Space Mapping Toolbox Manual

Version 1.0 by Jacob Søndergaard*
January 2003

Version 2.0 by Pernille Brock
September 2004

This manual describes a Matlab toolbox with space mapping optimization algorithms and test problems. The theoretical background for the algorithms in this toolbox are given in [9] and [4]. The earlier toolbox version 1.0 has been augmented with a new space mapping algorithm, and the new version 2.0 of the toolbox is covered by this manual.

The problems to be solved by the optimization algorithms in this toolbox have two models available: One model denoted the *fine model*, being the model of primary interest, and the other denoted the *coarse model*. The fine model is often expensive to evaluate, though this is not always the case with the simple test problems in this toolbox. It is expected that the coarse model somehow resembles the behaviour of the fine model. Further, it is expected that the coarse model is cheaper to evaluate than the fine model, and therefore it is most likely less accurate than the fine model.

The optimization algorithms employ the coarse model in the search for the fine model minimizer. This is done through a parameter mapping, the so-called space mapping, which in effect makes the coarse model behave as the fine model. We call this combination of the space mapping and the coarse model, the mapped coarse model or the surrogate model. Hence, in the space mapping technique, this mapped coarse model is to take the place of the fine model in search for a minimizer of the latter. For a more thorough introduction to the space mapping technique see [1].

This manual is divided into three sections. The first section introduces the test problems, and the second section introduces the algorithms. Both sections provide a brief description of the Matlab interface. The last section consists of two small examples of running the software.

We should note here that the optimization algorithms in this toolbox rely on the Matlab optimization toolbox [7] in order to run. The test problems

*Please direct communication to: Kaj Madsen, Informatics and Mathematical Modelling, Technical University of Denmark, DK-2800 Lyngby. Email: `km@imm.dtu.dk`

do not require the Matlab optimization toolbox. The toolbox version 1.0 has been developed with Matlab version 6.5 (R13), and version 2.0 has been upgraded to work with the latest version of Matlab 7.0 (R14). Though the toolbox should work with other recent versions of Matlab.

1 Space Mapping Optimization Test Problems

We first describe the common interface to the models, and thereafter we briefly introduce the individual test problems.

1.1 Interface

The definitions of the test problems are stored in the function `smprob`.

To obtain a structure for a given problem the call is

```
[prob, opts] = smprob(num, opts)
```

where the inputs are

`num`	the number of the wanted problem (see below),
`opts`	options for space mapping optimization algorithms (see `smopts`),

the outputs are

`prob`	structure with the problem definition,
`opts`	modified options.

The second input argument is optional, and is meant for the case where the user want to provide alternative default options, instead of those provided by `smopts`. The `opts` structure returned from `smprob` contains the problem specific parameters like initial trust region size etc.

The test problems are:

Tranmission Line Transformer problems
1 - two-section impedance transformer (TLT2)
2 - seven-section impedance transformer (TLT7)

Piston problem
3 - piston simulator (PISTON)

Rosenbrock problem
4 - rosenbrock function, with linear transformation (ROSEN)

Parallel Resonator problems
5 - exact linear mapping (RLCA)
6 - exact non-linear mapping (RLCB)
7 - inexact non-linear mapping (different topology) (RLCC)
8 - inexact non-linear mapping (RLCD)

Quadratic functions
9 - quada (coarse responses shifted up) (QUADA)
10 - quadb (coarse responses shifted down) (QUADB)

The test problems are placed in separate directories in the toolbox. In order to access the problems, the path variable of Matlab is *automatically* modified when first interfacing the test problems through `smprob`. This modified path variable is temporary for the session. If the changes should be permanent use Matlabs `pathtool` to perform the changes.

1.2 The Test Problems

We now briefly introduce the individual test problems. But first some general comments:

Even though the optimization methods in the toolbox are general for all norms, the test problems presented here are posed as minimax problems. For minimization problems in the ∞-norm, the test problem is still posed as a minimax problem and solved by a minimax optimization algorithm, but the original vector function $\tilde{f}$ is extended to $f = \left(\begin{smallmatrix} \tilde{f} \\ -\tilde{f} \end{smallmatrix} \right)$.

Unless explicitly noted in the description the space mapping (using the usual formulation) is not perfect, hence $p(x^*) \neq z^*$. As described in [9, Chapter 4], this condition is critical for the success of the original space mapping algorithms, see also Section 2 below.

All test examples are continuously differentiable in their parameters.

1.2.1 TLT2

The problem TLT2 concerns the design of a twosection capacitively-loaded $10 : 1$ impedance transformer. The exact physical origin of the problem is described in [1].

The designable parameters are the physical lengths of the two transmission lines. Eleven frequency points are simulated per sweep. The objective is to minimize the maximum input reflection coefficient over all simulated frequencies. The design specifications are that all input reflection coefficient responses should be below 50%.

Formally, the fine model response function is $f : \mathbb{R}^2 \mapsto \mathbb{R}^{11}$ and the specifications are

$$H(f(x)) = \max_{j=1}^{11}\{f_j(x)\} \leqslant 0.50\,.$$

The coarse model is as the fine model, except that coupling effects (modelled by capacitors) are not modelled.

1.2.2 TLT7

The TLT7 problem concerns the design of a seven-section capacitively-loaded impedance transformer. The exact physical origin of the problem is described in [1].

The designable parameters are the physical lengths of the seven transmission lines. 68 frequency points are simulated per sweep. The objective is to minimize the maximum input reflection coefficient over all simulated frequencies. The design specifications are that all input reflection coefficient responses should be below 7%.

Formally, the fine model response function is $f : \mathbb{R}^7 \mapsto \mathbb{R}^{68}$ and the specifications are

$$H(f(x)) = \max_{j=1}^{68} \{f_j(x)\} \leqslant 0.07 \, .$$

The coarse model is as the fine model, except that coupling effects (modelled by capacitors) are not modelled.

