Optimization Using Space Mapping

Pernille Brock

LYNGBY 2004
EKSAMENSPROJEKT
NR. 54

IMM

Trykt af IMM, DTU

Preface

This Master Thesis is submitted at IMM, DTU, with the supervision of Kaj Madsen, Professor, dr. techn., and Hans Bruun Nielsen, Ass. Professor.
I would like to thank Kaj Madsen and Hans Bruun Nielsen for their enthusiasm, and for always having time for a question or a discussion.
Also I would like to thank Ph. D. Student Frank Pedersen for being involved in the project and supplying many good ideas.

Lyngby, August 2, 2004

Pernille Brock

Abstract

The subject of this master thesis is non-linear optimization using the Space Mapping method with an interpolating surrogate model.
The Space Mapping method is useful in optimization problems, where the fine model we wish to optimize is very computationally expensive. The interpolating surrogate is based on a cheap coarse model and serves as a replacement for the expensive model in order to minimize the number of function evaluations.
An important part of the Space Mapping algorithm is the Parameter Extraction, which involves minimization of the residual between the surrogate and the fine model, which we aim to align. The Parameter Extraction problem does not always have a unique solution, and different formulations are presented in order to ensure this uniqueness.

The thesis provides a presentation of the mathematical theory followed by the Space Mapping algorithm. We then make a number of theoretical and practical investigations concerning different formulations of the residual defining the Parameter Extraction problem.
The step length in forward difference approximations is analyzed, and the optimal step length suited for the considered problems is found to be approximately 10^{-5}. We make an analysis of the solutions to underdetermined and overdetermined problems, hereby an analysis of the Marquardt equations and of least squares problems with and without weighting factors. We look at the effect of adding a regularization term to the residual vector and find, that this residual formulation corresponds to a special case of the Marquardt equations with the damping parameter $1 + \mu$.
The presented Space Mapping algorithm is tested in the various versions on three test problems, and the results are compared. The convergence is faster than with classical optimization algorithms. It is not possible to make general conclusions on the performance of the different algorithm versions based on the included test problems.

Key words: Space Mapping, non-linear optimization, interpolating surrogates, least squares problems, weighting factors, underdetermined and

overdetermined problems.

Resumé

Dette eksamensprojekt omhandler ikke-lineær optimering med brug af Space Mapping-metoden med interpolerende surrogater.
Space Mapping-metoden er anvendelig i optimeringsproblemer ved optimering af en fin model, som er meget dyr beregningsmæssigt. Det interpolerende surrogat er baseret på en billig grov model og erstatter den fine model i optimeringsprocessen, hvorved vi mindsker antallet af tidskrævende funktionsevalueringer.
Et vigtigt delproblem i forbindelse med Space Mapping-algoritmen er Parameter-Ekstraktion, som involverer minimering af residuet mellem surrogatet og den fine model, som vi ønsker at matche. Parameter-Ekstraktions-problemet har ikke altid en entydig løsning, og vi præsenterer forskellige formuleringer med det formål at sikre en entydig løsning.

Projektet præsenterer den matematiske teori efterfulgt af Space Mapping-algoritmen. Herefter laves en række teoretiske og praktiske undersøgelser vedrørende de forskellige formuleringer af residuet, som definerer Parameter-Ekstraktions-problemet.
Skridtlængden i differenstilnærmelser analyseres, og den optimale skridtlængde, som er velegnet til de her betragtede problemer, bestemmes til omkring 10^{-5}. Vi analyserer løsninger til underbestemte og overbestemte problemer, herunder Marquardts ligninger og mindste-kvadraters problemer med og uden vægtfaktorer. Vi betragter effekten af at medtage et regulariseringsled i residuet, og finder at en sådan residue-formulering svarer til et specialtilfælde af Marquardts ligninger med dæmpningsparameteren $1 + \mu$.
De forskellige versioner af den beskrevne Space Mapping-algoritme afprøves på tre testproblemer, og resultaterne sammenlignes. Konvergensen er hurtigere end for klassiske optimeringsmetoder benyttet direkte på den fine model. Det er ikke muligt, at foretage generelle konklusioner om algoritmens præstationer på basis af de her inkluderede testproblemer.

Nøgleord: Space Mapping, ikke-lineær optimering, interpolerende surrogater, mindste-kvadraters problemer, vægtfaktorer, underbestemte og overbestemte problemer.

Contents

Chapter 1

Introduction

1.1 Introduction to the Space Mapping Method

The Space Mapping method is an optimization method used for engineering design problems. The technique is useful, when the model that we wish to optimize is computationally expensive. In this case the use of a classical optimization method directly on the fine model would result in a large number of function evaluations, and is considered impossible in practice. The goal is to lower the number of time-consuming fine model evaluations.

The Space Mapping method relies on the existence of two functions modelling the same system: the fine model, which is very time-consuming to evaluate, and a coarse model, which is cheap to evaluate. We wish to construct a surrogate model based on the coarse model, and let the surrogate serve as a replacement for the fine model in the optimization process. The fine model is successively evaluated in order to construct an interpolating surrogate model, which is then used for optimization. The surrogate model is at least as accurate as the coarse model. By aligning the surrogate model with the fine model in more than one point we seek global as well as local agreement of the two models.

The interpolating surrogate is constructed as a composed mapping consisting of both an input and an output mapping. This mapping is the Space Mapping connecting the coarse model responses with the fine model responses. The design parameters are transformed by the input mapping, and the output mapping corrects the surrogate to ensure exact agreement of the responses. We align both function values and gradients of the surrogate model with the fine model and hereby wish, that the surrogate provides a good approximation in a large region of the design parameter space.

The Space Mapping-based optimization algorithm consist of two sub-problems:

- The optimization of the surrogate model.
- The update of the surrogate, the so-called Parameter Extraction, which determines the mapping parameters in order to ensure the agreement of the surrogate and the fine model.

1.1.1 Different Space Mapping Techniques

The original Space Mapping formulation is described in [4] and [5] and only involves an input mapping $\mathbf{P} : \mathbb{R}^n \rightarrow \mathbb{R}^n$, where:

$$\mathbf{P}(\mathbf{x}) = \arg \min_{\mathbf{z} \in \mathbb{R}^n} \|\mathbf{c}(\mathbf{z}) - \mathbf{f}(\mathbf{x})\|_2^2 \tag{1.1.1}$$

We refer to (1.1.1) as the original Space Mapping definition.
The Space Mapping methods with input mapping can be approached in different ways in order to ensure the uniqueness of the Space Mapping. A full overview and further discussions of the various versions are provided by Jacob Søndergaard in [7]. When only input mappings are used, we cannot be sure that the Space Mapping technique provides the fine model optimizer as a solution, unless certain theoretical conditions are met. These conditions are stated in [7], chapter 4.1. The exact match between the fine model and the coarse model response is therefore not likely in the original Space Mapping, even though the mapped coarse model can provide a good approximation to the fine model over a large region of the parameter space.
When introducing an additional mapping, an output mapping, to define the surrogate model, we can ensure the matching, and hereby overcome the residual misalignment. On that ground the Space Mapping techniques with both input and output mappings are to be preferred. With these techniques the uniqueness of the Parameter Extraction is still not ensured, which is a problem that can be solved in many ways.

This report will only work with the Space Mapping method with both input and output mappings, providing an interpolating surrogate that gives exact alignment with the fine model in the expansion point.

1.2 Problem Formulation

The main subject of this thesis is the Space Mapping method based on an interpolating surrogate. We wish to investigate different theoretical and practical aspects of the method, with the main concern on the solution to the Parameter Extraction problems. On this basis we present a Space Mapping algorithm and test the implementation of the algorithm on different test problems.

In the report we analyze the following subjects:

- The approximation error from using forward difference approximations as estimates for the derivatives, and hereby the optimal step length.
- Least squares problems.
    - The Marquardt equations.
    - The solution to the regularized problem.
    - The solutions to underdetermined and overdetermined least squares problems with and without weights.
- Different formulations of the residual in order to ensure uniqueness of the Parameter Extraction.
    - Reduction of the number of input mapping parameters.
    - The effect of using a regularization term in the Parameter Extraction problem.
    - The effect of using weighting factors in the Parameter Extraction problem.
    - The effect of using normalization factors in the Parameter Extraction problem.

The mathematical theory of the Space Mapping method with interpolating surrogate is introduced, and the Space Mapping algorithm is presented in pseudo-code. We then consider the theoretical and practical investigations of the subjects above. The various versions of the algorithm are tested numerically on three problems. Finally suggestions for future investigations are proposed by listing some unresolved matters.

1.3 Mathematical Introduction

We are aiming at solving an optimization problem of the form:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^n} \{H(\mathbf{f}(\mathbf{x}))\}$$

where $H : \mathbb{R}^m \to \mathbb{R}$ is a suitable objective function, and $\mathbf{x}^* \in \mathbb{R}^n$ is the optimal set of design parameters.
We assume, that two models of the same system are available: A fine but expensive model, given by $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ and a coarse but cheap model given by $\mathbf{c} : \mathbb{R}^n \to \mathbb{R}^m$. The function vectors from a given parameter set are also denoted response vectors.
The surrogate model $\mathbf{s} : \mathbb{R}^n \to \mathbb{R}^m$ is defined by a composite mapping: For

each of the m responses we define the input mapping $\mathbf{P}_i : \mathbb{R}^n \to \mathbb{R}^n$, which performs a linear transformation of the design parameters, and the output mapping $\mathbf{O} : \mathbb{R}^m \to \mathbb{R}^m$, which transforms the coarse model response. The aim is to align the surrogate with the fine model for all m responses.
The input and output mapping parameters for $i = 1, \dots m$ are:

$$\mathbf{A}_i \in \mathbb{R}^{n \times n}, \qquad \mathbf{b}_i \in \mathbb{R}^n, \qquad \alpha_i \in \mathbb{R}, \qquad \beta_i \in \mathbb{R}.$$

The linear transformation $\mathbf{P}_i$ for the ith response function is now defined as:

$$\mathbf{P}_i(\mathbf{x}) = \mathbf{A}_i \mathbf{x} + \mathbf{b}_i \tag{1.3.1}$$

and the output mapping O_i as:

$$O_i(\mathbf{y}) = \alpha_i \left(y_i - \bar{y}_i \right) + \beta_i \tag{1.3.2}$$

where $\bar{\mathbf{y}}$ is a constant vector. Gathering the input and output mappings we have:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{P}_m^{\mathrm{T}} \end{bmatrix}, \qquad \mathbf{O} = \begin{bmatrix} O_1 \\ \vdots \\ O_m \end{bmatrix}$$

The interpolating surrogate model is now defined by the composition:

$$\mathbf{s} = \mathbf{O} \circ \mathbf{c} \circ \mathbf{P} \tag{1.3.3}$$

When inserting the expressions for the input and output mappings we get the surrogate model for the ith response given by:

$$\begin{aligned} s_i(\mathbf{x}) &= O_i \left(c_i \left(\mathbf{P}_i(\mathbf{x}) \right) \right) \\ &= \alpha_i \left(c_i \left(\mathbf{P}_i(\mathbf{x}) \right) - c_i \left(\mathbf{P}_i(\bar{\mathbf{x}}) \right) \right) + \beta_i \\ &= \alpha_i \left(c_i \left(\mathbf{A}_i \mathbf{x} + \mathbf{b}_i \right) - c_i \left(\mathbf{A}_i \bar{\mathbf{x}} + \mathbf{b}_i \right) \right) + \beta_i \end{aligned}$$

We wish to align the responses of the surrogate model with the fine model in all m sampling points. In the kth iteration $\mathbf{x}^{(k)}$ we must thereby have:

$$\mathbf{s}^{(k)}(\mathbf{x}^{(k)}) = \mathbf{f}(\mathbf{x}^{(k)}) \tag{1.3.4}$$

where $\mathbf{s}^{(k)}$ denotes the surrogate used in the kth iteration. We furthermore want the surrogate model to approximate the fine model at previous iteration points. An additional criterion for choosing the mapping parameters is to aim for agreement of the Jacobians of the fine model (denoted $\mathbf{J}_f$) and the surrogate model (denoted $\mathbf{J}_s$) in the current iterate. This leeds to the two equations:

$$\mathbf{s}^{(k)}(\mathbf{x}^{(j)}) = \mathbf{f}(\mathbf{x}^{(j)}) \quad \text{for } j = 1, \ldots, k-1 \tag{1.3.5a}$$

$$\mathbf{J}_s^{(k)}(\mathbf{x}^{(k)}) = \mathbf{J}_f(\mathbf{x}^{(k)}) \tag{1.3.5b}$$

Equations (1.3.4) and (1.3.5) ensure the alignment of the surrogate model and the fine model both both wrt. the function responses and the Jacobians in the current iterate as well as wrt. the function responses in all previous iterates. The goal is to have both local and global agreement of the models. The local agreement is ensured by (1.3.4) and (1.3.5b), and the global agreement by (1.3.5a).

The initial values of the mapping parameters can be chosen as follows: We wish to start the iterations in the coarse model optimizer $\mathbf{z}^*$, so that $\mathbf{x}^{(1)} = \mathbf{z}^*$. In iteration 0 we therefore want the surrogate model to be identical to the coarse model, which is ensured by choosing the input and output mapping parameters as:

$$\left.\begin{aligned} \mathbf{A}_i^{(0)} &= \mathbf{I} \\ \mathbf{b}_i^{(0)} &= \mathbf{0} \\ \alpha_i^{(0)} &= 1 \\ \beta_i^{(0)} &= \alpha_i^{(0)} c_i(\mathbf{P}_i^{(0)}(\mathbf{x}^{(0)})) \end{aligned}\right\} \quad \text{for } i = 1, \ldots, m \tag{1.3.6}$$

In this way the ith response of the 0th surrogate becomes:

$$\begin{aligned} s_i^{(0)}(\mathbf{x}) &= \alpha_i^{(0)} \left(c_i \left(\mathbf{P}_i^{(0)}(\mathbf{x}) \right) - c_i \left(\mathbf{P}_i^{(0)}(\mathbf{x}^{(0)}) \right) \right) + \alpha_i^{(0)} c_i(\mathbf{P}_i^{(0)}(\mathbf{x}^{(0)})) \\ &= \alpha_i^{(0)} \left(c_i \left(\mathbf{P}_i^{(0)}(\mathbf{x}) \right) \right) \\ &= c_i(\mathbf{x}) \end{aligned} \tag{1.3.7}$$

Since the coarse model is assumed to be cheap to evaluate, the optimizer is found by a standard optimization algorithm.

In the following iterations the matching (1.3.4) is ensured by choosing the output mapping parameters α_i and β_i and the constant $\bar{\mathbf{x}}$ in an appropriate way. By putting $\bar{\mathbf{x}}^{(k)} = \mathbf{x}^{(k)}$ we have the ith surrogate in the kth iteration:

$$s_i^{(k)}(\mathbf{x}) = \alpha_i^{(k)} \left(c_i \left(\mathbf{P}_i^{(k)}(\mathbf{x}) \right) - c_i \left(\mathbf{P}_i^{(k)}(\mathbf{x}^{(k)}) \right) \right) + \beta_i^{(k)}$$

By inserting the iterate $\mathbf{x}^{(k)}$ in the surrogate function and then in (1.3.4) we find the value of $\beta_i^{(k)}$ to be:

$$\alpha_i^{(k)} \left(c_i \left(\mathbf{P}_i^{(k)}(\mathbf{x}^{(k)}) \right) - c_i \left(\mathbf{P}_i^{(k)}(\mathbf{x}^{(k)}) \right) \right) + \beta_i^{(k)} = f_i(\mathbf{x}^{(k)})$$
$$\Rightarrow \quad \beta_i^{(k)} = f_i(\mathbf{x}^{(k)})$$

The ith response of the interpolating surrogate is now given by:

$$s_i^{(k)}(\mathbf{x}) = \alpha_i^{(k)} \left(c_i \left(\mathbf{P}_i^{(k)}(\mathbf{x}) \right) - c_i \left(\mathbf{P}_i^{(k)}(\mathbf{x}^{(k)}) \right) \right) + f_i(\mathbf{x}^{(k)}) \tag{1.3.8}$$

which is valid for all $k > 0$.
Because of the choice of $\bar{\mathbf{x}}$ the match (1.3.4) only depends on the output parameter β_i, and the α_i's must be chosen appropriately based on (1.3.5).

In each iteration the next set of design parameters $\mathbf{x}^{(k+1)}$ are found by minimizing the surrogate (1.3.8) defined by the mapping parameters of the previous iteration:

$$\mathbf{x}^{(k+1)} = \arg \min_{\mathbf{x} \in \mathbb{R}^n} \left\{ H \left(\mathbf{s}^{(k)}(\mathbf{x}) \right) \right\} \tag{1.3.9}$$

It must be clarified, that the new iterate $\mathbf{x}^{(k+1)}$ is only accepted, if it produces a decrease in the objective function compared to the previous iterate, ie. if $H(\mathbf{f}(\mathbf{x}^{(k+1)})) < H(\mathbf{f}(\mathbf{x}^{(k)}))$. If this is not the case, the alignments (1.3.4) and (1.3.5b) must be made with respect to the previous (and so far best) iterate. The unaccepted iterate is only used in the global alignment equation (1.3.5a), and must not satisfy the gradient match. To handle such an uphill step regarding the fine model objective we use a trust region method for the surrogate optimization.

When the iterate $\mathbf{x}^{(k+1)}$ is now available, the $(k+1)$th set of mapping parameters must be found. The response alignment (1.3.4) is already ensured. In order to satisfy the additional matching (1.3.5) we define the residual function for the ith response:

$$\mathbf{r}_i^{(k+1)}(\mathbf{A}_i, \mathbf{b}_i, \alpha_i) = \begin{bmatrix} s_i^{(k+1)}(\mathbf{x}^{(1)}, \mathbf{A}_i, \mathbf{b}_i, \alpha_i) - f_i(\mathbf{x}^{(1)}) \\ \vdots \\ s_i^{(k+1)}(\mathbf{x}^{(k)}, \mathbf{A}_i, \mathbf{b}_i, \alpha_i) - f_i(\mathbf{x}^{(k)}) \\ \mathbf{J}_{s,i}^{(k+1)}(\mathbf{x}^{(k+1)}, \mathbf{A}_i, \mathbf{b}_i, \alpha_i) - \mathbf{J}_{f,i}(\mathbf{x}^{(k+1)}) \end{bmatrix} \tag{1.3.10}$$

where $\mathbf{J}_{s,i}$ and $\mathbf{J}_{f,i}$ are the gradients of f_i and s_i wrt. $\mathbf{x}$, ie. the transpose of the ith rows of the Jacobians of the fine resp. the surrogate model wrt. the $\mathbf{x}$ vector. We find the next set of mapping parameters by minimizing the residual:

$$\left\{ \mathbf{A}_i^{(k+1)}, \mathbf{b}_i^{(k+1)}, \alpha_i^{(k+1)} \right\} = \arg \min_{\mathbf{A}_i, \mathbf{b}_i, \alpha_i} \|\mathbf{r}_i^{(k+1)}(\mathbf{A}_i, \mathbf{b}_i, \alpha_i)\| \tag{1.3.11}$$

in some norm for all m responses. This process of updating the parameters is called Parameter Extraction.
The iterations continue in this way with optimization of the current surrogate followed by the Parameter Extraction, until the solution is found within a satisfying accuracy. Appropriate stopping criteria could be based on the relative change in the solution vector or in the objective function.

1.3.1 Overview of The Space Mapping Algorithm

Based on the previous section we can now summarize the Space Mapping method with the interpolating surrogate. The algorithm for solving the optimization problem is outlined as follows:

1. Given coarse model and fine model.
2. Set $k = 0$, choose initial guess for the coarse optimizer $\mathbf{x}^{(0)}$. Initialize input and output mapping parameters $\mathbf{A}_i^{(0)} = \mathbf{I}$, $\mathbf{b}_i^{(0)} = \mathbf{0}$, $\alpha_i^{(0)} = 1$ and $\beta_i^{(0)} = \alpha_i^{(0)} c_i(\mathbf{x}^{(0)})$ for $i = 1, \dots, m$.
3. Optimize the surrogate model (1.3.8) to find the next iterate $\mathbf{x}^{(k+1)}$ by solving (1.3.9).
4. Compute $\mathbf{f}(\mathbf{x}^{(k+1)})$ and $\mathbf{J}_f(\mathbf{x}^{(k+1)})$. Check stopping criteria and terminate if satisfied.
5. Update mapping parameters $\mathbf{A}_i^{(k+1)}$, $\mathbf{b}_i^{(k+1)}$ and $\alpha_i^{(k+1)}$ for $i = 1, \dots m$ by (1.3.11) with the residual given by (1.3.10).
6. Set $k = k + 1$ and go to step 3.

1.3.2 New Formulation of the Residual Vector

In the practical implementation of the Space Mapping algorithm we use different versions of the residual vector for the Parameter Extraction. This is done in order to ensure that the problem has a unique solution, and that this solution should satisfy the local and global agreement between the surrogate and the fine model, that we aim for.
On that ground we hereby define the residual:

$$\mathbf{r}_i^{(k+1)}(\mathbf{A}_i, \mathbf{b}_i, \alpha_i) = \begin{bmatrix} w_1 \cdot a_1 \cdot \left(s_i^{(k+1)}(\mathbf{x}^{(1)}, \mathbf{A}_i, \mathbf{b}_i, \alpha_i) - f_i(\mathbf{x}^{(1)}) \right) \\ \vdots \\ w_k \cdot a_k \cdot \left(s_i^{(k+1)}(\mathbf{x}^{(k)}, \mathbf{A}_i, \mathbf{b}_i, \alpha_i) - f_i(\mathbf{x}^{(k)}) \right) \\ \sigma \cdot d \cdot \left(\mathbf{J}_{s,i}^{(k+1)}(\mathbf{x}^{(k+1)}, \mathbf{A}_i, \mathbf{b}_i, \alpha_i) - \mathbf{J}_{f,i}(\mathbf{x}^{(k+1)}) \right) \end{bmatrix} \tag{1.3.12}$$

It is noted, that the dimension of $\mathbf{r}_i$ is $(k + n)$, when we have found $(k + 1)$ $\mathbf{x}$-iterates. The factors $a_1, \dots, a_k$ and d are normalization factors used for

avoiding scaling problems, in case the responses are not of the same order of magnitude.
The w-factors are weighting factors used in the first k elements of the residual.
The factor σ is a penalty factor only multiplied on the last n elements of the residual vector. The weighting factors and the penalty factor have the same effect, but we distinguish between them, because the factors have different purposes.
The aim of the weighting factors is to give an individual priority to each of the iteration points in the residual. In this way we can distinguish between points far from the current iterate and points closer to the current iterate and make the global agreement more or less accurate in a particular point.
The penalty factor is used to give the alignment of the gradients a certain priority compared to the function value alignments in the previous points. If we increase σ, we can ensure, that the gradients in the current point match.

(1.3.12) is equivalent to (1.3.10), if we put all factors $a_1, \ldots, a_k$, $w_1, \ldots, w_k$, d and σ equal to 1. The theory of section 1.3 and the summarized algorithm in 1.3.1 are hereby still valid, when we use the residual (1.3.12) instead of (1.3.10). The new residual is equivalent to the residual (1.3.10) multiplied by a diagonal matrix.

Throughout the rest of the report the residual we use is given by the definition (1.3.12). If nothing else is mentioned the factors $a_1, \ldots, a_k$, $w_1, \ldots, w_k$, d and σ have the value 1. The first k elements of the residual are referred to as the function value residual, whereas the last n elements are called the gradient residual.

1.4 Assumptions

In optimization problems there can often be several optimizers, both global and local. The Space Mapping method is not a global optimization method, and depending on the problem, we cannot be sure, that the found solution is the global optimizer, or even that this optimizer is unique.

A number of conditions must be satisfied in order to find a minimizer by the Space Mapping method. These conditions are discussed in details in [7], and are not the subject of this report.

We assume the following:

- The sets $\{\mathbf{x}^*\} = \arg\min\{H(\mathbf{f}(\mathbf{x}))\}$ and $\{\mathbf{z}^*\} = \arg\min\{H(\mathbf{c}(\mathbf{x}))\}$ are non-empty, ie. there exists at least one solution to both problems.

- The coarse model optimizer $\mathbf{z}^*$ and the fine model optimizer $\mathbf{x}^*$ are unique.
- All variables are real.
- The coarse model function and the fine model function are both continous and at least one time differentiable.
- The evaluation time for the coarse model is negligible.

1.5 Previous Work and Implementation

The MATLAB programs made in connection with this thesis are working in the existing SMIS (Space Mapping Interpolating Surrogate) framework implemented by Frank Pedersen. The framework is programmed in MATLAB, but also involves some Fortran subroutines collected in the F-package. This included F-package contains different algorithms for the solution of constrained and unconstrained non-linear optimization problems. A detailed description of the Fortran subroutines is found in [13].

The SMIS framework by Frank Pedersen includes a number of algorithms for solving optimization problems with the Space Mapping Method. The different versions of the algorithms are placed in their own directory corresponding to the particular formulation of the algorithm. The problems used to test the algorithms are also placed in each of their own directories. Furthermore the framework contains a number of directories with basic tools, such as forward difference approximations, plot functions etc. A new toolbox has been added, this is the `immoptibox` programmed by Hans Bruun Nielsen [12]. The framework can be augmented by adding a new algorithm or a new test problem placed in the proper new MATLAB directory in the proper MATLAB search path.

