $10 \times N$, where $N = 2$ is the problem dimension, see Figure 7.6a. Moreover, as documented by Figure 7.6b, the objective function is now substantially noisier than for the elastic case and hence more difficult to optimize.

7.3.3 Identification of softening parameters

The last stage of identification involves the discrete damage parameters: the limit normal traction $\bar{\bar{\sigma}}_f$ and the softening parameter $\bar{\bar{\beta}}$. Variable $\bar{\bar{\sigma}}_f$ represents a limit of hardening of material



(a) (b)

Figure 7.6: Objective function F_2: (a) Whole domain (b) Detail close to optimum.

and the appearance of a macroscopic crack. Determination of displacement $\bar{\bar{u}}_f$ corresponding to this event, however, is rather delicate. The most straightforward method is based on the comparison of the reference curve with a simulation for a very high value of $\bar{\bar{\sigma}}_f$. The point where these two curves start to deviate then defines the wanted value of $\bar{\bar{u}}_f$, see Figure 7.8a. A more reliable possibility could be based on a local optical measurement in the vicinity of notch [Claire et al., 2004] or acoustic emission techniques [Chen and Liu, 2004]. In our computations we consider local measurements of notch upper corners displacements v_3 and v_4, see Figure 7.7. As demonstrated by graph 7.8b, the 'local' value of $\bar{\bar{u}}_f$ corresponds to the 'global' quantity rather well.

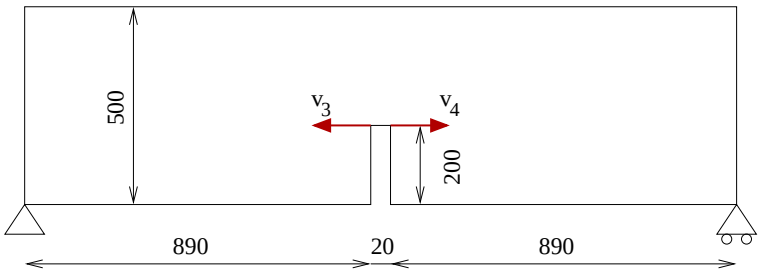


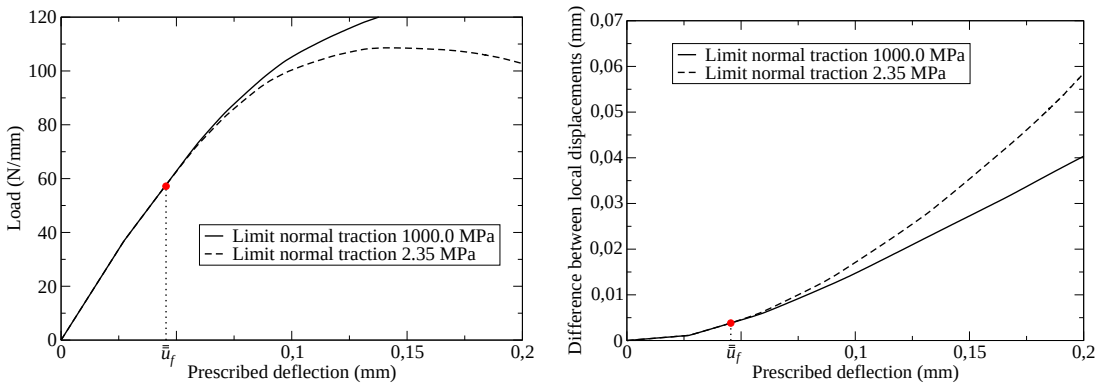
Figure 7.7: Displacement measured to express crack opening defined as $v_4 - v_3$.

After the specimen enters the softening regime, the shape of both local and global curves is fully governed by the softening parameter $\bar{\bar{\beta}}$. Therefore, its value can be fitted on the basis of the load corresponding to the deflection for which the softening is sufficiently active. This leads to the last objective function in the form

$$F_3 = \left(\bar{\bar{u}}_{f,ref} - \bar{\bar{u}}_f\right)^2 w_5 + \left(L_{ref}(u) - L(u)\right)^2 w_6 \quad ; \qquad u = 0.15mm. \tag{7.5}$$

Since all the other parameters are already fixed on the values determined during the previous optimization steps, this objective function depends again only on two parameters: the limit normal traction $\bar{\bar{\sigma}}_f$ and the softening parameter $\bar{\bar{\beta}}$.

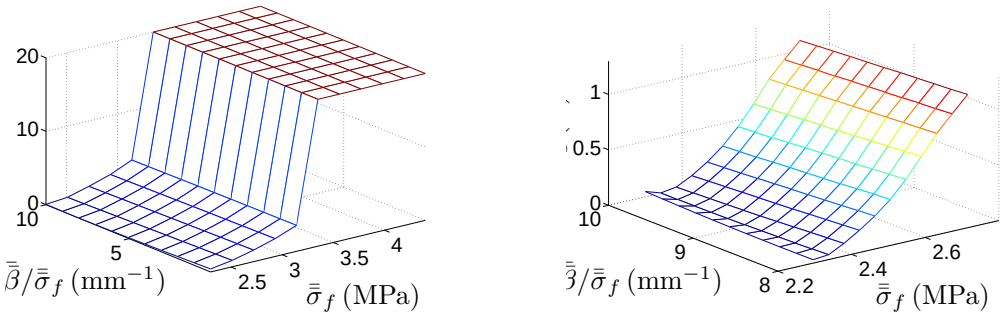
By analogy to the procedure described in the previous section, the penalization is applied to



(a) (b)

Figure 7.8: Comparison of diagrams with and without the spring of crack (a) Load-deflection diagram (b) Evolution of difference between chosen local displacements during the loading.

the cases where the specimen does not reach the softening regime until the end of simulations. This effect is well-visible from the surface of the objective function shown in Figure 7.9.



(a) (b)

Figure 7.9: Objective function F_3: (a) Whole domain (b) Detail close to optimal value.

7.4 Identification procedure verification

When assessing the reliability of the proposed identification procedure a special care must be given to stochastic nature of the optimization algorithm. The analysis presented herein is based on the statistics of 100 independent optimization processes, executed for each objective function. The termination criteria were set to:

- the number of objective function evaluations exceeded 155;
- the value of objective function smaller than a stopping precision was found.

F	Stopping precision	Successful runs	Maximal number of F's evaluation	Average number of F's evaluation
F_1	10^{-3}	100	32	16
F_2	10^{-2}	94	140	29
F_2	10^{-3}	80	140	47
F_3	10^{-2}	92	140	37
F_3	3.10^{-3}	76	143	47

Table 7.5: Summary of reliability study.

Parameter	Stopping precision on F	Average error [%]	Maximal error [%]
E	10^{-5}	0.41	1.23
ν	10^{-5}	0.16	2.20
$\bar{\sigma}_f$	10^{-2}	0.87	2.58
$\bar{K}$	10^{-2}	0.78	2.49
$\bar{\sigma}_f$	10^{-3}	0.30	0.59
$\bar{K}$	10^{-3}	0.49	1.54
$\bar{\bar{\sigma}}_f$	10^{-2}	0.47	1.32
$\bar{\bar{\beta}}$	10^{-2}	2.34	12.21
$\bar{\bar{\sigma}}_f$	3×10^{-3}	0.33	0.67
$\bar{\bar{\beta}}$	3×10^{-3}	0.26	2.68

Table 7.6: Influence of stopping precision on accuracy of identified parameters.

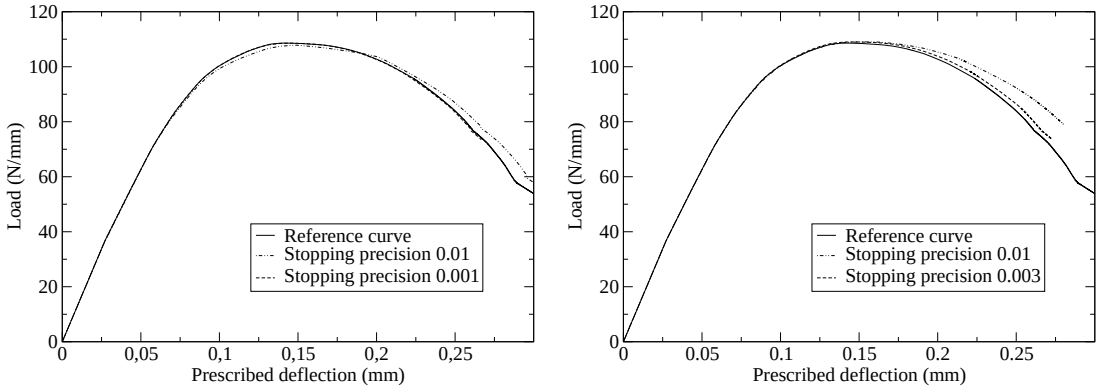
A particular optimization process was marked as 'successful', when the latter termination condition was met. Note that since the reference simulation instead of experimental data are used, the optimum for every objective function is equal to zero. Results of the performance study are summarized in Table 7.5 showing the success rate related to a stopping precision together with the maximum and average number of function evaluations.

Moreover, the different values of stopping precision allow us to investigate the relation between the accuracy of identified parameters and the tolerance of the objective function value. Table 7.6 shows a concrete outcome of such an analysis, where the maximal and average errors are calculated relatively to the size of the interval given by limit values for each parameter.

The results show that the maximal error for the elastic parameters E and ν is less than 3%, which is sufficient from the practical point of view. This is also documented by Figure 7.10, where no deviation of the reference and resulting curves is visible in the elastic range. For the hardening stage (parameters $\bar{\sigma}_f$ and $\bar{K}$) a similar precision is unfortunately not sufficient as shown by Figure 7.10a. Increasing the stopping precision to 10^{-3} reduces the error on parameters roughly by 50%, which is sufficient to achieve almost perfect fit of the reference curve.

Finally, a similar conclusion holds for the parameters $\bar{\bar{\sigma}}_f$ and $\bar{\bar{\beta}}$ describing the softening part of the experiment. The stopping precision equal to 10^{-2} is too coarse to achieve sufficient precision on parameters and needs to be reduced to 3×10^{-3}. The effect of increased accuracy

is then well visible in Figure 7.10b. This step completes the verification of the algorithm.



(a) (b)

Figure 7.10: Comparison of load-deflection diagrams: (a) Hardening parameters (b) Softening parameters.

7.5 Summary

We have proposed a sound identification procedure for material parameters of the constitutive model for representing the localized failure of massive structures. The most pertinent conclusions can be stated as follows:

i) The sequential identification approach employed for the uniaxial tensile test can be extended to the three-point bending test. The resulting algorithm is very straightforward and has a clear link with the structure of the constitutive model. Moreover, each of three stages uses only a part of the test simulation, which leads to substantial computational time savings.

ii) Due to the physical insight into the model, it was possible to construct simple objective functions F_1, F_2 and F_3 with a high sensitivity to the relevant parameters. This led to non-smooth and non-convex objective functions, which were optimized by robust soft-computing methods.

iii) The proposed identification procedure was verified on 100 independent optimization processes executed for each objective function. In the worst case, the reliability of the algorithm is 76% due to very small number of objective functions calls set in the termination condition. From our experience with evolutionary algorithms [Hrstka et al., 2003], such a result is rather satisfactory.

iv) As a result of a sequential character of the identification procedure, the errors in identified parameters accumulate. Therefore, the values need to be determined with higher accuracy then usually required in applications (i.e. 5%) and achievable by neural network-based inverse analysis [Kučerová et al., 2007].

v) The major difficulty of the proposed methods is to properly identify the three stages of structural behavior. From the point of view of method verification, where the reference load-deflection diagram is not noisy, the problem was successfully resolved. To fully accept the procedure, however, the experimental validation of the method appears to be necessary and will be subject of a future work.

Chapter 8

IDENTIFICATION OF MICROPLANE MODEL M4 PARAMETERS

> He who never made a mistake
> never made a discovery.
>
> Samuel Smiles

Concrete is one of the most frequently used materials in Civil Engineering. Nevertheless, as a highly heterogeneous material, it shows very complex non-linear behavior, which is extremely difficult to describe by a sound constitutive law. As a consequence, numerical simulation of response of complex concrete structures still remains a very challenging and demanding topic in engineering computational modelling.

One of the most promising approaches to modelling of concrete behavior is based on the microplane concept, see, e.g., [Jirásek and Bažant., 2001, Chapter 25] for general exposition and [Bažant and Caner, 2005] for the most recent version of this family of models. It leads a fully three-dimensional material law that incorporates tensional and compressive softening, damage of the material, supports different combinations of loading, unloading and cyclic loading along with the development of damage-induced anisotropy of the material. As a result, the M4 variant of the microplane model introduced in [Bažant et al., 2000] is fully capable of predicting behavior of real-world concrete structures once provided with proper input data, see e.g. [Němeček and Bittnar, 2004, J. Němeček and Bittnar, 2005] for concrete engineering examples. The major disadvantages of this model are, however, a large number of phenomenological material parameters and a high computational cost associated with structural analysis even in a parallel implementation [J. Němeček and Bittnar, 2002]. Although the authors of the model proposed a heuristic calibration procedure [Bažant et al., 2000, Part II], it is based on the trial-and-error method and provides only a crude guide for determination of selected material parameters. Therefore, a reliable and inexpensive procedure for the identification of these parameters is required. This Chapter present the sequential identification of microplane model parameters. More precisely, cascade neural networks (see Chapter 3) are developed in order to identify all dominant model parameters.

8.1 Microplane model M4 for concrete

In contrary to traditional approaches to constitutive modelling, which is based on description via second-order strain and stress *tensors* at individual points in the (x, y, z) coordinate sys-

tem, the microplane approach builds the descriptions on planes of arbitrary spatial orientations – so-called *microplanes*, related to a macroscopic point, see Figure 8.1. This allows to formulate constitutive equations in terms of stress and strain *vectors* in the coordinate system $(\mathbf{l}, \mathbf{m}, \mathbf{n})$ associated with a microplane oriented by a normal vector $\mathbf{n}$. The general procedure of evaluation of a strain-driven microplane model response for a given "macroscopic" strain tensor $\boldsymbol{\varepsilon}(\mathbf{x})$ can be described as follows: (i) for a given microplane orientation $\mathbf{n}$ normal "macroscopic" strain tensor $\boldsymbol{\varepsilon}(\mathbf{x})$ is projected onto the normal "microstrain" vector $\varepsilon(\mathbf{n})$ and the shear microstrains $\varepsilon(\mathbf{m})$ and $\varepsilon(\mathbf{l})$, (ii) the normal and shear microstresses $\sigma(\mathbf{n}), \sigma(\mathbf{m})$ and $\sigma(\mathbf{l})$ are evaluated using microplane constitutive relations, (iii) the "macroscopic" stress tensor $\boldsymbol{\sigma}(\mathbf{x})$ is reconstructed from the microscopic ones using the principle of virtual work, see, e.g., [Jirásek and Bažant., 2001, Chapter 25] for more details. In the particular implementation, 28 microplanes with a pre-defined orientation on the unit hemisphere is used to evaluate the response of the model.

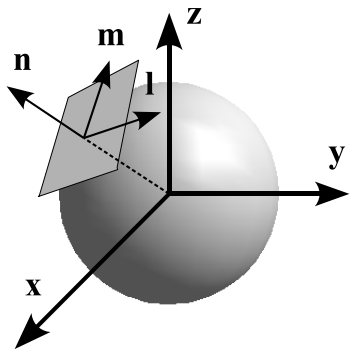


Figure 8.1: Concept of microplane modelling

To close the microplane model description, the appropriate microplane constitutive relation must be provided to realistically describe material behavior. The model examined in the current work is the microplane model M4 [Bažant et al., 2000]. The model uses volumetric-deviatoric split of the normal components of the stress and strain vectors, treats independently shear components of a microplane and introduces the concept of "boundary curves" to limit unrealistically high values predicted by earlier version of the model. As a result, the strain-to-stress map $\boldsymbol{\varepsilon}(\mathbf{x}) \mapsto \boldsymbol{\sigma}(\mathbf{x})$ is no longer smooth, which complicates the formulation of consistent tangent stiffness matrix [J. Němeček and Bittnar, 2002] and, subsequently, gradient-based approaches to material model parameters identification.

In overall, the microplane model M4 needs eight parameters to describe a certain type of concrete, namely: Young's modulus E, Poisson's ratio ν, and other six parameters (k_1, k_2, k_3, k_4, c_3, c_{20}), which do not have a simple physical interpretation, and therefore it is difficult to determine their values from experiments. The only information available in the open literature are the bounds shown in the Table 8.1.