1.2.3 PISTON

The PISTON problem is a data fitting problem, where a piston simulator should be fitted to a given target response. Here the piston simulator is a model which calculates the pressure over time at an oil producing one-dimensional well, relative to a fixed injection pressure. The target response is the fine model evaluated for a certain set of parameters, so the match of the model to the target response is exact at the optimal parameters. Because of this, we have chosen to formulate the problem as solving the nonlinear equations $f(x) = 0$ using the L_∞ merit function.

The fine model is a piston model with six sections of different reservoir permeabilities along the shaft of the well. Two of the six reservoir permeabilities are chosen as designable parameters. The coarse model is a piston model with two sections of different reservoir permeabilities along the shaft of the well. Both permeabilities in the coarse model are considered designable parameters. For both models, 20 simulation times are simulated per model evaluation.

Let the model response be $\tilde{f} : \mathbb{R}^2 \mapsto \mathbb{R}^{20}$ and let the target response be $y \in \mathbb{R}^{20}$. In the implementation we use the minimax merit for the nonlinear equations, instead of the L_∞ merit, by introducing the residuals $f = \binom{\tilde{f}-y}{y-\tilde{f}}$.

Formally, the deviations of the fine model response and the specifications are $f : \mathbb{R}^2 \mapsto \mathbb{R}^{40}$, and the optimization problem is

$$\min_x \max_{j=1}^{40} \{f_j(x)\} \, .$$

The deviation of the coarse model to the specifications is defined equivalently.

The PISTON problem is provided by Poul Erik Frandsen from Ticra Engineering Consultants, Copenhagen, Denmark.

1.2.4 ROSEN

The ROSEN problem involve solving the Rosenbrock equations, $f(x) = 0$, where $\tilde{f} : I\!R^2 \mapsto I\!R^2$,

$$\begin{aligned} f_1(x) &= 10 * (x_2 - x_1^2) \\ f_2(x) &= 1 - x_1 \end{aligned}$$

We formulate the problem as a minimax problem, by defining the fine model response function as $f = (f_1, f_2, -f_1, -f_2)^T$. Hence the problem is $\min_x \max_{j=1}^4 \{f_j(x)\}$.

We define the coarse model response as a linear transformation of the fine model response. Hence, $c(z) = f(Az + b)$, where

$$A = \begin{bmatrix} 1 & 2 \\ 5 & 0 \end{bmatrix}, \qquad b = \begin{bmatrix} -3 \\ 1 \end{bmatrix}.$$

We note that the space mapping between the coarse and the fine model is exact linear:

$$p(x) = A^{-1}(x - b),$$

since A is invertible.

It is easy to see that the mapping is perfect, i.e. $p(x^*) = z^*$, for this problem as the responses of both models vanish in their optimum.

1.2.5 RLC

The RLC problem concerns design of parallel RLC lumped resonators.

The coarse model is a parallel RLC lumped resonator with three designable parameters. 15 frequency points are simulated per sweep. The objective is to minimize the maximum deviation between the input reflection coefficient and some design specifications over all simulated frequencies. The specifications consists in a passband at the center frequencies and a stopband at all other frequencies.

The problem has four fine models that also model a parallel RLC lumped resonator, but the fine models also have some parasitic elements. The fine models are related to the same design problem (i.e. the same specifications) as the coarse model.

Here are the characteristics of the differences between the models:

$RLCA$: The fine model has an exact linear mapping to the coarse model.

$RLCB$: The fine model has an exact nonlinear mapping to the coarse model.

$RLCC$: The fine model has an inexact non-linear mapping (different topology) to the coarse model.

$RLCD$: The fine model has an inexact non-linear mapping to the coarse model.

The deviation of the fine model response to the specifications is $f : I\!R^3 \mapsto I\!R^{15}$, the problem is $\min \max_{j=1}^{15}\{f_j(x)\}$. The deviation of the coarse model to the specifications is defined equivalently.

1.2.6 QUAD

The QUAD problems, QUADA and QUADB, involve three quadratic functions. The fine model response is $f : I\!R^2 \mapsto I\!R^3$, $f = (f_1, f_2, f_3)^T$, where

$$\begin{aligned} f_1(x) &= 0.5\,x_1^2 + .1\,x_2^2 - 2\,x_2 - 2 \\ f_2(x) &= 0.2\,x_1^2 + 0.1\,x_2^2 + 2\,x_2 - 2 \\ f_3(x) &= 0.1\,x_1^2 - 3\,x_1 + 0.2\,x_2^2 - 2\,. \end{aligned}$$

The fine model is the same for both QUADA and QUADB.

The coarse model for QUADA is $c(z) = f(z + 0.1) + 0.1$ and the coarse model for QUADB is $c(z) = f(z - 0.1) + 0.1$.

The simple shift in the response functions causes that for neither problem the space mapping is perfect, $p(x^*) \neq z^*$.

2 Space Mapping Optimization Algorithms

The toolbox contains six algorithms based on space mapping technique. Two algorithms, namely `smo` and `smon`, are related to the original space mapping formulation. Three algorithms, namely `smh`, `smho` and `smhc`, are so-called hybrid space mapping algorithms, combining space mapping technique with classical Taylor based optimization. The last algorithm, `smis`, is a space mapping algorithm with interpolating surrogates. The first five algorithms employ only input mappings, while the latter employs both input and output mappings.

2.1 Interface

The algorithms have a common interface:

```
[xk, fk, Hfk, stop, trace] =

       smx(H, fine, coarse, x0, A, b, eq, opts, P1, P2, ...)
```

where `smx` is one of the following

`smo`	original space mapping,
`smon`	new space mapping formulation,
`smh`	hybrid space mapping,
`smho`	hybrid space mapping with orthogonal steps.
`smhc`	hybrid space mapping with response correction,
`smis`	space mapping with interpolating surrogate model.