All MATLAB code programmed during the working period of this report is available at IMM's homepage, see [15]. Some of the program files are modifactions or augmented versions of existing code, and some files are made from scratch.
Appendix A provides a short user's guide for the SMIS framework, yet only the implementations and test problems used in this report are included.

Chapter 2

Implementation of The Space Mapping Algorithm

In this chapter the Space Mapping algorithm will be outlined and discussed. The method can be parted into three algorithms: The main algorithm and two sub-algorithms. The main algorithm is summarized in the previous section, and is referred to as Algorithm 1. Each iteration in this algorithm consists of two optimization procedures:
The first optimization problem involves finding the next iterate by minimizing the surrogate model defined by the current mapping parameters. This sub-algorithm is called Algorithm 2.
The second optimization procedure - the Parameter Extraction - consists of m optimization problems each giving a new set of mapping parameters for the corresponding surrogate model response. Algorithm 3 is used in each of the m Parameter Extraction problems.

The three algorithms will be presented in pseudo-code in the next sections followed by comments on the involved parameters and procedures.

2.1 The Space Mapping Algorithm

The algorithms follow the theoretical introduction in section 1. Algorithm 1 builds on the MATLAB implementation by Frank Pedersen, but has been changed to a certain extent. The optimization problem in Algorithm 2 is solved by calling a Fortran subroutine from the F-package. The algorithm is identical to the original and has not been altered. The algorithm is not discussed in details and is presented here to give a full overview of the Space Mapping method. Algorithm 3 used in the Parameter Extraction is a new and different formulation compared to the existing framework by Frank Pedersen.

Before presenting the main Space Mapping algorithm we define:

p : Defines the norm used in the objective function $H : \|\cdot\|_{\mathtt{p}}$. Possible values are 1,2 or ∞, where the latter (minimax optimization) is used throughout this report.
Δ : Trust region radius used in Algorithm 2.
F : The objective function in the optimization problem.

The stopping criteria are defined by a number of optional parameters:

$\max_{f1}$: Maximal number of function evaluations in Algorithm 1.
$\max_{f2}$: Maximal number of function evaluations in Algorithm 2.
$\max_{f3}$: Maximal number of function evaluations in Algorithm 3.
ε_F : Used in stopping criterion for the objective function.
ε_K : Used in stopping criterion for the gradient matching.
ε_{hx} : Used in stopping criterion for the step length for $\mathbf{x}$-iterates.
ε_{hp} : Used in stopping criterion for the step length for $\mathbf{p}$-iterates.

The values for the parameters used in the stopping criteria must be defined in the problem setup-file, for more details see Appendix A.

2.1.1 The Main Algorithm

The surrogate $\mathbf{s}$ is given by (1.3.8) and the ith residual function $\mathbf{r}_i$ by the general formulation (1.3.12).
In the algorithm the superscript indexes $^{(k)}$ for iteration numbers are omitted to simplify the pseudo-code. The lower index 'new' corresponds to the upper index $^{(k+1)}$, for references to the formulation of the theory in section 1.3. It is assumed, that the surrogate model and the residual function in each iteration are defined by the current mapping parameters.

Algorithm 1: Main Algorithm for Space Mapping Iterations

$k = 0$; $stop = 0$; $\mathbf{x} \in \mathbb{R}^n$; $\Delta = 10^{-1} \cdot \|\mathbf{x}\|_2$
$\mathbf{A}_i = \mathbf{I}$; $\mathbf{b}_i = \mathbf{0}$; $\alpha_i = 1$; $\beta_i = \alpha_i c_i(\mathbf{x})$ `for i=1,...,m`
`while` not *stop*
  Find $\mathbf{h}_{new} = \arg\min_{\|\mathbf{h}\|_2 \le \Delta} \|\mathbf{s}(\mathbf{x}+\mathbf{h})\|_{\mathtt{p}}$ by Algorithm 2.
  Evaluate $\mathbf{x}_{new} = \mathbf{x} + \mathbf{h}_{new}$, $S_{new} = \|\mathbf{s}(\mathbf{x}_{new})\|_{\mathtt{p}}$ and $dS = S_{new} - F$
  Check stopping criteria $dS \ge 0$ and $\|\mathbf{h}_{new}\|_2 < \varepsilon_{hx} \cdot (\|\mathbf{x}\|_2 + \varepsilon_{hx})$
  Evaluate $\mathbf{f}_{new} = \mathbf{f}(\mathbf{x}_{new})$ and $\mathbf{J}_{f,new} = \mathbf{J}_f(\mathbf{x}_{new})$and $F_{new} = \|\mathbf{f}_{new}\|_{\mathtt{p}}$
  $dF = F_{new} - F$; $\rho = dF/dS$
  $k = k + 1$
  Add $\mathbf{x}_{new}$ and $\mathbf{f}_{new}$ to sorted internal datastructure
  $Active = |\Delta - \|\mathbf{h}_{new}\|_\infty| < 10^{-2}\Delta$
  `if` $dF < 0$
    $\mathbf{x} = \mathbf{x}_{new}$; $\mathbf{f} = \mathbf{f}_{new}$; $\mathbf{J}_f = \mathbf{J}_{f,new}$; $F = F_{new}$
  `end`
  Check stopping criteria $dF < \varepsilon_F$ and $k \ge \max_{f1}$
  `if` $\rho > 0.5$ & *Active*
    $\Delta = \Delta \cdot 2$
  `else if` $\rho < 10^{-4}$
    $\Delta = \Delta/3$
  `end`
  `for i = 1:m`
    Find $\{\mathbf{A}_{i,new}, \mathbf{b}_{i,new}, \alpha_{i,new}\} = \arg\min\{1/2 \cdot \mathbf{r}_i^{\mathrm{T}}\mathbf{r}_i\}$ by Algorithm 3.
    Set $\{\mathbf{A}_i, \mathbf{b}_i, \alpha_i\} = \{\mathbf{A}_{i,new}, \mathbf{b}_{i,new}, \alpha_{i,new}\}$
  `end`
`end`

Some remarks to Algorithm 1 are given below:

Initialization

The initial guess for the coarse model optimizer is $\mathbf{x}$, and the initialization of the parameters corresponds to the formulation in section 1.3. The elements of the corresponding surrogate model are given by (1.3.7) and are identical to the coarse model elements. The first optimization before entering the main loop hereby gives the coarse model optimizer. The value of the initial trust region is recommended to be $\Delta = 10^{-1} \cdot \|\mathbf{x}\|_2$ according to [13], but can be altered by the user in the problem setup-file, see Appendix A.

Optimization of the Surrogate

In the main loop the optimizer of the current surrogate function (defined by the current mapping parameters) is found. The step $\mathbf{h}_{new}$ and the objective function gain compared to the previous iterate is calculated. The formulation of the interpolating surrogate makes sure that $\mathbf{s}^{(k)}(\mathbf{x}^{(k)}) = \mathbf{f}(\mathbf{x}^{(k)})$, so that $S^{(k)} = F^{(k)}$.

Stopping Criteria

Two stopping criteria are checked at this point:
The change in the surrogate function must be negative, if not the new iterate is actually a worse solution than the previous one, and we want to exit. This stopping criterion is included as a safety to avoid an infinite loop. The optimization algorithm for the surrogate model uses a descent method, so we are ensured a decrease of dS. If $dS = 0$ we cannot improve the surrogate, and we exit the loop.
The second stopping criterion is concerning the relative step length and is defined by the optional parameter ε_{hx}. The formulation makes sure that the criterion is also useful, when $\|\mathbf{x}\|_2$ is close to zero. If the solution vector is too close to the previous one, we have not acchieved more information to get a new set of mapping parameters, and we are stuck at the current iterate.
Checking these two stopping criteria at this point of the algorithm makes sure, that unnecessary evaluations of the fine model are avoided.

Gain Ratio

We continue the main loop with evaluations of $\mathbf{f}$, $\mathbf{J}_f$ and the objective function F in the new iterate. The gain ratio ρ is the ratio between the true gain and the predicted gain. It serves as a measure for how well the surrogate model approximates the fine model, and is used for updating the trust region radius Δ.

Internal Data Structure

The iterate and the corresponding function vector and Jacobian are added to an internal data structure. The internal data structure contains:

F : The objective functions of the iterates number 1 to k sorted in ascending order.
X : Matrix with the iterates sorted according to F.
dX : Row vector with the norms of the distances from the sorted iterates in **X** to the best iterate.

By this sorting the first element in **F** will be the best objective function value so far, the first column in **X** will be the best iterate so far, and the first element in **dX** will be 0. The data is used in the Parameter Extraction problem and also for documentation and plotting after ended Space Mapping iterations.

The Active Flag

This flag is active ($= 1$), if the new solution is close to the boundary of the trust region, in the sense that the length of the step must be in the open interval $\|\mathbf{h}_{new}\|_\infty \in]0.99 \cdot \Delta \ , \ 1.01 \cdot \Delta[$. In theory it is impossible to have $\|\mathbf{h}_{new}\|_\infty > \Delta$, but in practice rounding errors can have an effect. An active flag equal to 1 indicates, that the trust region constraints are active, and the flag is used later for updating the trust region radius.

Update of the Iterate

If the objective function has decreased, we wish to accept the new iterate, and use this as an initial value in the next surrogate optimization.

Stopping Criteria

At this point two additional stopping criteria are checked:
We use the change in the objective function to formulate the stopping criterion: $dF < \varepsilon_F$. The relative change can be used in case F is not close to zero in the optimizer.
As a final safety towards an infinite loop we exit, if the number of main iterations has exceeded the maximum value $\max_{f1}$.

Update of the Trust Region

We use the following updating strategy for the trust region radius Δ:
If the gain ratio is larger than 0.5, and the new iterate is close to the bound-

ary of the trust region, we increase the trust region radius by a factor 2. A large value of ρ indicates, that the surrogate serves as a good approximation to the fine model. Since the active flag is 1, we have taken the largest step possible in the latest iteration. We would then like to increase Δ, and take longer steps.
If ρ is small, the surrogate is a poor approximation to the fine model, and we want to take smaller steps. A decrease of the trust region is made, if the gain ratio is smaller than the value 10^{-4}. This condition is quite strict, because of the fact that the surrogate has not yet been updated. By the update of the mapping parameters to follow, we hope that the surrogate model is improved. It is also important, that the trust region does not get too small, since the optimization procedure of the surrogate model would then get restricted.

Update of the Mapping Parameters

The mapping parameters are updated by Algorithm 3.

Each iteration in the main loop involves one evaluation of the fine model function vector and one evaluation of the fine model Jacobian. The iteration counter k is then equal to the number of $\mathbf{f}$- and $\mathbf{J}_f$-evaluations. Also a number of coarse model evaluations (through the surrogate evaluation) are performed. Since these are considered cheap compared to the fine model evaluations, we are only interested in the final amount of fine model evaluations.

2.1.2 The Algorithm for Surrogate Optimization

The algorithm used to solve the optimization of the surrogate is outlined in pseudo-code in the box below. Since we are mainly concerned with Algorithm 1 and 3 in this report, Algorithm 2 is only discussed briefly. We note that the iteration counter is used independently of the iteration counter of Algorithm 1.

Algorithm 2: Sub-algorithm for the Surrogate Optimization

Given global trust region radius Δ and initial parameter vector $\mathbf{x}^{(0)}$
Linear inequality constraints: $\hat{\mathbf{A}} = [\quad \mathbf{I}; \quad -\mathbf{I}]$
$\hat{\mathbf{b}} = [\quad -\mathbf{x}^{(0)} + \Delta; \quad \mathbf{x}^{(0)} + \Delta]$
Call to function `minnonlin`.
Call to Fortran subroutine (non-linear optimization in the norm p).
Initial local trust region radius $\Delta/4$
`stop if` $j \geq \max_{f2}$
`or if` $\mathbf{h} < \varepsilon_{hx} \cdot \|\mathbf{x}\|_2$

Initialization

The trust region radius Δ is given by Algorithm 1. To ensure that the optimizer of the surrogate function is inside the trust region, we introduce the linear inequality constraints given by:

$$\mathbf{x} + \left(-\mathbf{x}^{(0)} + \Delta\right) \geq \mathbf{0} \qquad \text{and} \qquad -\mathbf{x} + \left(\mathbf{x}^{(0)} + \Delta\right) \geq \mathbf{0}$$

which is equal to the conditions:

$$\mathbf{x} \geq \mathbf{x}^{(0)} - \Delta \qquad \text{and} \qquad \mathbf{x} \leq \mathbf{x}^{(0)} + \Delta$$

See the user's guide in Appendix A for more information on how to handle the case where the original minimization problem is constrained.

Function Call to `minnonlin` and Fortran Subroutine

This function is a helping function that, depending on the problem type, makes another function call to the proper Fortran subroutine, see [13]. Which optimization algorithm is used depends on the norm p, in which we want to minimize the surrogate function, and of wether the objective function is a scalar function or a vector function. Because of the trust region approach the optimization problem is always constrained.
Some of the Fortran subroutines use an optimization method with trust region, in this case the initial local trust region is set to $\Delta/4$. This local trust region has nothing to do with the global trust region Δ, which ensures, that the surrogate optimizer is not too far from the previous iterate.

Stopping Criteria

The Fortran subroutines require two optional parameters used in the stopping criteria of the algorithm: The maximum number of iterations allowed (max$_{f3}$, and the parameter ε_{hx} which is used in regard to the step length. The algorithm stops, when it suggests a step length $\mathbf{h}$, when $\mathbf{h} < \varepsilon_{hx} \cdot \|\mathbf{x}\|_2$.

2.1.3 The Algorithm for Parameter Extraction

The third algorithm, which performs the Parameter Extraction for each of the m response functions, is outlined below.

Algorithm 3: Sub-algorithm for Parameter Extraction

Given initial parameter vector $\mathbf{p}_k = [\mathbf{A}(:); \quad \mathbf{b}; \quad \alpha]$
The objective function is $\mathbf{r}$ and the last n rows of $\mathbf{r}$ are denoted $\mathbf{g}$.
The options in `marquardt` are given by `opts`.
$j = 0; \quad stop = 0; \quad \sigma = 1$
$K = \|\mathbf{g}\|_\infty; \quad K_r = \|\mathbf{r}\|_\infty$
`while` not stop
  $j = j + 1$
  Find $\mathbf{p}_{new} = \arg\min_{\mathbf{p}}\{1/2 \cdot \mathbf{r}(\mathbf{p})^{\mathrm{T}}\mathbf{r}(\mathbf{p})\}$ by `marquardt` with `opts`
  $\mathbf{r}_{new} = \mathbf{r}(\mathbf{p}_{new}); \quad K_{new} = \|\mathbf{g}_{new}\|_\infty; \quad K_{r,new} = \|\mathbf{r}_{new}\|_\infty$
  $\mathbf{h} = \mathbf{p}_{new} - \mathbf{p}; \quad Accept = K_{new} < K$
  `if` $Accept$
    $\mathbf{p} = \mathbf{p}_{new}; \quad K = K_{new}; \quad K_r = K_{r,new}$
  `end`
  `if` $K_{new} < \varepsilon_K$
    $stop = 1;$ `break`
  `else if` $\mathbf{h} < \varepsilon_{hp} \cdot (\|\mathbf{p}\|_2 + \varepsilon_{hp})$
    $stop = 2;$ `break`
  `else if` $j \geq \max_{f3}$
    $stop = 3;$ `break`
  `else if` $\sigma \geq 10^3$
    $stop = 4;$ `break`
  `end`
  `if` $\sigma < 10^3$
    $\sigma = \sigma \cdot 10$
  `end`
`end`

This algorithm is considered independently of Algorithm 1 and 2, and any references to iterations only concern the present Algorithm 3.

Initialization

We have given initial sets of the mapping parameters $\mathbf{A}$, $\mathbf{b}$ and α corresponding to an arbitrary response. The mapping parameters are arranged in the vector $\mathbf{p}_k$, where k denotes the mapping parameters defining the kth surrogate model. The residual function for the response is given by equation (1.3.12). The options used in the `marquardt`-function are set in the 5-element vector `opts` where:

`opts(1)` : Defines the initial value of the Marquardt parameter:
: $\mu_0 = \texttt{opts(1)} \max\{(\mathbf{J}_0^{\mathrm{T}}\mathbf{J}_0)_{(i,i)}\}$.
`opts(2)` : Parameter used in stopping criteria for the gradient:
: $\|F'(\mathbf{p})\|_\infty \leq \texttt{opts(2)}$.
`opts(3)` : Parameter used in stopping criteria for the step length:
: $\|dx\|_2 \leq \texttt{opts(3)}(\texttt{opts(3)} + \|\mathbf{p}\|_2)$.
`opts(4)` : Maximal number of iterations
`opts(5)` : Lower bound on μ: $\mu = \texttt{opts(5)} \max\{(\mathbf{J}^{\mathrm{T}}\mathbf{J})_{(i,i)}\}$.

The penalty factor σ used in the gradient residual is initialized to 1. K is a measure of the violation of the gradient match, and K_r of the matching of the full residual.

Optimization of the Residual

The current residual is optimized by the MATLAB-function `marquardt` implemented by Hans Bruun Nielsen to find the next parameter vector. This new solution is a result of a number of Marquardt steps, and the iterations ended because one of the stoppping criterias mentioned above was activated. The Marquardt algorithm is discussed further in section 3.2. The residual function vector is evaluated in the new iterate, as well as the new values of K and K_r and the step length.

The Accept Flag and Update of the Iterate

Since we consider the gradient match to be of great importance, we use the factor K to decide, wether the new set of parameters is better than the previous. The parameter vector is accepted (*Accept* $= 1$), only if the gradient match residual has been decreased compared to the previous iteration.

Stopping Criteria

We check four stopping criteria and exit the loop, if either of them are satisfied. The gradient match is considered satisfied, if K is smaller than the value ε_K. Secondly if the step between two consecutive iterates is relatively small, the step length criterion is activated. The third criterion stops the loop, if the number of iteration steps has exceeded the limit $\max_{f3}$. Finally we will only continue the iterations, while σ is below or equal to 10^3. This criterion is equivalent to using $\max_{f3} = 4$, if we choose the updating strategy below.

Update of the Penalty Factor

For the penalty factor σ we use a simple updating strategy: σ is increased by a factor 10 for each main iteration. In this way we force the gradient match to become of greater importance than the other elements of the residual.

Chapter 3

Theoretical and Practical Investigations

3.1 Finite Difference Approximation

To run the SMIS algorithm the user must supply two MATLAB-files with implementations of the coarse and fine model and their Jacobians. For some problems (including the TLT2 and TLT7 problems), no exact gradients are available, and the Jacobians are instead calculated by eg. difference approximations. In the test problems investigated in this report, the Jacobian is calculated by forward difference approximations by the MATLAB-function `diffjacobian`, see [15], implemented by Jacob Søndergaard. In the implementation of `diffjacobian` the step length is scaled according to the independent variable x, so that $h = \eta(1 + |x|)$. This formulation is useful in the case where $|x|$ is very small, and we get $h \simeq \eta$. The value of η originally had the fixed value $\eta = \sqrt{\varepsilon_M}$, where ε_M is the machine accuracy, but η can be changed by the user directly in the `m`-file.

By the investigation of the TLT2 problem, some interesting features regarding the model functions were dicovered. The fine, coarse and surrogate functions are smooth, but the gradients (partial derivatives) wrt. both $\mathbf{x}$ and $\mathbf{p}$ calculated from `diffjacobian` show a different picture, when the step length is small. This motivated an investigation of the step length in `diffjacobian`.

In the following sections the theory is simplified by looking at scalar functions whenever possible.

3.1.1 Optimal Step Length

The Jacobians of both the fine and coarse model are calculated by the MATLAB-function `diffjacobian`. In the following the influence of the step length used in the difference approximation will be analyzed based on a scalar example.

Given a scalar function $f : \mathbb{R} \rightarrow \mathbb{R}$ we compute a forward difference approximation to the gradient of f in the point x by:

$$D_F(x,h) = \frac{f(x+h) - f(x)}{h} \tag{3.1.1}$$

The Taylor expansion of f in the expansion point x is given by:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + O(h^3) \tag{3.1.2}$$

Inserting (3.1.2) in the forward difference approximation (3.1.1), we get:

$$D_F(x,h) = f'(x) + \frac{h}{2}f''(x) + O(h^2)$$

The truncation error E_T is now:

$$\begin{aligned} E_T &= D_F(x,h) - f'(x) \\ &= f'(x) + \frac{h}{2}f''(x) + O(h^2) - f'(x) \\ &= \frac{h}{2}f''(x) + O(h^2) \end{aligned} \tag{3.1.3}$$

and we see that the truncation error is $O(h)$ for $h \rightarrow 0$. If we also take the rounding errors into account, we get the floating point numbers $\bar{f}(x+h)$ and $\bar{f}(x)$ instead of $f(x+h)$ and $f(x)$:

$$\begin{aligned} \bar{f}(x+h) &= f(x+h)(1+\delta_1), \qquad && |\delta_1| \leq K\varepsilon_M \\ \bar{f}(x) &= f(x)(1+\delta_2), && |\delta_2| \leq K\varepsilon_M \end{aligned}$$

where the constant $K \geq 1$ and ε_M is the machine accuracy.
Inserting the above in the forward difference approximation (3.1.1) we get:

$$\begin{aligned} \bar{D}_F(x,h) &= \frac{\bar{f}(x+h) - \bar{f}(x)}{h} \\ &= \frac{f(x+h) - f(x)}{h} + \frac{\delta_1 f(x+h) - \delta_2 f(x)}{h} \\ &= D_F(x,h) + E_R \\ &= f'(x) + E_T + E_R \end{aligned} \tag{3.1.4}$$

where the rounding eror is denoted E_R. The worst possible rounding error is when $\delta_1 f(x+h)$ and $\delta_2 f(x)$ have opposite signs, giving:

$$\begin{aligned} |E_R| &\leq \frac{K\varepsilon_M\,(|f(x+h)|+|f(x)|)}{h} \\ &\simeq 2K|f(x)|\frac{\varepsilon_M}{h} \end{aligned} \tag{3.1.5}$$

The absolute total error is now given by the absolute difference between $\bar{D}_F$ and the real gradient:

$$\begin{aligned} |E| = |E_T + E_R| &= |\bar{D}_F(x,h) - f'(x)| \\ &\simeq Ah + O(h^2) + B\frac{\varepsilon_M}{h} \end{aligned} \tag{3.1.6}$$

The constants A and B depend on the function values and second derivatives in the neighbourhood of x, $A \simeq \frac{1}{2}|f''(x)|$ and $B \simeq 2K|f(x)|$.

Using the above we now consider the case when approximating the gradients wrt. the parameters in the $\mathbf{x}$-vector. The gradients appear in the Jacobians of the fine, coarse and surrogate models in the Parameter Extraction problem. We consider an arbitrary response function with no index on f and s. Each of the rows of the gradient residual (the last n rows of the residual from (1.3.12)) has the form:

$$g_i(\mathbf{x},\mathbf{p}) = s'_{x_i}(\mathbf{x},\mathbf{p}) - f'_{x_i}(\mathbf{x}) \tag{3.1.7}$$

$\mathbf{x}$ is a column vector holding the design parameters, and the vector $\mathbf{p}$ is holding the parameters $\mathbf{A}$, $\mathbf{b}$ and α. To simplify the calculations we consider only the ith row of the gradient residual as a function of the variable vectors $\mathbf{x}$ and $\mathbf{p}$ (keeping all other parameters than x_i fixed). Without loss of generality we also disregard the factors σ and d.

The exact function $g_i(\mathbf{x},\mathbf{p})$ is replaced by the approximated function $G_i(\mathbf{x},\mathbf{p})$ returned by the MATLAB-function, where the difference approximation wrt. x_i and the rounding errors gives the approximation to (3.1.7):

$$\begin{aligned} G_i(\mathbf{x},\mathbf{p}) &\simeq \frac{s(\mathbf{x}+h_x\mathbf{e}_i,\mathbf{p}) - s(\mathbf{x},\mathbf{p}) + B\varepsilon_M}{h_x} - f'_{x_i}(\mathbf{x}) \\ &= \left(s'_{x_i}(\mathbf{x},\mathbf{p}) - f'_{x_i}(\mathbf{x}) + \frac{h_x}{2}s''_{x_ix_i}(\mathbf{x},\mathbf{p}) + O(h_x^2) + B\frac{\varepsilon_M}{h_x}\right) \\ &= g_i(\mathbf{x},\mathbf{p}) + Ah_x + O(h_x^2) + B\frac{\varepsilon_M}{h_x} \end{aligned} \tag{3.1.8}$$

where $\mathbf{e}_i$ is a unit vector in the ith direction and h_x is the step length. The constant A depends on the second derivative of s, and B depends on the function values of s in the interval $x \in [x\ ,\ x+h]$.