In the present work, the computational model of a structure is provided by the object-oriented C++ finite element code **OOFEM 1.5** [Patzák and Bittnar, 2001, Patzák, WWW]. Spatial discretization is performed using linear brick elements with eight integration points. The

Parameter		Bounds
E	$\in$	$\langle 20.0, 50.0 \rangle$ GPa
ν	$\in$	$\langle 0.1, 0.3 \rangle$
k_1	$\in$	$\langle 0.00008, 0.00025 \rangle$
k_2	$\in$	$\langle 100.0, 1000.0 \rangle$
k_3	$\in$	$\langle 5.0, 15.0 \rangle$
k_4	$\in$	$\langle 30.0, 200.0 \rangle$
c_3	$\in$	$\langle 3.0, 5.0 \rangle$
c_{20}	$\in$	$\langle 0.2, 5.0 \rangle$

Table 8.1: Bounds for the microplane model parameters

arc-length method with elastic stiffness matrix is used to determine the load-displacement curve related to the analyzed experiment.

8.2 Sequential identification - verification

This section summarizes the individual steps of M4 material model identification. Following the heuristic calibration procedure suggested in [Bažant et al., 2000, Part II], we examine three specific experimental tests: (i) uniaxial compression, (ii) hydrostatic test and (iii) triaxial test. Advantage of these tests is their simplicity and availability in most experimental facilities. Moreover, authors in [Bažant et al., 2000] claim that these experiments are sufficient to determine all parameters of the microplane model M4. The results presented in this section can be understood as a verification of this claim.

The large number of identified parameters complicates the development of a universal inverse model capable to predict all these parameters with sufficient precision. Therefore, the cascade neural networks are created in a sequential way, where the previously predicted parameters are

i) either used as inputs to neural network created in following step;

ii) or fixed on the predicted values for simulations performed in order to prepare the training data for neural network in following step.

Each neural network is obtained by the procedure described in Chapter 5. More precisely, a fully connected two-layer perceptron with bias neurons is applied to map discrete values from stress-strain diagrams to microplane model parameters. Log-sigmoid functions defined by Equation (5.4) are consider as activation function in all neurons. The genetic algorithm GRADE (see Section 4.1.2 extended by CERAF strategy (see Section 4.1.3) is used for training the networks.

The error function optimized in the ANN's training process is defined as

$$E(w_{l,i,j}) = \max_{ntr} \left\{ \left\| \mathbf{x}^M - \tilde{\mathbf{x}}^M \right\| \right\}, \tag{8.1}$$

for all following calculations. ntr denotes the number of training pairs.

Training samples are generated by Latin Hypercube Sampling and optimized by Simulated Annealing in order to minimize the correlation among all samples. Those methods are implemented in FREET software [Novák et al., 2003]. Relevant values chosen from stress-strain diagram as ANN's inputs are chosen by hand, but with respect to the results of stochastic sensitivity analysis. For the latter one, the Pearson product-moment correlation coefficient defined by Equation (3.5) is evaluated between the discrete values of stresses (or strains) and corresponding values of microplane model parameters.

8.2.1 Uniaxial compression test

The most common experiment used for the determination of concrete parameters is the uniaxial compression test on cylindrical concrete specimens. In particular, the cylinder with a radius equal to 75 mm and the height of 300 mm is used. The set-up of the experiment, the finite element mesh as well as the deformed configuration predicted by the computational model are shown in Figure 8.2.

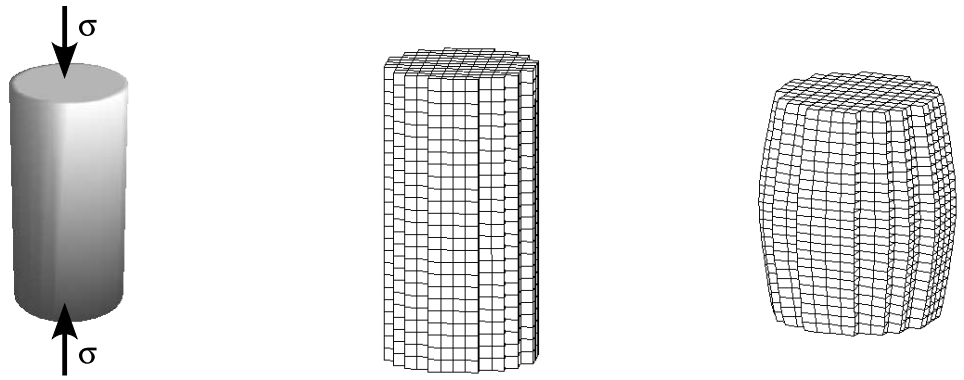


Figure 8.2: Uniaxial test. (a) Experiment setup, (b) Finite element mesh, (c) Deformed mesh

The LHS sampling procedure has been used to determine the set of 30 simulations resulting in a "bundle" of stress-strain curves shown in Figure 8.3. The evolution of stochastic sensitivity during the loading process is depicted in Figure 8.4. The results indicate that the most sensitive parameters are Young's modulus E, the coefficient k_1, Poisson's ratio ν (especially for the initial stages of loading) and, for the later stages of loading, the coefficient c_{20}. Therefore, one can expect that only these parameters can be reliably identified from this test.

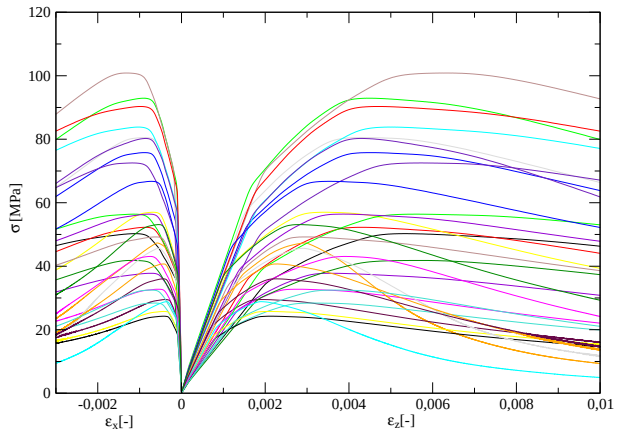


Figure 8.3: Bundle of simulated stress-strain curves for uniaxial compression test

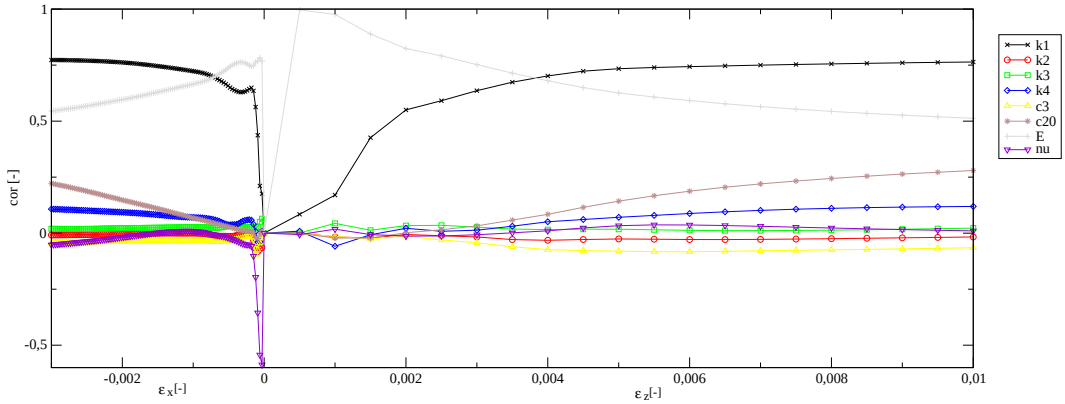


Figure 8.4: Sensitivity evolution for uniaxial compression test

Moreover, the impact of individual parameters on a position of a peak of stress-strain curves is computed. The results of a sensitivity analysis using Pearson's product moment correlation coefficient of peak coordinates [ϵ,σ] are listed in Table 8.2. Results indicate particularly strong influence of the k_1 parameter, which hopefully allows its reliable determination.

Based on the results of sensitivity analysis, the neural network training can be performed using a nested strategy. First, *Young's modulus* E with sensitivity ≈ 1 in the initial stage is easily identified. To this end, the following ANN's inputs are chosen: the values of stresses $\sigma_{z,1}$, $\sigma_{z,2}$ and $\sigma_{z,3}$ in the first three points of axial strain with extremal Pearson's correlation coefficient. The hidden layer contains two neurons only; the output layer consists of one neuron corresponding to the predicted value of Young's modulus E. For the ANN training the GRADE algorithm is used and calculation is stopped after 1,000,000 iterations of the algorithm.

The two-layer ANN trained using the GRADE algorithm is also used for the identification of other microplane model parameters. In Table 8.3 network's architectures and the choice of input values for the identification of each microplane parameter are presented.

The results of the identification for an independent set of ten stress-strain curves using the

Parameter	Pearson's coefficients	
	ϵ	σ
k_1	0.968	0.709
k_2	0.025	0.008
k_3	0.015	0.030
k_4	0.021	0.074
c_3	-0.019	-0.020
c_{20}	0.158	0.041
E	0.004	0.684
ν	0.129	0.000

Table 8.2: Pearson's coefficient as a sensitivity measure of individual parameters to the peak coordinates [ϵ,σ] of stress-strain curves

Parameter	ANN's layout	Input values
k_1	4 - 2 - 1	$\sigma_{x,2}$, $\sigma_{z,peak}$, $\epsilon_{z,peak}$, prediction of E
k_2	4 - 2 - 1	$\sigma_{z,20}$, $\sigma_{z,peak}$, $\epsilon_{z,peak}$, prediction of E
k_3	4 - 2 - 1	$\sigma_{z,20}$, $\sigma_{z,peak}$, $\epsilon_{z,peak}$, prediction of E
k_4	4 - 2 - 1	$\sigma_{z,20}$, $\sigma_{z,peak}$, $\epsilon_{z,peak}$, prediction of E
c_3	4 - 2 - 1	$\sigma_{z,20}$, $\sigma_{z,peak}$, $\epsilon_{z,peak}$, prediction of E
c_{20}	4 - 3 - 1	$\sigma_{x,100}$, $\sigma_{z,20}$, prediction of E, prediction of k_1
E	3 - 2 - 1	$\sigma_{z,1}$, $\sigma_{z,2}$, $\sigma_{z,3}$
ν	4 - 3 - 1	$\sigma_{x,1}$, $\sigma_{x,2}$, prediction of E, prediction of k_1

Table 8.3: Neural network architectures

proposed strategy are shown in Table 8.4.

Parameter	Absolute error		Relative error [%]	
	average	maximal	average	maximal
k_1	2.058e-06	4.678e-06	1.34	**2.76**
k_2	138.9	318.7	38.92	179.62
k_3	2.679	5.283	33.96	102.33
k_4	52.70	91.70	48.33	107.17
c_3	1.675	2.278	37.66	47.09
c_{20}	0.7547	1.4168	26.70	56.69
E	229.3	594.5	0.74	**1.79**
ν	0.006447	0.010361	2.93	**4.72**

Table 8.4: Errors in the estimated parameters obtained from ten independent tests

Note that obtained errors are in a close agreement with the results of sensitivity analysis. Except E, ν and k_1, the parameters of the model are identified with very high error values. Therefore, additional simulations are needed to obtain these values reliably.

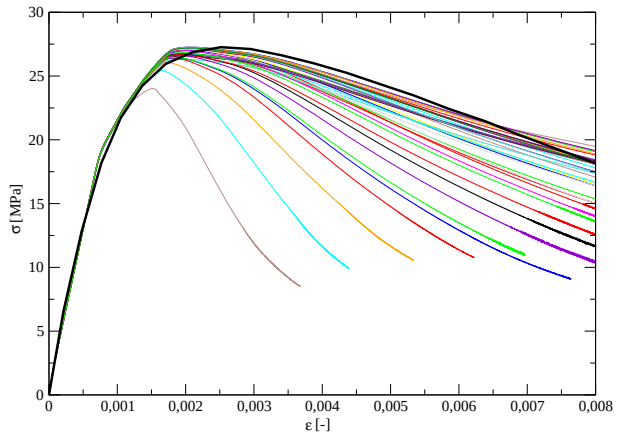


Figure 8.5: Bundle of simulated stress-strain curves for uniaxial compression with fixed values of Young's modulus, Poisson's ratio and k_1 parameter and one (bold black) measured stress-strain curve

At this point we have to fix already well-identified parameters to the optimized values and perform simulations for the rest of parameters. To minimize computational time, values for one uniaxial measurement presented later in Section 8.3.1 were used. Corresponding values predicted by previously learned neural networks were Young's modulus $E = 32035.5$ MPa and $k_1 = 0.000089046$. Poisson's ratio is set to $\nu = 0.2$ as a usual value of a wide range of concretes. Next, 40 new simulations varying the rest of unknown parameters are computed. From this suite, only 34 solutions are valid, i.e. these solutions were able to reach the desired value of axial strain $\varepsilon = 0.008$. The bundle of resulting curves is shown in Figure 8.5. Note that the black bold curve represents measured data. The evolution of Pearson's correlation coefficient during the experiment is shown in Figure 8.6.

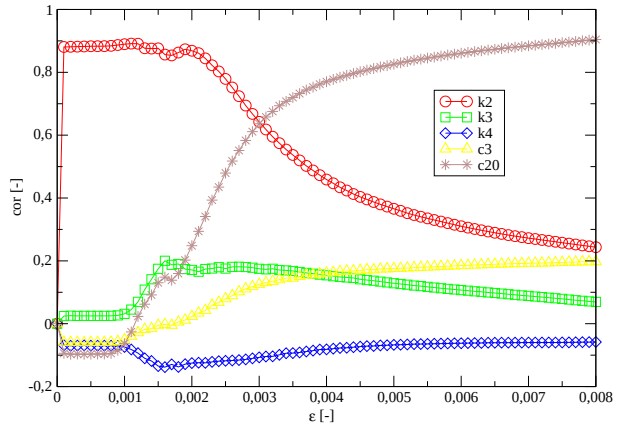


Figure 8.6: Evolution of Pearson's correlation coefficient during the loading test for fixed values of E, ν and k_1 parameters

The sensitivity shows very high influence of k_2 parameter at the beginning of the loading. If we inspect Figure 8.7, it is clear that the k_2 parameter influences the stress-strain curve only on a very narrow interval and hence the parameter k_2 cannot be identified from this test (the au-

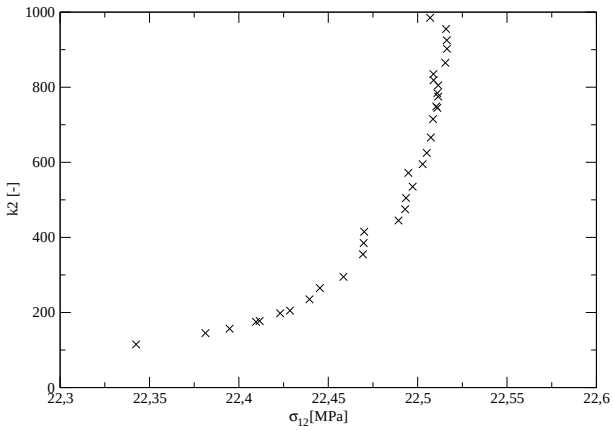


Figure 8.7: k_2 parameter as a function of the stress σ_{12} (corresponding to $\epsilon = 0.0011$)

thors of the microplane model indeed proposed k_2 to be estimated from the triaxial compression test). We are more interested in fitting data in the post-peak part. For post-peak curves, sensitivity analysis shows especially growing influence of c_{20} parameter. This can be demonstrated by a relation between the c_{20} parameter and a value of a stress (σ_{81}) at the end of our simulations, where the correlation coefficient reaches the value 0.904429. This relation is graphically illustrated in Figure 8.8.

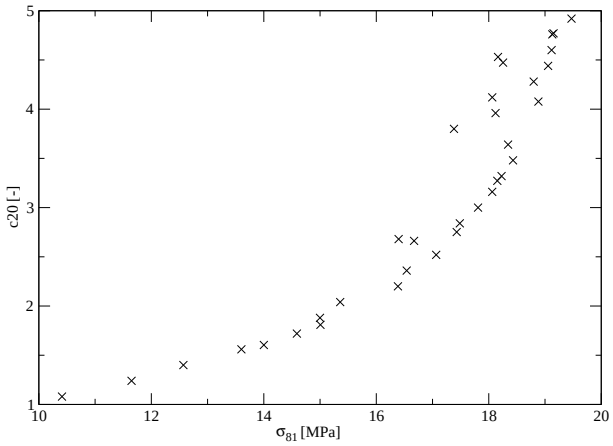


Figure 8.8: The c_{20} parameter as a function of a stress (σ_{81}) at the end of simulations

It is clearly visible that this relation is highly non-linear and any simple regression of this data is hardly possible. We applied the ANN with 3 input neurons chosen to get the best prediction of parameters based on post-peak curves. Therefore, one input value is a stress value at the peak σ_{peak} and the other two inputs are stress values σ_{61} and σ_{81} corresponding to strains $\epsilon = 0.006$ and $\epsilon = 0.008$, respectively. Two neurons in the hidden layer were used. Quality of ANN prediction is demonstrated in Figure 8.9. In particular, the exact prediction of the searched value corresponds to a point lying on the axis of the first quadrant. Values of predicted parameters are normalized here to the interval $\langle 0.15, 0.85 \rangle$. Dashed parallel lines bound a 5% relative error related to the size of the parameter's interval. Clearly, the identification procedure works with an error which does not exceed the selected 5% tolerance.