The mandatory arguments of the algorithms are the merit function `H`, the file handles to the `fine` and the `coarse` model and a starting point `x0`. Any parameters that should be passed directly to the fine and the coarse model can be specified in the place of `P1, P2, ...`.
The `smis` algorithm has an extra input argument `mmap`, which is the number of response functions for which the mapping parameter extraction is performed. In case of minimization in the ∞-norm, `mmap` is the length of $\tilde{f}$, otherwise the length of f.

The algorithms return the best iterate `xk`, the fine model response `fk` at `xk`, the merit `Hfk` of the response and the reason for stopping the algorithm `stop`. A fifth output option is a `trace` structure which contains a trace of important values gathered in the iteration process. The `trace` structure is mandatory, if the results should be visualized by the function `smplot`.

The user may supply the algorithm with linear constraints $\mathtt{A}\cdot\mathtt{x} \leqslant \mathtt{b}$, where the first `eq` rows are equality constraints. If the problem is unconstrained empty matrices may be passed. The toolbox provides no check for consistency of the constraints.

The constraints only apply to fine model parameters, e.g. in the trust region subproblems, hence the coarse model may be evaluated at any $z \in I\!R^n$. The only exception is in the initial phase of the algorithms, where the coarse model parameters are constrained in the search for the coarse model minimizer, z^*. This is needed because the first iterate is the coarse model minimizer, $x_0 = z^*$, i.e. the first point where the fine model is evaluated.

Specific options determining the behaviour of the optimization algorithms are passed in the structure `opts`. The default values for the structure `opts` are obtained from the function `smopts`. The structure contains all options used to determine the behaviour of the specific algorithms. If an empty matrix is passed instead of a structure, the default values are obtained from `smopts`.

The function `smopts` is called as follows

```
opts = smopts(key1, value1, key2, value2, ...)
opts = smopts(opts, key1, value1, key2, value2, ...)
```

See the source file for a more complete description of the options, than is presented in this manual. An existing structure with options can be passed as input to override default values. Individual default options can be overwritten by specifying new key-value pairs as input arguments.

We mentioned some of the more important options here:

Most of the optimization algorithms in this toolbox rely on trust region methodology to enforce convergence. Control of the trust region is determined by a number of options. The most important is `dx`, the initial trust region size, which is problem dependent. Refer to `smopts` for the other options related to the trust region handling.

The accuracy of the optimization result is determined by the option `epsilon`. The algorithms stop if the relative step length or trust region size becomes smaller than `epsilon`. Another option controlling when the algorithms stops is `kmax` which determines the maximum allowed number of fine model evaluations. So the algorithms are stopped if one of the following conditions are satisfied

$$\begin{aligned} k &\geqslant \texttt{kmax} \\ \|h\| &\leqslant \texttt{epsilon}(1 + \|x_k\|) \\ \mathrm{dx} &\leqslant \texttt{epsilon}(1 + \|x_k\|) \end{aligned}$$

where k counts fine model function evaluations, $\|h_k\|$ is the step length, dx is the size of the trust region and $\|x_k\|$ is the norm of the current iterate.

The last criterion is there to avoid an unnecessary extra iteration, solving a trust region problem with a trust region size that is less than the minimum allowed step length.
In the `smis` algorithm an additional stopping criterion is

$$H(f(x_k)) - H(f(x_k + h) \quad \leqslant \quad \texttt{epsilon}(1 + |H(f(x_k))|)$$

requiring a sufficient decrease in the objective function to continue.

A quick way to test-run the algorithms is through the `smrun` function, which is called by

```
[p, trace, opts] = smrun(num, algo, opts)
```

where `num` is the number of the test problem (see p. 3), and `algo` is the number of the algorithm to test:

1	SMO orginal space mapping formulation
2	SMON new space mapping formulation with mapped coarse model
3	SMH space mapping hybrid algorithm
4	SMHO space mapping hybrid algorithm with orthogonal steps
5	SMHC space mapping hybrid algorithm with response correction
6	SMIS space mapping algorithm with interpolating surrogate model
7	direct optimization of the fine model, 1st order method
8	direct optimization of the fine model, 2nd order method
9	direct optimization of the coarse model, 2nd order method

If the third input argument `opts` is missing in the call to `smrun`, the default options are used.
The variables `p` and `opts` are the structures obtained from `smprob`, and `trace` is the trace of the optimization process obtained by calling one of the algorithms. See Section 3 for examples showing the content of `trace` and `p`. For the choices 8 and 9 of `algo` there cannot be produced a trace variable.

A quick way to plot some of the results from algorithm 1-7 is through the `smplot` function, which is called by

```
smplot(p, trace, opts)
```

where `p` is the structure obtained by the function `smprob` (see p. 2), and `trace` is the structure containing the trace of the optimization process. `opts` is the structure defining the options, if the default values have been used in the algorithm, the argument can be omitted. For algorithm 1-7 at least three plots are provided:

1	The performance measured in the merit function $H(f(x_k)) - H(f(x^*))$
	The performance measured in the iterate $\|x_k - x^*\|_2$
2	The trust region radius and the step length
3	The fine model response function at x^* and at the best iterate

If the optimizer x^* is not available, the performance is measured according to the best solution available. Additional plots depend on the problem type and the optimization algorithm. For algorithm 1-5 the convergence of the space mapping compared to the perfect space mapping is plotted, for algorithm 3-5 also the transition parameter, and for algorithm 5 furthermore the response correction factors. For algorithm 6 in case of a two-dimensional problem the approximation errors from using a linear Taylor model and from using the optimal surrogate model are visualized in a three-dimensional figure.
For all algorithms a table with the level of accuracy $(H(f(x_k)) - H(f(x^*)))$ for each iteration is displayed in the Matlab command window.