By comparing (3.1.8) with the exact function (3.1.7), we get the total error for each of the gradient residual rows:

$$\begin{aligned} E_G(h_x) &= E_T + E_R \\ &\simeq Ah_x + O(h_x^2) + B\frac{\varepsilon_M}{h_x} \end{aligned} \tag{3.1.9}$$

where the differentiation is wrt. x_i for row number i, and the lower index of x has been omitted for simplicity.
When h_x is large the total error is dominated by the truncation error (the first two terms), whereas the rounding error (the last term) dominates for small h_x. To determine the optimal step length $h_{x,opt}$ in order to minimize the total error we differentiate (3.1.9) negligating the $O(h_x^2)$ term and get:

$$E'_G(h_x) = A - B\frac{\varepsilon_M}{h_x^2} \tag{3.1.10}$$

Equalizing (3.1.10) with 0 we have the optimal step length minimizing the error function:

$$E'_G(h_x) = 0 \qquad \Rightarrow \qquad h_{x,opt} = \sqrt{\varepsilon_M \frac{B}{A}} \tag{3.1.11}$$

In MATLAB the unit round-off is $\varepsilon_M \simeq 10^{-16}$, so the step length should be around $10^{-8}\sqrt{\frac{B}{A}}$ to give the smallest errors. In practice we don't distinguish between the x-variables and choose one suitable step length.
With typical values of the derivatives of the surrogate model of: $s_x(\mathbf{x}, \mathbf{p}) \sim 10^{-3}$, $s''_{xx}(\mathbf{x}, \mathbf{p}) \sim 10^{-4}$ we get the approximate value of the optimal step length $h_{x,opt} \sim 10^{-7}$. Here we have used the values for the constants $A = \frac{1}{2}10^{-4}$ and $B = 2 \cdot 10^{-3}$. The result agrees with figure 3.1.1, where the error function (3.1.9) for these values of A and B is plotted as a function of the step length h_x.

We now analyse the case where the difference approximation is made with the $\mathbf{p}$-parameters. We again consider the ith row of the gradient residual, and look at the gradient with respect to the jth parameter in the vector $\mathbf{p}$. All other variables are kept fixed.
We now have the (i, j)-element of the Jacobian:

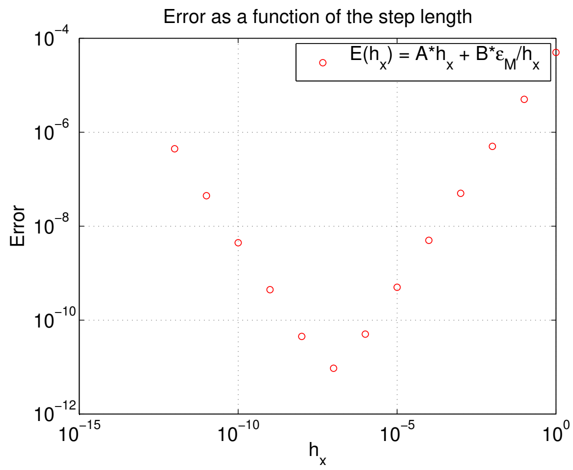

Figure 3.1.1: Errors as a function of the step length in the x-direction

$$
\begin{aligned}
(G_i)'_{p_j}(\mathbf{x},\mathbf{p}) &= \frac{G_i(\mathbf{x},\mathbf{p}+h_p\mathbf{e}_j) - G_i(\mathbf{x},\mathbf{p}) + D\varepsilon_M}{h_p} \\
&\simeq \frac{1}{h_p}\left[s'_{x_i}(\mathbf{x},\mathbf{p}+h_p\mathbf{e}_j) + \frac{h_x}{2}s''_{x_ix_i}(\mathbf{x},\mathbf{p}+h_p\mathbf{e}_j) + O(h_x^2) + B_1\frac{\varepsilon_M}{h_x}\right] \\
&\quad - \frac{1}{h_p}\left[s'_{x_i}(\mathbf{x},\mathbf{p}) + \frac{h_x}{2}s''_{x_ix_i}(\mathbf{x},\mathbf{p}) + O(h_x^2) + B_2\frac{\varepsilon_M}{h_x}\right] + D\frac{\varepsilon_M}{h_p} \\
&\simeq \frac{s'_{x_i}(\mathbf{x},\mathbf{p}+h_p\mathbf{e}_j) - s'_{x_i}(\mathbf{x},\mathbf{p})}{h_p} + D\frac{\varepsilon_M}{h_p} \\
&\quad + \frac{h_x}{2h_p}\left(s''_{x_ix_i}(\mathbf{x},\mathbf{p}+h_p\mathbf{e}_j) - s''_{x_ix_i}(\mathbf{x},\mathbf{p})\right) + 2B\frac{\varepsilon_M}{h_xh_p}
\end{aligned}
\tag{3.1.12}
$$

D is a constant depending on $G_i(\mathbf{x},\mathbf{p})$. We have assumed, that the truncation errors $O(h_x^2)$ almost cancel, and that the rounding error terms $B_1\frac{\varepsilon_M}{h_x}$ and $B_2\frac{\varepsilon_M}{h_x}$ in the worst case can be replaced by $2B\frac{\varepsilon_M}{h_x}$. The variation of $s''_{x_ix_i}$ is assumed small, so the first term in the last line cancels out. This reduces the approximation of $(G_i)'_{p_j}$ to:

$$
(G_i)'_{p_j}(\mathbf{x},\mathbf{p}) = s''_{x_ip_j}(\mathbf{x},\mathbf{p}) + Ch_p + O(h_p^2) + D\frac{\varepsilon_M}{h_p} + 2B\frac{\varepsilon_M}{h_xh_p} \tag{3.1.13}
$$

where the constant C depends on $s'''_{xpp}(\mathbf{x},\mathbf{p})$. The gradient of g_i wrt. to p_j is

$$
(g_i)'_{p_j}(\mathbf{p},\mathbf{x}) = s''_{x_ip_j}(\mathbf{p},\mathbf{x})
$$

We get the total error for an element in the Jacobian matrix as a function of the step length h_p:

$$\begin{aligned} E_{G'}(h_p) &= E_T + E_R \\ &\simeq Ch_p + O(h_p^2) \\ &\quad + D\frac{\varepsilon_M}{h_p} + 2B\frac{\varepsilon_M}{h_x h_p} \end{aligned} \tag{3.1.14}$$

where the truncation errors are gathered in the first line and the rounding errors in the second. For Jacobian matrix element (i, j) the differentiation is made wrt. the ith x-variable and the jth p-parameter.

When $h_x = 10^{-7}$ and the order of magnitude of $s'''_{xpp}(\mathbf{x}, \mathbf{p})$ is 10^{-3} the optimal step length in the p-direction is $h_{p,opt} \simeq 10^{-4}$. If we instead put $h_p = h_x$ we get an approximate optimal step length of $h_{p,opt} \simeq 10^{-5}$ as shown in figure 3.1.2 for $B = 2 \cdot 10^{-3}$, $C = \frac{1}{2}10^{-3}$ and $D = 10^{-3}$.

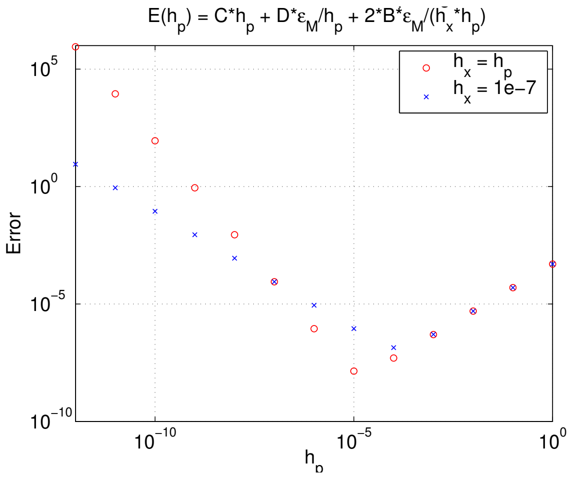

Figure 3.1.2: Errors as a function of the step length in the p-direction

3.1.2 Results from the TLT2 Problem

To make qualified values for the step lengths $h_{x,opt}$ and $h_{p,opt}$ we consider the gradient residual function and the partial derivatives wrt. the parameters in the $\mathbf{p}$-vector. These functions have been investigated in MATLAB for the TLT2 problem, which is considered in section 4.3. All computations are made with fixed $\mathbf{x} = \mathbf{z}^* = [90 \;,\; 90]^{\mathrm{T}}$ and the initial parameter vector $\mathbf{p} = [1 \;,\; 1 \;,\; 0 \;,\; 0 \;,\; 1]^{\mathrm{T}}$ corresponding to the diagonal matrix $\mathbf{A} = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$, $\mathbf{b} = \left[\begin{smallmatrix} 0 \\ 0 \end{smallmatrix}\right]$ and $\alpha = 1$. First an investigation of the function $G(\mathbf{p})$ made with the step length $h_x = 10^{-8}$ and $h_p = 10^{-8}$. The gradient residual is calculated in an area in the (x_1, x_2)-plane:

As figure 3.1.3 show the functions are smooth in the area of relevant x-values, and the gradient $g'_p(\mathbf{x}, \mathbf{p})$ is expected to be smooth too. Nevertheless

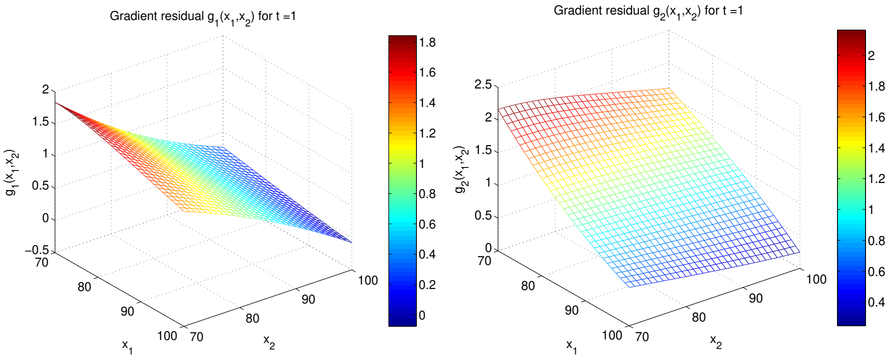

Figure 3.1.3: Gradient residuals

the figures 3.1.4 - 3.1.5 show something different.

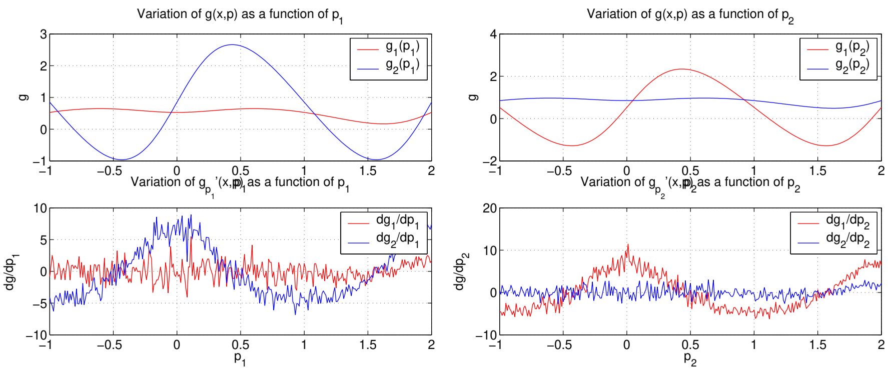

Figure 3.1.4: Functions of p_1 and p_2

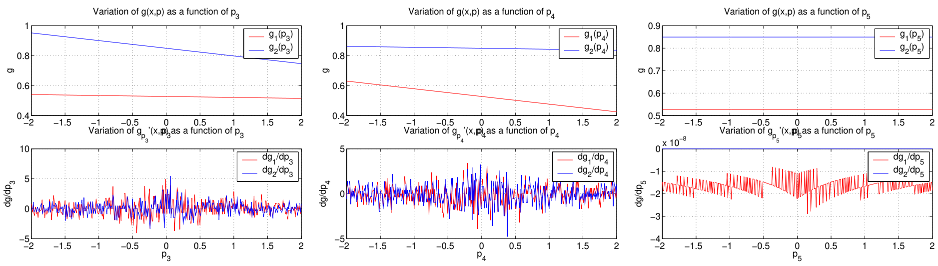

Figure 3.1.5: Functions of p_3, p_4 and p_5

The upper figures show the function $g(\mathbf{x}, \mathbf{p})$ depicted as a function of each of the five parameters in the $\mathbf{p}$-vector. The lower figures show the partial derivatives of $g(\mathbf{x}, \mathbf{p})$ with respect to the each of the five parameters. The functions have been calculated for a certain range of parameters with a spacing of 0.01. It is obvious, that the curves are not smooth for any of the five parameters (corresponding to elements of $\mathbf{A}$ and $\mathbf{b}$ and the α-parameter).

The same investigations made with a larger step length $h_x = 10^{-5}$ and $h_p = 10^{-5}$ show a much better picture:

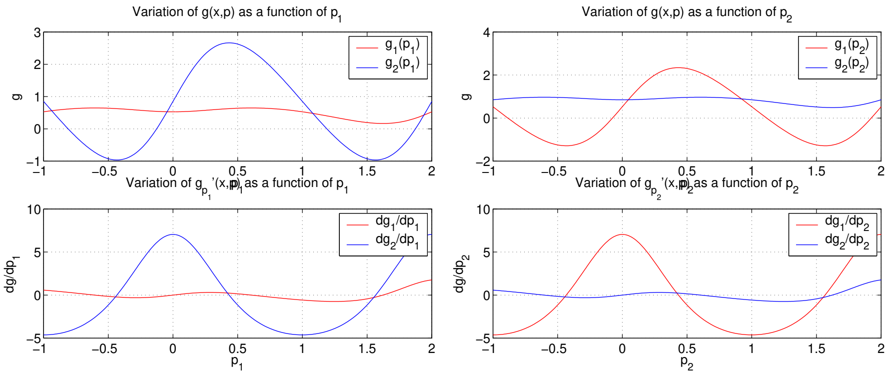

Figure 3.1.6: Functions of p_1 and p_2

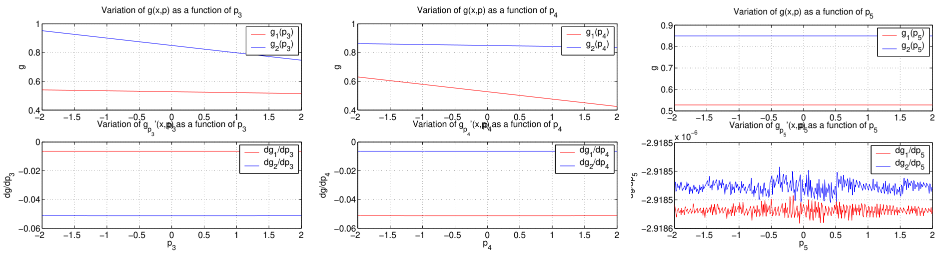

Figure 3.1.7: Functions of p_3, p_4 and p_5

The scaling of the ordinate axis of figure 3.1.7 (right) is noted, and even though the curve is still not smooth, the variation is considered of no practical importance.
This example shows, that the step length used in the difference approximations is of crucial importance. If the step length is too small, it seems that the rounding errors dominate, and the results thereby are not reliable. In two points very close to each other the calculated gradient varies extremely for the small step length, which is not the case for an increasing step length. A suitable step length in both the x- and the p-direction is chosen to 10^{-5}. This step length agrees with the theoretical optimal step lengths found in section 3.1.1, and is used in all further calculations.

3.2 The Marquardt Algorithm

The Marquardt algorithm is an important part of the Parameter Extraction problem solved by Algorithm 3 in section 2.1.3. The theory behind the Marquardt equation is briefly listed, and the solution is analysed through a Singular Value Decomposition (SVD). The cases of underdetermined and overdetermined equation systems are discussed.
We disregard the lower index corresponding to the response number.

The Marquardt method is a method for solving non-linear least squares problems. The aim is to minimize the norm of a vector function $\mathbf{r} : \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_r}$. We introduce the objective function $F(\mathbf{p})$ and wish to find:

$$\mathbf{p}^* = \arg \min_{\mathbf{p} \in \mathbb{R}^{n_p}} \{F(\mathbf{p})\}$$

where

$$F(\mathbf{p}) = \frac{1}{2} \sum_{i=1}^{n_r} (r_i(\mathbf{p}))^2 = \frac{1}{2} \mathbf{r}(\mathbf{p})^{\mathrm{T}} \mathbf{r}(\mathbf{p}) \tag{3.2.1}$$

Minimizing the objective function $F(\mathbf{p})$ is equivalent to finding the minimum of the norm of the vector function $\mathbf{r}(\mathbf{p})$.

When minimizing a linearized model of F based on a linear Taylor model of $\mathbf{r}$ we get the Gauss-Newton step by solving the equation:

$$\mathbf{J}_r(\mathbf{p})^{\mathrm{T}} \mathbf{J}_r(\mathbf{p}) \mathbf{h} = -\mathbf{J}_r(\mathbf{p})^{\mathrm{T}} \mathbf{r}(\mathbf{p}) \tag{3.2.2}$$

The matrix $\mathbf{J}_r(\mathbf{p})^{\mathrm{T}} \mathbf{J}_r(\mathbf{p})$ is symmetric and positive semidefinite. If the Jacobian has full rank, $\mathbf{J}_r(\mathbf{p})^{\mathrm{T}} \mathbf{J}_r(\mathbf{p})$ is positive definite, giving a unique solution to (3.2.2) and a step in a descent direction. We introduce the damping parameter μ and get the Marquardt equation:

$$\left(\mathbf{J}_r(\mathbf{p})^{\mathrm{T}} \mathbf{J}_r(\mathbf{p}) + \mu \mathbf{I}\right) \mathbf{h} = -\mathbf{J}_r(\mathbf{p})^{\mathrm{T}} \mathbf{r}(\mathbf{p}) \tag{3.2.3}$$

The matrix $\mathbf{J}_r(\mathbf{p})^{\mathrm{T}} \mathbf{J}_r(\mathbf{p}) + \mu \mathbf{I}$ is now guaranteed to be positive definite, and the Marquardt step is a descent direction.

3.2.1 Singular Value Decomposition

The Marquardt step is the solution to the equation:

$$\left(\mathbf{J}^{\mathrm{T}} \mathbf{J} + \mu \mathbf{I}\right) \mathbf{h} = -\mathbf{J}^{\mathrm{T}} \mathbf{r} \tag{3.2.4}$$

where the rank of the Jacobian is q. A singular value decomposition of $\mathbf{J}$ provides the matrix factorization:

$$\mathbf{J} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathrm{T}}$$

where the matrices $\mathbf{U}$, $\mathbf{V}$ and $\boldsymbol{\Sigma}$ are:

$$\mathbf{U} \in \mathbb{R}^{n_r \times n_r}, \qquad \mathbf{V} \in \mathbb{R}^{n_p \times n_p}, \qquad \boldsymbol{\Sigma} \in \mathbb{R}^{n_r \times n_p}.$$

The columns of $\mathbf{U}$ and $\mathbf{V}$ form orthonormal bases for the vector spaces $\mathbb{R}^{n_r}$, $\mathbb{R}^{n_p}$ respectively, and the matrix $\boldsymbol{\Sigma}$ is a 'diagonal' matrix with the singular values on the main diagonal $\boldsymbol{\Sigma}_{(i,i)} = \sigma_i, i = 1, \ldots \min\{n_r, n_p\}$ and zeros everywhere else. The singular values $\sigma_1, \ldots \sigma_{\min\{n_r,n_p\}}$ appear in decreasing order, and the first q values are strictly positive:

$$\sigma_1 \geq \cdots \geq \sigma_q > 0 \qquad \text{and} \qquad \sigma_{q+1} = \cdots = \sigma_{\min\{n_r,n_p\}} = 0$$

The jth column in $\mathbf{U}$ resp. $\mathbf{V}$ is denoted $\mathbf{u}_j$ resp. $\mathbf{v}_j$ and we can write the Jacobian and its transpose as a summation:

$$\mathbf{J} = \sum_{j=1}^{q} \sigma_j \mathbf{u}_j \mathbf{v}_j^{\mathrm{T}} \qquad \mathbf{J}^{\mathrm{T}} = \sum_{j=1}^{q} \sigma_j \mathbf{v}_j \mathbf{u}_j^{\mathrm{T}} \tag{3.2.5}$$

where the summation only runs to q, because of the fact that $\sigma_j = 0, j = q+1, \ldots \min\{n_r, n_p\}$. Hereby the matrix product $\mathbf{J}^{\mathrm{T}}\mathbf{J}$ becomes:

$$\begin{aligned}
\mathbf{J}^{\mathrm{T}}\mathbf{J} &= \sum_{i=1}^{q} \sigma_i \mathbf{v}_i \mathbf{u}_i^{\mathrm{T}} \sum_{j=1}^{q} \sigma_j \mathbf{u}_j \mathbf{v}_j^{\mathrm{T}} \\
&= \sum_{i=1}^{q} \sum_{j=1}^{q} \sigma_i \sigma_j \mathbf{v}_i \mathbf{u}_i^{\mathrm{T}} \mathbf{u}_j \mathbf{v}_j^{\mathrm{T}} \\
&= \sum_{j=1}^{q} \sigma_j^2 \mathbf{v}_j \mathbf{v}_j^{\mathrm{T}}
\end{aligned}$$

where we have used the characteristic feature of the orthonormal basis vectors:

$$\langle \mathbf{u}_i, \mathbf{u}_j \rangle = \mathbf{u}_i^{\mathrm{T}} \mathbf{u}_j = \begin{cases} 1 & \text{for } i = j, \\ 0 & \text{for } i \neq j \end{cases} \tag{3.2.6}$$

We can write the step $\mathbf{h} \in \mathbb{R}^{n_p}$ as a linear combination of the basis vectors $\mathbf{v}_j$, which span the space $\mathbb{R}^{n_p}$:

$$\mathbf{h} = \sum_{j=1}^{n_p} \eta_j \mathbf{v}_j \tag{3.2.7}$$

and the squared norm of $\mathbf{h}$ is given by the inner product:

$$||\mathbf{h}||_2^2 = \langle \mathbf{h}, \mathbf{h} \rangle = \langle \sum_{i=1}^{n_p} \eta_i \mathbf{v}_i, \sum_{j=1}^{n_p} \eta_j \mathbf{v}_j \rangle = \sum_{j=1}^{n_p} \eta_j^2 \tag{3.2.8}$$

Furthermore we can write the vector function $\mathbf{r} \in \mathbb{R}^{n_r}$ by use of the basis vectors $\mathbf{u}_j$:

$$\mathbf{r} = \sum_{j=1}^{n_r} \varphi_j \mathbf{u}_j$$

with the coefficients φ_j given by:

$$\begin{aligned} & \mathbf{u}_i^{\mathrm{T}} \mathbf{r} = \mathbf{u}_i^{\mathrm{T}} \sum_{j=1}^{n_r} \varphi_j \mathbf{u}_j && \text{for } i = 1, \dots, n_r \\ \Rightarrow \quad & \varphi_j = \mathbf{u}_j^{\mathrm{T}} \mathbf{r} && \text{for } j = 1, \dots, n_r \end{aligned}$$

Now we can use (3.2.5) and the above relation to write the right hand side of (3.2.4) as:

$$-\mathbf{J}^{\mathrm{T}} \mathbf{r} = -\sum_{j=1}^{q} \sigma_j \mathbf{v}_j \mathbf{u}_j^{\mathrm{T}} \mathbf{r} = -\sum_{j=1}^{q} \sigma_j \mathbf{v}_j \varphi_j \tag{3.2.9}$$

and the left hand side as:

$$\begin{aligned} \left(\mathbf{J}^{\mathrm{T}} \mathbf{J} + \mu \mathbf{I} \right) \mathbf{h} &= \left(\sum_{i=1}^{q} \sigma_i^2 \mathbf{v}_i \mathbf{v}_i^{\mathrm{T}} + \mu \mathbf{I} \right) \sum_{j=1}^{n_p} \eta_j \mathbf{v}_j \\ &= \sum_{i=1}^{q} \sum_{j=1}^{n_p} \sigma_i^2 \mathbf{v}_i \mathbf{v}_i^{\mathrm{T}} \eta_j \mathbf{v}_j + \mu \sum_{j=1}^{n_p} \eta_j \mathbf{v}_j \\ &= \sum_{j=1}^{q} \sigma_j^2 \eta_j \mathbf{v}_j + \mu \sum_{j=1}^{n_p} \eta_j \mathbf{v}_j \\ &= \sum_{j=1}^{q} (\sigma_j^2 + \mu) \eta_j \mathbf{v}_j + \mu \sum_{j=q+1}^{n_p} \eta_j \mathbf{v}_j \end{aligned} \tag{3.2.10}$$

The only unknowns in this equation are the coefficients $\eta_j, j = 1, \dots n_p$. Equating ((3.2.9)) and ((3.2.10)) gives:

$$\sum_{j=1}^{q} (\sigma_j^2 + \mu) \eta_j \mathbf{v}_j + \mu \sum_{j=q+1}^{n_p} \eta_j \mathbf{v}_j = -\sum_{j=1}^{q} \sigma_j \varphi_j \mathbf{v}_j \tag{3.2.11}$$

Multiplicating (3.2.11) with $\mathbf{v}_i^{\mathrm{T}}$ for $i = 1, \ldots, n_p$ results in n equations of the form:

$$\mathbf{v}_i^{\mathrm{T}} \sum_{j=1}^{q} (\sigma_j^2 + \mu)\eta_j \mathbf{v}_j + \mathbf{v}_i^{\mathrm{T}} \mu \sum_{j=q+1}^{n_p} \eta_j \mathbf{v}_j = -\mathbf{v}_i^{\mathrm{T}} \sum_{j=1}^{q} \sigma_j \varphi_j \mathbf{v}_j \quad \text{for } i = 1, \ldots, n_p$$

which by the property (3.2.6) of the orthonormal basis vectors give the n_p equations:

$$\begin{aligned} (\sigma_j^2 + \mu)\eta_j &= -\sigma_j \varphi_j && \text{for } j = 1, \ldots, q \\ \mu \eta_j &= 0 && \text{for } j = q+1, \ldots, n_p \end{aligned}$$

We hereby get the coefficients η_j:

$$\eta_j = -\frac{\sigma_j \varphi_j}{(\sigma_j^2 + \mu)} \qquad \text{for } j = 1, \ldots, q \tag{3.2.12a}$$

$$\eta_j = 0 \qquad \text{for } j = q+1, \ldots, n_p \tag{3.2.12b}$$

From the above expressions we see, that if $q < n_p$ we lack information about the last $n_p - q$ coefficients of (3.2.12a), and the coefficients are set to zero. This is the case when the columns of the Jacobian are linearly dependent.