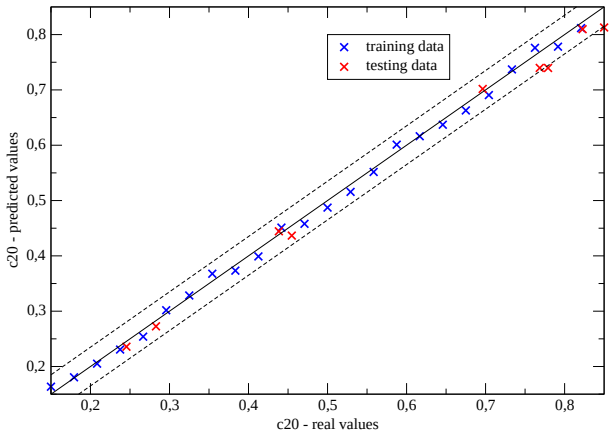


Figure 8.9: Quality of ANN predictions of c_{20} parameter

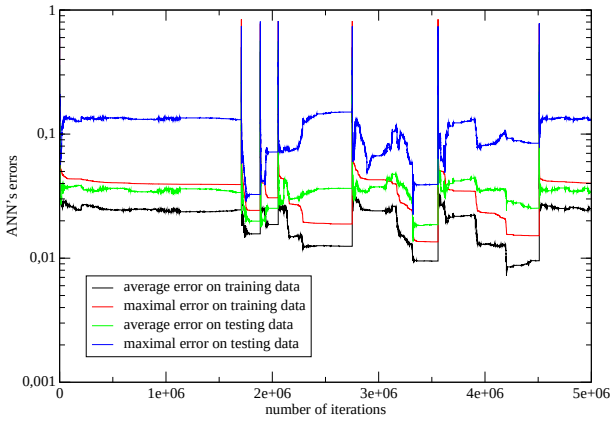


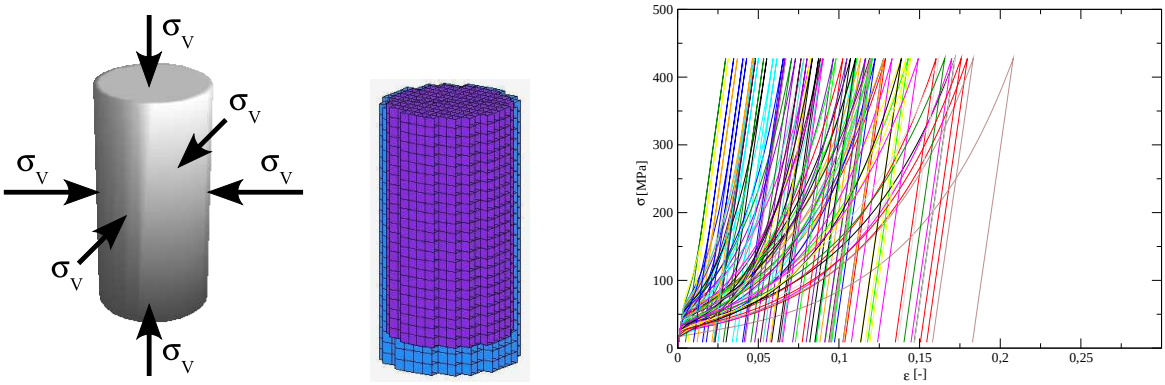
Figure 8.10: Evolution of ANN's errors during the training in prediction of c_{20} parameter

The attention is also paid to the over-training of the ANN. To control this aspect, the evolution of errors in ANN's predictions during the training process on the training and testing data are monitored (see Figure 8.10). Recall that if the errors on the testing set are much higher than on the training set, we suppose such an ANN to be over-trained. Even though this seems to be the case of the current ANN, we attribute such a behavior to the fact that there are more training data (in our case 25) then 11 neural weights optimized by the algorithm. Also note that the typical restarting of the optimization process caused by the multi-modal optimization strategy CERAF presented in Section 4.1.3, is clearly visible in Figure 8.10.

8.2.2 Hydrostatic compression test

The next independent test used for the identification problem is the hydrostatic compression test, where a concrete cylinder is subjected to an increasing uniform pressure. Axial and two lateral deformations (in mutually perpendicular directions) are measured. The experimental setup is shown in Figure 8.11a–b. Again, to improve identification precision, the parameters E, ν and k_1 are supposed to be fixed to the previously identified values. The "bundle" of stress-strain

curves obtained using the LHS sampling for 70 samples is depicted in Figure 8.11c and the corresponding sensitivity evolution in Figure 8.12. Note that the maximal value of a hydrostatic pressure for all these tests is 427.5 MPa.



(a) (b) (c)

Figure 8.11: Hydrostatic test. (a) Experiment setup, (b) Initial and deformed finite element mesh, (c) Stress-strain curves

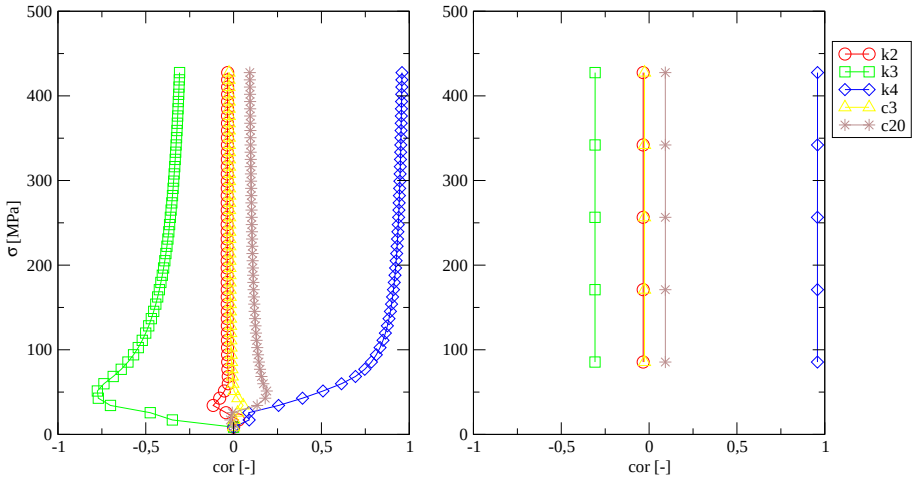


Figure 8.12: Evolution of Pearson's correlation coefficient during the hydrostatic compression test for loading (left) and unloading (right) branch

The sensitivity information reveal that this test can be used to identify parameter k_3 from the loading branch while a combination of loading and/or unloading data can be used for k_4 parameter identification. Moreover, the correlation between the strain at the peak of curves and k_4 parameter is so high that one can expect their relation to be almost linear. This is, however, not the case as illustrated by Figure 8.13 showing the value of k_4 parameter as a function of a strain ε. In spite of a high value of the correlation coefficient equal to 0.958586, the noise of these data seems to be very high and a use of a linear regression introduces a high error in k_4 parameter prediction.

To identify the k_3 parameter, which shows high correlation near to the end of elastic stage,

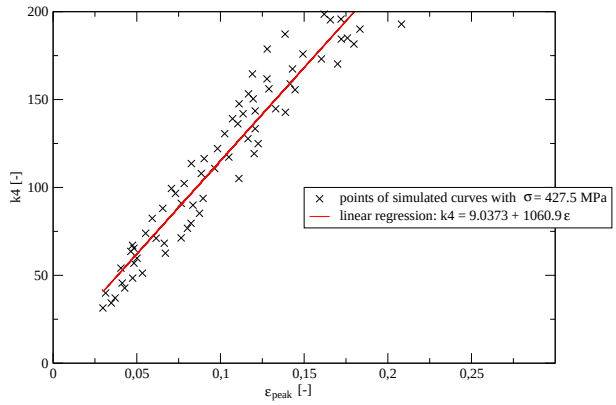


Figure 8.13: k_4 parameter as a function of a strain of a peak

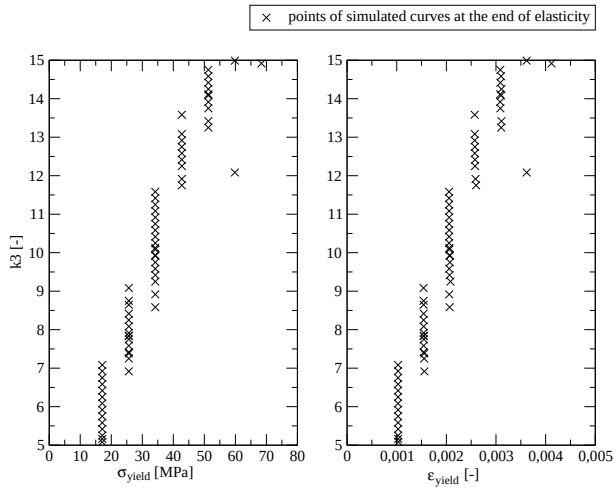


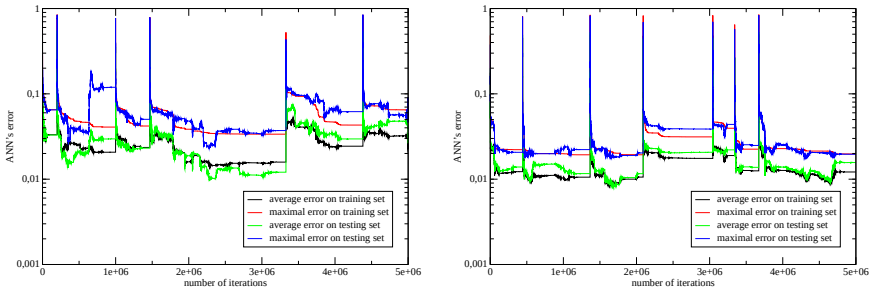
Figure 8.14: k_3 parameter as a function of a position of the end of an elastic stage

we evaluate the correlation coefficient between this parameter and a position of the end of an elastic stage. k_3 and ϵ_{yield} correlation is 0.95078 and k_3 and σ_{yield} = 0.951873. In Figure 8.14, values of k_3 parameter as a function of these coordinates are presented. The discrete values of ϵ_{yield} are caused by the size of a time step at the beginning of simulations. The noise in data is very high again and it is not possible to reliably use a linear regression. Because all other parameters have very small correlation, we suppose that the noise of the parameter k_3 is caused by k_4 parameter and vice-versa. In other words, these noises could be caused by some level of correlation between the parameters k_3 and k_4. Hence we decided to apply an artificial layered neural network again. The first 60 simulations prepared by the LHS method were used for training and remaining (randomly chosen) 10 simulations for testing. Particular choice of input values as well as architectures of ANN's is shown in Table 8.5. To eliminate unknown correlation between parameters k_3 and k_4, their values are used also as inputs into ANNs.

The architectures of ANNs were chosen manually to get the best precision in predictions and also to avoid the over-training of the ANN's. Therefore, it is possible to show the evolution of ANN's errors (see Figure 8.15) during the training. A training process with 5000000 iterations

Parameter	ANN's layout	Input values
k_3	5 - 2 - 1	k_4, ϵ_{yield}, $\epsilon_{load,2}$, $\epsilon_{load,5}$,ϵ_{peak}
k_4	3 - 2 - 1	k_3, ϵ_{peak}, $\epsilon_{unload,4}$

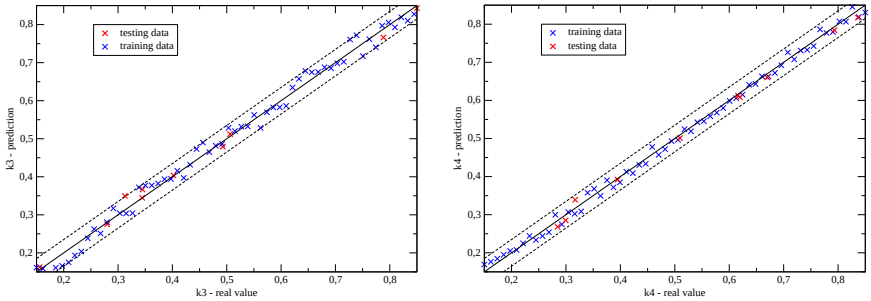
Table 8.5: Neural network architectures for hydrostatic test



(a) (b)

Figure 8.15: Evolution of ANN's errors during the training process in prediction of (a) k_3 parameter and (b) k_4 parameter

takes approximately 20 minutes. Quality of ANN's predictions is demonstrated in Figure 8.16. Values of predicted parameters are again normalized into the interval $\langle 0.15, 0.85 \rangle$.



(a) (b)

Figure 8.16: Quality of ANN prediction of (a) k_3 parameter and (b) k_4 parameter

In this way, two ANNs or, in other words, two implicit functions are prescribed. One defines a value of k_3 parameter depending on a value of k_4 parameter and some other properties of a stress-strain curve, the second define a value of k_4 parameter depending on a value of k_3 parameter and some other properties of a stress-strain curve. Once we get some "measured" data and we fix all properties of a stress-strain curve, we get a system of two non-linear equations for k_3 and k_4. We can solve this system, e.g., graphically. Both relations are depicted in Figure 8.17 for one independent stress-strain curve ($k_3 = 7.84293$, $k_4 = 155.551$). Their intersection defines searched parameters, $k_3 = 8.15687$ and $k_4 = 154.072$ in this particular case. The precision of the proposed strategy is visible in comparison of corresponding stress-strain curves, see Figure 8.18. Note, that E, ν and k_1 were the same as in previous section and remaining parameters, i.e. k_2, c_3 and c_{20}, were chosen randomly.[1]

[1] Theoretically, the value of c_{20} is known in this stage from the previous identification step. Nevertheless, the

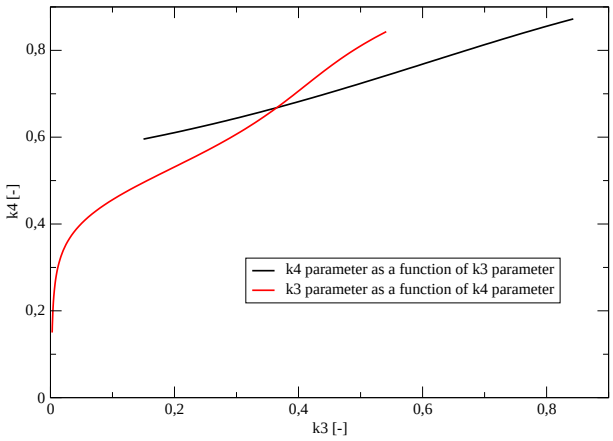


Figure 8.17: Relations of k_3 and k_4 parameters

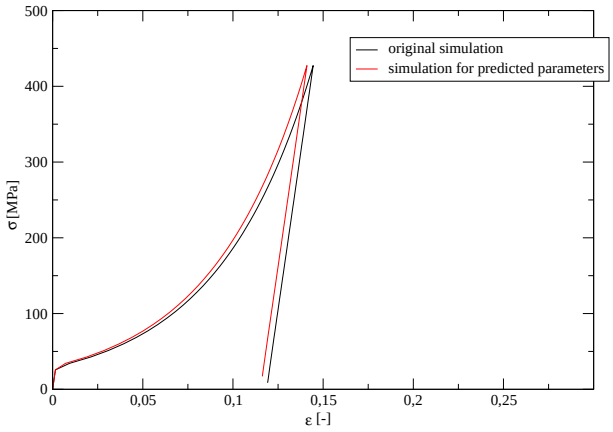


Figure 8.18: Comparison of original simulation and simulation for predicted k_3 and k_4 parameters

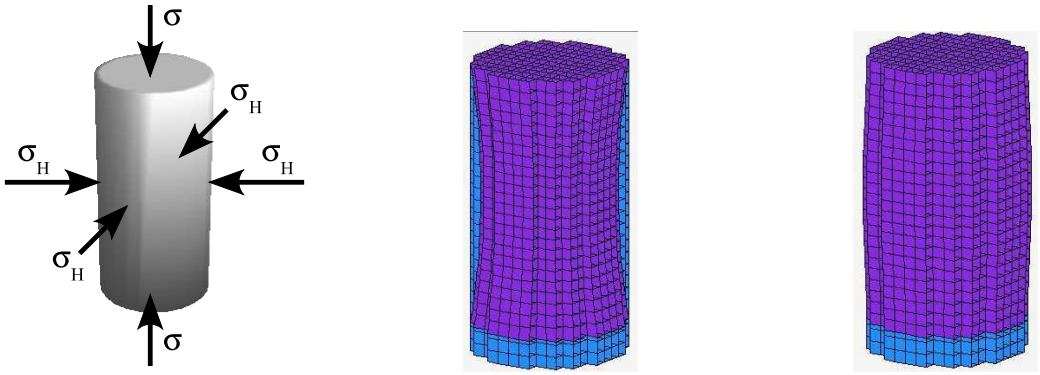
8.2.3 Triaxial compression test

The last experiment, used for the purpose of parameter identification, is a triaxial compression test. To this end, a specimen is subjected to the hydrostatic pressure σ_H. After the peak value of σ_H is reached, the axial stress is proportionally increased. The "excess" axial strain $\varepsilon = \varepsilon_T - \varepsilon_H$, where ε_T and ε_H denote the total and hydrostatic axial strain, is measured as a function of the overall stress σ. The test setup is shown in Figure 8.19.