Since the `smplot` function relies on the existence of the trace variable, it can not be used in case `algo` is 8 or 9.

Before we describe the algorithms we first give a brief theoretical overview of space mapping theory.

2.2 Theoretical Overview

The main problem consists in finding the minimizer x^* (assumed unique) of the fine model,

$$x^* = \arg\min_x H(f(x)) \; , \tag{1}$$

where $f : I\!\!R^n \mapsto I\!\!R^m$ is the vector response function representing the fine model, and $H : I\!\!R^m \mapsto I\!\!R$ is a convex merit function, usually a norm. We denote x^* the *fine model minimizer*. We note that f is assumed so expensive that using a classical Taylor based optimization method is infeasible, so finding x^*, or an approximation to it, is nontrivial.

A related problem is finding the minimizer z^* (assumed unique) of the coarse model,

$$z^* = \arg\min_{z \in I\!\!R^n} H(c(z)) \, ,$$

with $c : I\!\!R^n \mapsto I\!\!R^m$ being the vector response function representing the coarse model. We denote z^* the *coarse model minimizer*. Since c is assumed cheap to evaluate the gradient is available (e.g. by finite difference approximation), hence finding z^* is a trivial problem for a classical Taylor based optimization method.

For the space mapping algorithms with only input mappings, the space mapping $p : I\!R^n \mapsto I\!R^n$ linking the parameter space of the fine and the coarse model is usually defined as solving the so-called *parameter extraction problem*,

$$p(x) = \arg\min_{z \in I\!R^n} \|c(z) - f(x)\|_2 .$$

This definition of the space mapping may lead to nonuniqueness in the parameter extraction problem, so several alternative definitions are available in the toolbox:

Regularization with regard to the distance to z^*,

$$p_\lambda(x) = \arg\min_z \left\{ \; (1-\lambda)\,\|c(z) - f(x)\|_2^2 + \lambda\,\|z - z^*\|_2^2 \; \right\} \; , \tag{2}$$

for some value of $0 \leqslant \lambda < 1$.

Regularization with regard to the distance to x,

$$p_\lambda(x) = \arg\min_z \left\{ \; (1-\lambda)\,\|c(z) - f(x)\|_2^2 + \lambda\,\|z - x\|_2^2 \; \right\} \; , \tag{3}$$

for some value of $0 \leqslant \lambda < 1$.

Regularization using gradient information,

$$p_\lambda(x) = \arg\min_z \left\{ \; (1-\lambda)\,\|c(z) - f(x)\|_2^2 + \lambda\,\|c'(z) - f'(x)\|_F^2 \; \right\} \; , \tag{4}$$

for some value of $0 \leqslant \lambda < 1$. In the optimization algorithms $f'(x)^T$ is approximated by a secant approximation $D \in I\!R^{m \times n}$ during iterations, so this D is used instead of the true Jacobian matrix in (4).

In the implementation the above regularized problems are solved as normal nonlinear least-squares problems, exemplified here by (2),

$$p_\lambda(x) = \arg\min_z \left\| \begin{array}{c} \sqrt{(1-\lambda)}\,(c(z) - f(x)) \\ \sqrt{\lambda}\,(z - z^*) \end{array} \right\|_2^2 \; ,$$

for some value of $0 \leqslant \lambda < 1$. As the Jacobian of c is assumed available, the gradient for this least-squares objective function is available, at least for (2) and (3). In the case of (4) though, the gradient of the least-squares objective function depends on the second derivatives of c. So, as second order information is not available, the gradient of the least-squares objective function is found by finite difference approximation.

The space mapping algorithm `smis` employs both input and output mappings. The space mapping-based interpolating surrogate model $s : I\!R^n \mapsto I\!R^m$ is defined by input mappings $P_i : I\!R^n \mapsto I\!R^n$, transforming the parameter space of the coarse model, and by output mappings $O_i : I\!R^m \mapsto I\!R$, transforming the coarse model responses, for $i = 1, \ldots, m$. The output mappings ensure exact interpolation between the fine model responses and the surrogate model responses in the expansion point. Each of the responses are considered individually.

The space mapping is hereby linking the coarse model and the surrogate model, and is defined by the so-called mapping parameters. These are found by solving m mapping parameter extraction problems, formulated as nonlinear least-squares problems,

$$p_k = \arg\min_p \left\| \begin{matrix} w_1\left(s(x_1, p) - f(x_1)\right) \\ \vdots \\ w_1\left(s(x_{k-1}, p) - f(x_{k-1})\right) \\ w_2\left(s'_x(x_k, p) - f'_x(x_k)\right) \end{matrix} \right\|_2^2$$

where w_1 and w_2 are weighting factors. The formulation aims for alignment of the fine model and the surrogate model at the previous iteration points, and also includes gradient information at the best iterate. Since second order information for the coarse model is not available, the gradient of the least-squares objective function is found by finite difference approximation. It should also be mentioned, that the mapping parameter extraction problems may be nonunique.

With this theoretical introduction we are now in a position to introduce the algorithms.

2.2.1 The Original Space Mapping Formulation

The original space mapping technique involves solving the nonlinear equations

$$p(x) = z^*\,,$$

for $x \in \mathbb{R}^n$. The algorithm implemented in the toolbox function `smo` addresses this problem, by solving the least-squares formulation of the problem,

$$\min_{x\in\mathbb{R}^n} \|p(x) - z^*\|_2\,. \tag{5}$$

Another space mapping technique, equivalent with the original formulation in some ways, is to minimize the mapped coarse model,

$$\min_{x\in\mathbb{R}^n} H(c(p(x)))\,. \tag{6}$$

The algorithm implemented in the toolbox function `smon` solves this problem.

The solutions of (5) and (6) are not necessarily the solution x^* of the main problem (1). In fact we can only be certain that the solution is x^* if the space mapping is perfect, $p(x^*) = z^*$.

