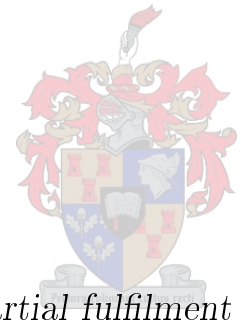
Automated Space-Mapping Framework for Electromagnetic Device Optimisation

by

David William Wolsky



Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Electronic) in the Faculty of Engineering at Stellenbosch University

Supervisor: Prof. D. De Villiers

April 2019

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: April 2019

Abstract

A space-mapping (SM) framework that allows an automated approach to solving computer-aided design (CAD) optimisation problems for electromagnetic (EM) devices is presented. Direct optimisation of detailed, high-fidelity/fine EM models can be computationally expensive and can restrict the adoption of optimisation for large systems.

SM allows the incorporation of low-fidelity/coarse models that are quick to evaluate, without sacrificing the accuracy of results. The SM framework builds up a surrogate model from a coarse model that is aligned programmatically to the fine model. Optimisation is carried out using the surrogate model. If the surrogate is evaluated far away from where the alignment took place, the results may diverge. A trust-region (TR) is introduced as a method of improving the robustness of the framework. The TR governs the bounds of the optimisation space.

Four types of SM are implemented within the automated framework: input, output, implicit and frequency SM. Literature using some of these techniques is investigated, and a detailed analysis on the original SM implementation and a frequency SM approach is included. A basic TR implementation, from literature, is also investigated in detail.

The methodology used to develop the automated framework is explained, and MATLAB implementation details for each stage are discussed. Model alignment and surrogate building for each of the SM techniques are discussed. The user's interface to the TR enhanced SM optimisation system is detailed. The available high-fidelity solvers are FEKO and CST, while those for low-fidelity are AWR-MWS and MATLAB.

A microstrip stub example is used to demonstrate input, implicit and frequency SM. FEKO and AWR-MWS are used for these examples. A microstrip double folded stub filter is taken from literature and used to evaluate the system. This bandstop example has three design variables and is required to meet three S-parameter goals. An additive input and implicit SM approach is chosen to solve this problem. Each iteration is analysed and the SM framework successfully meets specification within four fine model evaluations.

Finally, improvements to the automated framework are presented. A general mathematical model is suggested for unit-testing, and an object orientated design is suggested.

Opsomming

'n Ruimteafbeelding (SM) raamwerk wat 'n outomatiese benadering vir die oplos van rekenaar gesteunde ontwerp (CAD) optimeringsprobleme vir elektromagnetiese (EM) toestelle toelaat word aangebied. Direkte optimering van gedetailleerde, hoëtrou EM modelle kan bewerkingsintensief wees, en kan die gebruik van optimering in groot stelsels beperk.

SM laat die inkorporasie van lae-vertroue/growwe modelle toe wat vinnig is om te evalueer, sonder om die akkuraatheid van die resultate in te boet. Die SM raamwerk bou 'n surrogaatmodel vanaf die growwe model op, wat programmaties belyn word met die hoëtrou/fyn model. Optimering word uitgevoer deur van die surrogaatmodel gebruik te maak. As die surrogaat ver van enige punt waar belyning plaasgevind het geëvalueer word, mag die resultate divergeer. A vertrouegebied (TR) word voorgestel as 'n metode om die robuustheid van die raamwerk te versterk. Die TR beheer die grense van die optimeringsruimte.

Vier tipes SM word geïmplementeer binne die outomatiese raamwerk: intree, uittree, implisiete en frekwensie SM. Literatuur wat gebruik maak van party van die tegnieke word bestudeer, en 'n gedetailleerde analise van die oorspronklike SM implementasie en 'n frekwensie SM implementasie word ingesluit. 'n Basiese TR implementasie, van die literatuur, word ook in detail ondersoek. Die metodiek wat gebruik is om die outomatiese raamwerk te ontwikkel word verduidelik, en MATLAB implementeringsdetails vir elke stadium word bespreek. Die gebruikerskoppelvlak na die TR-verbeterde SM optimerings stelsel word bespreek. Die beskikbare hoëtrou oplossers is FEKO en CST, terwyl, vir die growwe modelle, AWR-MWS en MATLAB gebruik word.