For the Gauss-Newton step without the damping parameter, the equation system giving the step is:

$$\mathbf{J}^{\mathrm{T}}\mathbf{J}\mathbf{h} = -\mathbf{J}^{\mathrm{T}}\mathbf{r} \tag{3.2.13}$$

By using the SVD we can write (3.2.13) as:

$$\sum_{j=1}^{q} \sigma_j^2 \eta_j \mathbf{v}_j = -\sum_{j=1}^{q} \sigma_j \varphi_j \mathbf{v}_j \tag{3.2.14}$$

The coefficients in (3.2.7) are now given by:

$$\sigma_j^2 \eta_j = -\sigma_j \varphi_j \qquad \text{for } j = 1, \ldots, q \tag{3.2.15}$$

We have no information for the last $n_p - q$ coefficients, and these can be chosen freely. It follows from (3.2.8), that the length of the step $\mathbf{h}$ can be expressed in terms of the coefficients η_j. The shortest step is found by putting all free coefficients equal to zero. This step is the minimum norm solution to (3.2.13).

$$\eta_j = -\frac{\varphi_j}{\sigma_j} \qquad \text{for } j = 1, \ldots, q \tag{3.2.16a}$$

$$\eta_j = 0 \qquad \text{for } j = q+1, \ldots, n_p \tag{3.2.16b}$$

Comparing the solution to the Marquardt equation ($\mu \neq 0$) and the minimum norm solution to (3.2.13) we see, that for $\mu << \sigma_q$ the solutions are nearly identical. In this case the Marquardt step gives a new iterate closest to the previous iterate, which can be desirable to avoid ending up with a solution very far from the initial one.

The Marquardt algorithm used in Algorithm 3 is the implementation by Hans Bruun Nielsen [12]. The initial value for the damping parameter μ is given in the input argument vector `opts`, in the sense that μ_0 is scaled according to the maximal diagonal element of the matrix $\mathbf{J}(\mathbf{p}^{(0)})^{\mathrm{T}}\mathbf{J}(\mathbf{p}^{(0)})$. The `opts`-vector also provides a lower bound on μ. This ensures, that we do not have $\mu \to 0$, and thereby that the Marquardt equations always give a small regularization effect.

Underdetermined System of Equations

In the case $n_r < n_p$ where the residual has less rows than the number of parameters, the parameter solution is not well-defined. If the Jacobian at the optimizer has full rank, then $q = n_r$ and the solution has $n_p - n_r$ free parameters, and the Parameter Extraction problem has an $(n_p - n_r)$-infinity of solutions. The size of the damping parameter determines the Marquardt step in each iteration. For a small μ-value compared to the smallest singular value of $\mathbf{J}$ the solution is very close to the minimum norm solution, which is the solution closest to the previous one.

Overdetermined System of Equations

In the case of a quadratic Jacobian matrix, the solution is unique, provided that the Jacobian has full rank. If not, the rank as before determines the number of free parameters.
If we have more residual elements than parameters, we may not be able to satisfy all equations. The least squares solution provided by the Marquardt algorithm is the solution, that minimizes the residual.
When regularizing the residual function wrt. the initial solution as described in section 3.3 we always have more residual rows than unknown parameters,

and the rank of the Jacobian is guaranteed to be full.

3.3 Regularization

In the Space Mapping algorithm the Parameter Extraction involves the minimization of the residual given by (1.3.12). We can not always be sure, that the solution to the Parameter Extraction is unique. Different techniques can be used to constrain the solution space, see for example [7] and [3] for other strategies and further discussion.
One approach is to choose the solution closest to the initial parameters. By adding a regularization term to the residual vector, the distance to the former solution is penalized. In this section we provide an analysis of the solution to the regularized equation system.

In the general case we consider the residual corresponding to an arbitrary response, and ommit the lower index to simplify the equations. The regularization is made using $\mathbf{p}_k$ - the mapping parameters defining the kth surrogate model - and the modified residual function is denoted $\hat{\mathbf{r}}$:

$$\hat{\mathbf{r}}(\mathbf{p}) = \hat{\mathbf{r}} = \begin{bmatrix} \mathbf{r}(\mathbf{p}) \\ (\mathbf{p} - \mathbf{p}_k) \end{bmatrix} \tag{3.3.1}$$

The first vector function $\mathbf{r}$ is the original residual function (1.3.12) consisting of n_r rows, and the second term is the regularization term, which is added to the vector to ensure, that the optimizer $\mathbf{p}^*$ is close to the initial solution $\mathbf{p}_k$, which is the mapping parameter vector corresponding to the kth surrogate. When minimizing (3.3.1) by Marquardt we have:

$$\begin{aligned} F(\mathbf{p}) &= \frac{1}{2} \sum_{i=1}^{n_r+n_p} (\hat{r}_i(\mathbf{p}))^2 \\ &= \frac{1}{2} \sum_{i=1}^{n_r} (r_i(\mathbf{p}))^2 + \frac{1}{2} \sum_{i=1}^{n_p} (\mathbf{p}_i - \mathbf{p}_{i,k})^2 \\ &= \frac{1}{2}\mathbf{r}(\mathbf{p})^{\mathrm{T}}\mathbf{r}(\mathbf{p}) + \frac{1}{2}(\mathbf{p} - \mathbf{p}_k)^{\mathrm{T}}(\mathbf{p} - \mathbf{p}_k) \end{aligned} \tag{3.3.2}$$

The Jacobian matrix corresponding to (3.3.1) is given by:

$$\mathbf{J}_{\hat{r}}(\mathbf{p}) = \mathbf{J}_{\hat{r}} = \begin{bmatrix} \mathbf{J}_r \\ \mathbf{I} \end{bmatrix} \tag{3.3.3}$$

where $\mathbf{J}_r \in \mathbb{R}^{n_r \times n_p}$ is the Jacobian of the function $\mathbf{r}$ and $\mathbf{I} \in \mathbb{R}^{n_p \times n_p}$ is the identity matrix. The Marquardt step with the damping parameter μ is found by the equation system:

$$\left(\mathbf{J}_{\hat{r}}^{\mathrm{T}}\mathbf{J}_{\hat{r}} + \mu\mathbf{I}\right)\mathbf{h} = -\mathbf{J}_{\hat{r}}^{\mathrm{T}}\hat{\mathbf{r}} \tag{3.3.4}$$

where the next iterate is given by $\mathbf{p}_{new} = \mathbf{p} + \mathbf{h}$ The matrix multiplication $\mathbf{J}_{\hat{r}}(\mathbf{p})^{\mathrm{T}}\mathbf{J}_{\hat{r}}(\mathbf{p})$ gives:

$$\begin{aligned}\mathbf{J}_{\hat{r}}^{\mathrm{T}}\mathbf{J}_{\hat{r}} &= \begin{bmatrix}\mathbf{J}_r^{\mathrm{T}} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{J}_r \\ \mathbf{I}\end{bmatrix} \\ &= \mathbf{J}_r^{\mathrm{T}}\mathbf{J}_r + \mathbf{I}\end{aligned}$$

which inserted in (3.3.4) produces the left hand side of the equation system:

$$\left(\mathbf{J}_{\hat{r}}^{\mathrm{T}}\mathbf{J}_{\hat{r}} + \mu\mathbf{I}\right)\mathbf{h} = \left(\mathbf{J}_r^{\mathrm{T}}\mathbf{J}_r + (1+\mu)\mathbf{I}\right)\mathbf{h} \tag{3.3.5}$$

The right hand side is given by:

$$\begin{aligned}-\mathbf{J}_{\hat{r}}^{\mathrm{T}}\hat{\mathbf{r}} &= -\begin{bmatrix}\mathbf{J}_r^{\mathrm{T}} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{r} \\ (\mathbf{p}-\mathbf{p}_k)\end{bmatrix} \\ &= -\left(\mathbf{J}_r^{\mathrm{T}}\mathbf{r} + (\mathbf{p}-\mathbf{p}_k)\right) \end{aligned} \tag{3.3.6}$$

Inserting (3.3.5) and (3.3.6) in (3.3.4) gives the Marquardt step $\mathbf{h}$ as the solution to:

$$\left(\mathbf{J}_r^{\mathrm{T}}\mathbf{J}_r + (1+\mu)\mathbf{I}\right)\mathbf{h} = -\left(\mathbf{J}_r^{\mathrm{T}}\mathbf{r} + (\mathbf{p}-\mathbf{p}_k)\right) \tag{3.3.7}$$

It is seen from (3.3.7), that the last term on the right hand side vanishes for $\mathbf{p} = \mathbf{p}_k$, corresponding to the first Marquardt step.
In each iteration the Marquardt step $\mathbf{h} = \mathbf{p} - \mathbf{p}_k$ is found by solving (3.3.7). In the first iteration the equation system corresponds to minimizing the function $\mathbf{r}(\mathbf{p})$ by the Marquardt method with the damping parameter $1+\mu$. In the following steps the damping parameter is still $1+\mu$, but the right hand side is changed, because the regularization is made wrt. the initial set of parameters. This corresponds to an increase of the gradient $\mathbf{J}_r^{\mathrm{T}}\mathbf{r}$ as we move further away from the initial solution $\mathbf{p}_k$.

By adding the regularization term to the residual vector, we ensure that the found solution is not too far from the initial solution. Since the number of unknowns is n_p and the number of rows in the residual vector and it's Jacobian is now $n_r + n_p$, we have an overdetermined equation system for all iterations. In the optimizer $\mathbf{p}^*$ the gradient of the regularized residual function is zero, which means that:

$$\mathbf{J}_r(\mathbf{p}^*)^{\mathrm{T}}\mathbf{r}(\mathbf{p}^*) + (\mathbf{p}^* - \mathbf{p}_k) = \mathbf{0} \Rightarrow \qquad \mathbf{J}_r(\mathbf{p}^*)^{\mathrm{T}}\mathbf{r}(\mathbf{p}^*) = -(\mathbf{p}^* - \mathbf{p}_k)$$

The gradient of the original residual function is equal to the negative distance from the optimizer to the initial parameter vector. If the optimizer is far from the initial vector we can not necessarily expect, that the original residual match is good. This motivates the use of weighting factors in the residual.

When taking the weighting factors in the residual function into consideration the regularized residual vector is given by:

$$\hat{\mathbf{r}}(\mathbf{p}) = \begin{bmatrix} \mathbf{W}\,\mathbf{r}(\mathbf{p}) \\ \mathbf{V}(\mathbf{p}-\mathbf{p}_k) \end{bmatrix} \tag{3.3.8}$$

with the residual corresponding to an arbitrary response

$$\mathbf{r}(\mathbf{p}) = \begin{bmatrix} a_1 \cdot \left(s^{(k+1)}(\mathbf{x}^{(1)},\mathbf{p}) - f(\mathbf{x}^{(1)})\right) \\ \vdots \\ a_k \cdot \left(s^{(k+1)}(\mathbf{x}^{(k)},\mathbf{p}) - f(\mathbf{x}^{(k)})\right) \\ d \cdot \left(\mathbf{J}_s^{(k+1)}(\mathbf{x}^{(k+1)},\mathbf{p}) - \mathbf{J}_f(\mathbf{x}^{(k+1)})\right) \end{bmatrix} \tag{3.3.9}$$

The matrices $\mathbf{W} \in \mathbb{R}^{n_r \times n_r}$ and $\mathbf{V} \in \mathbb{R}^{n_p \times n_p}$ are diagonal matrices with the weight factor for element number i in the ith row:

$$\mathbf{W} = \begin{bmatrix} w_1 & & & & & \mathbf{0} \\ & \ddots & & & & \\ & & w_k & & & \\ & & & \sigma & & \\ & & & & \ddots & \\ \mathbf{0} & & & & & \sigma \end{bmatrix} \qquad \mathbf{V} = \begin{bmatrix} v_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & v_{n_p} \end{bmatrix}$$

The Marquardt step is found from the equation system:

$$\left(\mathbf{J}_r^{\mathrm{T}}\mathbf{W}^2\mathbf{J}_r + \mathbf{V}^2 + \mu\mathbf{I}\right)\mathbf{h} = -\mathbf{J}_r^{\mathrm{T}}\mathbf{W}^2\mathbf{r} + \mathbf{V}^2(\mathbf{p}-\mathbf{p}_k) \tag{3.3.10}$$

3.4 Variable Number of Mapping Parameters

The uniqueness of the solution to the Parameter Extraction depends on the number of mapping parameters in relation to the number of elements in the residual. But as discussed in section 3.2 it really depends on the rank of the Jacobian of the residual. In this section we consider the input mapping separately to analyze the effect of the number of mapping parameters on the surrogate model and it's derivatives.

In the following we consider an arbitrary response function with no subscript index. The equations are visualized in the case $n = 2$, which can be generalized to any value of n. The input mapping proposed in the Space Mapping formulation (1.3.1) is a linear mapping:

$$\mathbf{z} = \mathbf{P}(\mathbf{x}, \mathbf{p}) = \mathbf{A}\mathbf{x} + \mathbf{b} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{3.4.1}$$

The $\mathbf{z}$-vector is the transformed design parameter vector, which is inserted in the coarse model function. We have included the vector $\mathbf{p}$ in the formulation, since the mapping also depends on the parameter vector.
We arrange the input mapping parameters in the vector $\mathbf{p} \in \mathbb{R}^{(n(n+1))}$ with the elements of $\mathbf{A}$ inserted rowwise (the output mapping parameter α is not considered here):

$$\mathbf{p} = \begin{bmatrix} A_{11} & A_{12} & A_{21} & A_{22} & b_1 & b_2 \end{bmatrix}^{\mathrm{T}} \tag{3.4.2}$$

If we instead consider the mapping as a function of the parameter vector $\mathbf{p}$, we can write (3.4.1) as:

$$\mathbf{z} = \mathbf{H}(\mathbf{x})\mathbf{p}, \qquad \mathbf{H}(\mathbf{x}) = \begin{bmatrix} x_1 & x_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1 & x_2 & 0 & 1 \end{bmatrix} \tag{3.4.3}$$

where the matrix $\mathbf{H} \in \mathbb{R}^{n \times (n(n+1))}$ depends on the x-variables. The matrix has at most one element different from zero in each column. Because of the trailing identity matrix, it is obvious, that the rank of $\mathbf{H}$ is n, and consequently for a fixed $\mathbf{x}$, there are infinitely many $\mathbf{p}$-vectors, that satisfy (3.4.3). We can choose n^2 variables freely.
It must be clarified, that the equation for the input mapping is not solved directly in the form (3.4.1), since the Parameter Extraction is done by minimizing the residual function[1].

In solving the Parameter Extraction problem we need the Jacobian of the surrogate model. Hereby we use the derivatives of the mapping wrt. both the x- and the p-variables. By the formulations (3.4.1) and (3.4.3) the Jacobians of $\mathbf{P}(\mathbf{x}, \mathbf{p})$ wrt. $\mathbf{x}$ and wrt. $\mathbf{p}$ are easily seen to be:

$$\mathbf{P}'_x(\mathbf{x}, \mathbf{p}) = \mathbf{A}, \qquad \mathbf{P}'_p(\mathbf{x}, \mathbf{p}) = \mathbf{H}(\mathbf{x}) \tag{3.4.4}$$

The surrogate model is given by the composed mapping $s = O \circ c \circ \mathbf{P}$:

$$\begin{aligned} s(\mathbf{x}, \mathbf{p}) &= \alpha\left(c\left(\mathbf{P}(\mathbf{x}, \mathbf{p})\right) - c\left(\mathbf{P}(\bar{\mathbf{x}}, \mathbf{p})\right)\right) + f(\bar{\mathbf{x}}) \\ &= \alpha\left(c(\mathbf{z}) - c(\bar{\mathbf{z}})\right) + f(\bar{\mathbf{x}}) \end{aligned} \tag{3.4.5}$$

[1]Compare with the original Space Mapping formulation (1.1.1), where (3.4.3) is solved directly

We use the chain rule to differentiate wrt. $\mathbf{x}$:

$$\begin{aligned} s'_x(\mathbf{x},\mathbf{p}) &= \alpha\left(c'_z(\mathbf{z})\cdot\mathbf{P}'_x(\mathbf{x},\mathbf{p})\right) \\ &= \alpha\left(c'_z(\mathbf{z})\cdot\mathbf{A}\right) \end{aligned} \tag{3.4.6}$$

and wrt. $\mathbf{p}$:

$$\begin{aligned} s'_p(\mathbf{x},\mathbf{p}) &= \alpha\left(c'_z(\mathbf{z})\cdot\mathbf{P}'_p(\mathbf{x},\mathbf{p})\right) \\ &= \alpha\left(c'_z(\mathbf{z})\cdot\mathbf{H}\right) \end{aligned} \tag{3.4.7}$$

Here we have inserted the expressions from (3.4.4). The gradients are row vectors.

The input mapping depends on both $\mathbf{x}$ and $\mathbf{p}$ and for a given set of $\mathbf{x}$ and $\mathbf{p}$, we get a certain vector $\mathbf{z} = \mathbf{P}(\mathbf{x},\mathbf{p})$. An arbitrary change in one of the x-elements results in a new mapping vector $\mathbf{z}+\Delta$. But this new vector could also have been the result of a change in the $\mathbf{p}$-vector. The only difference is the scaling of the change, and the fact that changing one x-element effects all the elements of $\mathbf{z}$, while changing one p-element only has an effect on one element of $\mathbf{z}$.
In other words: we cannot dinstinguish between changes in the surrogate response as a result of altering the design parameters or altering the mapping parameters.

If we also consider the output mapping the $\mathbf{p}$-vector has an extra element, α. The gradient of the surrogate wrt. $\mathbf{p}$ should the be augmented with the element:

$$\begin{aligned} s'_\alpha(\mathbf{x},\mathbf{p}) &= \frac{\partial}{\partial\alpha}\left[\alpha\left(c\left(\mathbf{P}(\mathbf{x},\mathbf{p})\right) - c\left(\mathbf{P}(\bar{\mathbf{x}},\mathbf{p})\right)\right) + f(\mathbf{x})\right] \\ &= c(\mathbf{z}) - c(\bar{\mathbf{z}}) \end{aligned} \tag{3.4.8}$$

The length of $\mathbf{p}$ is now n^2+n+1.

The Parameter Extraction problem consists of m minimization problems and uses many surrogate model function (and consequently coarse model) evaluations. Even though we consider the coarse model computationally fast to evaluate, the Parameter Extraction is time-consuming for large problems. This is a reason for wishing to reduce the number of parameters.

In order to reduce the size of the parameter vector we can use different approaches. One approach is to reduce $\mathbf{A}$ to a diagonal matrix and keep the constant vector $\mathbf{b}$.
In the case $n=2$ we now have:

$$\mathbf{P}'_x(\mathbf{x},\mathbf{p}) = \mathbf{A} = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix}, \qquad \mathbf{P}'_p(\mathbf{x},\mathbf{p}) = \mathbf{H}(\mathbf{x}) = \begin{bmatrix} x_1 & 0 & 1 & 0 \\ 0 & x_2 & 0 & 1 \end{bmatrix} \tag{3.4.9}$$

where $n_p = 2n$. The new reduced matrix $\mathbf{H}$ is the result of eliminating the second and the third column of the old $\mathbf{H}$, since we have eliminated the second and third element of $\mathbf{p}$. It is straightforward to produce the matrices $\mathbf{A}$ and $\mathbf{H}$ corresponding to an arbitrary element elimination of the full parameter vector $\mathbf{p} \in \mathbb{R}^{(n(n+1))}$.

When using the diagonal matrix approach we reduce the number of input parameters by a factor $\frac{n+1}{2}$. Accordingly we can expect a reduction of the calculation time, which can be useful in problems of large dimension.

A different approach could be to alter the number of parameters for every main iteration, with the aim of the possibility of repeatedly having a unique solution to the Parameter Extraction. The length of the residual vector is increased by 1 for each main iteration, starting at n for $k = 0$, and we can augment the $\mathbf{p}$-vector with one extra element. The order of the elements can be chosen in many ways, a suggestion would be to begin with $\mathbf{p} = [\alpha]$ and augment with the diagonal elements of $\mathbf{A}$ one by one followed by the $\mathbf{b}$-elements. The initial value for the mapping parameters would still be $\mathbf{A}_i^{(0)} = \mathbf{I}$, $\mathbf{b}_i^{(0)} = \mathbf{0}$, $\alpha_i^{(0)} = 1$, and this initial value is valid, until the parameter is contained in $\mathbf{p}$.

3.5 The Penalty Factor

The solution to the Parameter Extraction problem is not always unique, and there are many different approaches to overcome this problem. A well known approach in optimization problems is to penalize the equations, that we with guarantee wish to satisfy by multiplying with a large factor. It these penalized residual elements are not very small (or zero if possible), then the objective function $1/2 \cdot \mathbf{r}^{\mathrm{T}}\mathbf{r}$ is large. Depending of course on the scaling of the residual elements we are sure, that the penalized equations are satisfied in the solution, because the solution minimizes the residual.
In order to determine the effect of the penalty factor σ introduced in equation (1.3.12) on the solution we again consider the cases, where the system of equations is underdetermined and overdetermined.
The formulation of the residual origins from the aim to align the surrogate model with the fine model, and it depends on the models of the given prob-

lem, if an exact match is possible in the cases, where we have either more parameters than residual elements or the same number of both. Usually it is possible to satisfy all equations exactly, but one can find examples of problems, where it is not. We assume that the problems considered here are consistent. In case of a full rank consistent problem the weighting of the residual elements has no effect on the solution. When the problem is rank deficient the solution space for the unweighted problem is the same as for the weighted problem, and the two problems have identical minimum norm solutions.
In the case of an non-consistent problem, the weighting factors can have an effect.

Underdetermined System of Equations

If there are more parameters n_p than residual elements n_r, the problem does not have a unique solution. The Jacobian of $\mathbf{r}$ is denoted $\mathbf{J}_r$ and has the rank q, where $q \leq n_r$. There will then be an $(n_p - q)$-infinity of solutions to the Parameter Extraction. For each of the infinitely many solutions we are then sure, that all n_r equations are satisfied, ie. $\mathbf{r}(\mathbf{p}^*) = \mathbf{0}$. Algorithm 3 provides the solution defined by the Marquardt parameters μ.
The multiplication with the weigthing factors now results in the residual $\mathbf{Wr}$ and the Jacobian $\mathbf{WJ}_r$, where $\mathbf{r}$ and $\mathbf{J}_r$ refer to equation (3.3.9), and where the diagonal matrix $\mathbf{W}$ is given by (3.5.1):

$$\mathbf{W} = \begin{bmatrix} w_1 & & & & & \mathbf{0} \\ & \ddots & & & & \\ & & w_k & & & \\ & & & \sigma & & \\ & & & & \ddots & \\ \mathbf{0} & & & & & \sigma \end{bmatrix} \tag{3.5.1}$$

In case of a consistent problem the solution to the penalized problem is the same as the unpenalized solution. In this case the penalty factors have no effect on the solution.

Overdetermined System of Equations

Provided that $\mathbf{J}_r$ has full rank in the case where $n_r = n_p$, the solution to the Parameter Extraction is unique. All the equations are satisfied exactly, and since $\mathbf{r}(\mathbf{p}^*) = \mathbf{0}$, then also $\mathbf{W} \cdot \mathbf{r}(\mathbf{p}^*) = \mathbf{0}$.
If there are more equations than unknown parameters $(n_r > n_p)$, we find the

least squares solution by Algorithm 3. We cannot be sure to satisfy all equations, and the weight of each equation determines, which solution is found, since this solution minimizes $1/2 \cdot \mathbf{r}^{\mathrm{T}}\mathbf{r}$. A large penalty factor will result in a solution that is guaranteed to satisfy the gradient residual, provided that the scaling of the residual elements is not bad. If the problem on the other hand is consistent, the penalty factor has no influence.

3.6 The Weighting Factors

Since the weighting factors function in the same way as the penalty factor discused in the previous section, the analysis regarding the systems of equations is also valid in the present section.

In the new Space Mapping formulation we wish the mapping parameters to minimize the residual function (1.3.12), consisting of the function value residual and the gradient residual. We assume, that we have made $k+1$ iterations in the main algorithm. The first iterate is $\mathbf{x}^{(0)}$. The residual depends on k of the $k+1$ iterates, as well as the gradient wrt. the design parameters in the best iterate, since the function values of the fine and surrogate models in the best iterate already match. The length of the residual vector is denoted n_r and depends on the number of main iterations and the number of design parameters ($n_r = k+n$). With use of the regularization term n_r also depends on the number of mapping parameters, in this case $n_r = k+n+n_p$.

We introduce weighting factors for the function value residual rows denoted w_i for $i = 1, \ldots, k$. The gradient residual is weighted with the penalty factor σ, which is discussed in section 3.5.