At this point, we assume that parameters E, ν, k_1, k_3 and k_4 are known from previous identifications[2]. Next, 70 simulations (60 training and 10 testing) of the triaxial compression test are computed by varying three remaining parameters k_2, c_3 and c_{20}. The bundle of stress-strain

numerical simulation of the unidirectional experiment takes substantially more time to complete, see Section 8.4 for an example, and therefore the independence of c_{20} allows us to proceed with the inverse analysis even though the first phase is not finished.

[2] i.e. $E = 32035.5$ MPa, $\nu = 0.2$, $k_1 = 0.000089046$, $k_3 = 8.15687$ and $k_4 = 154.072$.



(a) (b) (c)

Figure 8.19: Triaxial compression test. (a) Experiment setup, (b) Initial and deformed mesh at the end of hydrostatic loading, (c) Initial and deformed mesh at the end of total loading

curves for $\sigma_H = 34.5$ MPa is shown in Figure 8.20 together with the evolution of Pearson's correlation coefficient during the experiment in Figure 8.21.

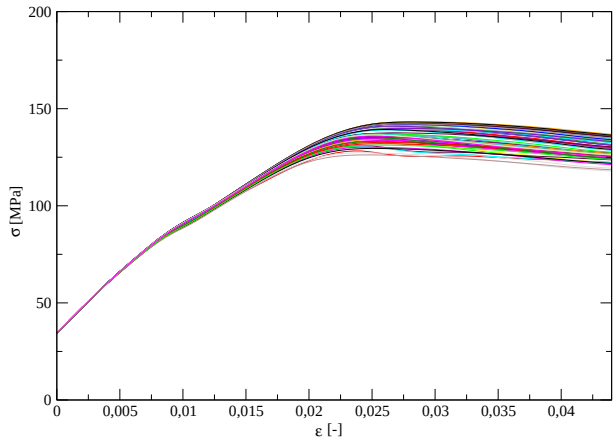


Figure 8.20: Bundle of simulated stress-strain curves for triaxial compression test

Parameter	**Pearson's coefficient** ϵ	σ
k_2	0.585	0.791
c_3	-0.067	-0.088
c_{20}	0.664	0.329

Table 8.6: Pearson's coefficient as a sensitivity measure of individual parameters to the peak coordinates [ϵ,σ] of stress-strain curves

In addition, the correlation coefficient between microplane parameters and stress and strain values of peaks is computed. These results are shown in Table 8.6. It is visible that maximal correlation is between k_2 parameter and the value of the stress σ_{29} corresponding to the strain

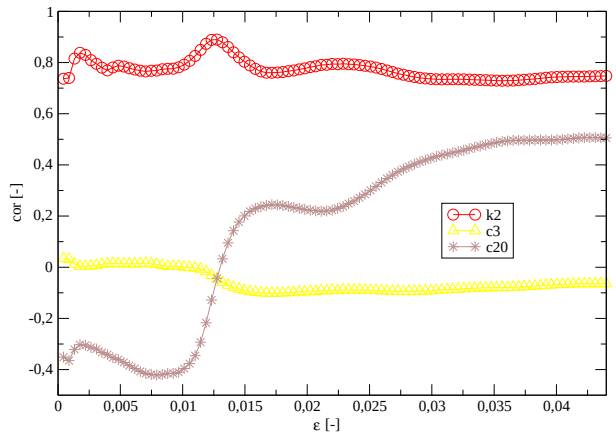


Figure 8.21: Evolution of Pearson's correlation coefficient during the triaxial compression test

equal to $\epsilon_{29} = 0.01276$. This correlation is 0.88956 and at the same time, the correlation between these σ_{29} values and c_{20} parameter is very small, therefore c_{20} parameter does not influence the relation between k_2 parameter and σ_{29}. Figure 8.22 shows that only small values of c_{20} parameter disturb this relation. In particular, points related to the c_{20} parameter smaller then 1 lead to oscillatory dependence.

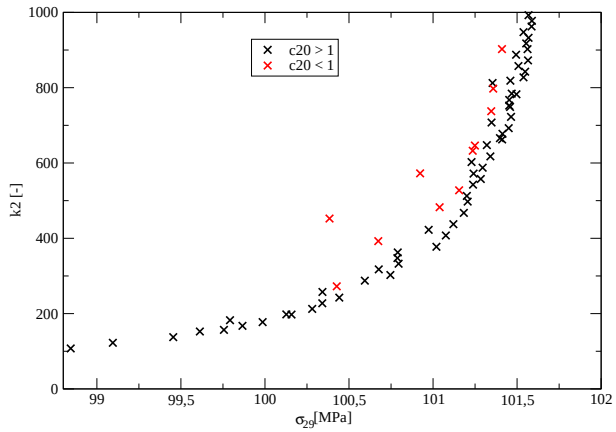


Figure 8.22: k_2 parameter as a function of the stress value σ_{29}

Because the highest correlation for k_2 parameter is again at the beginning of the loading and after our experiences in identification parameters from the uniaxial compression test, we were again afraid of small significance of k_2 parameter to the shape of curves. Also σ_{29} does not seem to be significant. Therefore we made several short computations with randomly chosen fixed value of c_{20} parameter to filter out its influence. We have got a bundle of curves showing similar spread of values as curves in Figure 8.20. Therefore, it can be concluded that these differences are probably caused by k_2 parameter only and a neural network for k_2 parameter identification can be designed. Because the bundle of curves varies mostly in the post-peak part and we would like to get a predictor capable to fit this part of a curve properly, we use σ_{peak} and σ_{100} as input values. The latter one correspond to the end of our simulations, where $\epsilon = 0.044$. We also add the third input value – σ_{29} – because of its small correlation with c_{20} parameter.

Two neurons in the hidden layer were used. Quality of the ANN prediction is demonstrated in Figure 8.23. Under- and over-fitting issues were again checked by errors evaluations during the training process, see Figure 8.24.

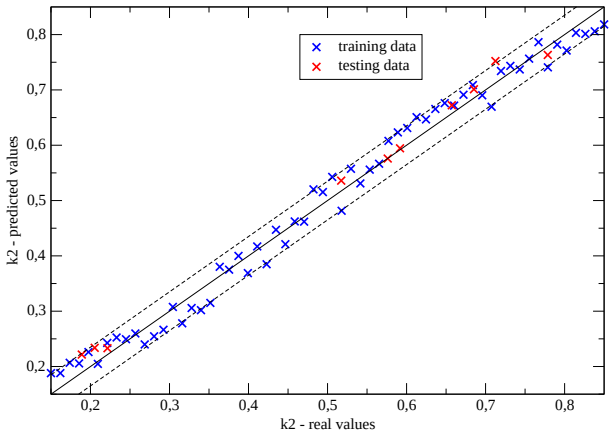


Figure 8.23: Quality of ANN prediction of k_2 parameter.

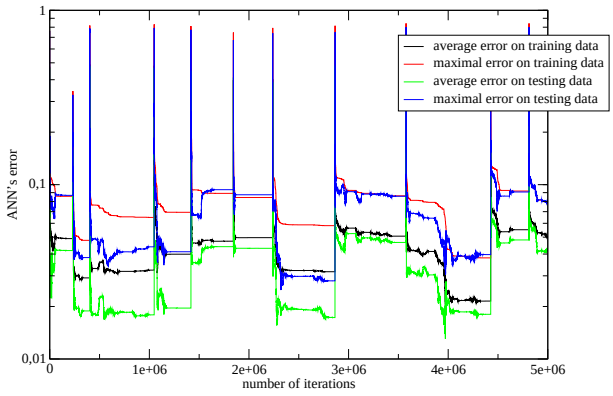


Figure 8.24: Evolution of ANN's errors during the training in prediction of k_2 parameter

Almost perfect precision of the proposed strategy is visible in comparison of corresponding stress-strain curves for $k_2 = 748.857$ and its prediction equal to 767.777 and randomly chosen parameters c_{20} and c_3, see Figure 8.25.

8.3 Application to measured data - validation

In previous sections, we have shown that the proposed methodology is able to identify all but one (c_3) parameters from computer-simulated curves. To demonstrate the applicability of the proposed procedure, a real simulation should be examined.

However, only limited experimental data from uniaxial compression tests are available to author which leaves us with only one uniaxial stress-strain curve to be identified. Other mea-

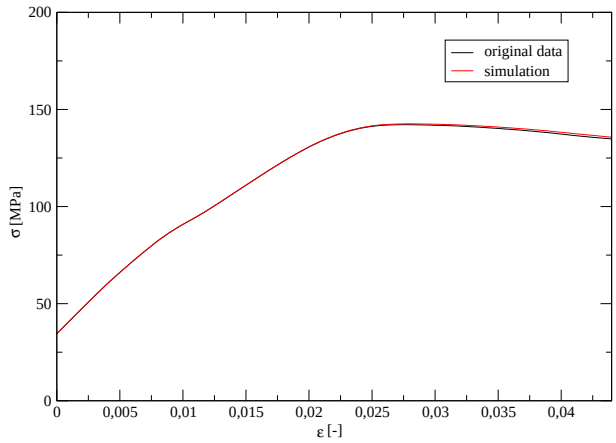


Figure 8.25: Comparison of original simulation and simulation for predicted parameters of triaxial compression test

surements from hydrostatic compression test and triaxial compression test are available in literature, e.g. in [Bažant et al., 2000, PartII].

In this section will be shown the results of proposed identification strategy applied to measured data. Since the measurements from different loading tests are obtained for different concretes, this Section does not represent a validation of proposed identification strategy in general, but only the validation of application of particular inverse models.

8.3.1 Uniaxial compression test

As was mentioned previously in Section 8.2.1, Young's modulus $E = 32035.5$ MPa, Poisson's ratio $\nu = 0.2$ and $k_1 = 0.000089046$ are predicted by the neural network for a particular real measurement. Next, 30 simulations are computed varying the parameter c_{20}, see Figure 8.5. If we zoom into the loading part of a stress-strain curve, it is clear, that the real measurement is far from all simulated data, see Figure 8.26. This part is influenced by high correlation of k_2 parameter and therefore, it is clear that the k_2 parameter cannot be obtained from this test. Finally we applied trained ANN to predict the c_{20} parameter for the measured data and the obtained values was $c_{20} = 5.27065$. This value is out of the interval specified for this parameter, but it is not surprising since it is visible in Figure 8.5 that measured data somewhat deviate from the simulated bundle of curves. The final comparison of measured data and a simulation for predicted values of E, ν, k_1 and c_{20} parameters is shown in Figure 8.27. The rest of unknown parameters are same as in previous sections.

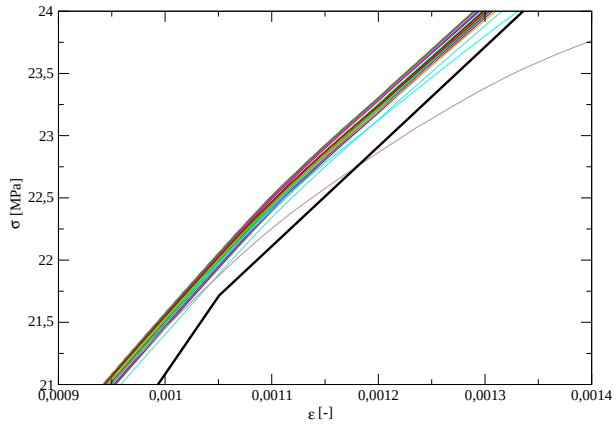


Figure 8.26: Bundle of simulated stress-strain curves for uniaxial compression and one (bold black) measured stress-strain curve under zoom

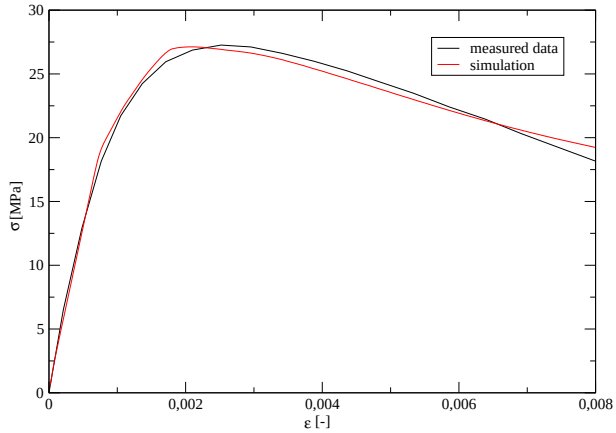


Figure 8.27: Comparison of measured data and results of final simulation.

8.3.2 Hydrostatic compression test

Experimental data from hydrostatic compression test could be found e.g. in [Bažant et al., 2000, Part II]. These data are obtained by authors Green and Swanson (1973). The stress-strain diagram represent the relation of hydrostatic pressure σ and axial deformation ε. The results from simulation perform for microplane model parameters obtained by trial-and-error method are there also available in comparison with measured data. Since we suppose that values of Young's modulus, Poisson's ratio and k_1 parameter could be reliably obtained from identification of uniaxial compression test (these results are nevertheless not available for the concrete observed here), the values of these parameters are taken directly from the article [Bažant et al., 2000, Part II]. The goal is then to identify values of parameters k_3 and k_4.

For neural network training, 60 simulations were performed using the data generated by FREET software [Novák et al., 2003] and 10 independent simulations for randomly chosen parameters were prepared for ANN's testing. The bundle of resulting stress-strain diagrams could be compared with measured data in Figure 8.28.

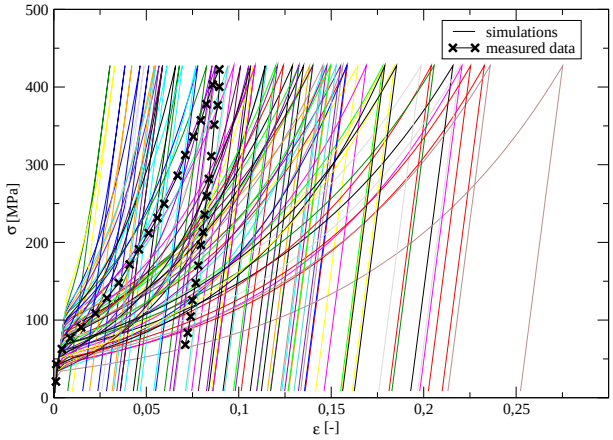


Figure 8.28: Comparison of measured data and results of 70 simulations of hydrostatic compression test.

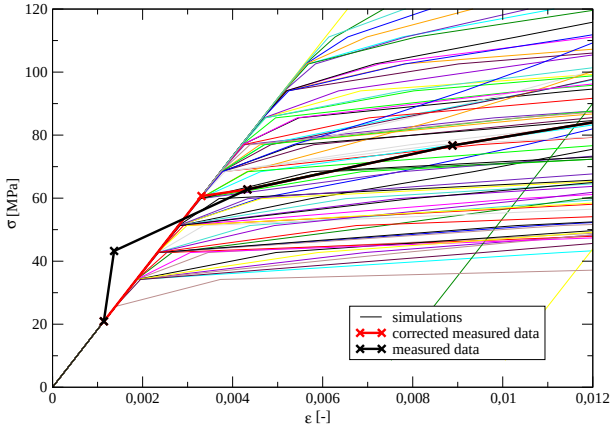


Figure 8.29: Detail in comparison of measured data and results of 70 simulations of hydrostatic compression test.

In Figure 8.29 is shown in detail the elastic part of stress-strain diagrams. It is clearly visible, that all the simulations are equal in the elastic part, i.e. there no influence of k_3 or k_4 parameter in this stage of loading, which is governed by Young's modulus, Poisson's ratio and k_1 parameter. It is also visible that their values were probably established such that the obtained simulations cross the first point in experimental data. The second point is nevertheless remote from all simulations. This is caused probably by the noise in experimental data or by an error in experiment. Therefore this point in experimental data was manually "corrected" as it is shown in Figure 8.29.

For k_4 parameter, the same ANN was trained as described in Section 8.2.2. The inputs to ANN for prediction of k_3 parameter were chosen in a little bit different way in order to put more importance to the shape of post-elastic loading part of stress-strain diagram. Two neural networks were trained with the inputs and the topology described in Table 8.7. Both neural networks have among the inputs the value of k_4 parameter and values of deformation at the end of the loading and at the end of the elastic part of the diagram. For the first ANN is then

	Topology	Inputs
ANN1	$4+2+1$	k_4, ϵ_{peak}, ϵ_{yield}, $\epsilon_{load,25}$
ANN2	$5+2+1$	k_4, ϵ_{peak}, ϵ_{yield}, $\epsilon_{load,16}$, $\epsilon_{load,36}$

Table 8.7: Description of two neural networks trained to predict k_3 parameter

parameter	Training data		Testing data	
	Average error	Maximal error	Average error	Maximal error
k_3	1.40	2.59	1.71	3.07
k_4	1.51	2.52	1.21	2.13

Table 8.8: Error in ANN's predictions relative to the definition interval of the parameters in [%].

added the value of deformation in the middle of the loading part of the diagram, whereas for the second ANN are added two values of deformation corresponding to $1/3$ and $2/3$ of the maximal load. Therefore, the second ANN puts more impact on the shape of loading part of the diagram than the first one.