'n Mikrostrook stomplyn voorbeeld is gebruik om die gebruik van intree, implisiete en frekwensie SM toe te lig. FEKO en AWR-MWS word vir hierdie voorbeelde gebruik. 'n Mikrostrook dubbelgevoude stomplyn filter word van die literatuur geneem en gebruik om die stelsel te evalueer. Hierdie bandstop voorbeeld het drie ontwerpsveranderlikes en daar word verwag dat drie S-parameter doelfunksies bereik word. 'n Optellings intree en implisiete SM benadering is gekies om hierdie probleem op te los. Elke iterasie is geanaliseer en die SM raamwerk haal suksesvol die spesifikasie binne vier fyn model evaluasies.

Ten slotte word verbeterings aan die outomatiese raamwerk voorgelê. 'n

Algemene wiskundige model word voorgestel vir eenheidstoetse, en toekomstige werk wat objek georiënteerde ontwerp voorstel word bespreek.

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations:

- My supervisor Prof. D. De Villiers who was always motivating an encouraging even when I didn't feel motivated and encouraged.
- Altair Development S.A. for providing the time and space to pursue this project.

Contents

List of Figures

List of Tables

Nomenclature

Optimisation Varaibles

$\boldsymbol{x}$	Design variables
$\boldsymbol{x}^*$	Optimal solution
$\bar{\boldsymbol{x}}$	Estimate of the optimal solution ($\boldsymbol{x}^*$)
$\boldsymbol{f}$	Vector of frequencies
N_n	Number of design variables
N_m	Number of output responses

Loop terminology

i	Iteration step for main loop
N_i	Maximum number of iterations for various loops
k	Iteration step for trust-region loop
N_k	Maximum number of iterations for various loops

Space Mapping Terms

$\boldsymbol{R}$	Response of a model
$\boldsymbol{P}$	Mapping relating two or more sets of model parameters
$\boldsymbol{D}$	Original space-mapping model set
$\boldsymbol{A}$	Multiplicative output space mapping matrix
$\boldsymbol{B}$	Multiplicative input space mapping matrix
$\boldsymbol{c}$	Additive input space mapping vector
$\boldsymbol{G}$	Multiplicative implicit space mapping matrix
$\boldsymbol{d}$	Additive output space mapping vector
$\boldsymbol{p}$	Bound for implicit space mapping
$\boldsymbol{F}$	Frequency space mapping matrix
N_q	Number of implicit/preassigned variables
N_c	Number of fine models available.

Space Mapping Subscripts

f	Fine or high fidelity model
c	Coarse or low fidelity model
s	Surrogate model
p	Implicit or preassigned parameter

Trust-region Terms

$\boldsymbol{s}$	Trial point for trust-region analysis
ρ	Trial point response ratio
Δ	Radius of the trust-region

MATLABOptimisation Problem Definitions

objective	Objective function
$\mathbf{x_0}$	Initial point for $\mathbf{x}$
$\mathbf{A_{ineq}}$	Matrix for linear inequality constraint
$\mathbf{b_{ineq}}$	Vector for linear inequality constraint
$\mathbf{A_{eq}}$	Matrix for linear equality constraint
$\mathbf{b_{neq}}$	Vector for linear equality constraint
lb	Vector of lower bounds
ub	Vector for upper bounds

Mathematical terms

NaN	Not a number

Chapter 1

Introduction

Bandler, in his 1969 paper, postulates that a fully automated design and optimisation is surely one of the ultimate goals for computer-aided design (CAD) [1]. He takes this even further and suggests that the more human intervention that is required to come to an acceptable design is a measure of how ignorant the designer was in setting up the problem, and in specifying goals and constraints in a meaningful way [1]. The use of CAD within the radio frequency (RF) and microwave circuit disciplines has indeed become standard practice for many engineers [2]. Electromagnetic (EM) solvers are ever-increasingly being used for design verification and are themselves improving in accuracy (higher fidelity) [3]. While EM solvers improve in accuracy and efficiency, industry continually pushes computational boundaries requiring finer meshes, for intricate designs, and analysing the effects within a multi-physics context [4, chap. 3, pg. 34]. For complex problems, the cost of solving them directly can be prohibitive and hamper adoption of the using these solvers in an optimisation context as this inevitably require solving the high-fidelity models multiple times [3, 5].