Linear Weight Function

In the implementation of the SMIS framework by Frank Pedersen, the weighting factors are given by a linear function depicted in figure 3.6.1.

where ε_{res} is a given threshold value (the figure shows $\varepsilon_{res} = 0.25$). The weighting function decreases linearly from 1 to 0 as the distance to the best iteration point grows, giving the expression for the weighting factors $w_1, \ldots, w_k$:

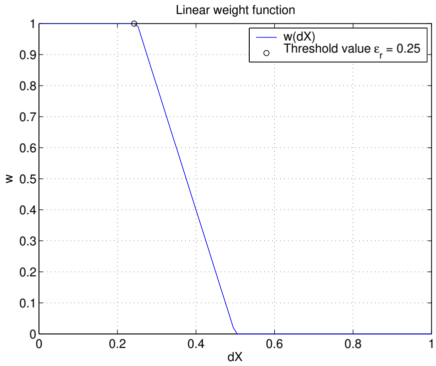

Figure 3.6.1: Linear weight function

$$w_i = \begin{cases} 1 & \text{for } dX_i < \varepsilon_{res} \\ 2 - \frac{dX_i}{\varepsilon_{res}} & \text{for } dX_i < 2\varepsilon_{res} \\ 0 & \text{for } dX_i \geq 2\varepsilon_{res} \end{cases}$$

Gauss Distributed Weight Function

Another approach is to let the weighting factors depend on the number of residual rows n_r. The number of unknown parameters is n_p, hence we need at least $n_p - n + 1$ iterations to ensure, that the problem is not underdetermined.
We choose a weight function as a Gauss curve.

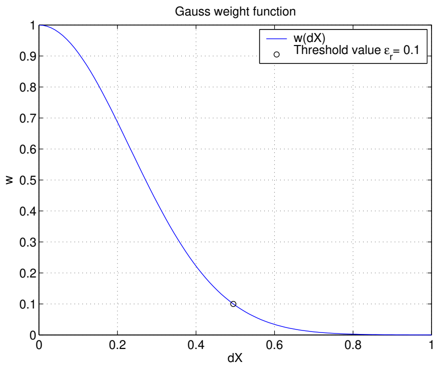

Figure 3.6.2: Gauss distributed weight function

Here the weighting factors are calculated from the expression:

$$w_i = \begin{cases} 1 & \text{if } n_r < n_p \\ \exp(-\gamma \cdot dX_i^2) & \text{if } n_r \geq n_p \end{cases} \tag{3.6.1}$$

where the factor γ is determined by $\gamma = -\frac{\ln(\varepsilon_{res})}{d\bar{X}^2}$, with $d\bar{X}$ corresponding to the $(n_p - n)$th best iterate. This gives the weighting factor ε_{res} for this particular residual element. In this ways the weighting factors for the residual elements corresponding to the best iterates are kept above or equal to the threshold value ε_{res}, and all other points are considered low priority. The weighting function for $\varepsilon_r = 0.10$ is seen in figure 3.6.2. If there are not enough rows in the residual to match the number of unknown parameters, all residual rows are weighted equally with a factor 1.

If the Parameter Extraction problem is overdetermined, the weighting factors make sure that the best points are given the highest priority. As mentioned before the weighting factors do not influence the solution, if the system is consistent. The threshold value is used to decide, how important a role the 'unnecessary' iterates should play in finding the optimal parameter set.
If the Parameter Extraction problem is underdetermined, it is usually possible to satisfy all equations exactly. In this case the weighting factors have no effect on the solution, since if $\mathbf{r}(\mathbf{p}^*) = \mathbf{0}$ then also $\mathbf{W} \cdot \mathbf{r}(\mathbf{p}^*) = \mathbf{0}$. The strategy for determining the weighting factors in the case $n_r < n_p$ is therefore not important, as long as the w's are not equal to zero.

3.7 The Normalization Factors

The normalization factors $a_1, \ldots, a_k$ and d are multiplied on each of the residual elements as given by (1.3.12). This is done to avoid scaling problems in the residual function.
The normalization factors theoretically have the same effect as the weighting factors w and the penalty factor σ. But the normalization factors are not used to constrain the problem by penalizing some chosen elements, but simply to ensure that all the equations are weighted equally in the residual vector. If the residual elements are of very different orders of magnitude, the weighting factors will not have the effect that we aim for.

A strategy for choosing the normalization factors is to put:

$$a_j = \frac{1}{\sqrt{\varepsilon_M} + \|f_i(\mathbf{x}_j)\|_2} \qquad \text{for } j = 1, \dots, k \tag{3.7.1}$$

$$d = \frac{1}{\sqrt{\varepsilon_M} + \|\mathbf{J}_{f,i}(\mathbf{x}_0)\|_2} \tag{3.7.2}$$

$$v = \frac{1}{\sqrt{\varepsilon_M} + \|\mathbf{p}_k\|_2} \tag{3.7.3}$$

where the iterates $\mathbf{x}_0, \dots, \mathbf{x}_k$ are the sorted iterates corresponding to section 3.6. The factor v in (3.7.3) is used for scaling the regularization term, if this is included in the residual formulation.

Each of the first k residual elements is the difference between the surrogate and the fine model response in k different iteration points. We scale these residual elements according to the norm of the fine model response in the iterate.

The gradient residual is scaled with the normalization factor d, which is found by means of the norm of the fine model gradient. To avoid infinite scaling factors, when $\|f_i(\mathbf{x}_i)\|_2$ or $\|\mathbf{J}_{f,i}(\mathbf{x}_0)\|_2$ are very small, we add $\sqrt{\varepsilon_M}$ to the denominator of all normalization factors.

Finally the regularization term, when present, is scaled according to the initial parameter vector $\mathbf{p}_k$, which holds the mapping parameters corresponding to the kth surrogate model.

Chapter 4

Test Problems

4.1 Introduction

The Space Mapping method performed by the three algorithms in section 2.1 has been tested on various test problems. The two test problems TLT2 and TLT7 are from the SMIS framework by Frank Pedersen. Another test problem is the Rosenbrock function, which has been tested in its classical form as well as in an augmented version.
The Space Mapping method can produce very different results, depending on the implementation and the formulation of the residual. In all test runs the implementation of the three algorithms correspond to the desriptions in chapter 2.
The tests have been performed on the SUN Fire 3800 server on the IMM system with the following data: 8 CPU, 16 GB RAM and the clock frequency 1200 MHz.

A very important factor for the general performance of the Space Mapping method is connected with the Parameter Extraction problems. As described in the previous chapter the formulation of the residual can be varied by the use of normalization factors, weighting factors and regularization. These three approaches can be combined with the reduction of the parameter vector. There are many options to choose from and every one of these options have an influence on the results.
The test investigations presented here are concerned only with the formulation of the Parameter Extraction problems. Through the work with the implementation and the following test runs of the algorithms, the effects of a certain approach has been somewhat clarified. On the basis of this the options have been chosen to provide different scenarios, which define the algorithms used. All scenarios have been used on the three problem types. Before presenting the test results we define the profiles of the scenarios.

In every test run we must define four tolerance options for use in the stopping critera for the three algorithms. These options are:

ε_F : Stopping criterion for the objective function.
ε_{hx} : Stopping criterion for the step length for $\mathbf{x}$-iterates.
ε_{hp} : Stopping criterion for the step length for $\mathbf{p}$-iterates.
ε_K : Stopping criterion for the gradient residual match.

In the setup file it is currently only possible to set two tolerance parameters ε_1 and ε_2, where the first is the desired accuracy for the main problem, and the latter is the desired accuracy, when solving the Parameter Extraction problem. We use $\varepsilon_1 = \varepsilon_F = \varepsilon_{hx}$ and $\varepsilon_2 = \varepsilon_{hp} = \varepsilon_K$.
Furthermore the maximal number of function evaluations in each of the algorithms is needed. These values are set according to the particular problem. The options in `marquardt` must also be defined cf. p. 19.
The tolerance options and `marquardt`-options in some cases have an effect on the results. On that ground we use either one of the following sets of options, depending on the problem:

- $\varepsilon_1 = 10^{-14}$, $\varepsilon_2 = 10^{-4}$ and `opts= [1e-8 1e-4 1e-4 200 1e-12]`.
- $\varepsilon_1 = 10^{-14}$, $\varepsilon_2 = 10^{-14}$ and `opts= [1e-8 1e-14 1e-14 200 1e-12]`.

The initial trust region radius for all test problems is $\Delta^{(0)} = 10^{-1} \cdot \|\mathbf{x}^{(0)}\|_2$. The parameter η, that defines the step length in `diffjacobian`, is fixed at $\eta = 10^{-5}$ for both forward difference approximations wrt. x and wrt. p.
The updating of the penalty factor is done only by the strategy in Algorithm 3. Finally we only consider minimax optimization corresponding to $p = \infty$ in Algorithm 1 and 2, whereas Algorithm 3 corresponds to minimization in the 2-norm.

4.1.1 The Test Scenarios

For showing the effects of a given approach regarding the residual definition, we compare the test runs from different residual profiles. We here present an overview of the investigated test scenarios.

Regularization

To show the effects of including the regularization term in the residual vector we compare the test runs with the profiles:

- Regularization of the residual vector as described in section 3.3. The regularization term is added to the residual vector (1.3.12) which now has $n_r = k + n + n_p$ elements.

- No regularization of the residual vector. The residual has the formulation (1.3.12) and consists of $n_r = k + n$ elements.

Both cases are including complete normalization of the residual elements corresponding to $a_j = \frac{1}{\sqrt{\varepsilon_M}+\|f_i(\mathbf{x}_j)\|_2}$ for $j = 1, \ldots, k$ and $d = \frac{1}{\sqrt{\varepsilon_M}+\|\mathbf{J}_{f,i}(\mathbf{x}_0)\|_2}$. When the regularization term is present, it is multiplied with the normalization factor $v = \frac{1}{\sqrt{\varepsilon_M}+\|\mathbf{p}_i^{(0)}\|_2}$.

The Normalization Factors

For investigating the influence of the normalization factors we test the following cases:

- Normalization of all residual elements: The normalization factors are given by $a_j = \frac{1}{\sqrt{\varepsilon_M}+\|f_i(\mathbf{x}_j)\|_2}$ for $j = 1, \ldots, k$ and $d = \frac{1}{\sqrt{\varepsilon_M}+\|\mathbf{J}_{f,i}(\mathbf{x}_0)\|_2}$.
- Partly normalization of the residual elements: Only the gradient residual is scaled corresponding to $a_1, \ldots, a_k = 1$ and $d = \frac{1}{\sqrt{\varepsilon_M}+\|\mathbf{J}_{f,i}(\mathbf{x}_0)\|_2}$.
- No normalization of the residual elements: $a_i = 1$ for $i = 1, \ldots, k$ and $d = 1$.

The Weighting Factors

The effect of the weighting factors are considered and we compare the scenarios:

- No weighting of the function value residual, $w_i = 1$ for $i = 1, \ldots, k$.
- Weighting of the residual elements corresponding to the Gauss distributed weight function, and the strategy in section 3.6 with the weights $w_1, \ldots, w_k$ given by (3.6.1).

The two cases are combined with a complete normalization of the residual elements.

The Number of Mapping Parameters

The last test scenario concerns the number of mapping parameters. We consider two cases:

- With a full-size parameter vector: The number of mapping parameters is $n_p = n^2 + n + 1$.
- With a reduced parameter vector corresponding to using a diagonal input mapping matrix $\mathbf{A}$: The number of mapping parameters is $n_p = 2n + 1$.

In the problem setup-file the size of $\mathbf{p}$ is controlled by the flag `diagA`. For `diagA` $= 0$ the input mapping matrix $\mathbf{A}$ is full, and for `diagA` $= 1$ we have a diagonal $\mathbf{A}$.

4.1.2 Visualization of The Results

The results of the MATLAB runs of the test problems will be presented in different ways. The SMIS framework contains different plotting programs, which can be called after the ended Space Mapping iterations for a test problem. The following plots are used to show the results:

Convergence of the Iteration Sequence

The test problems all have known optimizers of the fine and the coarse model, which makes it possible to see if the Space Mapping method converges or not. The optimizers $\mathbf{x}^*$ and $\mathbf{z}^*$ must be provided in the problem setup-file. Two measures for the convergence to the optimizer are plotted with different plot symbols as a function of the iteration number:

$\square$ $\|\mathbf{x}^{(k)} - \mathbf{x}^*\|_2$
$\Diamond$ $F(\mathbf{x}^{(k)}) - F(\mathbf{x}^*)$

These plots are comparable with the results of [1]. The formulations are still valid, when $\mathbf{x}^*$ and $F(\mathbf{x}^*)$ are equal to zero.

Approximation Error

The surrogate model provides an alternative to the fine model, which is cheaper to evaluate than the fine model. The surrogate model is used as a local approximation to the fine model, and the region, where the model approximation is good, is of interest. With the new Space Mapping formulation with both input and output mappings, the surrogate and the fine model match exactly in the expansion point. To evaluate the results we are interested in the model matching in the region around the expansion point. The model matching of the surrogate model can be compared with the matching of a Taylor model, which is used for approximation in classical optimization methods.

In classical optimization with first order information available, we use a Taylor model to approximate the objective function. The Taylor expansion is:

$$\mathbf{f}(\mathbf{x}+\mathbf{h}) = \mathbf{f}(\mathbf{x}) + \mathbf{J}_f(\mathbf{x})\mathbf{h} + O(\|\mathbf{h}\|_2^2)$$

and when only using the first order information, we get the linearized model:

$$\mathbf{l}(\mathbf{h}) = \mathbf{f}(\mathbf{x}) + \mathbf{J}_f(\mathbf{x})\mathbf{h}$$

The approximation error is then given by $\mathbf{f}(\mathbf{x}+\mathbf{h}) - \mathbf{l}(\mathbf{h})$. A measure of the approximation error suitable for plotting is:

$$E_l(\mathbf{h}) = \|\mathbf{f}(\mathbf{x}+\mathbf{h}) - \mathbf{l}(\mathbf{h})\|_2$$

and it is obvious, that the error grows quadratically as we move from the expansion point.

In the Space Mapping Method we use the surrogate model as an approximation to the objective function. The norm of the difference between the surrogate and the fine model can be used as a measure of the approximation error given by:

$$E_s(\mathbf{h}) = \|\mathbf{f}(\mathbf{x}+\mathbf{h}) - \mathbf{s}(\mathbf{x}+\mathbf{h})\|_2$$

Because of the new Space Mapping formulation with the output mapping we are ensured an exact function value match in the expansion point. Furthermore the residual function used in the Parameter Extraction guarantees a satisfactory gradient match in this point.

For two-dimensional problems ($n = 2$) the two approximation errors E_l and E_s can be visualized in three-dimensional plots, with the errors plotted in a region of the (x_1, x_2)-plane.

Direct Optimization

The SMIS framework also contains two algorithms, which can be used for direct optimization based on first order derivatives of the fine model. The first algorithm `direct` is an implementation of a first order method which uses approximations to the first order derivatives from a Broyden update. The second algorithm `directd` exploits the gradients returned directly from the model functions.

The iteration sequence from the direct optimization can be used to compare the Space Mapping method with the interpolating surrogate with a classical optimization method.

4.2 The Rosenbrock Problem

4.2.1 Introduction

This test problem is based on the well-known non-linear function, the Rosenbrock function, a vector function $\mathbb{R}^2 \to \mathbb{R}^2$ ($n = m = 2$) with the optimizer $\mathbf{x}^* = [1\ ,\ 1]^{\mathrm{T}}$. In this test problem the coarse model is identical with the Rosenbrock function:

$$\mathbf{c}(\mathbf{z}) = \begin{bmatrix} 10(z_2 - z_1^2) \\ 1 - z_1 \end{bmatrix}$$

with the optimizer $\mathbf{z}^* = [1, 1]^{\mathrm{T}}$. The fine model is a transformed version of the coarse model defined by a linear transformation of the design parameters:

$$\mathbf{f}(\mathbf{u}) = \begin{bmatrix} 10(u_2 - u_1^2) \\ 1 - u_1 \end{bmatrix} \qquad \mathbf{u}(\mathbf{x}) = \mathbf{C}\mathbf{x} + \mathbf{d}$$

where $\mathbf{C} \in \mathbb{R}^{2\times 2}$ and $\mathbf{d} \in \mathbb{R}^2$. The fine model optimizer $\mathbf{x}^*$ satisfies $\mathbf{u}(\mathbf{x}^*) = \mathbf{z}^*$ giving:

$$\mathbf{z}^* = \mathbf{C}\mathbf{x}^* + \mathbf{d} \qquad \Rightarrow \qquad \mathbf{x}^* = \mathbf{C}^{-1}(\mathbf{z}^* - \mathbf{d})$$

and the solution $\mathbf{x}^*$ is unique, provided that $\mathbf{C}$ is nonsingular.
Since we know the transformation given by $\mathbf{C}$ and $\mathbf{d}$, the input mapping parameters $\mathbf{A}_i = \mathbf{C}$ and $\mathbf{b}_i = \mathbf{d}$ for $i = 1, 2$ will give an exact match between $\mathbf{P}_i(\mathbf{x})$ and $\mathbf{u}(\mathbf{x})$. With the output mapping parameters $\alpha_1, \alpha_2 = 1$ the surrogate model is exactly equal to the fine model, since for $i = 1, 2$:

$$\begin{aligned} s_i(\mathbf{x}) &= \alpha_i \left(c_i\left(\mathbf{P}_i(\mathbf{x})\right) - c_i\left(\mathbf{P}_i(\bar{\mathbf{x}})\right)\right) + f_i(\mathbf{u}(\bar{\mathbf{x}})) \\ &= 1 \cdot c_i\left(\mathbf{P}_i(\mathbf{x})\right) \\ &= f_i(\mathbf{u}(\mathbf{x})) \end{aligned}$$

But as shown in section 3.4, there are infinitely many solutions to the input mapping parameters $\mathbf{A}_i$ and $\mathbf{b}_i$ giving $\mathbf{P}_i(\mathbf{x}^*) = \mathbf{z}^*$, and because of the approach in the Parameter Extraction problem we cannot be sure to find the particular solution, where $\mathbf{A}_i = \mathbf{C}$ and $\mathbf{b}_i = \mathbf{d}$ for $i = 1, 2$. In fact the mapping parameters for the two responses are probably not equal.
As we shall see later the Rosenbrock function is special, since the second response function only depends on x_1.

4.2.2 Linear Transformation

In this example we use the following transformation of the fine model:

$$\mathbf{u}(\mathbf{x}) = \mathbf{C}\mathbf{x} + \mathbf{d} = \begin{bmatrix} 1.1 & -0.2 \\ 0.2 & 0.9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -0.3 \\ 0.3 \end{bmatrix} \tag{4.2.1}$$

The corresponding fine model optimizer rounded to 4 decimals is $\mathbf{x}^* = [1.2718\ ,\ 0.4951]^{\mathrm{T}}$.

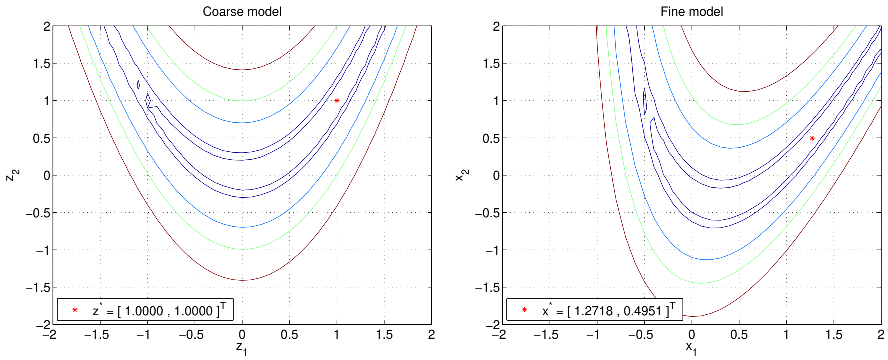

Figure 4.2.1: Contour plots of the Rosenbrock function: coarse model (left) and fine model (right)

We show the level curves of the coarse and fine models in figure 4.2.1, where the objective function is $F = \|\cdot\|_2^2$. The fine model is very similar to the coarse model (the original Rosenbrock function) and the characteristic banana shape is still present.

In all the test results with the Rosenbrock function we use the options $\varepsilon_1 = 10^{-14}$, $\varepsilon_2 = 10^{-14}$ and `opts` = `[1e-8 1e-14 1e-14 200 1e-12]`. The initial guess for the coarse model optimizer is $\mathbf{x}^{(0)} = [-1.2\ ,\ 1.0]^{\mathrm{T}}$. We show the iteration sequences also after the algorithm has converged to the optimizer.

Effect of the Regularization

The convergence is very fast for both the case with regularization and the case without. The performances are shown in figures 4.2.2 and 4.2.3.

We notice the absence of the points of iteriation 6 in figure 4.2.2, because the value 0 is not visible in the semilogarithmic plot.
There are small differences between the iteration sequences of the two test

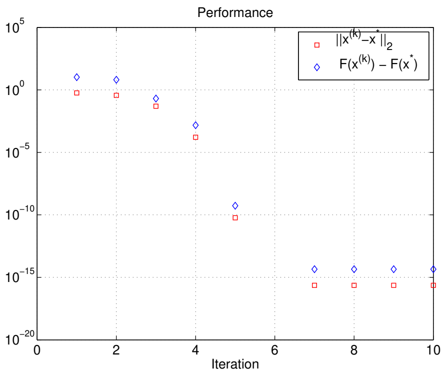

Figure 4.2.2: With regularization

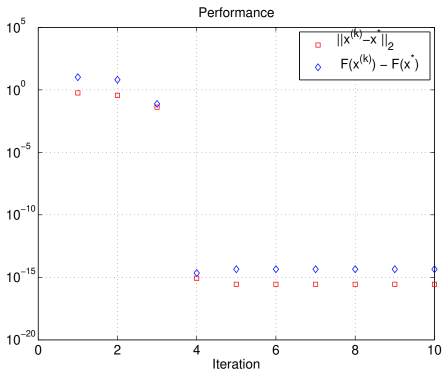

Figure 4.2.3: Without regularization

runs: The convergence is a little faster without the regularization term added to the residual vector. When we solve the unregularized problem, we have $n_p = 7$ and there is a possibility of an overdetermined Parameter Extraction problem in iteration 6 and onwards. For this problem the underdetermined Parameter Extraction problems do not have a negative effect on the convergence rate, since we find the fine model optimizer before iteration 6.

The iteration points corresponding to figure 4.2.3 are shown in the (x_1, x_2)-plane with the objective function $F = \|\mathbf{f}(\mathbf{x})\|_2^2$. It is noted that the first iteration point plotted is the initial guess for the coarse model optimizer. The first evaluation of the fine model is made in the point $\mathbf{x}^{(1)}$ which is the second point plotted in figure 4.2.3.

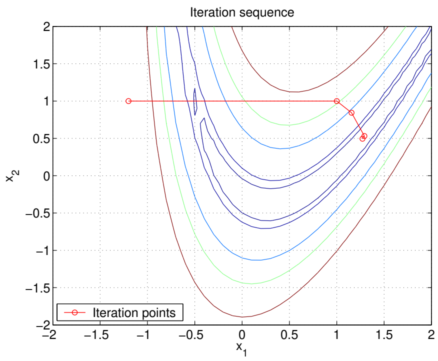

Figure 4.2.4: Sequence of iteration points

With the Space Mapping algorithm we avoid the iteration sequence moving through the narrow valley with small values of the objective function.

Effect of the Normalization Factors

The next test runs are made with all normalization factors equal to 1. The iteration sequences in figures 4.2.5-4.2.6 are almost identical with figures 4.2.2-4.2.3, and we conclude, that the normalization factors have practically no effect in the Rosenbrock problem.

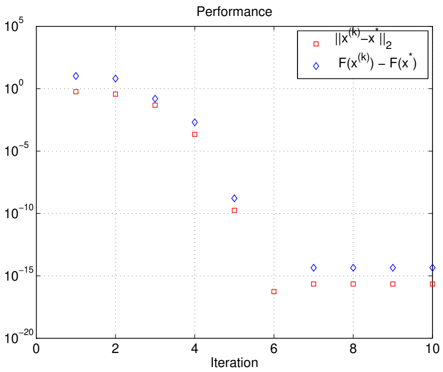

Figure 4.2.5: With regularization and without normalization

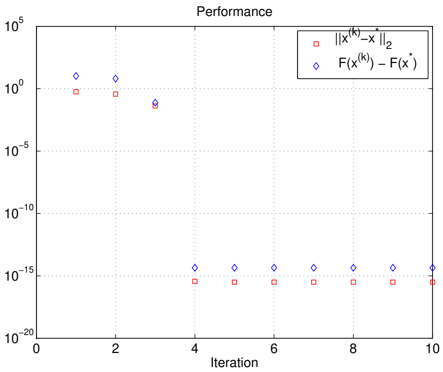

Figure 4.2.6: Without regularization and without normalization

Effect of the Weighting Factors

We can not use this problem for testing the effect of the weighting factors. The weighting factors from the Gauss distributed weight function approach are only different from zero from iteration number 6. At this point the solution is already found.

Effect of the Number of Mapping Parameters

In this case the results are very different, when we use the reduced parameter vector instead of the complete. We test the performance of the algorithm in the following three cases:

- With regularization
- Without regularization
- Without regularization and with weighting factors

All three cases are with normalization of the residual elements. The results are shown in figures 4.2.7, 4.2.8 and 4.2.9.