The errors in predictions for training and testing data are noted in Table 8.8. The Table contains the errors corresponding to first ANN trained for k_3 parameters. The errors of the second network were a negligibly higher, hence it is not necessary to present them.

Two neural networks obtained to predict k_3 and k_4 parameters with inputs taken from measured data represent system of two non-linear equations, which could be solved graphically. Relation given by neural network for k_4 prediction with all inputs fixed to values obtained from measured diagram except the input value of k_3 parameter, which states as variable, is shown in Figure 8.30. The relation depicted in this Figure is defined in similar manner by first neural network trained to predict k_3 parameter. The intersection of presented curves defines the predicted values of k_3 and k_4 parameters corresponding to measured data, in this case it is $k_3 = 12.34$ and $k_4 = 98.19$.[3]

Figure 8.31 then shows equivalent relations for k_3 and k_4 parameters, but here obtained by second neural network trained to predict k_3 parameter. Values of predicted parameters are in this case: $k_3 = 11.20$ and $k_4 = 91.85$.

In both cases, the predicted values differs from values stated by authors in [Bažant et al., 2000], where $k_3 = 9$ and $k_4 = 82$. The comparison of measured data with simulated diagrams obtained for parameters given in [Bažant et al., 2000] and for two couples of parameters predicted by neural networks is depicted in Figure 8.32.

It is not so easy to judge which predicted simulation really better correspond to measured data. Simulation proposed in [Bažant et al., 2000] relatively well fit the measured data at the end of elasticity and at the end of loading. Nevertheless, there is a significant error in the middle of the loading part of the diagram. Simulation corresponding to first the prediction of k_3

[3] Relations between k_3 and k_4 parameters is in Figure 8.30 shown in normalized intervals (0.15; 0.85).

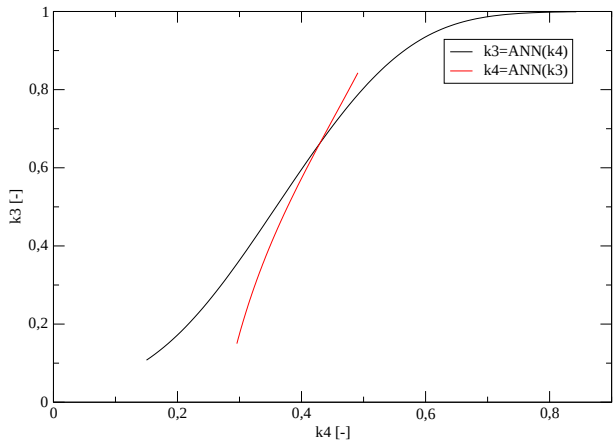


Figure 8.30: Relations of k_3 and k_4 parameters for measured data, black curve correspond to first ANN trained to predict k_3 parameter with four inputs.

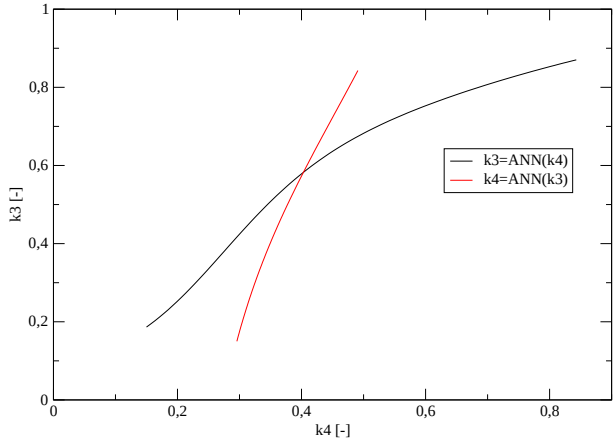


Figure 8.31: Relations of k_3 and k_4 parameters for measured data, black curve correspond to second ANN trained to predict k_3 parameter with five inputs.

parameter relatively well fit the measured data in the middle of the loading diagram and is worse near the end of elasticity and the end of the loading. Simulation corresponding to the second prediction of k_3 parameter could represent a compromise between the previous simulations. To show some more objective comparison of presented simulations, the error between measured data and simulated curves could be evaluated as a sum of differences between deformation values in discrete points corresponding to measured data, i.e.

$$E = \sum_{i}^{N} |\epsilon_i - \tilde{\epsilon}_i|, \tag{8.2}$$

where N is a number of measured points, ϵ_i corresponds to measured deformation corresponding to the i-th measured point and $\tilde{\epsilon}_i$ corresponds to simulated deformation corresponding to the i-th measured point. Values of the error defined by Equation 8.2 are written in Table 8.9.

From the comparison presented in Table 8.9 it is clearly visible, that simulations performed

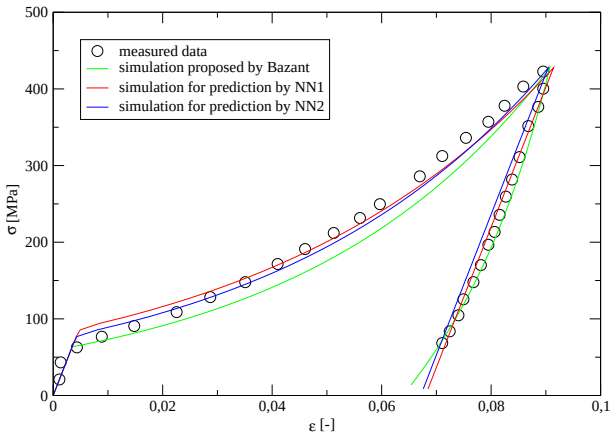


Figure 8.32: Comparison of measured data and simulated diagrams of hydrostatic compression test for predicted parameters.

Prediction	Error E
trial-and-error [Bažant et al., 2000]	0.0975
ANN1	0.0556
ANN2	0.0602

Table 8.9: Comparison of errors of predicted simulations.

for parameters predicted by both neural networks fit the measured data better than the simulation done for parameters obtained by trial-and-error method presented in [Bažant et al., 2000]. Moreover, in Figure 8.32 it is also visible, that simulations predicted by neural networks are handicapped because of simplification of microplane model M4 implementation to OOFEM software. This simplification disables the simulated curves to be non-linear in unloading part of diagram and that leads to higher error of these simulations in this part of diagram.

8.3.3 Triaxial compression test

Similarly to the hydrostatic compression test, also for triaxial compression test we can use measured data from literature, e.g. in [Bažant et al., 2000, Part II]. These data are obtained by Balmer (1949). Again, the presented measured data are accompanied by the simulation performed by the authors of [Bažant et al., 2000] for parameters values established by trial-and-error method. The triaxial compression test is supposed to be the last experiment needed to identify parameters, which could be identified by uniaxial or hydrostatic compression test, i.e. k_2 parameter. Therefore, Young's modulus, Poisson's ratio, k_1, k_3 and k_4 parameters are supposed to be known and their values are taken form [Bažant et al., 2000].

Five different measurements for triaxial compression test are available in [Bažant et al., 2000, Part II] corresponding to five different levels of hydrostatic pressure σ_H applied to specimens ($\sigma_H \in \{34.5, 68.9, 103.4, 137.9, 172.4\}$ MPa).

For neural network training, 60 simulations were performed using the data generated by FREET software [Novák et al., 2003] and 10 independent simulations for randomly chosen parameters were prepared for neural network testing. The bundle of resulting stress-strain diagrams could be compared with measured data in Figure 8.33.

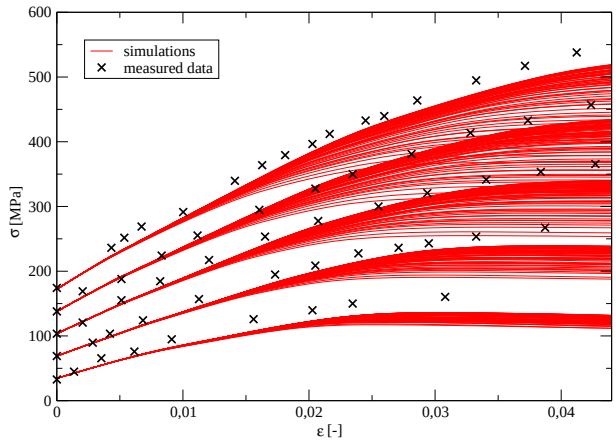


Figure 8.33: Comparison of measured data and results of 70 simulation of triaxial compression test.

In Figure 8.33 it is clearly visible, that measured data are remote from all simulated curves. It could be caused by wrong limits for k_2 parameter, i.e. $k_2 \in (100; 1000)$. New simulations were performed for $k_2 \in (100; 2000)$. From Figure 8.34, it is nevertheless visible, that the higher values of k_2 parameter have not a significant impact to the shape of stress-strain diagram.

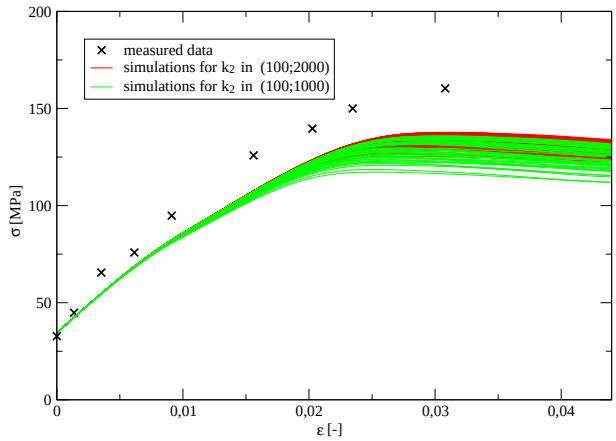


Figure 8.34: Comparison of measured data and results of 70 simulation of triaxial compression test for new interval given for k_2 parameter.

Another reason for the difference between measured data and the bundle of simulated curves could probably be attributed to incorrect values of fixed parameters, i.e. Young's modulus,

parameter	Training data		Testing data	
	Average error	Maximal error	Average error	Maximal error
k_2	2.63	5.82	2.37	6.23

Table 8.10: Error in ANN's predictions relative to the definition interval of the k_2 parameter in [%].

Poison's ratio, k_1, k_3 or k_4 parameter. Nevertheless, the goal of this Section is not to identify these parameters from triaxial compression test.

A neural network with the same architecture as proposed in Section 8.2.3 was trained on simulated data and then applied to predict value of k_2 parameter for measured data. Errors of ANN's predictions on training and testing samples are written in Table 8.10.

The prediction of the neural network for measured data is $k_2 = 1193$. It is not surprising, that the neural network needs to extrapolate for measured data and its prediction exceed the limit given for k_2 parameter. Nevertheless, layer neural networks are in general states as a good approximators, but they are week in extrapolation. Moreover, in this case it is a question, whether it is possible to find appropriate value k_2 parameter to fit measured data, since it is probable that more important error is hidden in values of other parameters.

Figure 8.35 shows the comparison of measured data, simulation given in [Bažant et al., 2000, Part II] and simulation for parameter value predicted by neural network.

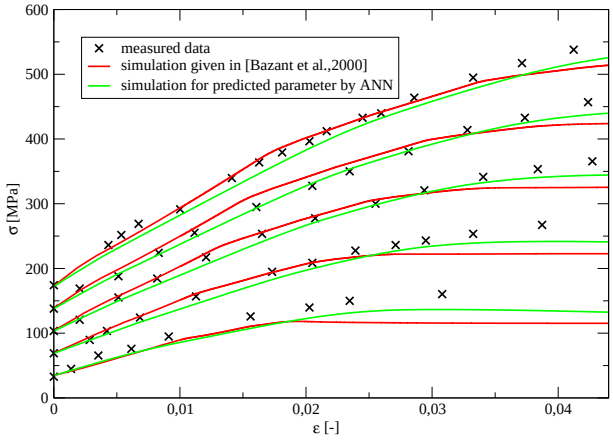


Figure 8.35: Comparison of measured data and simulated diagrams of hydrostatic compression test for predicted parameters.

Similarly to previous Section, the errors between measured data and simulated curves could be calculated. Resulting values for all five experiments corresponding to different levels of hydrostatic pressure σ_H are written in Table 8.11.

Errors in Table 8.11 are very similar for both simulations. It is possible to conclude, that

σ_H [MPa]	[Bažant et al., 2000]	ANN's predictions
34.5	742	780
68.9	2288	2298
103.4	3255	3245
137.9	4080	4030
172.4	5930	5775

Table 8.11: Comparison of errors of predicted simulations.

even for input data not captured by the training set, the neural network was able to predict reasonable value for k_2 parameter and resulting simulation is comparable with that one obtained by trial-and-error method.

8.4 Summary

In this Chapter, an example of the engineering problem, which is difficult to be solved by traditional procedures, was solved using soft computing methods. Particularly, cascade neural networks were used to estimate required microplane material model parameters in a sequential way. As the training procedure, the genetic algorithm-based method GRADE extended by CERAF strategy was used. A number of needed simulations is reduced by the application of the Latin Hypercube Sampling method accompanied by the optimization by Simulated Annealing. The sensitivity analysis shows not only the influence of individual parameters but also approximately predicts the errors produced by neural networks.

Parameter	**Test**	**ANN's topology**	**ANN's inputs**
E	Uniaxial compression	$3+2+1$	$\sigma_{z,1}$, $\sigma_{z,2}$, $\sigma_{z,3}$
ν	Uniaxial compression	$4+3+1$	σ_{x_1}, σ_{x_2}, E, k_1
k_1	Uniaxial compression	$4+2+1$	$\sigma_{x,2}$, $\sigma_{z,peak}$, $\epsilon_{z,peak}$, E
k_2	Triaxial loading	$3+2+1$	σ_{peak}, σ_{29}, σ_{100}
k_3	Hydrostatic loading	$5+2+1$	k_4, ϵ_{yield}, $\epsilon_{load,2}$, $\epsilon_{load,5}$, ϵ_{peak}
k_4	Hydrostatic loading	$3+2+1$	k_3, ϵ_{peak}, $\epsilon_{unload,4}$
c_3	$\times$	$\times$	
c_{20}	Uniaxial compression	$3+2+1$	σ_{peak}, σ_{61}, σ_{81}

Table 8.12: Final status of M4 identification project

Results, see Table 8.12, confirm the claims made by authors [Bažant et al., 2000] of the microplane M4 model on individual parameters fitting. Only the parameter c_3 remains undetermined but the parameter c_3 should be almost constant for the wide range of concretes and our computations confirm almost zero impact of this parameter on stress-strain curves. The Section 8.2 contains the verification of entire identification procedure of all microplane model parameters. Nevertheless the validation of the complete process could not done, since the experimental data for all three loading test performed on one concrete are not available to the author.

Therefore, the validation demonstrated in Section 8.3 was done only for particular identification steps.

The rather severe disadvantage of the microplane model, and also of the proposed methodology, is an extreme demand of computational time. A suite of 30 uniaxial tests consumes approximately 25 days on a single processor PC with the Pentium IV 3400 MHz processor and 3 GB RAM. If we run tests in parallel on 7 computers, the needed time is less than 4 days. The hydrostatic and triaxial tests are less demanding, by running in parallel on 7 computers the required time is less than one day for each test.

Because the identification procedure consists of developing cascade neural networks, the most of created inverse model should be recalculated for any new measured data. Fortunately, the most time consuming simulations of uniaxial compression test necessary for training of first three neural networks predicting Young's modulus, Poisson's ratio and k_1 could be used repeatedly for any new measurement. Then the proposed methodology still needs to compute 30 uniaxial tests to properly identify c_{20} parameter and a set of 30 hydrostatic and triaxial tests to fit k_3, k_4 and k_2. This drawback will be the subject of future work.

In Section 8.3.2 was presented the comparison of measured data with simulation obtained by trial-and-error method given in [Bažant et al., 2000] and with two simulations obtained for predicted parameters by neural networks. The errors evaluated as a difference between measured data and simulated curves are written in Table 8.9 and show that both simulations performed for predicted parameters fit the measured data better than the simulation obtained by trial-and-error method.

Section 8.3.3 shows the case, were the neural network need to extrapolate and is able to predict reasonable values even for remote input data which are comparable with that one obtained by trial-and-error method.

Chapter 9

CONCLUSIONS

> Success is the ability to go from one failure to another with no loss of enthusiasm.
>
> Sir Winston Churchill

The proposed thesis 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 optimization methods suitable for forward mode of identification is presented in Chapter 2. As a most robust and reliable methods could be considered evolutionary algorithms. Results presented in Chapter 4 has shown that genetic algorithms are very robust and very reliable methods especially in combination with some niching strategy. Also from the point of view of accuracy, the genetic algorithms will overcome other optimization methods, since they are able to find the optimum with any given precision. The principal disadvantage of these methods is, nevertheless, a huge number of objective function evaluations.