In the description of the test problems above it is stated which of the test problems that have a perfect mapping. Due to this drawback, the results of the functions `smo` and `smon` are not directly comparable with the functions implementing the hybrid space mapping framework, described next.

2.2.2 Hybrid Space Mapping Algorithms

The toolbox contains three functions, namely `smh`, `smho` and `smhc`, implementing the hybrid space mapping framework described in [6]. Basically the algorithms rely on a model of the form

$$s(x) = w\, c(p(x)) + (1 - w)\, l(x)$$

where $0 \leqslant w \leqslant 1$ is a transition parameter, $c \circ p$ is a mapped coarse model and l is linear Taylor model of the fine model. An exception is the algorithm in `smhc` which uses a form of the mapped coarse model where the responses are corrected to match the fine model using a secant method.

All the algorithms start with $w = 1$ and end with $w = 0$, provided enough iterations. Thereby a switch from the mapped coarse model to the linear Taylor model takes place.

With k being the iteration counter, it is proven in [6] that the main condition for convergence of this class of algorithms is that

$$w_k = \mathrm{dx}_k \cdot o(1)$$

where dx_k is the size of the trust region and $o(1) \to 0$ for $k \to \infty$.

Two of the algorithms, namely `smh` and `smhc`, use a gradual switching strategy, whereas the third algorithm `smho` switches abruptly from $w = 1$ to $w = 0$ at a certain point in the iteration process.

The algorithms use linear Taylor model with secant approximations to the derivatives for both the space mapping p and l. So the last stage of the three algorithms involves sequential linear programming, where the linear model has inexact derivatives. To help speed up the convergence, the options `dofinitediff` and `maxuphill` (refer to the source of `smopts`) can force the algorithms to correct the linear model by a finite difference approximation. Further we should note that the option `initd` controls the way that the initial approximation to the derivatives of the fine model is obtained.

2.2.3 The Space Mapping Algorithm with Interpolating Surrogate model

This new method employs both input and output mappings to construct an interpolating surrogate model, with the ith response given by

$$s_i(x) = \alpha_i \left(c_i(P_i(x)) - c_i(P_i(x_k))\right) + f_i(x_k)$$

where x_k is the best iterate so far. The ith input mapping P_i is a linear transformation of the design parameters

$$P_i(x) = A_i x + b_i$$

The output mapping parameter α_i transforms the coarse model response. The formulation ensures exact alignment of the surrogate model responses and the fine model responses in the expansion point x_k. The mapping parameters for the ith response are gathered in the vector p_i, which has n^2+n+1 elements. The total number of unknown mapping parameters in each main iteration is then $m \cdot (n^2+n+1)$.

To speed up the calculations the option `diag_a` can be set to 1, which makes the input mapping parameter matrix A_i a diagonal matrix for all responses. The total number of unknown mapping parameters is now $m \cdot (2n+1)$. The default value of `diag_a` is 0.

It must be mentioned, that in case of ∞-norm minimization, the mapping parameter extraction is only performed for $mmap = m/2$ responses, since the other half is identical due to the symmetry of the fine model function $f = \left(\begin{smallmatrix} \tilde{f} \\ -\tilde{f} \end{smallmatrix}\right)$.

The residual formulation includes weighting factors. The value of the weighting factor w_1 is relative to the weighting factor w_2, which has the fixed value 1. The ratio between w_1 and w_2 is set by the option `weight_f`, which has the default value 1. The value of `weight_f` is recommended to be ≤ 1. Both weighting factors are furthermore normalized according to the residual elements.

Another option concerning the `smis` algorithm is `eta`, which determines the step length used in the forward difference approximation for the gradient of the residual, when solving the mapping parameter extraction problems. The recommended value is `eta` $= 10^{-5}$.

2.3 The Optimization Algorithms

2.3.1 SMO

The function `smo` implements the original space mapping technique solving the problem in (5) using a trust region secant method. The secant method involves a linear Taylor model of the space mapping with a secant approximation to the Jacobian matrix.

2.3.2 SMON

The function `smon` implements the alternative space mapping technique solving the problem (6), using a trust region method with sequential linear approximations to p by a secant method.

2.3.3 SMH

The function `smh` implements the hybrid space mapping algorithm, with a gradual switching between the mapped coarse model and the linear Taylor model of the fine model.

The control of w is determined by the options `w_min`, `w_reduce` and `max_w_not_reduced`. If either a proposed step is not accepted or if the number of iterations where w has not been changed reaches $\max\{n, \texttt{max\_not\_reduced}\}$ then w is updated. The updating formula is

$$w_{k+1} = w_k \cdot \texttt{w\_reduce} \cdot \min\{\mathrm{dx}_{k+1}, 1\}\,,$$

where dx is the size of the trust region. If w by updating gets below `w_min` then w is set to zero.

2.3.4 SMHO

The function `smho` implements a hybrid space mapping algorithm with orthogonal updating steps of the space mapping approximation.

If the space mapping fails within the first n iterations the algorithm evaluates the fine model at a step in a direction orthogonal to previous steps, this is in order to improve the quality of the space mapping secant approximation. Which of the orthogonal directions that is chosen and the length of the step in that direction can be controlled by the options `ortho_met`, `ortho_scale_type` and `ortho_scale`. If a single orthogonal step is not sufficient, further steps are taken, until the fine model has been evaluated at most n times. Thereafter the algorithm switches to a linear Taylor model of the fine model.

If the space mapping steps are successful the algorithm keeps taking space mapping steps until at most $\texttt{max\_w\_not\_reduced} + n$ steps have been taken. Thereafter the algorithm is forced to switch to the linear Taylor model of the fine model.

2.3.5 SMHC

The function `smhc` implements a hybrid space mapping algorithm with response correction of the mapped coarse model.