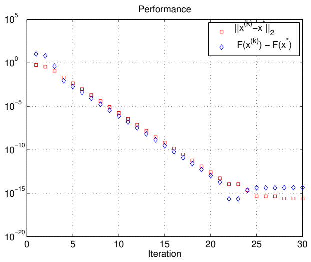

Figure 4.2.7: With regularization

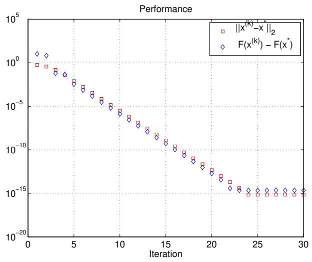

Figure 4.2.8: Without regularization

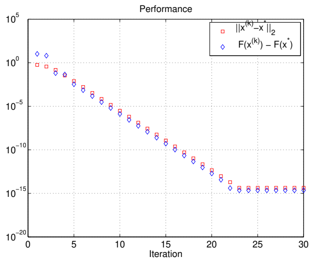

Figure 4.2.9: Without regularization and with weighting

The convergence is much slower in all three cases compared to figures 4.2.2 and 4.2.3. There is only a little difference in the iteration sequences in figures 4.2.7, 4.2.8 and 4.2.9 in the last iterations, when we are close to the optimizer. The convergence rate is the same for all three cases.
The table below shows the values of $\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\|_2 / \|\mathbf{x}^{(k)} - \mathbf{x}^*\|_2$ for $k = 1, \dots, 24$ corresponding to the results of figure 4.2.7.

k	$\frac{\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\|_2}{\|\mathbf{x}^{(k)}-\mathbf{x}^*\|_2}$	k	$\frac{\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\|_2}{\|\mathbf{x}^{(k)}-\mathbf{x}^*\|_2}$	k	$\frac{\|\mathbf{x}^{(k+1)}-\mathbf{x}^*\|_2}{\|\mathbf{x}^{(k)}-\mathbf{x}^*\|_2}$
1	6.4062e-01	9	2.0771e-01	17	2.0771e-01
2	3.4325e-01	10	2.0770e-01	18	2.0768e-01
3	1.6551e-01	11	2.0770e-01	19	2.0758e-01
4	2.1301e-01	12	2.0769e-01	20	2.0725e-01
5	2.0869e-01	13	2.0769e-01	21	2.0764e-01
6	2.0793e-01	14	2.0769e-01	22	1.0000e+00
7	2.0778e-01	15	2.0769e-01	23	2.0577e-01
8	2.0774e-01	16	2.0770e-01	24	1.9426e-01

We note that the asymptotic error constant is approximately 0.2 from iteration 4 to 21. The results indicate linear convergence.

In the case of the reduced parameter vector we have $n_p = 5$ unknown parameters in every Parameter Extraction problem. The transformation of the fine model parameters is defined by a non-diagonal matrix $\mathbf{C}$, and apparently this creates problems, when aligning the surrogate model with the fine model.

In figure 4.2.10 the iteration points from figure 4.2.8 are seen in the contour plot of $F = \|\mathbf{f}(\mathbf{x})\|_2^2$. It is similar to figure 4.2.4, except for the fact that a lot of points are clustered near the optimizer $\mathbf{x}^* = [1.2718\ ,\ 0.4951]^{\mathrm{T}}$.

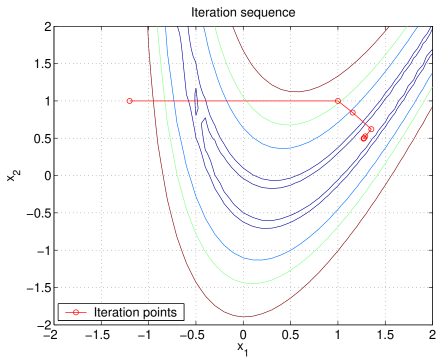

Figure 4.2.10: Sequence of iteration points

Optimal Mapping Parameters

The Rosenbrock function is special in the sense, that the two response functions are qualitatively different. The first response is a quadratic function and depends on both z_1 and z_2, whereas the second is linear and only depends on z_1. The Jacobian matrix is:

$$\begin{bmatrix} -20z_1 & 10z_2 \\ -1 & 0 \end{bmatrix}$$

Since $\partial c_2/\partial z_2 = 0$ and with reference to section 3.4 equation (3.4.7) this means that:

$$\begin{aligned} \frac{\partial s_2(\mathbf{x},\mathbf{p})}{\partial p} &= \alpha_2 \left(c'_{2,z}(\mathbf{z}) \cdot \mathbf{H} \right) \\ &= \alpha_2 \begin{bmatrix} -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1 & x_2 & 0 & 1 \end{bmatrix} \\ &= -\alpha_2 \begin{bmatrix} x_1 & x_2 & 0 & 0 & 1 & 0 \end{bmatrix} \end{aligned}$$

We have no information of the mapping parameters A_{21}, A_{22} and b_2 concerning z_2, since all the partial derivatives wrt. these parameters are zero. This influences the Parameter Extraction for response function number 2. The variables are never changed during the residual optimization, and the final values are $A_{21} = 0$, $A_{22} = 1$ and $b_2 = 0$ corresponding to the initial values.

This theory is confirmed when looking at the results from the Space Mapping algorithm. The optimal parameter sets are for the regularized case and a full parameter vector (figure 4.2.2):

$$\mathbf{A}_1 = \begin{bmatrix} 0.9567 & -0.1489 \\ 0.0370 & 0.9930 \end{bmatrix} \qquad \mathbf{b}_1 = \begin{bmatrix} -0.0678 \\ 0 \end{bmatrix} \qquad \alpha_1 = 0.9899$$

$$\mathbf{A}_2 = \begin{bmatrix} 1.0329 & -0.1878 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{b}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \alpha_2 = 1.0649$$

For the case when $\mathbf{A}$ is reduced to a diagonal matrix we get the optimal mapping parameters with use of the regularization term (figure 4.2.7):

$$\mathbf{A}_1 = \begin{bmatrix} 0.7811 & 0 \\ 0 & 1.1721 \end{bmatrix} \qquad \mathbf{b}_1 = \begin{bmatrix} 0.1609 \\ 0 \end{bmatrix} \qquad \alpha_1 = 1.1091$$

$$\mathbf{A}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{b}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \alpha_2 = 1$$

The results with no regularization (figure 4.2.8) are not qualitatively different:

$$\mathbf{A}_1 = \begin{bmatrix} 0.7923 & 0 \\ 0 & 1.3003 \end{bmatrix} \qquad \mathbf{b}_1 = \begin{bmatrix} 0.2548 \\ 0 \end{bmatrix} \qquad \alpha_1 = 0.9997$$

$$\mathbf{A}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{b}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \alpha_2 = 1$$

It is noted, that the mapping parameters for response function 2 are identical with the initial mapping parameters in both cases. This probably has something to do with the fact, that the second response function is linear and only depends on the first variable.

Direct Optimization

We finally present the results from direct optimization of the fine model by the two algorithms `direct` and `directd` from the SMIS framework implemented by Frank Pedersen.

Both iteration sequences in figure 4.2.11 converge very slowly, which is also seen from the plots in figure 4.2.12.

From the given initial guess $\mathbf{x}^{(0)} = [-1.2\ ,\ 1]$ the iterates move through the valley with a large number of small steps towards the optimizer. This behaviour of the iteration sequence is avoided for the Space Mapping algorithm.

It is obvious, that the Space Mapping algorithm is much more efficient than the classical optimization algorithms considered here.

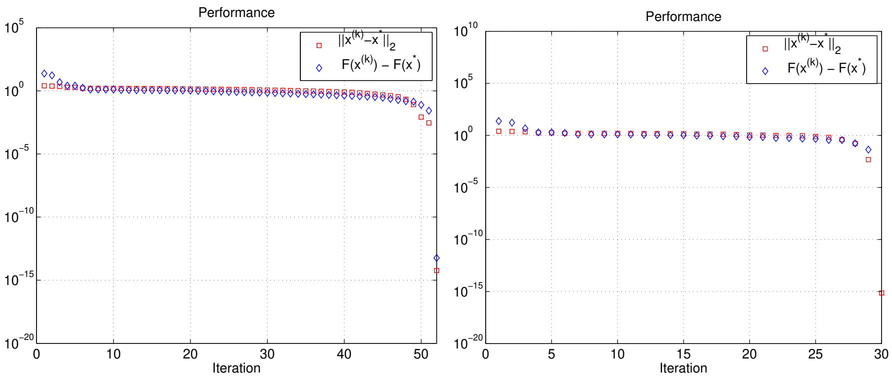

Figure 4.2.11: Performance of direct optimization of the fine model ('direct' left, 'directd' right)

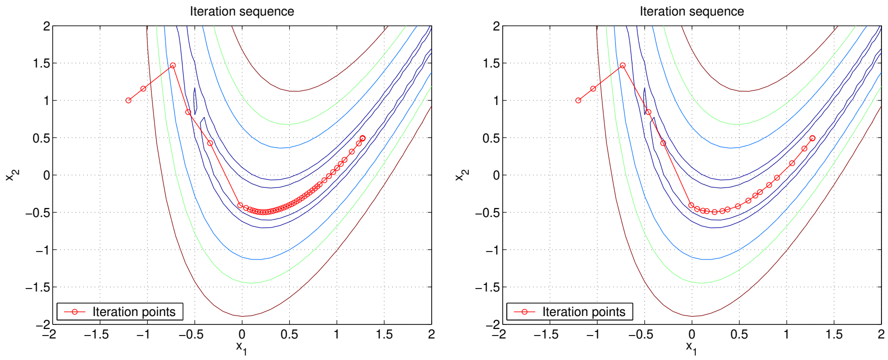

Figure 4.2.12: Iteration sequences for direct optimization of the fine model ('direct' left, 'directd' right)

Summary of the Results

- The regularization seems to have slightly negative effect on the convergence speed.
- The normalization factors have practically no effect on the convergence speed.
- The effect of the weighting factors is not possible to investigate for this problem, because the optimizer is found, before the chosen weighting strategy has any influence.
- The reduction of the mapping parameters results in a much slower convergence rate.
- The optimal mapping parameters are practically not influenced by the regularization term.
- Several of the input mapping parameters are not changed from the initial values. This behaviour is caused by the character of the problem.

4.2.3 The Augmented Rosenbrock Function

To test the Space Mapping algorithm on a problem with a known solution but of larger dimension than the classical Rosenbrock function we add two variables and three equations to give the augmented Rosenbrock function $\mathbb{R}^4 \to \mathbb{R}^5$ with $n = 4$ and $m = 5$:

$$\mathbf{c}(\mathbf{z}) = \begin{bmatrix} 10(z_2 - z_1^2) \\ 1 - z_1 \\ 10(z_3 - z_4^2) \\ 1 - z_3 \\ z_1^2 + z_2^2 + z_3^2 + z_4^2 - 4 \end{bmatrix}$$

The coarse model optimizer is $\mathbf{z}^* = [1\ ,\ 1\ ,\ 1\ ,\ 1]^{\mathrm{T}}$, where we have restricted the solution space to $z_i \geq 0$ for $i = 1, 2, 3, 4$, since there is also an optimizer in $\mathbf{z}^* = [1\ ,\ 1\ ,\ 1\ ,\ -1]^{\mathrm{T}}$.

The fine model is again similar to the coarse model except for the parameter transformation $\mathbf{u}(\mathbf{x})$:

$$\mathbf{f}(\mathbf{u}) = \begin{bmatrix} 10(u_2 - u_1^2) \\ 1 - u_1 \\ 10(u_3 - u_4^2) \\ 1 - u_3 \\ u_1^2 + u_2^2 + u_3^2 + u_4^2 - 4 \end{bmatrix}$$

with the linear transformation of the $\mathbf{x}$-vector given by the matrix $\mathbf{C} \in \mathbb{R}^{4\times 4}$ and the vector $\mathbf{d} \in \mathbb{R}^4$. We choose the transformation corresponding to:

$$\mathbf{u}(\mathbf{x}) = \mathbf{C}\mathbf{x} + \mathbf{d} = \begin{bmatrix} 1.1 & -0.2 & 1.1 & 0.2 \\ 0.2 & 0.9 & -0.2 & 0.9 \\ 1.1 & 0.2 & 1.1 & -0.2 \\ -0.2 & 0.9 & 0.2 & 0.9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} -0.3 \\ 0.3 \\ -0.3 \\ 0.3 \end{bmatrix}$$

The fine model optimizer is now:

$$\begin{bmatrix} 0.5909 & 0.3889 & 0.5909 & 0.3889 \end{bmatrix}^{\mathrm{T}}$$

We begin the iteration proces with the initial guess for the coarse model optimizer $\mathbf{x}^{(0)} = [-1.2\ ,\ 1.0\ ,\ -1.2\ ,\ 1.0]^{\mathrm{T}}$. All test runs are made with $\varepsilon_1 = 10^{-14}$, $\varepsilon_2 = 10^{-14}$ and `opts = [1e-8 1e-14 1e-14 200 1e-12]`.

Effect of the Regularization

In figure 4.2.13 we show the iteration sequence as a result of running the Space Mapping algorithm with regularization of the Parameter Extraction problem. Figure 4.2.14 shows the results without regularization. Both test runs are with full input mapping matrices **A**.

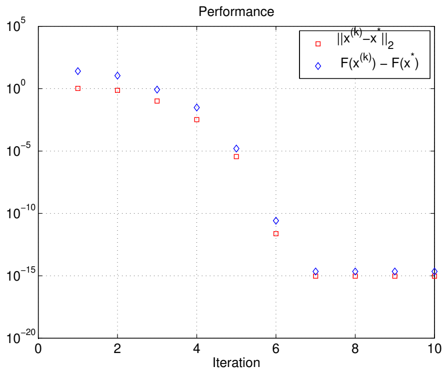

Figure 4.2.13: With regularization

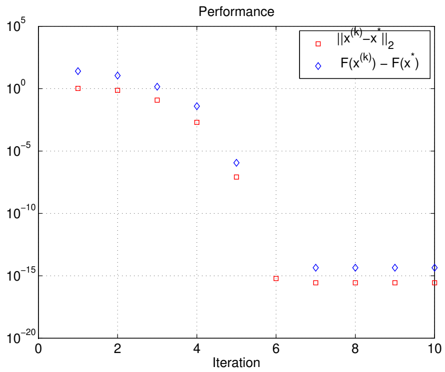

Figure 4.2.14: Without regularization

Again we get a slightly faster convergence for the case with no regularization, in which the optimizer is found in 6 iterations. We have $n_p = 21$ unknown parameters. The scenario with no regularization term is thereby underdetermined at least until the 18th iteration, but this has no effect since the optimizer is found in the 6th iteration.

Effect of the Normalization Factors

The normalization factors do practically not effect the results, when we solve the regularized Parameter Extraction problems, which is shown in figure 4.2.15. In the case without regularization the normalization factors have a slight positive effect, as it is seen in figure 4.2.16 compared to figure 4.2.14. The convergence within the accuracy of 10^{-15} is now made in 7 iterations.

Effect of the Weighting Factors

The influence of the weighting factors is not possible to investigate for this problem, since the weighting function strategy will not have influence before iteration 18, where the algorithm has converged to the optimizer.

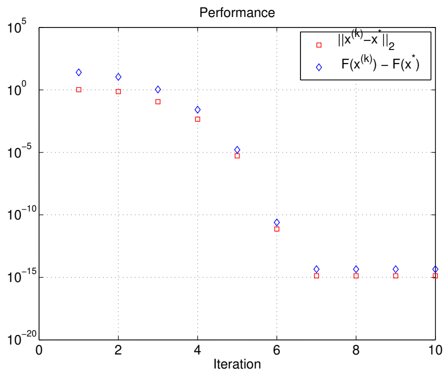

Figure 4.2.15: With regularization and without normalization

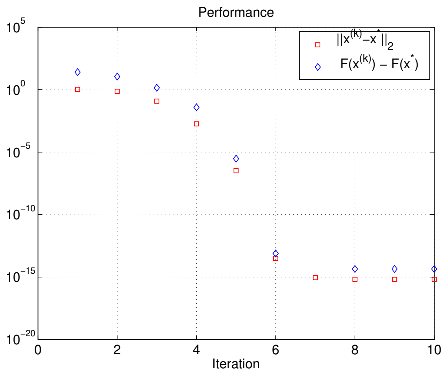

Figure 4.2.16: Without regularization and without normalization

Effect of the Number of Mapping Parameters

The algorithm does not converge when only a diagonal matrix is used in the input mapping. This applies for all test scenarios.
The number of mapping parameters is $n_p = 9$ in this case. We must conclude, that it is not possible to find a mapping only consisting of these 9 parameters, that aligns the surrogate model with the fine model when the transformation $\mathbf{u}(\mathbf{x})$ is defined by a non-diagonal matrix $\mathbf{C}$. In the next subsection we look at the solution from a full parameter vector. It turns out, that several of the input mapping matrices are actually similar to diagonal matrices except for one or two rows.
The definition of the transformation $\mathbf{u}(\mathbf{x})$ makes the reduction of the parameter vector unsuitable for this problem.

Optimal Mapping Parameters

We now consider the optimal mapping parameters. These can be compared to the results of the original Rosenbrock function, since the first two response functions are identical. Also the added third and fourth response functions have similarities to the two first. The optimal mapping parameters for this augmented Rosenbrock function therefore has the same features as described in the previous section 4.2.2.

We bring the sets of optimal mapping parameters below for the case corresponding to figure 4.2.14.

i	$\mathbf{A}_i$				$\mathbf{b}_i$	α_i
1	1.1768	0.0552	1.1768	-0.0552	-0.4595	0.8738
	-0.0973	1.5905	-0.5551	0.4694	0	
	0	0	1	0	0	
	0	0	0	1	0	
2	1.0672	-0.1940	1.0672	0.1940	0	1.0307
	0	1	0	0	0	
	0	0	1	0	0	
	0	0	0	1	0	
3	1	0	0	0	0	0.7448
	0	1	0	0	0	
	1.4855	0.2296	1.4682	-0.3074	0	
	-0.2317	1.0428	0.2317	1.0428	0.3290	
4	1	0	0	0	0	1.0307
	0	1	0	0	0	
	1.0672	0.1940	1.0672	-0.1940	0	
	0	0	0	1	0	
5	1.0832	0.2075	0.2989	0.2550	0.0831	1.1934
	0.2044	1.1034	0.1611	0.3426	-0.0242	
	0.2974	0.1610	1.1579	0.2364	0.0262	
	0.2537	0.3435	0.2343	0.8789	0.0652	

We note the following:

- Response functions 2 and 4 are only dependent on one design parameter each. Since we have no information on the rest of the parameters, only the mapping parameters concerning the particular design parameter have been changed from the initial values.
- The 1st and the 3rd response functions depend exclusively on two of the four design parameters, consequently only the rows of $\mathbf{A}_1$ and $\mathbf{A}_3$ corresponding to the two involved design parameters are different from the initial values.
- For response functions $1, \ldots, 4$ some of the elements of $\mathbf{A}_i$ have the same (absolute) values.
- The 5th response function depends on all four design parameters, and all of the 21 mapping parameters have been altered during the iterations.

We also notice that none of the mapping parameters correspond to the solution $\mathbf{A}_i = \mathbf{C}$, $\mathbf{b}_i = \mathbf{d}$ and $\alpha_i = 1$.

Summary of the Results

- The regularization seems to have a slightly negative effect on the convergence speed.
- The normalization factors have practically no effect on the convergence rate.
- The effect of the weighting factors is not possible to investigate for this problem, because the optimizer is found before the chosen weighting strategy has any influence.
- The algorithm does not converge with a reduction of the mapping parameters. This is caused by the definition of the problem.
- The optimal mapping parameters are closely related to the features of the response functions. Several of the mapping parameters are not changed compared to the initial value.

4.3 The TLT2 Problem

4.3.1 Introduction

The TLT2 problem is a design problem of a two-section capacitively-loaded impedance transformer. In this problem we wish to determine the optimal lengths of the two transmission lines. The model is sampled at 11 frequency points giving 11 input reflection coefficient responses. With $n = 2$ and $m = 11$ we have the fine model function $\mathbf{f} : \mathbb{R}^2 \rightarrow \mathbb{R}^{11}$ and wish that the optimal response function satisfies the design specifications:

$$\|\mathbf{f}(\mathbf{x})\|_\infty = \max_{1 \leq i \leq 11} |f_i(\mathbf{x})| \leq 0.50$$

The main goal is to find the optimizer of the fine model, even though the design specifications are satisfied earlier in the optimization process.
The coarse model is a simplification of the fine model, not taking the coupled effects into account. For more details on the physical aspects see [7].

The coarse and the fine model optimizers are:

$$\mathbf{z}^* = \begin{bmatrix} 90 \\ 90 \end{bmatrix} \qquad \mathbf{x}^* = \begin{bmatrix} 74.233241580781367 \\ 79.265787256540477 \end{bmatrix}$$

The fine model optimizer is an estimate from some optimization algorithm available from the original setup file from the SMIS framework by Frank Pedersen.
These design parameters give the responses as shown in figure 4.3.1.

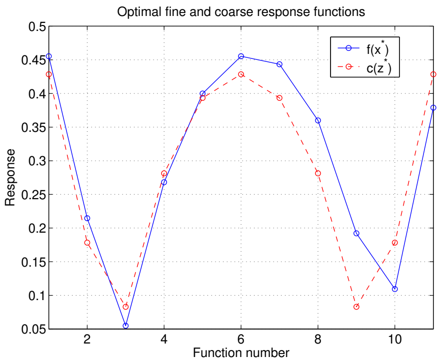

Figure 4.3.1: Coarse and fine model response in their respective optimizers

The response in the optimal surrogate model is identical to the response in the fine model optimizer. The initial guess for the coarse model optimizer

is set to $\mathbf{x}^{(0)} = [100\ ,\ 60]$. The MATLAB-files `ztran2c.m` and `ztran2f.m` are implementations of the coarse model and the fine model. The Jacobians of both model functions are calculated by `diffjacobian`.
In the following we present the results from running the Space Mapping algorithm in the various scenarios. Each test run has a calculation time of about two to five minutes.

4.3.2 The Results of the Test Runs

Effect of the Regularization

Here we run the tests with a full mapping parameter vector corresponding to a full matrix $\mathbf{A}$, ie. the number of parameters is $n_p = 7$. The tolerances for the main problem and the subproblems are set to $\varepsilon_1 = 10^{-14}$ and $\varepsilon_2 = 10^{-4}$. The options used in the Marquardt algorithm are `opts= [1e-8 1e-4 1e-4 200 1e-12]`. The test runs produce the following iteration sequences.

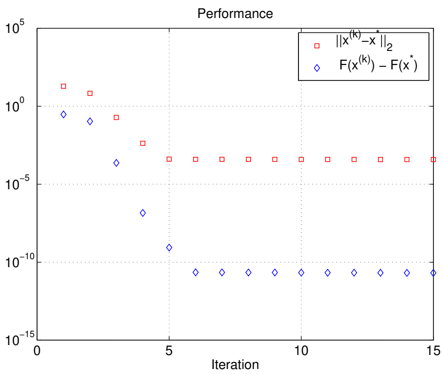

Figure 4.3.2: With regularization

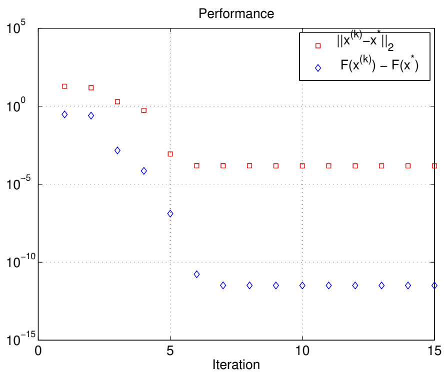

Figure 4.3.3: Without regularization

The desired accuracy ε_2 of the Parameter Extraction problems has a rather large value compared to ε_1, and correspondingly the options used in the stopping criteria for `marquardt` are of the same size. This is chosen because it gives the best results. If we alter the options to $\varepsilon_2 = 10^{-14}$ and the `opts`-vector to `[1e-8 1e-14 1e-14 200 1e-12]` the algorithm actually converges slower as shown in the figures 4.3.4 and 4.3.5.

This seems strange, since one would expect a higher accuracy of the mapping parameters, resulting in a better surrogate model and thereby a faster convergence. But this is not the case. It is not obvious why this behaviour is present. An explanation could be, that it is not possible to satisfy the

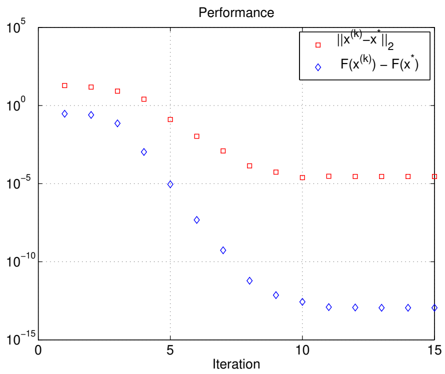

Figure 4.3.4: With regularization

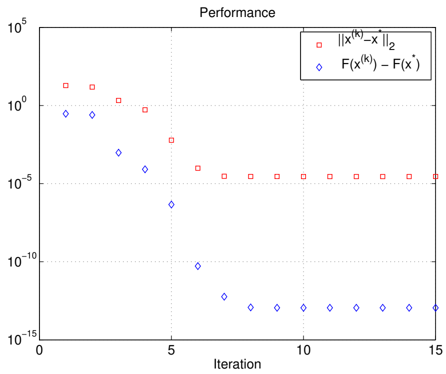

Figure 4.3.5: Without regularization

gradient match to the desired accuracy, and the `marquardt`-algorithm keeps iterating, actually producing a worse set of mapping parameters.
In the case of no regularization there is a possibility of having an overdetermined problem from the 6th iteration. At this point of the iteration, the iterate is already very close to the optimizer, and it apparently is no problem that the regularization term is not added.