Different optimisation techniques have been applied within the EM field [1, 6–8]. A typical direct optimisation can be described as follows. Firstly let the design parameters of the model be represented listed as an N_n dimensional vector labelled $\boldsymbol{x}_f$. That is to say

$$\boldsymbol{x}_f \in \Re^{N_n \times 1} . \tag{1.1}$$

The response of the model, given the design parameters, form an N_m dimensional vector $\boldsymbol{R}_f$, where

$$\boldsymbol{R}_f \in \Re^{N_m \times 1} . \tag{1.2}$$

Typically, a response can be scattering-parameters (S-parameters), directivity or a power quantity. These quantities correspond to a set of frequencies, in this case N_m is the number of frequencies points selected. The subscript f represents a link to the fine model. Fine models are typically very accurate/high in fidelity and give results that are comparable to real-world measurements. This

type of simulation/modelling can be computationally expensive, especially for computational electromagnetic (CEM) simulations [5]. An optimal solution has some criteria that makes it better than some other point. Goals within a system are defined to capture this. A goal could be, for example, when looking at s-parameters in filter design, that the response decays quick enough, or for directivity that the side-lobes are as low as possible. A given goal is pulled into a function U where a single value is given for how well a goal is met. This is called the objective or error function. The optimisation routine will try to minimise this function and is described mathematically as

$$\boldsymbol{x}_f^* = \arg\min_{\boldsymbol{x}_f} U\left(\boldsymbol{R}_f(\boldsymbol{x}_f)\right). \tag{1.3}$$

where $\boldsymbol{x}_f^*$ is then that optimal point where U is a minimum [5]. In Figure 1.1 a flow diagram of a typical fine model optimisation routine is shown. Here i represents the iteration count starting at zero,

$$i = 0, 1, 2, \ldots . \tag{1.4}$$

The initial design, that is set up by the user, has input parameters $\boldsymbol{x}_f^{(0)}$. It is common that the search space is constrained and the optimiser is to only work within this feasible region. The constraints can arise because of project specification, where a system needs to fit into a particular volume, or due to solver criteria to ensure the model remains valid. If the error function has been minimised, then the current fine model parameters are at the optimal point $\boldsymbol{x}_f^* = \boldsymbol{x}_f^{(i)}$. If U is not yet at a minimum, then the iteration count is incremented by one, $i = i + 1$ (or using a shorter version $i++$). An updated set of design variables are calculated $\boldsymbol{x}_f^{(i)}$ and a fine model evaluated takes place. The evaluation uses a high-fidelity solver to get the most accurate results possible. This is often external to the optimiser itself and thus drawn outside the grey optimiser block in Figure 1.1.

Several techniques have been used to try overcome the simulation runtime bottle-necks in high-fidelity simulations. These include table lookup routines [1, 9], surface modelling and multidimensional interpolations [10], model-reduction techniques [11] and artificial neural networks [12].

In their 1994 paper Bandler *et al.* introduce a concept called space-mapping (SM) in an attempt to reduce the number of accurate fine model evaluations required to complete a successful optimisation [13]. Here a mapping is built up by pairing high-fidelity EM simulations with circuit simulations. The circuit models are low-fidelity models that have a low CPU cost but are not as accurate [4, chap. 8.4, pg. 160]. They still have underlying *physical* characteristics that are shared with full-wave EM solvers, but there will still be discrepancies between the results [5]. The low-fidelity model is also referred to as the coarse model.