Chapter 4 contains the detailed description of two proposed genetic algorithms: the SADE algorithm and the GRADE algorithm and their comparison in optimization of a set of twenty mathematical objective functions. A niching strategy CERAF was also introduced to enhance mentioned genetic algorithms in order to solve multi-modal objective functions. The combination of GRADE algorithm with the CERAF strategy was found very robust, reliable and efficient in comparison with the SADE algorithm. Only SADE algorithm was chosen to be compared with GRADE algorithm, since other comparisons of SADE algorithm were published elsewhere, see [Hrstka and Kučerová, 2004] for comparison of SADE algorithm with the differential evolution, a standard binary genetic algorithm and an extended binary genetic algorithm or see [Hrstka et al., 2003] for comparison of SADE algorithm with a real-valued augmented simulated annealing (RASA), an integer augmented simulated annealing (IASA) and also differential evolution on two mathematical and two engineering tasks. From these two previous comparisons, the SADE algorithm was chosen as a most suitable algorithm for engineering tasks. One important aspect was also the small number of extern parameters of SADE

algorithm in comparison with all the other algorithms.

An example of genetic algorithms application is presented in Section 5.2. The SADE algorithm is applied to artificial neural network training and is compared with more traditional method of backpropagation. The results have shown that a genetic algorithm clearly outperforms the backpropagation training, mainly because of genetic optimization's higher resistance to fall into local extremes. Two engineering applications of SADE algorithm to optimization of periodic unit cell for unidirectional fiber composite and to optimal design of reinforced concrete beam are published in [Hrstka et al., 2003]. An application of GRADE algorithm to optimal design and optimal control of structure undergoing large displacements and rotations is presented in Chapter 6. As a most recent application of GRADE algorithm could be mentioned the optimization of statistically equivalent periodic unit cell for an asphalt mixture presented in [Valenta et al., 2007].

If the objective function has only a limited number of local extremes, some meta-model of the function could be constructed and optimized instead of the original objective function in order to reduce the number of time-consuming objective function evaluations. One particular methodology based on interpolation of the error function by radial basis function network with adaptive refining is proposed in Section 4.2 and compared with the GRADE algorithm extended by CERAF strategy in optimization of a set of twenty mathematical functions. Proposed methodology could be considered as a very efficient optimization method for objective function with a limited number of local extremes and with small number of variables (< 10). Also higher requirements on the optimum accuracy could significantly increase the number of objective function evaluations as documented in Section 7.4, but the possibility to predefine the desired precision is still in a particular way maintained.

One application of the proposed methodology to the problems of optimal design and optimal control is presented in Chapter 6 together with the applications of GRADE algorithm and gradient based diffuse approximation. In agreement with the results presented in Chapter 4.2, the RBFN interpolation based methodology clearly outperformed the GRADE algorithm in case of traditional formulation of optimal control, where only two variables were optimized. In was also shown, that gradient based diffuse approximation is very efficient method in giving the preliminary guess of the optimum. Nevertheless, the retrieval of the optimum with higher accuracy extremely increase the number of function calls and even GRADE algorithm become more efficient. In the case of simultaneous formulation of optimal design leading to optimization of design variables together with state variables, the number of optimized variables was too high and only GRADE algorithm was successfully applied.

The second application of the RBFN interpolation based methodology to parameters identification of continuum-discrete damage model capable of representing localized failure is presented in Chapter 7. Due to the physical insight into the model, it was possible to construct simple objective functions F_1, F_2 and F_3 with a high sensitivity to the relevant parameters. This led to non-smooth and non-convex objective functions. Moreover, the optima need to be determined with higher accuracy then usually required in applications (i.e. 5%) in order to enable the sequential character of the identification procedure. With all these requirements, the optimization became quite complex task, nevertheless, it was successfully solved by proposed RBFN

interpolation based method.

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] or [Ibrahimbegovic et al., 2005].

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 Chapter 3. 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.

Last proposed methodology represents an example of such approach and is described in Chapter 5 in very detail. Particularly, a multi-layer perceptron is applied for inverse model development, Latin Hypercube sampling extended by simulated annealing is used for training data preparation and stochastic sensitivity based on Pearson product moment correlation coefficient taken into account for the choice of appropriate MLP's inputs. The application of the proposed methodology on parameters identification of microplane model is presented in Chapter 8.

BIBLIOGRAPHY

[Andre et al., 2000] Andre, J., Siarry, P., and Dognon, T. (2000). An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization. *Advances in Engineering Software*, 32(1):49–60.

[Audze and Eglais, 1977] Audze, P. and Eglais, V. (1977). New approach for planning out of experiments. *Problems of Dynamics and Strengths*, 35:104–107.

[Auer et al., 2005] Auer, P., Burgsteiner, H., and Maass, W. (2005). A learning rule for very simple universal approximators consisting of a single layer of perceptrons. submitted for publication.

[Babuska and Oden, 2004] 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.

[Bažant and Caner, 2005] Bažant, Z. and Caner, F. (2005). Microplane model M5 with kinematic and static constraints for concrete fracture and anelasticity. Part I: Theory, Part II: Computation. *Journal of Engineering Mechanics-ASCE*, 131(1):31–40, 41–47.

[Bažant et al., 2000] Bažant, Z. P., Caner, F. C., Carol, I., Adley, M. D., and Akers, S. A. (2000). Microplane model M4 for concrete. Part I: Formulation with work-conjugate deviatoric stress, Part II: Algorithm and calibration. *Journal of Engineering Mechanics*, 126(9):944–953,954–961.

[Brancherie and Ibrahimbegovic, 2007] Brancherie, D. and Ibrahimbegovic, A. (2007). Novel anisotropic continuum-discrete damage model capable of representing localized failure. part i: theoretic formulation and numerical implementation. *Computers & Structures*. Submitted for publication.

[Breitkopf et al., 2002] Breitkopf, P., Knopf-Lenoir, C., Rasineux, A., and Villon, P. (2002). Efficient optimization strategy using hermite diffuse approximation. In Mang, H., editor, *Proceedings of Fifth World Congress on Computational Mechanics*, Vienna, Austria.

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

[Chen and Liu, 2004] Chen, B. and Liu, J. (2004). Experimental study on ae characteristics of three-point-bending concrete beams. *Cement and Concrete Research*, 34:391–397.

[Claire et al., 2004] Claire, D., Hild, F., and Roux, S. (2004). A finite element formulation to identify damage fields: the equilibrium gap method. *Int. J. Numer. Meth. Engng*, 61:189–208.

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

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

[Drchal et al., 2003] Drchal, J., Kučerová, A., and Němeček, J. (2003). Using a genetic algorithm for optimizing synaptic weights of neural networks. *CTU Report*, 7(1):161–172.

[Fairbairn et al., 2000] 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.

[Furukawa and Yagawa, 1997] Furukawa, R. and Yagawa, G. (1997). Inelastic constitutive parameter identification using an evolutionary algorithm with continuous individuals. *International Journal for Numerical Methods in Engineering*, 40:1071–1090.

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

[González et al., 2004] 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].

[Grefenstette, 1987] Grefenstette, J. (1987). Genetic algorithms and their applications. In Grefenstette, J., editor, *Proceedings of the Second International Conference on Genetic Algorithms and their Applications*. Lawrence Erlbaum Associates.

[Halgamuge et al., 1994] Halgamuge, S. K., Mari, A., and Glesner, M. (1994). Fast perceptron learning by fuzzy controlled dynamic adaption of network parameters. In *Fuzzy Systems in Computer Science*, pages 129–139. Vieweg, Braunschweig.

[Hartman et al., 1990] Hartman, E. J., Keeler, J. D., and Kowalski, J. M. (1990). Layered neural networks with Gaussian hidden units as universal approximations. *Neural Computation*, 2(2):210–215.

[Haykin, 1998] Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 2nd edition.

[Hertz et al., 1991] Hertz, J., Krogh, A., and Palmer, R. (1991). *An Introduction to the Theory of Neural Computation*. Addison Wesley.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. MIT Press.

[Hopfield, 1982] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proc. of the National Academy of Science, USA*, 79:2554–2558.

[Hrstka, WWW] Hrstka, O. (WWW). Homepage of SADE. `http://klobouk.fsv.cvut.cz/~ondra/sade/sade.html`.

[Hrstka and Kučerová, 2000] Hrstka, O. and Kučerová, A. (2000). Searching for optimization method on multidimensional real domains. In *Contributions to Mechanics of Materials and Structures*, volume 4 of *CTU Reports*, pages 87–104. Czech Technical University in Prague.

[Hrstka and Kučerová, 2004] 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.

[Hrstka et al., 2003] Hrstka, O., Kučerová, A., Lepš, M., and Zeman, J. (2003). A competitive comparison of different types of evolutionary algorithms. *Computers & Structures*, 81(18–19):1979–1990.

[Iacono et al., 2006] 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.

[Ibrahimbegović, 1994] Ibrahimbegović, A. (1994). Stress resultant geometrically nonlinear shell theory with drilling rotations - part 1: a consistent formulation. *Comput. Methods Appl. Mech.Eng.*, 118:265–284.

[Ibrahimbegović, 1995] Ibrahimbegović, A. (1995). Finite element implementation of reissner's geometrically nonlinear beam theory: three dimensional curved beam finite elements. *Comput. Methods Appl. Eng.*, 122:10–26.

[Ibrahimbegovic, 2006] Ibrahimbegovic, A. (2006). *Mécanique non linéaire des solides déformables : Formulation théorique et résolution numérique par éléments finis*. Lavoisier, Paris, France.

[Ibrahimbegović and Frey, 1993] Ibrahimbegović, A. and Frey, F. (1993). Finite element analysis of linear and non-linear planar deformations of elastic initially curved beams. *International Journal for Numerical Methods in Engineering*, 36:3239–3258.

[Ibrahimbegovic et al., 1991] Ibrahimbegovic, A., Frey, F., Fonder, G., and Massonnet, C. (1991). A variational formulation of shallow shells. In Onate, E. e. a. e., editor, *Finite elements in the 1990's*, pages 68–79. Springer, Berlin. A Book dedicated to O. C. Zienkeiwicz.

[Ibrahimbegovic et al., 2005] Ibrahimbegovic, A., Grešovnik, I., Markovič, D., Melnyk, S., and Rodič, T. (2005). Shape optimization of two-phase inelastic material with microstructure. *Engineering Computations*, 22(5/6):605–645.

[Ibrahimbegović et al., 2004] Ibrahimbegović, A., Knopf-Lenoir, C., Kučerová, A., and Villon, P. (2004). Optimal design and optimal control of structures undergoing finite rotations and elastic deformations. *International Journal for Numerical Methods in Engineering*, 61(14):2428–2460.

[Ibrahimbegović and Mamouri, 2000] Ibrahimbegović, A. and Mamouri, S. (2000). On rigid components and joint constraints in nonlinear dynamics of flexible multibody systems imploying 3D geometrically exact beam model. *Comp. Methods Appl. Mech. Eng.*, 188:805–831.

[Iman and Conover, 1980] Iman, R. L. and Conover, W. (1980). Small sample sensitivity analysis techniques for computer models, with an application to risk assessment. *Communications in Statistics - Theory and Methods*, 9(17):1749–1842.

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

[J. Němeček and Bittnar, 2005] J. Němeček, P. Padevět, B. P. and Bittnar, Z. (2005). Effect of transversal reinforcement in normal and high strength concrete columns. *Materials and Structures*, 38(281):665–671.

[J. Němeček and Bittnar, 2002] J. Němeček, B. Patzák, D. R. and Bittnar, Z. (2002). Microplane models: computational aspects and proposed parallel algorithm. *Computers & Structures*, 80(27–30):2099–2108.

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

[Jirásek and Bažant., 2001] Jirásek, M. E. and Bažant., Z. P. (2001). *Inelastic Analysis of Structures*. John Wiley & Sons.

[Johnson et al., 1990] Johnson, M., Moore, L., and Ylvisaker, D. (1990). Minimax and maximin distance designs. *J. Statist. Plann. Inverence*, 26:131–148.

[Jolliffe, 2002] Jolliffe, I. T. (2002). *Principal Component Analysis*. Springer–Verlag, 2^{nd} edition.

[Karakasis and Giannakoglou, 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].

[Keršner et al., 2007] 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.

[Kohonen, 1982] Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69.

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

[Kucerova et al., 2007] Kucerova, A., Brancherie, D., Ibrahimbegovic, A., Zeman, J., and Bittnar, Z. (2007). Novel anisotropic continuum-discrete damage model capable of representing localized failure of massive structures. part ii: identification from tests under heterogeneous stress field. *Computers & Structures*. submitted for publication.

[Kučerová, 2003] Kučerová, A. (2003). Optimisation de forme et contrle de chargement des structures elastique soumis de rotations finis en utilisant les algorithmes gntiques. Master's thesis, Ecole Normale Supérieure de Cachan, France.

[Kučerová et al., 2006] Kučerová, A., Brancherie, D., and Ibrahimbegović, A. (2006). Material parameter identification for damage models with cracks. In *Proceedings of the Eighth International Conference on Computational Structures Technology*, pages on CD–ROM. Civil-Comp Press Ltd.

[Kučerová et al., 2005] Kučerová, A., Lepš, M., and Skoček, J. (2005). Large black-box functions optimization using radial basis function networks. In Topping, B. H. V., editor, *Proceedings of Eighth International conference on the Application of Artificial Intelligence to Civil, Structural and Environmental Engineering*, pages pages on CD–ROM, Stirling, United Kingdom. Civil-Comp Press.

[Kučerová et al., 2007] 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.

[le Cun et al., 1989] le Cun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.

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

[Lehký and Novák, 2005] 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.

[Lepš, 2005] Lepš, M. (2005). *Evolutionary Algorithms and Intelligent Tools in Engineering Optimization*, chapter Single and Multi-Objective Optimization in Civil Engineering, pages 320–341. Southampton: WIT Press.

[Lepš and Šejnoha, 2003] Lepš, M. and Šejnoha, M. (2003). New approach to optimization of reinforced concrete beams. *Computers & Structures*, 81(18–19):1957–1966.

[Luenberger, 1984] Luenberger, D. (1984). Linear and nonlinear programming. *Addison-Wesley Publ.*

[Mahfoud, 1995a] Mahfoud, S. W. (1995a). A comparison of parallel and sequential niching methods. In Eshelman, L., editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 136–143, San Francisco, CA. Morgan Kaufmann.

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

[Mahnken, 2004] Mahnken, R. (2004). *Encyclopedia of Computational Mechanics Part 2. Solids and Structures*, chapter Identification of Material Parameters for Constitutive Equations. John Wiley & Sons, Ltd.

[Mahnken and Stein, 1996] 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 et al., 2006] Maier, G., Bocciarelli, M., Bolzon, G., and Fedele, R. (2006). Inverse analyses in fracture mechanics. *International Journal of Fracture*, 138:47–73.

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

[McKay et al., 1979] McKay, M. D., J., B. R., and Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21:239–245.

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

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

[Minsky and Papert, 1969] Minsky, M. and Papert, S. (1969). *Perceptrons*. MIT Press, Cambridge, MA.

[Montgomery, 2005] Montgomery, D. C. (2005). *Design and Analysis of Experiments*. John Wiley and Sons, 6^{th} edition.

[Most et al., 2007] 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.

[Myers and Montgomery, 1995] 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.

[Nakayama et al., 2004] 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].

[Narazaki and Ralescu, 1991] Narazaki, H. and Ralescu, A. L. (1991). A synthesis method for multi-layered neural network using fuzzy sets. In *IJCAI-91: Workshop on Fuzzy Logic in Artificial Intelligence*, pages 54–66, Sydney.

[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.

[Novák and Lehký, 2006] 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.

[Novák et al., 2007] 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.

[Novák et al., 2003] Novák, D., Vořechovský, M., and Rusina, R. (2003). Small-sample probabilistic assessment - software FREET. In *Proceedings of 9th International Conference on Applications of Statistic and Prabability in Civil Engineering - ICASP 9*, pages 91–96, San Francisco, USA. Millpress, Rotterdam.

[Němeček and Bittnar, 2004] Němeček, J. and Bittnar, Z. (2004). Experimental investigation and numerical simulation of post-peak behavior and size effect of reinforced concrete columns. *Materials and Structures*, 37(267):161–169.

[Park and Sandberg, 1991] Park, J. and Sandberg, I. W. (1991). Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2):246–257.

[Park and Sandberg, 1993] Park, J. and Sandberg, I. W. (1993). Approximation and radial-basis-function networks. *Neural Computation*, 5(3):305–316.

[Patzák, WWW] Patzák, B. (WWW). Homepage of OOFEM. `http://www.oofem.org`.

[Patzák and Bittnar, 2001] Patzák, B. and Bittnar, Z. (2001). Design of object oriented finite element code. *Advances in Engineering Software*, 32(10–11):759–767.