Effect of the Normalization Factors

The normalization factors have an important influence on both the iteration sequence and on the surrogate model in the optimizer. The next figures show the performance of the algorithm with three cases of normalization corresponding to section 4.1.1:

- Normalization of all residual elements
- Only normalization of the gradient residual
- No normalization

First we consider the case of no regularization.

We see that the convergence is a little slower in case 2, where $a_1, \ldots, a_k = 1$, but $d \neq 1$. With no normalization the solution is not as good compared to the estimate of $\mathbf{x}^*$.
In the figures 4.3.9-4.3.11 we see the approximation errors E_s (light grid) and E_l (dark grid) corresponding to the three cases of scaling.

Here there is a big difference in the surrogate models: Case 1 with complete scaling is a very bad approximation to the fine model. But when we

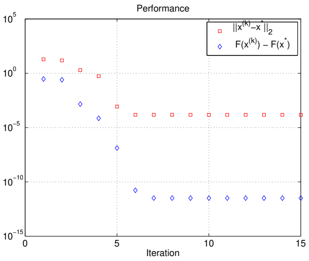

Figure 4.3.6: With normalization

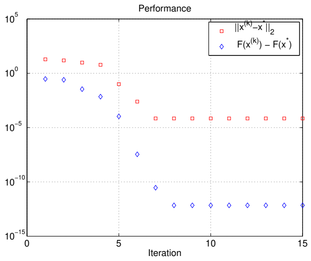

Figure 4.3.7: Only normalization of gradient residual

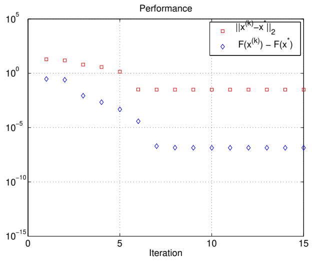

Figure 4.3.8: Without normalization

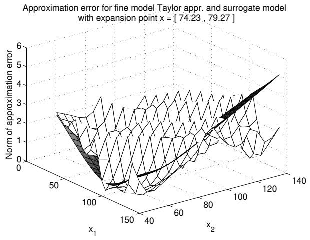

Figure 4.3.9: With normalization

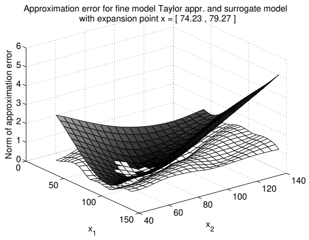

Figure 4.3.10: Only normalization of gradient residual

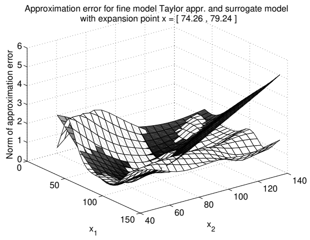

Figure 4.3.11: Without normalization

use only the normalization factor d for the gradient residual, the model is well-behaved and provides a much better approximation to the fine model. The third case with no scaling of the residual vector at all, provides a better result than the first case, but the second case is still to prefer.

For comparison we view the iteration sequences in the case of the regularized residual.

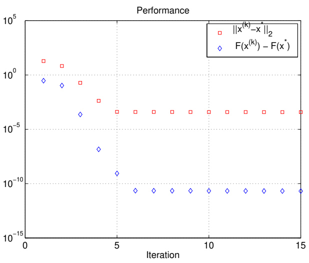

Figure 4.3.12: With normalization

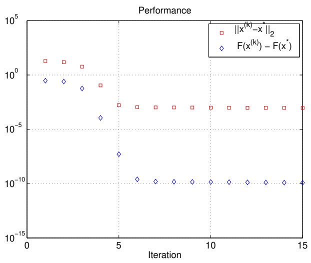

Figure 4.3.13: Only normalization of gradient residual

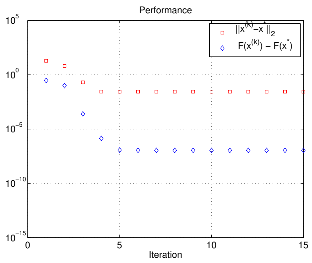

Figure 4.3.14: Without normalization

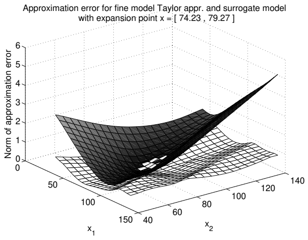

Figure 4.3.15: With normalization

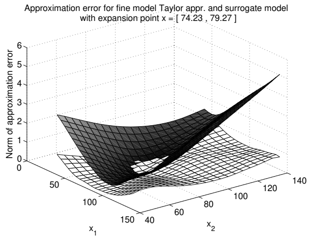

Figure 4.3.16: Only normalization of gradient residual

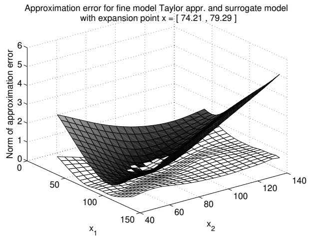

Figure 4.3.17: Without normalization

As seen from figures 4.3.12-4.3.14 the normalization factors are of no relevant importance of the performance in the case of regularization. The solution is again less accurate in the third case without any normalization. Also the surrogate model approximation errors in figures 4.3.15-4.3.17 are almost unaffected by the scaling.

We instead consider the case where we put $a_1, \ldots, a_k$ equal to 1, but still have the normalization factor $d \neq 1$, we get the results in figure 4.3.18:

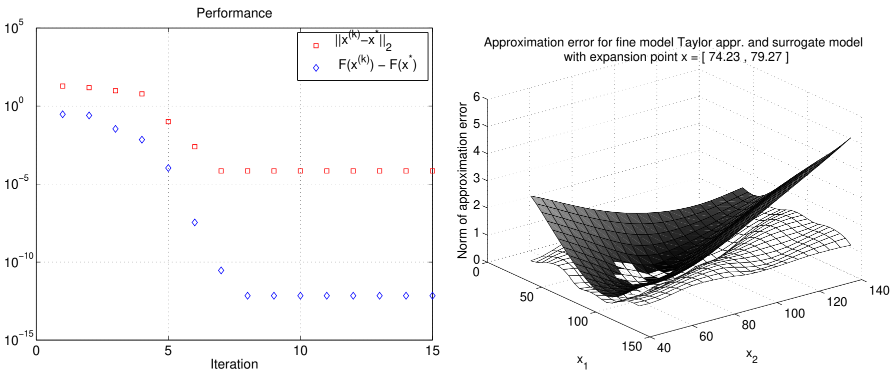

Figure 4.3.18: With normalization only of gradient residual

We see that the partly normalization is resulting in a significantly better surrogate approximation error, which is now better that the approximation error from using a linear Taylor model.
We conclude that for this particular problem, the scaling of the residual elements is important for the quality of the surrogate approximation, but not for the convergence.

Effect of the Weighting Factors

Here we test some effects of the weighting factors by looking at the case with no regularization of the residual. The weighting factors have no effect on the iteration sequence, as it is seen from the figures 4.3.19 and 4.3.20.

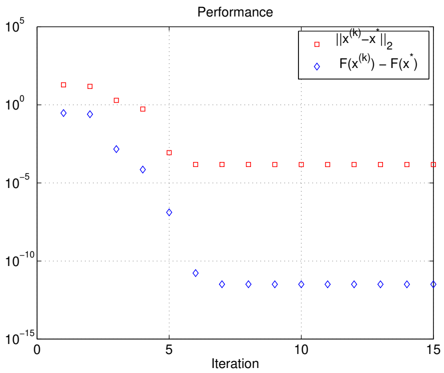

Figure 4.3.19: With normalization and without weighting

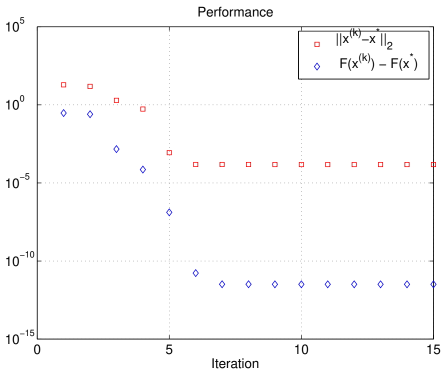

Figure 4.3.20: With normalization and with weighting

The number of unknown parameters is $n_p = 7$, which means, that the weighting factors will possibly influence the results from iteration 6 and onwards. But the surrogate model approximation error corresponding to the weighted case is similar to figure 4.3.9, and serves as a poor approximation to the fine model.

At last we consider the weighting strategy combined with only normalization of the gradient residual. The results from this scenario are shown in figure 4.3.21.

Now the surrogate is again well-behaved, which is caused by putting the normalization factors $a_1, \ldots, a_k = 1$.

We conclude, that the weighting factors in this test problem are practically without influence on the results - both regarding performance of the Space Mapping algorithm and regarding the quality of the surrogate model approximation in the optimizer.
Referring to section 3.1 we can only expect the results to be within the accuracy of the maximal errors from not using the exact gradients. The tolerance used in these test results is $\varepsilon_1 = 10^{-14}$, which is probably too strict. We therefore conclude, that the problem is not suited for investigating the weighting factors.

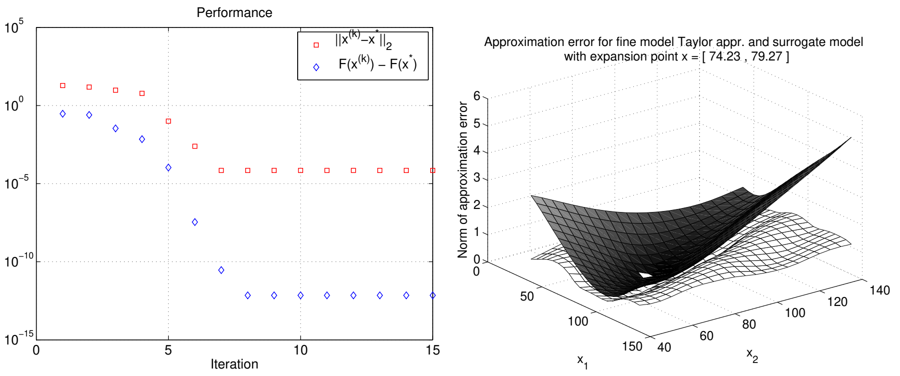

Figure 4.3.21: With normalization only of gradient residual and with weighting

Effect of the Number of Mapping Parameters

We here bring the results of the test runs with the diagonal input mapping parameter matrix $\mathbf{A}$. In this case we have $n_p = 5$ elements in the parameter vector $\mathbf{p}$. There is a possibility of having an overdetermined system in iteration 4. The test runs are made with the tolerance parameters $\varepsilon_1 = 10^{-14}$, $\varepsilon_1 = 10^{-4}$ and `opts= [1e-8 1e-4 1e-4 200 1e-12]`. Figures 4.3.22 and 4.3.23 show the results with a diagonal input mapping matrix in the regularized and the unregularized case.

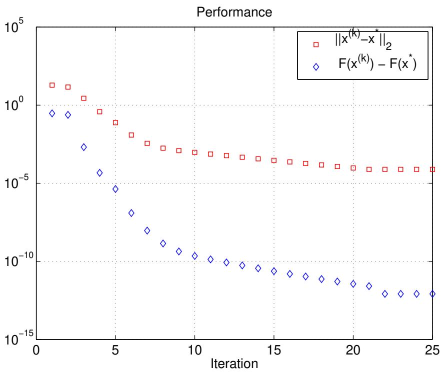

Figure 4.3.22: With regularization

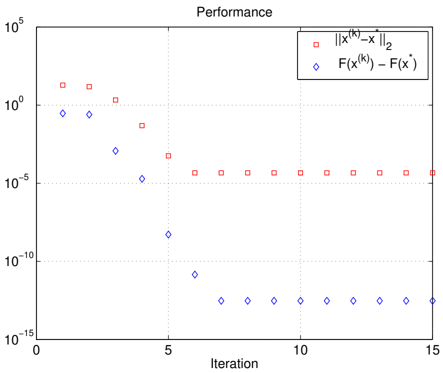

Figure 4.3.23: Without regularization

We see that the convergence is slower compared to figures 4.3.12 resp. 4.3.6 both with and without the regularization term added, when only considering the reduced parameter vector. The corresponding approximation errors for the surrogate model and the Taylor model in the optimizer are depicted in figures 4.3.24 and 4.3.25.

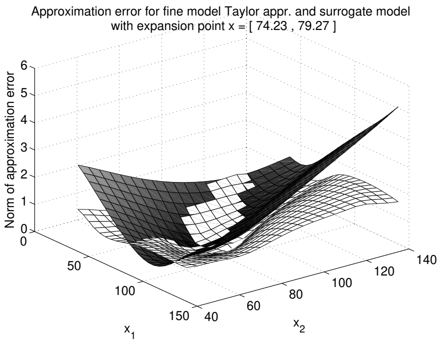

Figure 4.3.24: With regularization

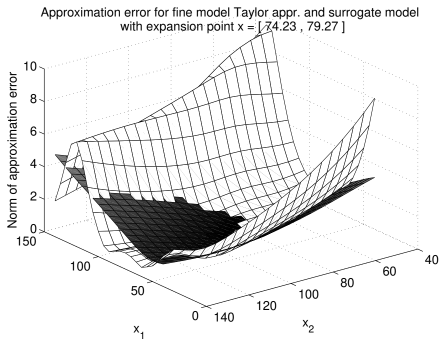

Figure 4.3.25: Without regularization

Again the surrogate model approximation error is large for the unregularized case compared to the regularized. In figure 4.3.25 the surrogate approximation is not as good as the approximation with a linear Taylor model in most af the design parameter region.

Also in this case the tolerance options are of great importance. We run the algorithm with smaller tolerances: $\varepsilon_1 = \varepsilon_2 = 10^{-14}$ and the `marquardt`-options `[1e-8 1e-14 1e-14 200 1e-12]`. As the results in figures 4.3.26 and 4.3.27 show, the convergence is now as fast as with the full parameter vector. This applies for both the case with regularization and the case without regularization.

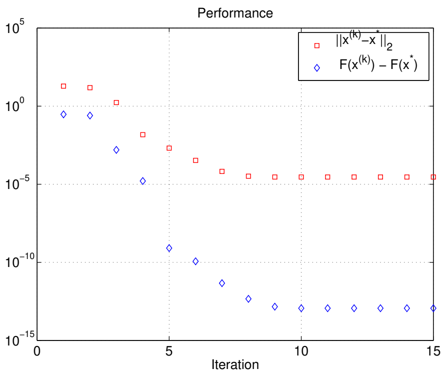

Figure 4.3.26: With regularization

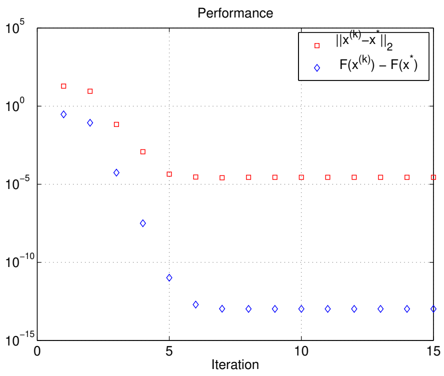

Figure 4.3.27: Without regularization

The approximation errors E_s and E_l corresponding to the regularized and unregularized tests, are seen in figures 4.3.28 and 4.3.29. The approximation error for the surrogate model is not better, when using a smaller tolerance value.

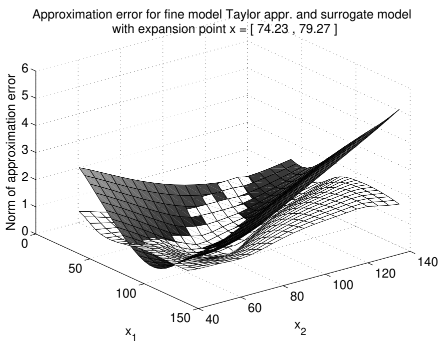

Figure 4.3.28: With regularization

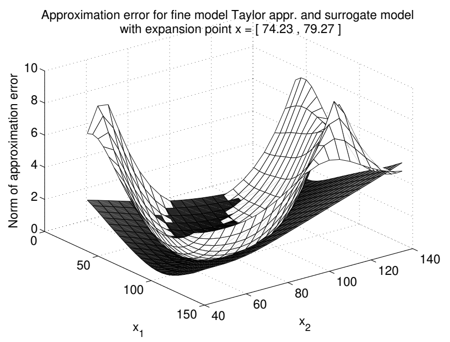

Figure 4.3.29: Without regularization

We conclude that the optional tolerance values are of great importance in this problem. A smaller tolerance here results in faster convergence, but the surrogate approximation errors are not effected positively by the smaller tolerance. In order to ensure good surrogate approximations over a large region of the design parameter space, we must put the normalization factors $a_1, \ldots, a_k$ equal to 1.

Optimal Mapping Parameters

Finally we look at the values of the mapping parameters in the optimal surrogate model. We have the initial mapping parameters $\mathbf{A}_i = \mathbf{I}$, $\mathbf{b}_i = \mathbf{0}$ and $\alpha_i = 1$ for all response functions $i = 1, \ldots, 11$ in iteration 0. It is interesting to see how different the optimal mapping parameters are from these values. We consider two test scenarios:

- With regularization and with normalization (figure 4.3.15)
- Without regularization and with normalization (figure 4.3.9)

In the first case a representative matrix $\mathbf{A}$ in the optimal surrogate model is:

$$\mathbf{A} = \begin{bmatrix} 1.09 & 0.10 \\ 0.05 & 1.07 \end{bmatrix}$$

The elements of $\mathbf{b}_i$ are of order of magnitude 10^{-3}, and all values of the output mapping parameters α vary in the interval $[0.75 \ , \ 1.45]$. But for response function number 2 we have:

$$\mathbf{A}_2 = \begin{bmatrix} -0.06 & -1.27 \\ -1.11 & 0.34 \end{bmatrix} \qquad \mathbf{b}_2 = \begin{bmatrix} -0.024 \\ 0.018 \end{bmatrix}$$

which is not close to the identity matrix. We conclude that most of the matrices $\mathbf{A}$ are not very different from the initial identity matrix, and most of the elements in $\mathbf{b}$ are close to 0. The surrogate model with these mapping parameters is well-behaved as we have seen in figure 4.3.15.

But in the case of no regularization we find a much more varying picture. Some of the $\mathbf{A}_i$'s are close to the identity matrix some are not, though all $\mathbf{A}_i$-elements have absolute values between 0 and 3.5. Again response function number 2 is special:

$$\mathbf{A}_2 = \begin{bmatrix} -0.37 & -0.31 \\ 2.12 & 1.70 \end{bmatrix} \qquad \mathbf{b}_2 = \begin{bmatrix} -93.91 \\ -65.52 \end{bmatrix} \qquad \alpha_2 = -12.85$$

Some $\mathbf{b}$-vectors have elements close to 0, but as the result above shows there are also examples of extremely different b-values. The α_i's vary between -12.85 and 1.42. The surrogate model approximation error shown in figure 4.3.9 has extreme variation, which could be a consequence of the variations of the mapping parameters.
This behaviour of the mapping parameters is probably connected to the missing regularization term. In the regularization case we force the mapping parameters to be close to the previous set, and in this way, we cannot end up with results very far from the initial values. We conclude, that if we do not regularize wrt. the previous parameter set, we could end up with a solution very far from the previous.

Finally we consider the mapping parameters corresponding to figure 4.3.10. Here the approximation error for the surrogate is not as large as before. The mapping parameters for response function 2 are now:

$$\mathbf{A}_2 = \begin{bmatrix} 4.08 & 4.52 \\ -1.06 & 2.73 \end{bmatrix} \qquad \mathbf{b}_2 = \begin{bmatrix} -231.01 \\ -72.83 \end{bmatrix} \qquad \alpha_2 = -0.38$$

Generally the mapping parameters in this case still vary much from response to response, but apparently the surrogate approximation is better.

Direct Optimization

For comparing the Space Mapping algorithm with a classical optimization algorithm, we bring the following results from direct optimization of the fine model by the two algorithms `direct` and `directd` from the SMIS framework implemented by Frank Pedersen.

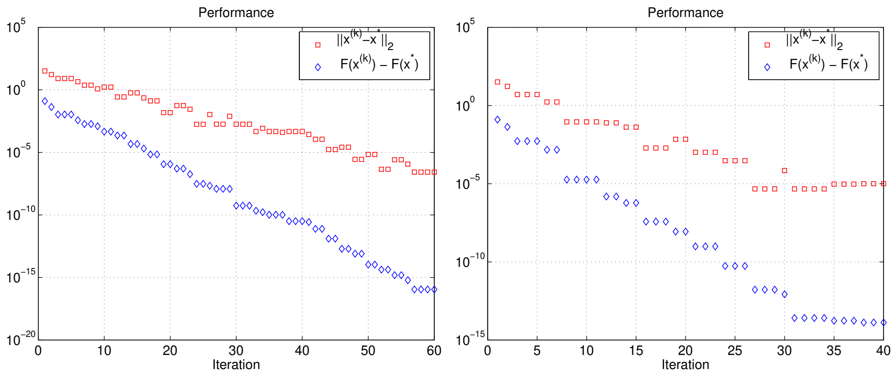

Figure 4.3.30: Performance of direct optimization of the fine model (direct left, directd right)

The `direct`-algorithm uses a Broyden updated approximation of the first order partial derivatives, while the other uses the Jacobian directly from the fine model function.
It is obvious, that the Space Mapping algorithm is much more efficient than the classical optimization algorithms.

Summary of the Results

By running the different versions of the Space Mapping algorithm on this test problem, we have seen, that the results are very different. It is difficult to conclude, why the results look as they do, and impossible to generalize the behaviour to other problem types. But for this particular problem the results show the following:

- The regularization seems to have a positive effect on both the convergence speed and the optimal surrogate aproximation.
- The normalization factors have only little effect on the convergence speed, but a big influence on the quality of the surrogate approximation, when we don't use regularization.
- The weighting factors do not seem to have a noticable effect on either the convergence or the surrogate approximation.
- The reduction of the mapping parameters still provides good convergence results, although the tolerance options have an effect.
- The optimal mapping parameters are influenced by the regularization

term.

- The tolerance options have an effect on the iteration sequences.

4.4 The TLT7 Problem

4.4.1 Introduction

This design problem concerns a seven-section capacitively-loaded impedance transformer. The problem is similar to the TLT2 problem, but the dimensions are larger. Here we aim to find the optimal lengths of seven transmission lines, and we sample the model at 68 frequencies. With $n = 7$ and $m = 68$ we now have $f : \mathbb{R}^7 \to \mathbb{R}^{68}$ and the design specifications:

$$\|\mathbf{f}(\mathbf{x})\|_\infty = \max_{1 \le i \le 11} |f_i(\mathbf{x})| \le 0.07$$

We hereby wish to keep all 68 input reflection coefficient responses below the value 0.07. The main goal is to find the optimizer of the fine model, even though the design specifications are satisfied earlier in the optimization process.

The optimizers $\mathbf{x}^*$ and $\mathbf{z}^*$ of the respective models are:

$$\mathbf{z}^* = \begin{bmatrix} 90 \\ 90 \\ 90 \\ 90 \\ 90 \\ 90 \\ 90 \end{bmatrix} \qquad \mathbf{x}^* = \begin{bmatrix} 81.65148220976137 \\ 85.52171481989218 \\ 87.54698544710705 \\ 88.62554722596209 \\ 89.25567599948434 \\ 89.95304587498956 \\ 84.87655093368730 \end{bmatrix}$$

with the corresponding the response functions in the figure below.

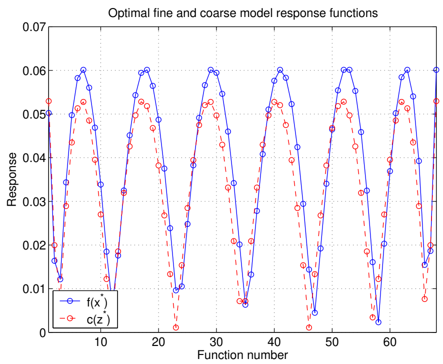

Figure 4.4.1: Coarse and fine model response in their respective optimizers

The response in the optimal surrogate model is identical to the response in the fine model optimizer. The fine model optimizer is an estimate from some

optimization algorithm available from the original setup file from the SMIS framework by Frank Pedersen, and this $\mathbf{x}^*$ is used for comparison with the test results.
The initial guess for the coarse model optimizer is equal to the coarse optimizer.
The coarse and fine models are implemented in the MatLab-files `ztran7c.m` and `ztran7f.m`. The test runs of the TLT7 problem take at least two hours when the .