The combined model for this algorithm is

$$s_k(x) = w_k(g_k \,.\!* \; [c(p_k(x)) - c(p(x_k))] + f(x_k)) + (1 - w_k)l_k(x)$$

where $g \in \mathbb{R}^m$ are the correction factors and $*$ is element-wise multiplication.

The correction factors are found by the secant update

$$g_{k+1}^{(j)} = \frac{f^{(j)}(x_{k+1}) - f^{(j)}(x_k)}{c^{(j)}(p(x_{k+1})) - c^{(j)}(p(x_k))}\,, \quad j = 1, \ldots, m\,,$$

where the superscript (j) indicates the jth element of the vector.

The transition parameter w is controlled as in the `smh` algorithm described above.

2.3.6 SMIS

The function `smis` implements a space mapping algorithm with interpolating surrogate model. The interpolating surrogate model employs individual linear input and output mappings for each response, and is given by:

$$s_k(x) = \begin{bmatrix} \alpha_1\,(c_1(P_1(x)) - c_1(P_1(x_k))) + f_1(x_k) \\ \vdots \\ \alpha_m\,(c_m(P_m(x)) - c_m(P_m(x_k))) + f_m(x_k) \end{bmatrix}$$

where x_k is the best iterate so far.

2.4 Auxiliary Functions

Both the main directory and the directory `private` contains a number of auxiliary functions. An overview of directories and functions is provided in the ps-file `overview`. We briefly introduce the most important ones.

`parameter_extraction` For a given x the function solves the parameter extraction problem, determining $p(x)$. The user can choose between four different space mapping definitions through the option `petype`. The starting point for the parameter extraction problem is specified by the option `pestart`, four possibilities exist. Further, for the regularization formulations, the value of λ can be specified by the option `lambda`.

`combined_model` For a given x the function calculates the response of the combined model $s_k(x) = wc(p_k(x)) + (1 - w)l_k(x)$, where $c(p_k(x))$ is the mapped coarse model (see `mapped_model` below) and $l_k(x) = D(x - x_k) + f(x_k)$ is a linear Taylor model of the fine model.

`combined_corrected_model` For a given x the function calculates the response of the combined model $s_k(x) = w(g \,.\!*\, (c(p_k(x)) - c(p(x_k))) + f(x_k)) + (1 - w)l_k(x)$, where the first part is the response corrected model, with $c(p_k(x))$ being the mapped coarse model (see `mapped_model` below), and $l_k(x) = D(x - x_k) + f(x_k)$ is a linear Taylor model of the fine model.

`mapped_model` For a given x the function calculates the response of the mapped coarse model $c(p_k(x))$, where $p_k(x) = B(x - x_k) + p(x_k)$ is a linear Taylor model of the space mapping.

`mapping_parameter_extraction` For a given x the function solves the parameter extraction problem, determining $p(x)$. The user can choose between four different space mapping definitions through the option `petype`. The starting point for the parameter extraction problem is specified by the option `pestart`, four possibilities exist. Further, for the regularization formulations, the value of λ can be specified by the option `lambda`.

`surrogate_model` For a given x the function calculates the response vector of the surrogate model $s(x, p)$, where p is a holding input and output mapping parameters for all m responses. The function is only used in the minimization of the surrogate model, giving the next iterate.

`surrogate_model_i` For a given x the function calculates the ith response of the surrogate model $s_i(x, p_i)$, where p_i is holding the input and output mapping parameters for the ith response. The function is used only in the solving of the mapping parameter extraction problems.

`smtrlinear` For a given x the function finds a minimizer of a given linear model subject to linear trust region constraints (infinity norm trust region) and, if any, user provided linear constraints. Formally the problem solved is

$$\begin{array}{lll} \min_x H(l_k(x)) & & \\ \text{s.t.} \quad \|x - x_k\|_\infty & \leqslant & \mathrm{dx}_k \\ \qquad\quad Ax & \leqslant & b \end{array}$$

where the first `eq` rows of the user constraints are equality constraints.

`smdirect` Direct, classical Taylor based optimization. Solves the problems of the general type

$$\begin{aligned} &\min_x H(s(x)) \\ &\text{s.t.} \quad Ax \leqslant b \end{aligned} \tag{7}$$

where s is a nonlinear vector response function. The first `eq` rows of the linear constraints are equality constraints. Exact gradient information is assumed available.

`direct` Direct, classical Taylor based optimization with inexact gradient information. Solves (7) using a trust region algorithm with secant gradient approximations.

3 Examples

We now give two small examples to show how the toolbox can be used. In these examples the `MINCIN` subroutine [5] is used to solve the minimax subproblems, other results may be obtained when the minimax algorithm in Matlab's Optimization Toolbox [7] is used instead.

3.1 Quick Run

Assume that we want to run the TLT2 problem with the hybrid space mapping algorithm `smh`. The easiest way to do this is by calling `smrun`,

```
[p, trace] = smrun(1, 3);
```

(the semicolon suppresses the display of the output variables). After the algorithm has finished the iteration process we now have a trace variable with the results (the contents of the variable `p` is shown in the next example). The `trace` variable is a Matlab structure. We list the contents from this run:

```
trace =
      k: [1x66 double]
      x: [2x66 double]
      z: [2x66 double]
      f: [11x66 double]
      h: [1x66 double]
     dx: [1x66 double]
      w: [1x66 double]
    rho: [1x65 double]
    obj: [1x66 double]
```

We see that there are nine fields containing the trace of variables in the 66 steps taken by the algorithm. For example the field `trace.f` contains all fine model responses evaluated by the algorithm. In the field `trace.x` are the corresponding fine model parameters. The field `trace.z` contains the space mapped parameters, $z = p(x)$.