[Pichler et al., 2003] 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.

[Pyrz and Zairi, 2007] Pyrz, M. and Zairi, F. (2007). Identification of viscoplastic parameters of phenomenological constitutive equations for polymers by deterministic and evolutionary approach. *Modelling Simul. Mater. Sci. Eng.*, 15:85–103.

[Quagliarella, 2003] 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).

[Queipo et al., 2005] 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.

[Rafiq and Southcombe, 1998] Rafiq, M. Y. and Southcombe, C. (1998). Genetic algorithms in optimal design and detailing of reinforced concrete biaxial columns supported by a declarative approach for capacity checking. *Computers & Structures*, 69:443–457.

[Rajasekaran et al., 1996] Rajasekaran, S., Febin, M. F., and Ramasamy, J. V. (1996). Artificial fuzzy neural networks in civil engineering. *Computers & Structures*, 61(2):291–302.

[Saastamoinen et al., 1998] Saastamoinen, A., Pietilä, T., Värri, A., Lehtokangas, M., and Saarinen, J. (1998). Waveform detection with rbf network – application to automated eeg analysis. *Neurocomputing*, 20:1–13.

[Sacks et al., 1989] Sacks, J., Shiller, S. B., and Welch, W. J. (1989). Designs for computer experiments. *Technometrics*, 34:15–25.

[Shewry and Wynn, 1987] Shewry, M. and Wynn, H. (1987). Maximum entropy design. *J. Appl. Statist.*, 14(2):165–170.

[Simpson et al., 2001] 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.

[Storn, 1996] Storn, R. (1996). On the usage of differential evolution for function optimization. In *NAPHIS 1996*, pages 519–523. Berkeley.

[Storn, WWW] Storn, R. (WWW). Homepage of Differential Evolution. `http://www.icsi.berkeley.edu/~storn/code.html`.

[Storn and Price, 1995] Storn, R. and Price, K. (1995). Differential Evolution : A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, University of Berkeley.

[Toropov and Yoshida, 2005] 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.

[Toropov et al., 2007] Toropov, V. V., Bates, S. J., and Querin, O. M. (2007). Generation of extended unifrom latin hypercube design of experiments. In Topping, B., editor, *Proceedings of the Ninth International Conference of the Application of Artificial Intelligence to Civil, Structural and environmental Engineering*. Civil-Comp Press, Stirling, Scotland.

[Tsoukalas and Uhrig, 1997] Tsoukalas, L. H. and Uhrig, R. E. (1997). *Fuzzy and neural aproaches in engineering*. John Wiley, New York.

[Valenta et al., 2007] Valenta, R., Šejnoha, J., and Šejnoha, M. (2007). Construction of a statistically equivalent periodic unit cell for an asphalt mixture. In Topping, B. H. V., editor, *Proceedings of The Eleventh International conference on Civil, Structural and Environmental Engineering Computing*, pages on CD–ROM. Civil-Comp Press.

[Varcol and Emmerich, 2005] 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.

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

[Villon, 1991] Villon, P. (1991). *Contribution à l'optimisation*. PhD thesis, Université de Technologie de Compiègne, Compiègne, France.

[Wang et al., 2002] 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.

[Waszczyszyn and Ziemianski, 2005] 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.

[Waszczyszyn and Ziemiański, 2006] 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.

[Weigend and Gershenfeld, 1994] Weigend, A. S. and Gershenfeld, N. A. (1994). *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison Wesley.

[Wineberg and Christensen, 2007] Wineberg, M. and Christensen, S. (2007). Genetic and evolutionary computation conference. In *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*, pages 3765–3791. ACM Press New York, NY, USA.

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

Appendix A

LIST OF FUNCTIONS APPLIED FOR GENETIC ALGORITHMS TESTING

The following set of mathematical test functions was taken from the article [Andre et al., 2000] and the same set of functions was also applied for the comparison of genetic algorithms published in the article [Hrstka and Kučerová, 2004].

A.1 Mathematical formulation of test functions

- F1:

$$f(x) = 2(x - 0.75)^2 + \sin(5\pi x - 0.4\pi) - 0.125 \tag{A.1}$$

where

$$0 \leqslant x \leqslant 1$$

- F3:

$$f(x) = -\sum_{j=1}^{5} [j \sin[(j+1)x + j]] \tag{A.2}$$

where

$$-10 \leqslant x \leqslant 10$$

- Branin:

$$f(x, y) = a(y - bx^2 + cx - d)^2 + h(1 - f)\cos x + h \tag{A.3}$$

where

$$a = 1, b = 5.1/4\pi^2, c = 5/\pi, d = 6,$$
$$h = 10, f = 1/8\pi, -5 \leqslant x \leqslant 10, 0 \leqslant y \leqslant 15$$

- Camelback:

$$f(x, y) = \left(4 - 2.1x^2 + \frac{x^4}{3}\right) x^2 + xy + (-4 + 4y^2)y^2 \tag{A.4}$$

where

$$-3 \leqslant x \leqslant 3, -2 \leqslant y \leqslant 2$$

- Goldprice:

$$f(x,y) = \begin{array}{l}[\ \ 1+(x+y+1)^2(19-14x+3x^2-14y+6xy+3y^2)]\cdot \\ [\ \ 30+(2x-3y)^2(18-32x+12x^2+48y-36xy+27y^2)]\end{array} \tag{A.5}$$

where

$$-2 \leqslant x \leqslant 2, -2 \leqslant y \leqslant 2$$

- PShubert 1 and 2:

$$f(x,y) = \left\{\sum_{i=1}^{5} i\cos[(i+1)x+i]\right\}\cdot \left\{\sum_{i=1}^{5} i\cos[(i+1)y+i]\right\} + \beta[(x-1.42513)^2+(y+0.80032)^2] \tag{A.6}$$

where

$$-10 \leqslant x \leqslant 10, -10 \leqslant y \leqslant 10,$$

for PShubert1: $\beta = 0.5$ for PShubert2: $\beta = 1.0$

- Quartic:

$$f(x,y) = \frac{x^4}{4} - \frac{x^2}{2} + \frac{x}{10} + \frac{y^2}{2} \tag{A.7}$$

where

$$-10 \leqslant x \leqslant 10, -10 \leqslant y \leqslant 10$$

- Shubert:

$$f(x,y) = \left\{\sum_{i=1}^{5} i\cos[(i+1)x+i]\right\}\cdot \left\{\sum_{i=1}^{5} i\cos[(i+1)y+i]\right\} \tag{A.8}$$

where

$$-10 \leqslant x \leqslant 10, -10 \leqslant y \leqslant 10$$

- Hartman 1:

$$f(x_1,x_2,x_3) = -\sum_{i=1}^{4} c_i e^{-\sum_{j=1}^{3} a_{ij}(x_i-p_{ij})^2} \tag{A.9}$$

where

$$0 \leqslant x_i \leqslant 1, i = 1,\ldots,3$$

$$x = (x_1,\ldots,x_3), p_i = (p_{i1},\ldots,p_{i3}), a_i = (a_{i1},\ldots,a_{i3})$$

i	a_{ij}			c_i	p_{ij}		
1	3.0	10.0	30.0	1.0	0.36890	0.1170	0.2673
2	0.1	10.0	35.0	1.2	0.46990	0.4387	0.7470
3	3.0	10.0	30.0	3.0	0.10910	0.8732	0.5547
4	0.1	10.0	35.0	3.2	0.03815	0.5743	0.8828

- Shekel 1,2 and 3:

$$f(x) = -\sum_{i=1}^{m} \frac{1}{(x-a_i)^T(x-a_i)+c_i} \tag{A.10}$$

where

$$0 \leqslant x_j \leqslant 10,$$

for Shekel1: $m = 5$, for Shekel2: $m = 7$, for Shekel3: $m = 10$

$$x = (x_1, x_2, x_3, x_4)^T, a_i = (a_{i1}, a_{i2}, a_{i3}, a_{i4})^T$$

i	a_{ij}				c_i
1	4.0	4.0	4.0	4.0	0.1
2	1.0	1.0	1.0	1.0	0.2
3	8.0	8.0	8.0	8.0	0.2
4	6.0	6.0	6.0	6.0	0.4
5	3.0	7.0	3.0	7.0	0.4
6	2.0	9.0	2.0	9.0	0.6
7	5.0	5.0	3.0	3.0	0.6
8	8.0	1.0	8.0	1.0	0.7
9	6.0	2.0	6.0	2.0	0.5
10	7.0	3.6	7.0	3.6	0.5

- Hartman 2:

$$f(x_1, \ldots, x_6) = -\sum_{i=1}^{4} c_i e^{-\sum_{j=1}^{6} a_{ij}(x_i - p_{ij})^2} \tag{A.11}$$

where

$$0 \leqslant x_j \leqslant 1, j = 1, \ldots, 6$$

$$x = (x_1, \ldots, x_6), p_i = (p_{i1}, \ldots, p_{i6}), a_i = (a_{i1}, \ldots, a_{i6})$$

i	a_{ij}						c_i
1	10.00	3.00	17.00	3.50	1.70	8.00	1.0
2	0.05	10.00	17.00	0.10	8.00	14.00	1.2
3	3.00	3.50	1.70	10.00	17.00	8.00	3.0
4	17.00	8.00	0.05	10.00	0.01	14.00	3.2

i	p_{ij}					
1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	0.2348	0.1451	0.3522	0.2883	0.3047	0.6650
4	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

- Hosc 45:

$$f(x) = 2 - \frac{1}{n!}\prod_{i=1}^{n} x_i \tag{A.12}$$

where

$$x = (x_1, \ldots, x_n), 0 \leqslant x_i \leqslant i, n = 10$$

- Brown 1:

$$f(x) = \left[\sum_{i \in J}(x_i - 3)\right]^2 + \sum_{i \in J}[10^{-3}(x_i - 3)^2 - (x_i - x_{i+1}) + e^{20(x_i - x_{i+1})}] \tag{A.13}$$

where

$$J = \{1, 3, \ldots, 19\}, -1 \leqslant x_i \leqslant 4, 1 \leqslant i \leqslant 20, x = (x_1, \ldots, x_{20})^T$$

- Brown 3:

$$f(x) = \sum_{i=1}^{19}[(x_i^2)^{(x_{i+1}^2+1)} + (x_{i+1}^2)^{(x_i^2+1)}] \tag{A.14}$$

$$x = (x_1, \ldots, x_{20})^T, -1 \leqslant x_i \leqslant 4, 1 \leqslant i \leqslant 20$$

- F5n:

$$f(x) = (\pi/20) \cdot \left\{10\sin^2(\pi y_1) + \sum_{i=1}^{19}[(y_i - 1)^2 \cdot (1 + 10\sin^2(\pi y_i + 1))] + (y_{20} - 1)^2\right\} \tag{A.15}$$

where

$$x = (x_1, \ldots, x_{20})^T, -10 \leqslant x_i \leqslant 10, y_i = 1 + 0.25(x_i - 1)$$

- F10n:

$$f(x) = (\pi/20) \cdot \left\{10\sin^2(\pi x_1) + \sum_{i=1}^{19}[(x_i - 1)^2 \cdot (1 + 10\sin^2(\pi x_{i+1}))] + (x_{20} - 1)^2\right\} \tag{A.16}$$

where

$$x = (x_1, \ldots, x_{20})^T, -10 \leqslant x_i \leqslant 10$$

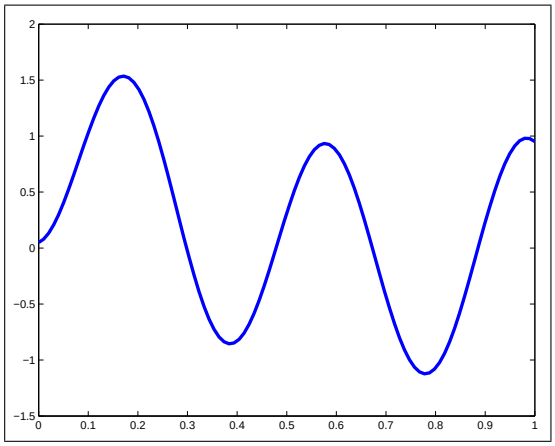
- F15n:

$$f(x) = (1/10) \cdot \left\{\sin^2(3\pi x_1) + \sum_{i=1}^{19}[(x_i - 1)^2(1 + sin^2(3\pi x_{i+1}))] + (1/10)(x_{20} - 1)^2[1 + \sin^2(2\pi x_{20})]\right\} \tag{A.17}$$

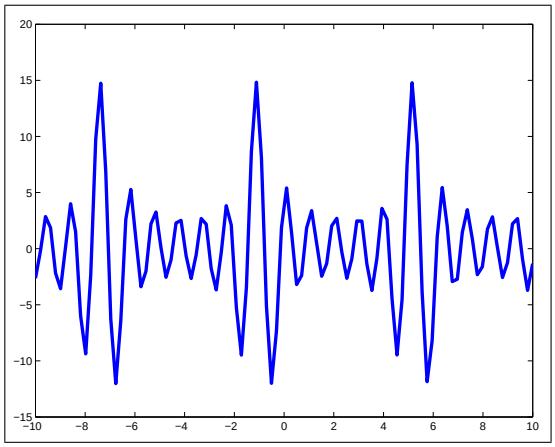
where

$$x = (x_1, \ldots, x_{20})^T, -10 \leqslant x_i \leqslant 10$$

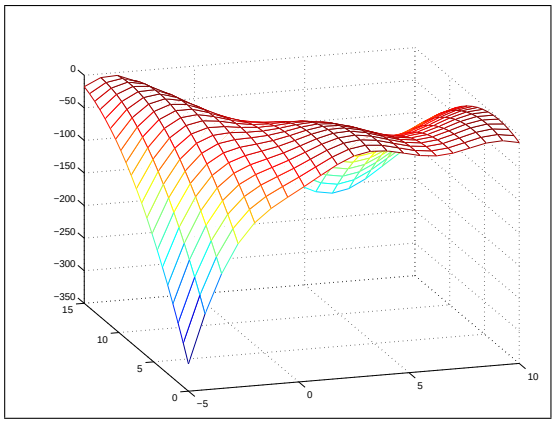
A.2 Graphical ilustration of test function with one or two variables