The test problem is treated in [1] and [2]. Here a different scaling is used, which implies, that the optimal set of design parameters is approximately 1000 times smaller. Furthermore the problem is approached in a different way. The fine and coarse model responses are complex, and the tests from the article use both the real and the complex parts to produce response vectors of the double length, ie. 136 responses per evaluation. The results in this report only uses the absolute value. In this way the surrogate model contains the double amount of information, which could perhaps lead to a better surrogate model. It is therefore unlikely, that the results in this report are identical to the results of [1] and [2].
The following test runs of the TLT7 problem are all with the tolerance options $\varepsilon_1 = 10^{-14}$ and $\varepsilon_2 = 10^{-14}$. The options used in the Marquardt algorithm are `opts=` `[1e-8 1e-14 1e-14 200 1e-12]`.

4.4.2 The Results of the Test Runs

Effect of the Regularization

The algorithm does not converge to the optimizer in the case where we use the regularized residual in the Parameter Extraction. When solving the unregularized Parameter Extraction problem we get the performance results of the algorithm depicted in figure 4.4.2. The result is with normalization of the residual elements and a full size parameter vector.

The optimizer is found in 10 iterations within the accuracy 10^{-16} of the objective function. The number of unknown mapping parameters in each of the Parameter Extraction problems is $n_p = 57$. In the case of no regularization this means, that until iteration 51 we will have less equations than unknowns. This is obviously not an obstacle, since we find the optimizer in 12 iterations.
It is not known, why the test scenario with the regularized Parameter Extraction problem does not converge to the optimizer.

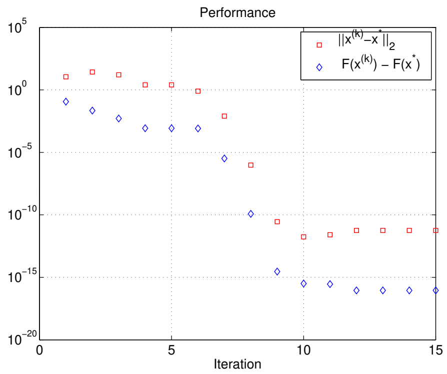

Figure 4.4.2: Without regularization

Effect of the Normalization Factors

In figure 4.4.3 we see the iteration sequence for the case of no normalization of the residual vector, corresponding to $a_1, \ldots, a_k = 1$ and $d = 1$. The test is computed with full parameter vector.

The convergence is a little faster than the result from figure 4.4.2.

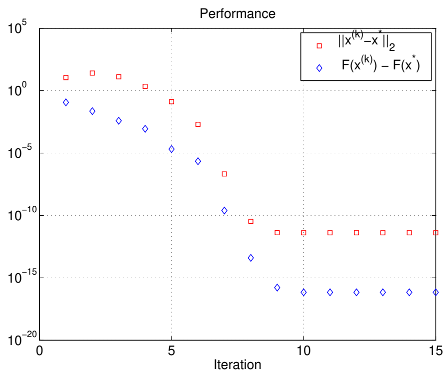

Figure 4.4.3: Without normalization

Effect of the Weighting Factors

With no reduction of the number of unknown mapping parameters in the Parameter Extraction we have $n_p = 57$, and hereby the weighting factors may influence the results from iteration number 51. Because the optimizer is already found at this point, we can not use this test scenario to investigate the influence of the weighting.

Instead we look at the scenario with reduction of the parameter vector. In this case we have $n_p = 15$, and some of the weighting factors will be different from 1 from iteration 9 and onwards, if we use the strategy from section 3.6. Figure 4.4.5 on the right shows the iteration sequence with the use of weighting strategy. For comparison the results without weighting are shown in figure 4.4.4 on the left.

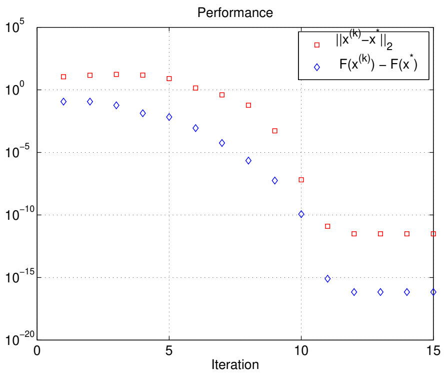

Figure 4.4.4: Without weighting

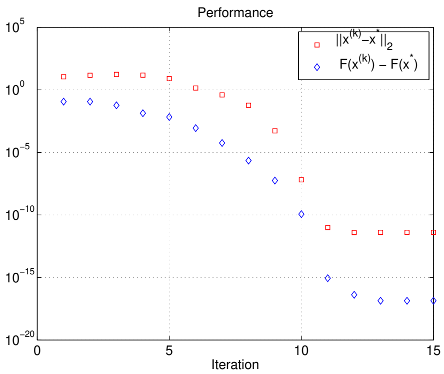

Figure 4.4.5: With weighting

The figures are almost identical, and the weighting factors are practically without influence in this problem.

Effect of the Number of Mapping Parameters

We now show the results from using a reduced parameter vector, corresponding to a diagonal matrix for the input mapping matrix **A**. Figure 4.4.6 is for the regularized case, and figure 4.4.7 for the unregularized case.

The algorithm does not converge in the first case with regularization. In the second case the optimizer is found in 12 iteration steps with an accuracy of 10^{-15} on the fine model objective function. It is not known, why the test run with regularization does not converge to the optimizer.
There is practically no difference between the iterations with the full parameter vector (figure 4.4.2) and the reduced parameter vector (figure 4.4.7). The algorithm works well when solving the unregularized Parameter Extraction problem, and it has no effect whether the problems are underdetermined or not.

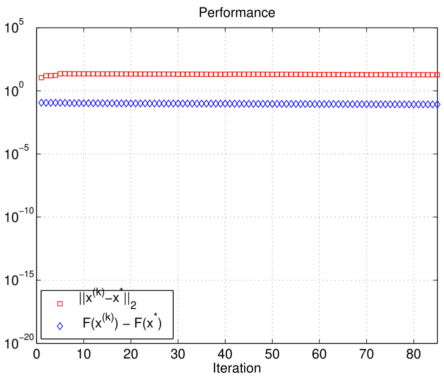

Figure 4.4.6: With regularization

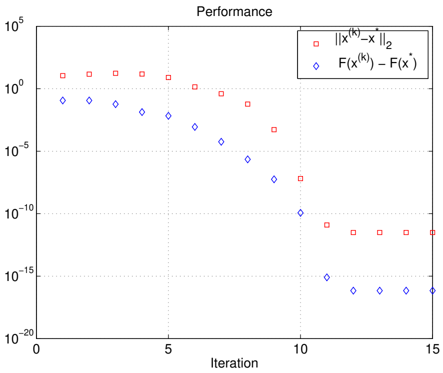

Figure 4.4.7: Without regularization

Optimal Mapping Parameters

This problem is of larger dimension than the other test problems, and we have not investigated the optimal mapping parameters.

Direct optimization

Below we see the performance of the `directd`-algorithm.

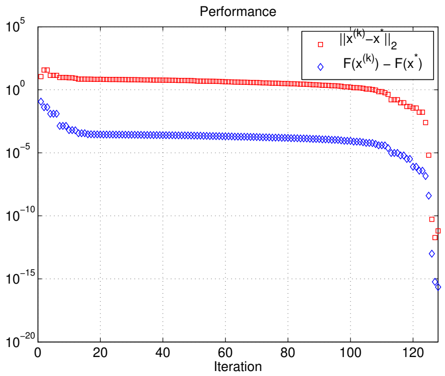

Figure 4.4.8: Performance of direct optimization of the fine model

The classical optimization method converges in 127 iterations, and we conclude that the Space Mapping technique is very useful in this problem, since it reduces the number of fine model evaluations with approximately a factor 10.

Summary of the Results

- When the Parameter Extraction problem is regularized we have no convergence to the optimizer, both in the case of a full and a reduced parameter vector.
- The normalization factors have little effect on the convergence speed.
- The weighting factors have practically no influence on the results in the case of a reduced parameter vector.
- The reduction of the mapping parameters has no effect on the performance in this test problem.
- It is not possible to get as good convergence results as the results from [1] and [2], probably because the problem is not identical to the one solved in this report.

Chapter 5

Future Work

5.1 Improvements of the SMIS Implementation

A number of changes of the implementation could be made in order to make the SMIS framework more flexible and user-friendly. Also the accuracy of the computations could be improved for some problems. The current framework consists of many directories and files linked together in a certain structure. The suggested changes will therefore influence many of the included files, which will effect the duration of the work involved.

In the current implementation the gradients of the surrogate model wrt. the parameter vector are calculated by forward difference approximations. In order to minimize the truncation errors, it would be an advantage to exploit the exact gradients, if they are available.

The MATLAB-file `diffjacobian` used to calculate the forward difference approximation. In the current implementation the parameter η defining the relative step length is fixed at a certain value, and can only be altered by modifying the MATLAB-file directly.

The user friendliness could be improved by including more optional parameters to the problem setup-file. The setup-file should define the following optional parameters:

Δ	Initial trust region radius for Algorithm 1.
$\max_{f1}$	Maximal number of function evaluations in Algorithm 1.
$\max_{f2}$	Maximal number of function evaluations in Algorithm 2.
$\max_{f3}$	Maximal number of function evaluations in Algorithm 3.
η_x	Step length in `diffjacobian` when calculating Jacobian wrt. $\mathbf{x}$.
η_p	Step length in `diffjacobian` when calculating Jacobian wrt. $\mathbf{p}$.
ε_F	Used in the stopping criterion for the fine model objective function.
ε_K	Used in the stopping criterion for the gradient matching.
ε_{hx}	Used in the stopping criterion for the step length for $\mathbf{x}$-iterates.
ε_{hp}	Used in the stopping criterion for the step length for $\mathbf{p}$-iterates.
`opts`	Options for `marquardt` (`[1e-8 1e-4 1e-4 200 1e-12]`).
`diagA`	Parameter defining the number of mapping parameters (`0`: full matrix $\mathbf{A}$, `1`: diagonal matrix $\mathbf{A}$)

Some of the above options are already used in the SMIS framework, ie. Δ, $\max_{f1}$, $\max_{f2}$, and `diagA`, and also two tolerance parameters corresponding to ε_F, ε_K, ε_{hx} and ε_{hp}.

We also list some possible changes in order to increase the accuracy of the computations:

- Introducing the step length parameter η as an input parameter to the function `diffjacobian`.
- Making it possible to exploit exact gradients, if they are available.

5.2 Suggestions for Further Investigations

As the investigations in this report have shown, there are a lot of unresolved matters concerning the solution of the Parameter Extraction problem. The problem can be defined in various ways, as well as different methods can be chosen to perform the Parameter Extraction. In case there is more than one solution, we can have an influence on which solution is returned by the optimization algorithm, if we choose a particular formulation of the residual.

We now list some suggestions for future investigations:

- Testing the algorithm with more test problems.
- Investigating the effects of the optional tolerance parameters on the results, hereby the tolerance parameters in relation to the errors expected from not using exact gradients of the models, cf. section 3.1.

- Providing a set of recommendable tolerance parameters.
- Testing different strategies for updating the penalty factor.
- Testing other weighting factor strategies.
- Further investigations of how the number of mapping parameters influence the results.
- Further analysis on the optimal mapping parameters.
- Testing different strategies concerning the number of mapping parameters, eg. releasing a parameter in every main iteration.
- Further analysis of the test results, eg. looking at the termination criteria for the sub-algorithms.
- Further analysis of the iteration sequences.
- Defining different residual formulations, eg. letting the formulation depend on the number of iterates available. Possibly omitting the gradient residual in the first iterations in order to minimize the number of fine model evaluations.

Chapter 6

Conclusion

In this thesis a Space Mapping algorithm has been presented and tested on some test problems. We have made a number of investigations in order to make a robust implementation and analyze the character of the Parameter Extraction problem in regard to the solutions, we can expect to find.

The step length used in the finite difference approximations is found to be of great importance, in case the exact gradients of the model are not available. The step length influences both the truncation errors and the rounding errors, and based on results from the TLT2 problem and the theoretical error functions, we find a suitable value for the step length to be approximately 10^{-5}.

The number of equations in the Parameter Extraction problems depends on, how many main iterations we have made. A large part of the Parameter Extraction problems are therefore underdetermined, and consequently there are infinitely many solutions. There is also the possibility of having an overdetermined problem, in which case we may be able to effect the solution by means of the residual formulation.

The formulation of the residual including the normalization factors, the weighting factors and the penalty factor makes it possible to give each of the residual elements an individual priority. In this way we can more or less choose which order of local and global agreement the surrogate model should satisfy. The solution we find is influenced by the choice of the residual formulation.
It must be clarified, that we are not always sure, that the various factors even influence the results. This depends entirely on the character of the problem, as discussed in section **??**.

We have chosen to define the Parameter Extraction problems as least squares problems, and solve them by the Marquardt algorithm. The Marquardt method is well-suited for the problem, since the damping parameter controls the step length and ensures, that the Marquardt equations have a unique solution.
If the rank of $\mathbf{J}_r$is not full, the solution depends on the damping parameter μ. For small μ-values the Marquardt solution is nearly identical to the minimum norm solution, which is the solution closest to the previous one. In this case the Marquardt equations give the same effect as a form of regularization wrt. the previous solution.
If we regularize the problems, we are guaranteed, that the Parameter Extraction problems are overdetermined in all iterations. The regularization term ensures, that we find a solution close to the previous solution. We have shown, that solving the Parameter Extraction problem for the regularized residual formulation corresponds to a special case of the Marquardt equations with the damping parameter $1+\mu$ and a changed right hand side. The regularization means, that we can not be sure to satisfy the gradient match, if the solution is far from the previous.

The algorithm with the various residual formulations is tested on three problems: The Rosenbrock problem in its original form and in the augmented version provides a theoretical example, and the TLT2 and TLT7 problems are examples of engineering design problems of more practical character.

The test results show, that the algorithm works well in case of both solving the regularized and the unregularized problem. In several cases the latter is actually resulting in a faster convergence to the optimizer, and the solution is found, before there is a possibility of having overdetermined Parameter Extraction problems. We conclude, that for the considered test problems it is no obstacle, if the Parameter Extraction problems are underdetermined.

The Rosenbrock problem shows some interesting features concerning the optimal mapping parameters. The Rosenbrock function is special, because the second response function is linear and only depends on one of the design parameters. This has consequences for the mapping parameters through the iteration sequence, in the sense that several of the mapping parameters are not changed form the initial values.
For the augmented Rosenbrock problem the same features are found for the optimal mapping parameters.

In the engineering design problem TLT2 we get good convergence results for both the residual version with and without the regularization term.

But when we consider the approximation errors from using the surrogate model defined by the optimal mapping parameters, the two versions give very different results.
The regularized problem provides good surrogate approximations and generally, the surrogate model is better than a classical Taylor model approximation - not only over a large region of the design parameter space, but also near the expansion point. For the scenario without regularization of the residual we get a very poor surrogate model, even though the algorithm has converged to the fine model optimizer.
It shows, that for this problem the normalization factors have an influence. We achieve acceptable results both regarding convergence and the quality of the surrogate model approximation for scaling of only the gradient residual. It is not possible to conclude anything in general on the normalization factors, but in this case they must be omitted to ensure a suitable surrogate model.
The importance of the quality of the surrogate model approximation of course depends on the design problem: If we are only interested in the optimal design parameters, the surrogate model is perhaps unimportant. But in modelling problems we may be interested in replacing the fine model with the surrogate model and examining this instead at a low computational expense. In such a case an unaccurate surrogate approximation is not useful.

The last test problem TLT7 is of larger dimension. The results here show fast convergence in the case of the unregularized Parameter Extraction problems, and the reduction of the mapping parameters have no influence on the results. But when the regularization term is added, there is no convergence. It is not possible to explain this behaviour.

It applies for all the considered test problems, that the Space Mapping algorithm provides fast convergence results compared to classical optimization methods.

Appendix A

Short User's Guide for The SMIS Framework

In the following chapter a brief overview of the SMIS framework in MATLAB is given.

A.1 The Problem Setup-file

The setup-file for the given problem is placed in the corresponding problem directory. The setup-file contains information on the fine and coarse model optimizers and linear equality/inequality constraints for the optimization problem.

These are defined by:

$\mathbf{A}_{(1:leq,:)}\mathbf{x} + \mathbf{b}_{(1:leq,:)} = \mathbf{0}$ and $\mathbf{A}_{(leq+1:l,:)}\mathbf{x} + \mathbf{b}_{(leq+1:l,:)} \geq \mathbf{0}$

The setup-file must return a structure S containing the following fields:

`n`	The number of design parameters.
`m`	The number of n parameters.
`fine`	The `m`-file implementing the fine model function.
`coarse`	The `m`-file implementing the coarse model function.
`parf`	Vector with the sampling points for the fine model.
`parc`	Vector with the sampling points for the coarse model.
`paro`	Structure with information used for plotting.
`xast`	The fine model optimizer.
`zast`	The coarse model optimizer.
`A`	l-by-n matrix defining the constraints ($l = \texttt{size}(\texttt{A}, 1) - leq$)
`b`	l column vector defining the constraints.
`leq`	Number of linear equality constraints.
`x`	Initial guess for coarse model optimizer.
`delta`	Initial trust region radius for surrogate optimization problem.
`eps1`	Used in stopping criterion for the main problem.
`eps2`	Used in stopping criterion for the Parameter Extraction problems.
`maxfun1`	Maximal number of function evaluations in Algorithm 1.
`maxfun2`	Maximal number of function evaluations in Algorithm 2 (also used in Algorithm 3).
`p1`	Norm defining the objective function of the main problem.
`p2`	Norm defining the objective function of the Parameter Extraction problems. When using the algorithm version with `marquardt` this option has no effect.
`icontr`	Parameter used for controlling the computation: 1: starts algorithm, 2: starts algorithm and prints information.
`diagA`	Parameter defining number of mapping parameters (`0`: full matrix **A**, `1`: diagonal matrix **A**)

The options for `marquardt` are defined in the file 'lmm' as eg. `opts=([1e-8 1e-14 1e-14 200 1e-12])`.

A.2 Calling the Space Mapping Algorithm

Before starting the main algorithm the fine and coarse model must each be implemented in an `m`-file with the syntax:

`[f,df] = fine(x,t)` and `[c,dc] = coarse(x,t)`

The files must be placed in the proper problem directory with the setup-file from the previous section.

The Space Mapping iterations are started by the `jobcontrol` program by the following call in the MATLAB command window (the command must be executed in the MATLAB directory 'setup'):

```
jobcontrol(prob,job,datafile)
```

The problem 'prob' is solved by the algorithm framework specified by 'job', and the results are written to the `mat`-file 'datafile' containing a structure `T` with information on the iteration performance. The datafile is put in the sub-directory 'output'.

The job types used in this report are:

12 The original SMIS4-algorithm implemented by Frank Pedersen.
13 The modified SMIS5-algorithm implemented with the `marquardt`-algorithm with regularization
14 The modified SMIS5m-algorithm implemented with the `marquardt`-algorithm without regularization.
15 The modified SMIS5mw-algorithm implemented with the `marquardt`-algorithm without regularization and with weighting factors

A.3 Plotting and Viewing Data

When the `jobcontrol` command is executed succesfully, the results of the space mapping iterations can be plotted by the call:

```
jobcontrol(prob,job,datafile)
```

where the job number is replaced by '10' or '17'.

Both jobs provide two plots:

Choosing the job '10' Figure 1 is a semilogarithmic convergence plot with the relative norm of the difference between the current iteration and the optimizer plotted for each iteration number. Furthermore the figure shows the relative difference between the norm of function values of the current and the optimal iteration measured in the `p1`-norm.
In figure 1 from job '17' the points are not plotted relative to the best iter-

ate, which makes it more useful in the case, where the optimizer and/or the optimal funtion vector is close to or equal to 0.
The second figure plots for both job types the trust region radius and the norm of the step length for each iteration.

The data structure `T` can be loaded into the workspace or a file by changing to the MATLAB path 'output' and typing the command:

`load datafile`.

Symbols and Notation

$\mathbf{x}$: Design parameter vector for fine and surrogate model
$\mathbf{z}$: Design parameter vector for fine and surrogate model
$\mathbf{f}$: Fine model vector function
$\mathbf{c}$: Coarse model vector function
$\mathbf{s}$: Surrogate model vector function
$\mathbf{r}$: Residual vector function
$\hat{\mathbf{r}}$: Regularized residual vector function
$\mathbf{g}$: Gradient residual vector function
$\mathbf{O}$: Output mapping
$\mathbf{P}$: Input mapping function
$\mathbf{p}$: Mapping parameter vector
$\mathbf{A}$: Input mapping matrix
$\mathbf{b}$: Input mapping vector
α : Output mapping parameter
β : Output mapping parameter
$\mathbf{h}$: Step vector
$\mathbf{e}$: Unit vector
$\mathbf{J}$: Jacobian matrix
$\mathbf{I}$: Identity matrix
$\mathbf{l}$: Linearized vector function
K : Measure of gradient match violation
H : Objective function
F : Objective function for fine model or in the Parameter Extraction problem
S : Objective function for surrogate model
n : Number of design parameters
m : Number of response functions
n_p : Number of mapping parameters
n_r : Number of residual elements
γ : Factor in weighting function
$\mathbf{W}$: Diagonal weight matrix for residual function
$\mathbf{V}$: Diagonal weight matrix for regularisation term
w : Weighting factor in $\mathbf{W}$
v : Weighting factor in $\mathbf{V}$

a : Normalization factor for function value residual
d : Normalization factor for gradient residual
$\mathbf{X}$: Matrix with sorted $\mathbf{x}$-iterates
$\mathbf{dX}$: Vector with sorted distances for $\mathbf{x}$-iterates
$\mathbf{F}$: Vector with sorted distances for $\mathbf{x}$-iterates
E : Error function
μ : Damping parameter in Marquardt equation
$\hat{\mathbf{A}}$: Matrix defining linear constraints in Algorithm 2
$\hat{\mathbf{b}}$: Vector defining linear constraints in Algorithm 2
Δ : Trust region radius
ε_M : Machine accuracy
ε : Tolerance value
η : Factor defining the relative step length
D_F : Forward difference approximation

Subscript Indexes

i, j : Vector/matrix index i, j
f, s : Corresponding to fine model resp. surrogate model
x, p : Differentiation wrt. x resp. p
$r, \hat{r}$: Corresponding to residual resp. regularized residual

Superscript Indexes

* : Optimizer
$^{(k)}$: Iteration number

Bibliography

[1] John W. Bandler, Daniel M. Hailu, Kaj Madsen and Frank Pedersen: *A Space Mapping Interpolating Surrogate Algorithm for Highly Optimized EM-Based Design of Microwave Devices.* Accepted for publication, IEEE Trans. Microwave Theory Tech., 2004.

[2] John W. Bandler, Qingsha S. Cheng, Sameh A. Dakroury, Daniel H. Gebré-Mariam, Kaj Madsen, Ahmed S. Mohamed and Frank Pedersen: *Space Mapping Interpolating Surrogates for Highly Optimized EM-Based Design of Microwave Devices.* IEEE MTT-S Int. Microwave Symp. Dig., Forth Worth, TX, 2004, pp. 1565-1568.

[3] John W. Bandler, Qingsha Cheng, Sameh A. Dakroury, Ahmed S. Mohamed, Mohamed H. Bakr, Kaj Madsen and Jacob Søndergaard: *Space Mapping: State-of-the-Art.* IEEE Transactions on Microwave Theory and Techniques, Vol. 52, No. 1, January 2004.

[4] J. W. Bandler, R. M. Biernacki, S. H. Chen, P. A. Grobelny, and R. H. Hemmers: *Space mapping technique for electromagnetic optimization.* IEEE Trans. Microwave Theory Tech., vol. 42, pp. 2536-2455, Dec. 1994.

[5] J. W. Bandler, R. M. Biernacki, S. H. Chen, R. H. Hemmers, and K. Madsen: *Electromagnetic optimization exploiting agressive space mapping.* IEEE Trans. Microwave Theory Tech., vol. 43, pp. 2874-2882, Dec. 1995.

[6] Frank Pedersen: *Discussion.*

[7] Jacob Søndergaard: *Optimization Using Surrogate Models - by the Space Mapping Technique.* IMM-PHD-2003-111, Kgs. Lyngby, 2003.

[8] Kaj Madsen, Hans Bruun Nielsen, Ole Tingleff: *Methods for Non-linear Least Squares Problems.* IMM, Department of Mathematical Modelling, DTU, 7. 7. 1999.

[9] Kaj Madsen, Hans Bruun Nielsen, Ole Tingleff: *Optimization with Constraints.* IMM, Department of Mathematical Modelling, DTU, 12. 10. 2001.

[10] Lars Eldén, Linde Wittmeyer-Koch, Hans Bruun Nielsen: *Introduction to Numerical Computation.* January 2004.

[11] Hans Bruun Nielsen: *Checking Gradients.* IMM, 21. 9. 2000.

[12] Hans Bruun Nielsen: *immoptibox*. Available August 2004 at the homepage: http://www.imm.dtu.dk/~hbn/immoptibox/.

[13] Kaj Madsen, Hans Bruun Nielsen, Jacob Søndergaard: *Robust Subroutines for Non-linear Optimization*. IMM, Department of Mathematical Modelling, DTU, 31. 1. 2002.

[14] Gene H. Golub, Charles F. van Loan: *Matrix Computations*. The John Hopkins University press, Third edition 1996

[15] The SMIS framework is available at the homepage: http://www.imm.dtu.dk/~km/.