Now we could for example check how close the best fine model response found by the algorithm is to the optimal response of the fine model. First the best objective function value found by the algorithm:

```
>> min(max(trace.f))
ans =
         0.455324591088871
```

Then the objective function value of the optimal response:

```
>> max(p.fast)
ans =
          0.45532459108887
```

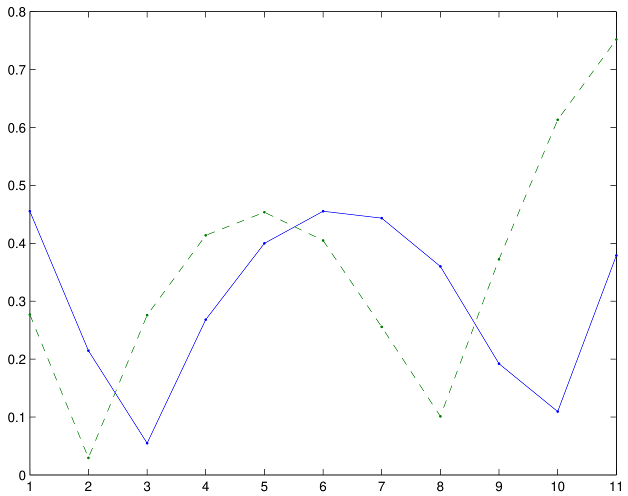
We see that the solutions only differ in the 15th decimal. In fact the difference is around full calculating accuracy.

If we want to plot the initial response (i.e. the response in the starting point $x_0 = z^*$) and the best response found by the algorithm, we first obtain the index of the best response:

```
>> [fmin idx] = min(max(trace.f))
fmin =
         0.455324591088871
idx =
    63
```

Then we plot the responses:

```
>> plot(1:p.m, trace.f(:,idx), '.-', ...
        1:p.m, trace.f(:, 1), '.--')
```



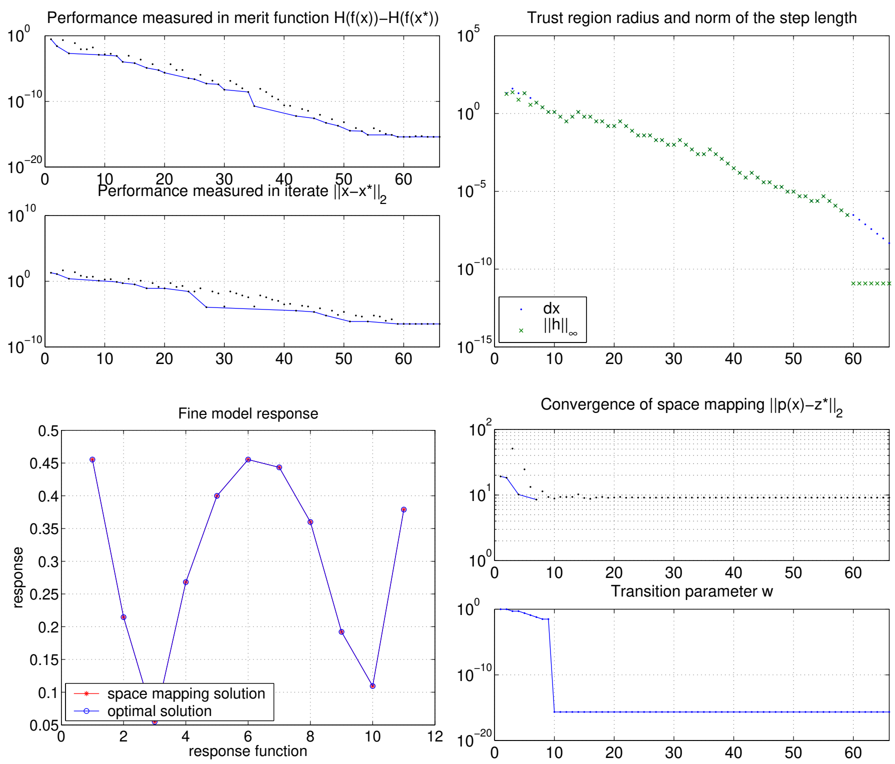
We see that the first response violated the specifications, since the maximum value of the response is above 0.5 (refer to the problem description above).

3.2 Quick plots

A quick way to make some relevant plots of the results contained in the trace variable is by the command

```
smplot(p, trace)
```

For the quick run of the TLT2 problem with the `smh` algorithm we get the following four Matlab figures:



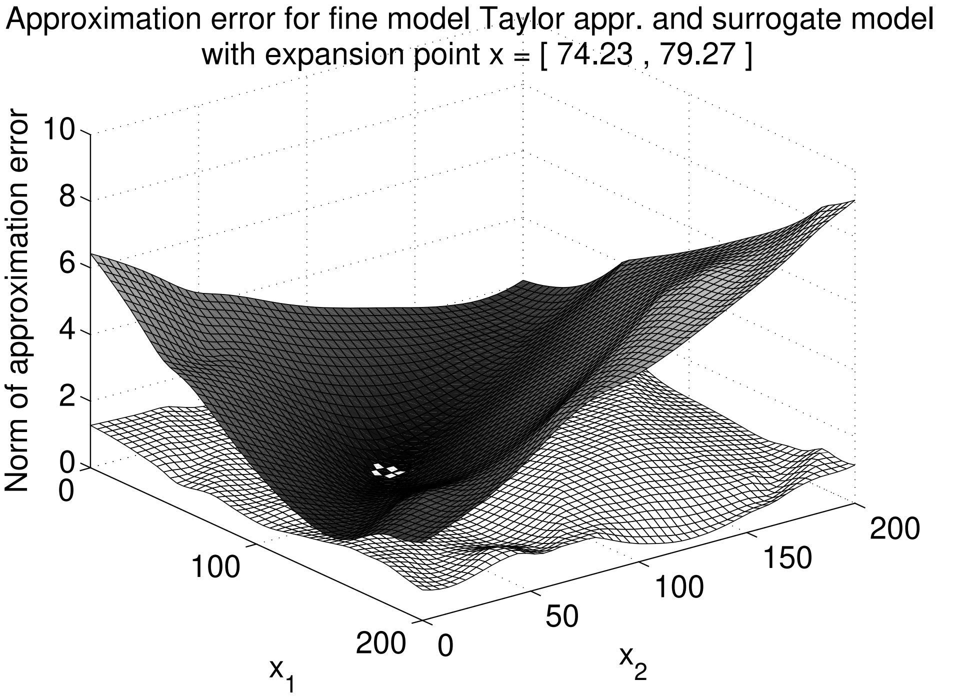
If we instead use the `smis` algorithm on the TLT2 problem, we get convergence with an accuracy of 10^{-13} in 7 iterations, if we run the algorithm with the command:

```
[p, trace, opts] = smrun(1, 6, smopts('weight_f', 0.001));
```