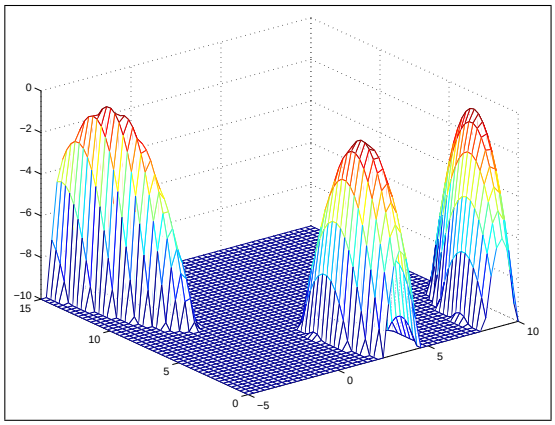
F1



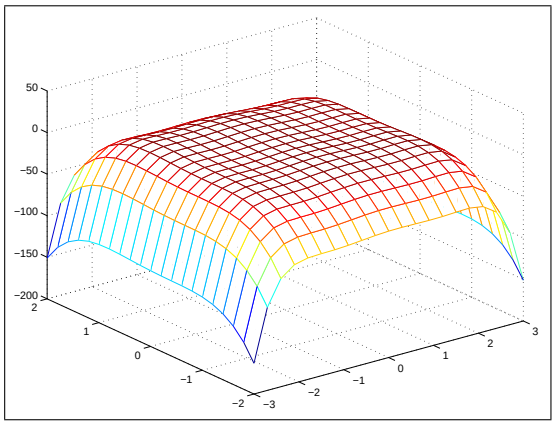
F3



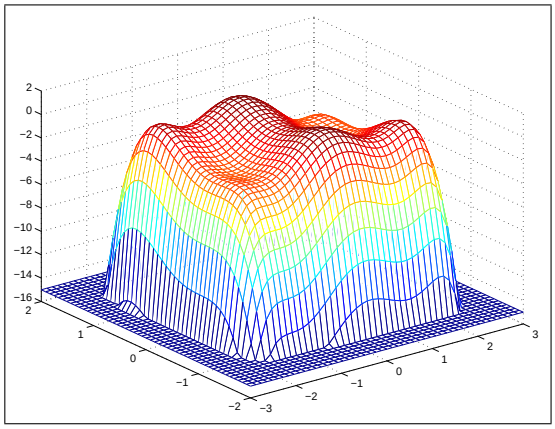
Branin



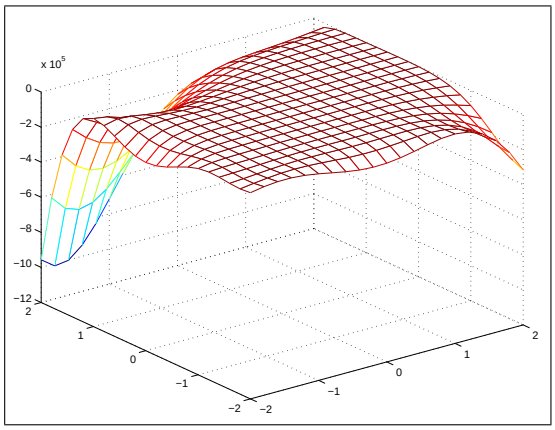
Branin - detailed



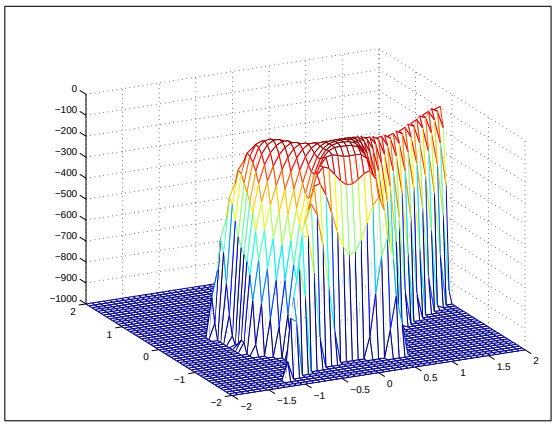
Camelback



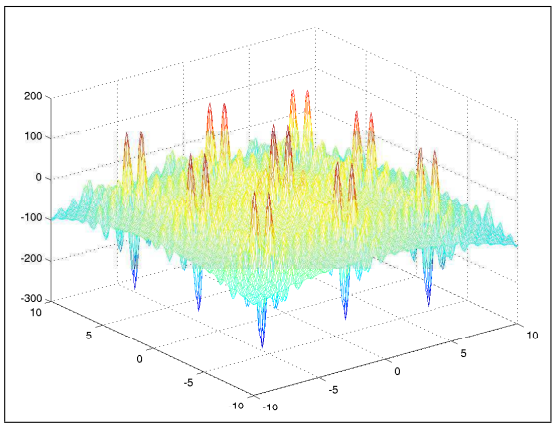
Camelback - detailled



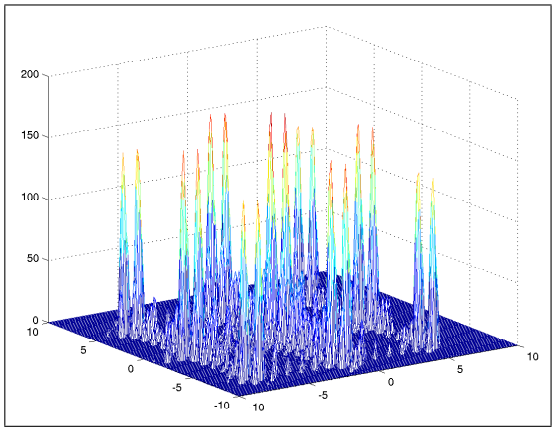
Goldprice



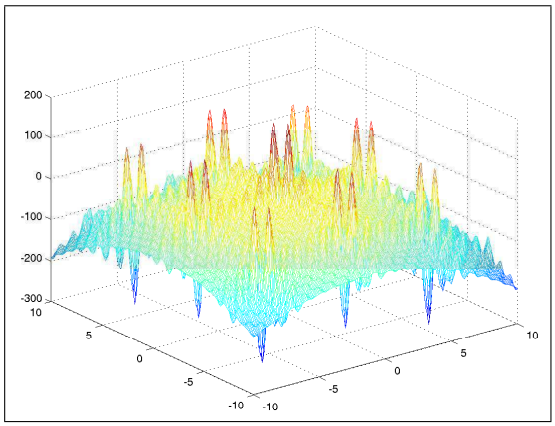
Goldprice - detailled



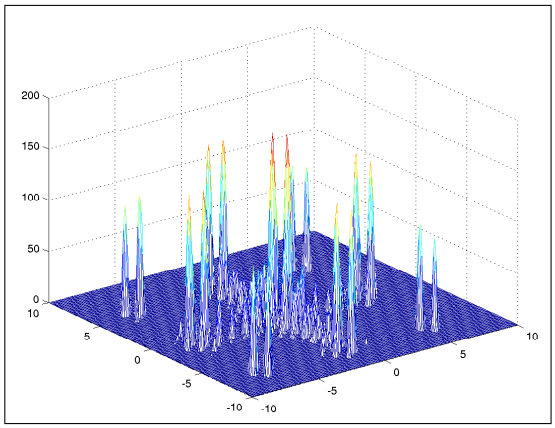
PShubert1



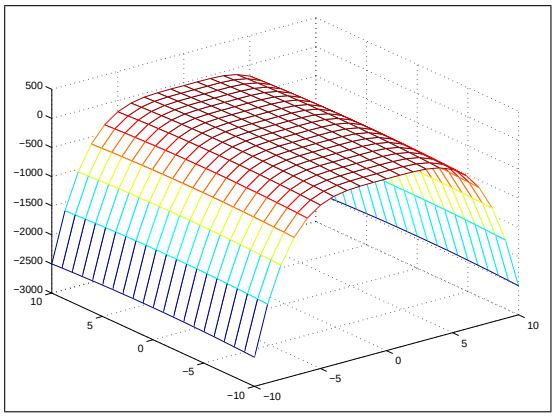
PShubert1 - detailled



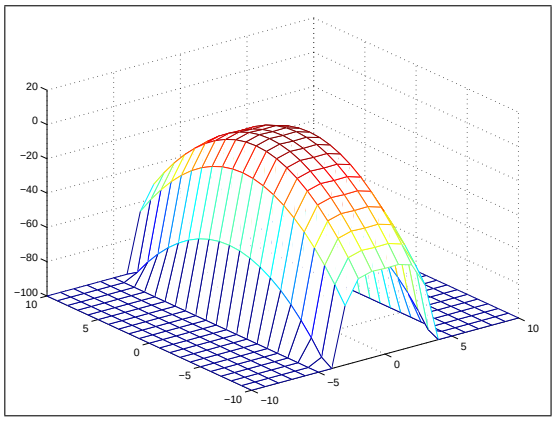
PShubert2



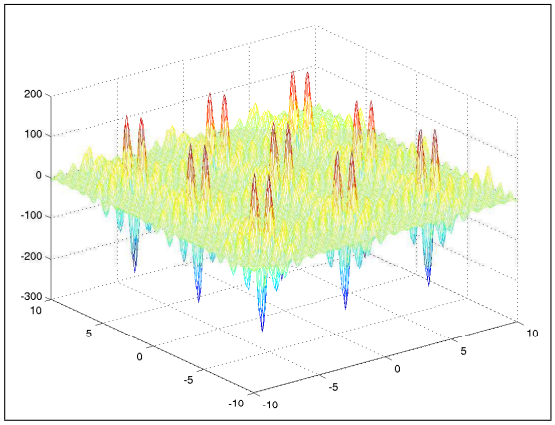
PShubert2 - detailled



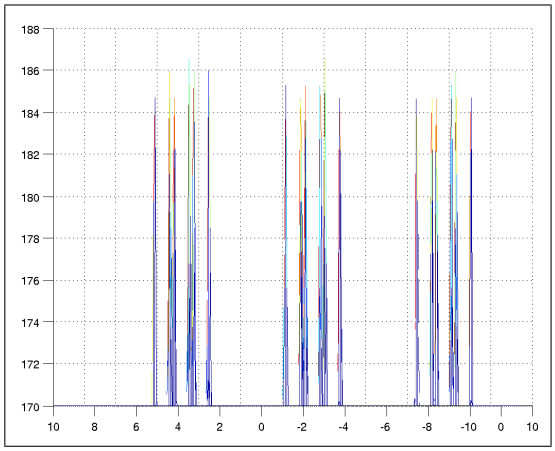
Quartic



Quartic - detailled



Shubert

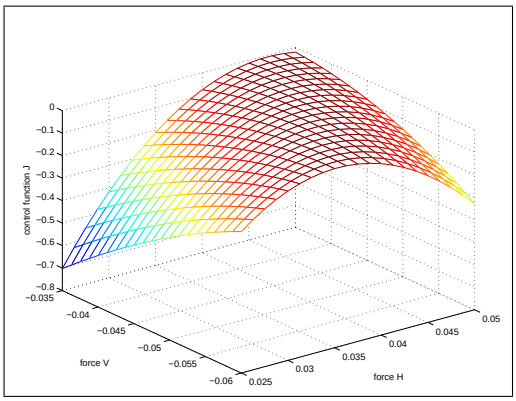


Shubert - detailled

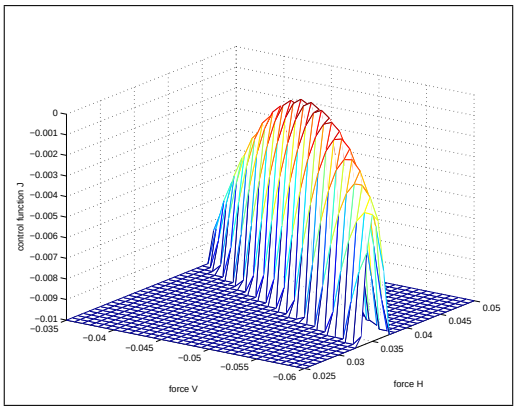
Appendix B

OBJECTIVE FUNCTION CONTOURS CORRESPONDING TO PROBLEM OF OPTIMAL CONTROL OF B LETTER STRUCTURE

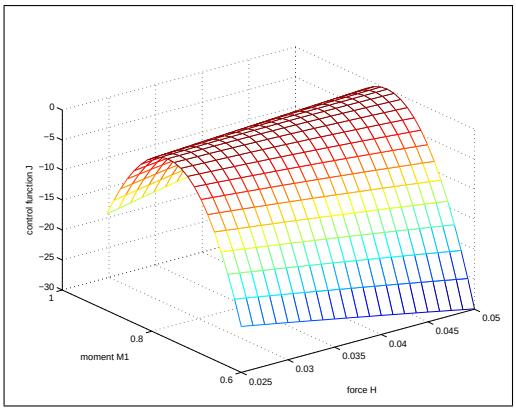
An illustrative representation of the objective function contours in different subspaces of control variables are given in following figures.



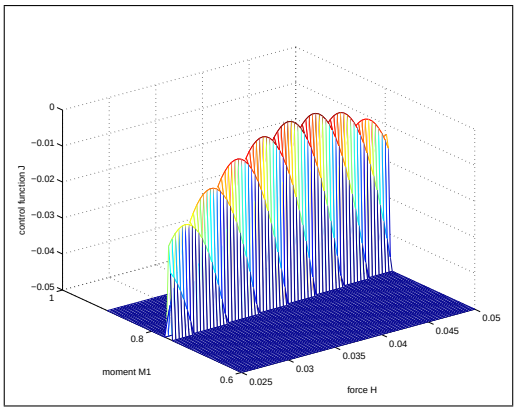
$H - V$ subspace



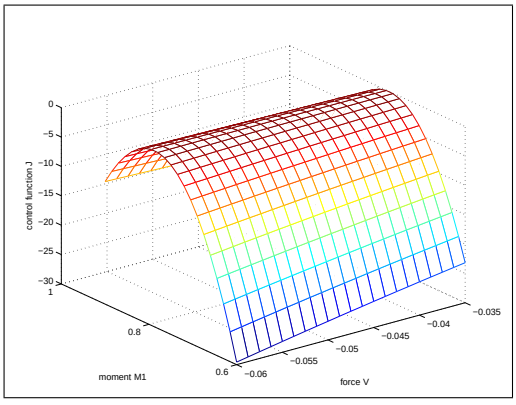
$H - V$ subspace



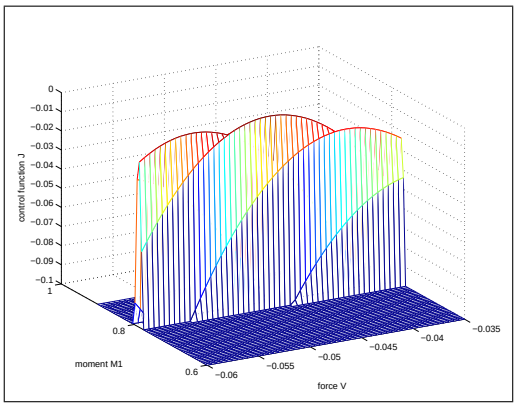
$H - M_1$ subspace



$H - M_1$ subspace

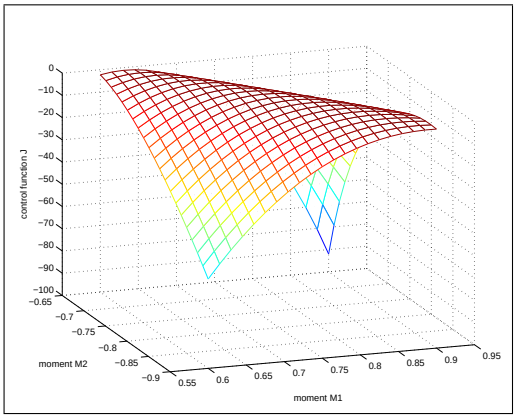


$V - M_1$ subspace

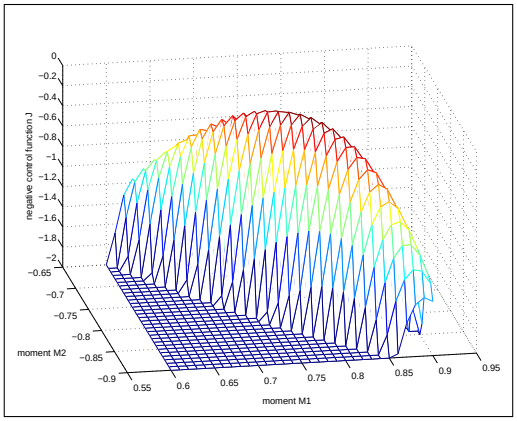


$V - M_1$ subspace

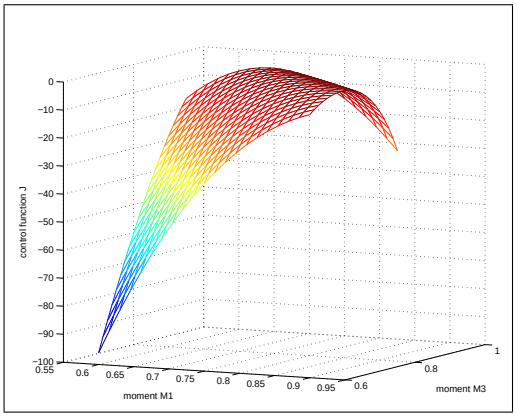
Figure B.1: Multibody system deployment: contours of the cost function in different subspaces.



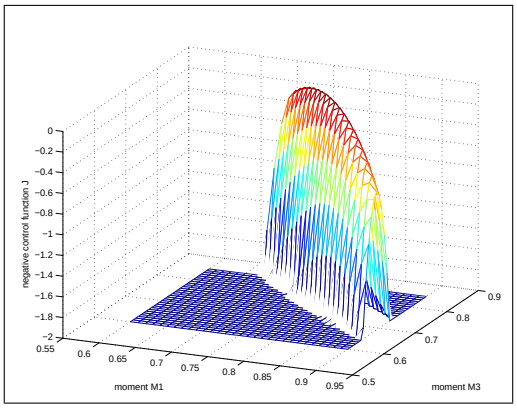
$M_1 - M_2$ subspace



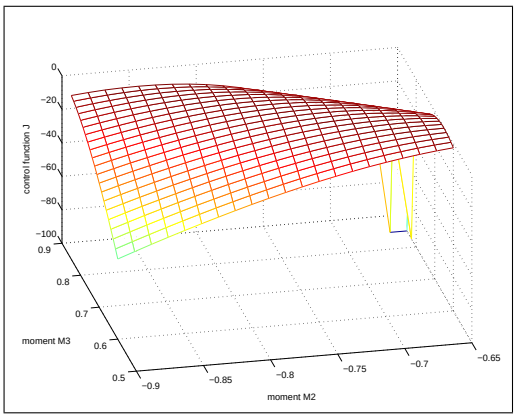
$M_1 - M_2$ subspace



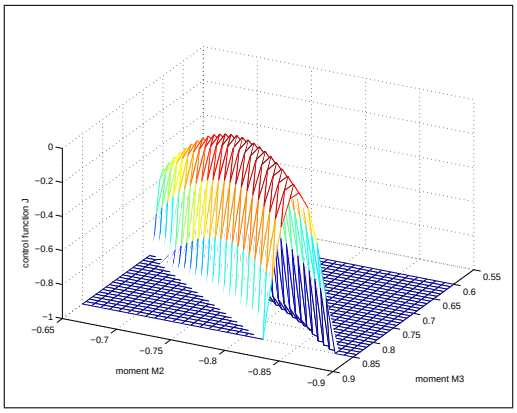
$M_1 - M_3$ subspace



$M_1 - M_3$ subspace



$M_2 - M_3$ subspace



$M_2 - M_3$ subspace

Figure B.2: Multibody system deployment: contours of the cost function in different subspaces.