The value of `weight_f` is here changed from the default value 1 to the value 0.001. To produce plots of the results we now type the command:

```
smplot(p, trace, opts)
```

Note that the extra input argument `opts` is added, because we want to use the same options as before.
`smplot` now produces four figures: the first three figures similar to the ones from the `smh` run, but the fourth is replaced by a 3-dimensional plot of the approximation errors:



Furthermore `smplot` for this algorithm produces three more figures showing the norm of the residual vector for each of the 11 response functions as the iterations proceed. Finally the following table is displayed in the Matlab command window:

```
k   Level of accuracy
---------------------
1   2.9662e-01
2   1.7145e-01
3   5.9678e-02
4   1.1477e-04
5   5.0508e-08
6   1.0198e-11
7   1.4766e-14
```

where the level of accuracy is measured by the difference $H(f(x_k)) - H(f(x^*))$ for each iteration.

3.3 Examining a Problem

Now let us examine the structure with the problem returned from `smprob`.

```
>> p = smprob(1)
p =
      xast: [2x1 double]
      zast: [2x1 double]
        x0: [2x1 double]
      fine: @tlt2f
    coarse: @tlt2c
         n: 2
     fopts: [1x1 struct]
         m: 11
      fast: [11x1 double]
         A: [4x2 double]
         b: [4x1 double]
        eq: 0
         H: 'minimax'
```

We see that the problem is a minimax problem (`H = 'minimax'`). Further we see that it is a two-dimensional problem ($n = 2$ and `xast` $= x^* \in \mathbb{R}^2$). There are 11 response functions ($m = 11$ and `fast` $= f(x^*) \in \mathbb{R}^{11}$). The fine model handle refers to the function `tlt2f`.

3.4 Results from the `smis` algorithm

The `smis` algorithm has been tested on some of the test problems available in this toolbox. The performance is shown in the following table, which displays the number of fine model evaluations needed to obtain the given level of accuracy.

Level of accuracy	Test problem no.					
	1	2	3	4	9	10
10^{0}	1	1	1	5	1	1
10^{-1}	3	2	3	5	2	2
10^{-2}	3	3	3	5	2	2
10^{-3}	3	5	4	5	2	2
10^{-4}	4	5	4	6	2	2
10^{-5}	4	6	4	6	2	2
10^{-6}	5	7	5	6	3	3
10^{-7}	5	8	5	6	3	3
10^{-8}	6	-	5	6	3	3
10^{-9}	6	-	6	6	3	3
10^{-10}	6	-	6	6	4	4
10^{-11}	7	-	6	6	4	4
10^{-12}	7	-	7	7	-	-
10^{-13}	7	-	7	7	-	-
10^{-14}	-	-	7	7	-	-
stop	ND	ND	SL	SL	ND	ND

The level of accuracy at the kth iteration is measured by:

$$\text{Level of accuracy} = H(f(x_k)) - H(f(x^*))$$

where the fine model is evaluated once per iteration. The reason for stopping the algorithm is also visible in the table:

ND the algorithm stopped because of insuffiecient decrease in the fine model objective
SL the algorithm stopped because the step length or the trust region radius was too small

Performance results for the remaining algorithms can be seen in [9].

References

[1] M.H. Bakr, J.W. Bandler, K. Madsen, J. Søndergaard, *An Introduction to the Space Mapping Technique*, Optimization and Engineering, vol. 2, no. 4, pp. 369–384, 2001.

[2] M.H. Bakr, W.J.R. Hoefer, *Electromagnetic Synthesis: A Novel Concept in Microwave Design*, 31st European Microwave Conference Dig., London, England, 2001.

[3] M.H. Bakr, W.J.R. Hoefer, *Electromagnetic Synthesis: A Novel Concept in Microwave Design*, IEEE Trans. Microwave Theory Tech., 2003.

[4] P. Brock, *Optimization Using Space Mapping*, Master Thesis No. 54, IMM, DTU, Lyngby, 2004.
The thesis may be downloaded from the department website: `http://www.imm.dtu.dk/`

[5] J. Hald, K. Madsen, *Combined LP and Quasi-Newton Methods for Minimax Optimization*, Mathematical Programming, vol. 20, pp. 49–62, 1981.

[6] K. Madsen, J. Søndergaard, *Convergence of Hybrid Space Mapping Algorithms*, submitted, Optimization and Engineering, 2003.

[7] Matlab^{TM} version 6.5 and Matlab^{TM} Optimization Toolbox version 2.2., The MathWorks, Inc., 3 Apple Hill Drive, Natick MA 01760-2098, 2003.

[8] G. Matthaei, L. Young, E.M.T. Jones, *Microwave Filters, Impedance-Matching Networks, and Coupling Structures*, Norwood, MA, Artech House, p. 545, 1980.

[9] J. Søndergaard, *Optimization Using Surrogate Models — by the Space Mapping Technique*, Ph.D. Thesis, IMM, DTU, Lyngby, 2003.
The thesis may be downloaded from the department website: `http://www.imm.dtu.dk/`