When the low and high-fidelity models have a strong similarity in characteristics, then there is normally a one-to-one mapping between the design

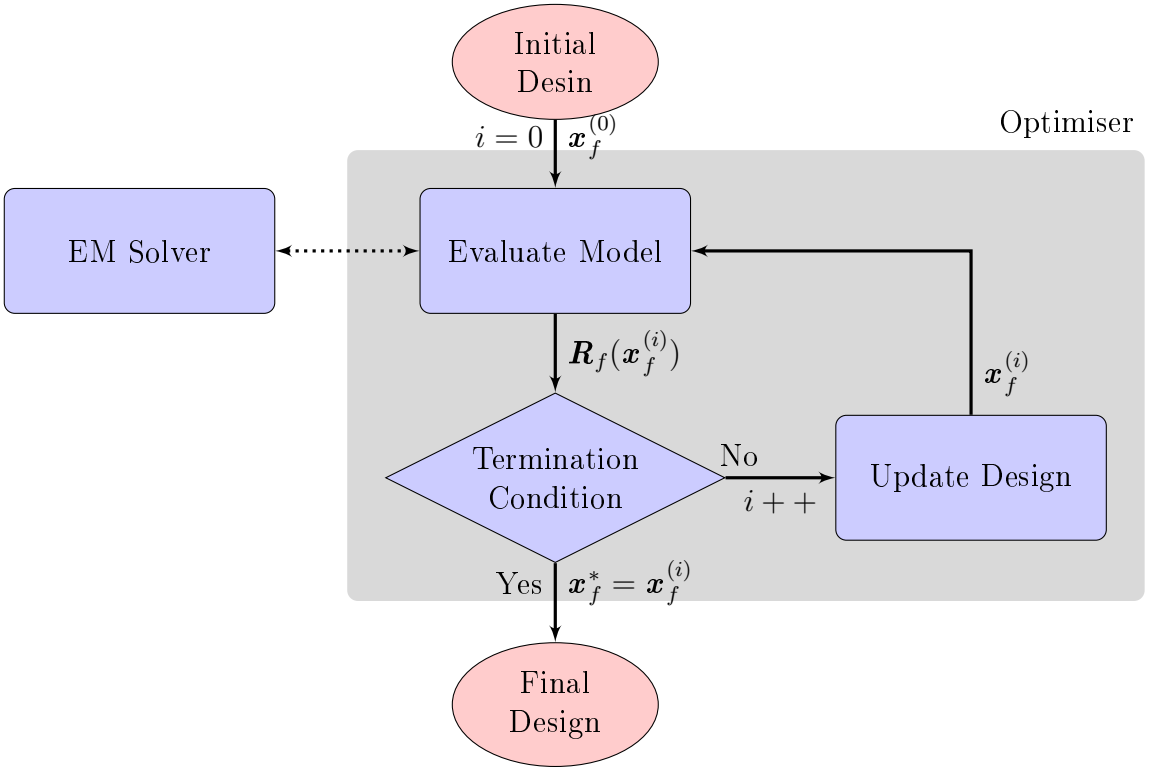


Figure 1.1: Flow diagram showing a direct high-fidelity optimisation

parameters used in the models. Therefore, the input parameter vector, for the coarse model is defined as

$$\boldsymbol{x}_c \in \Re^{N_n \times 1}, \tag{1.5}$$

where N_n is the same size as in (1.1). The response of the coarse model $\boldsymbol{R}_c$ is also expected to be the same size as the fine model response,

$$\boldsymbol{R}_c \in \Re^{N_m \times 1}, \tag{1.6}$$

where N_m is the number of samples (typically frequency) points. To compensate for the differences between the models, a parameter extraction (PE) or calibration phase is carried out [13]. Once this mathematical representation, reducing the differences has been built up, it can be applied to the other coarse models and transform them into a more accurate form. The updated coarse model is called a surrogate model and its response is given by $\boldsymbol{R}_s$. The process of updating the coarse model to get a surrogate model response that closely resembles the fine model response is called alignment. It is the minimisation of the difference/error between the surrogate and fine model responses. The error between the surrogate and the fine model response is given by

$$\epsilon = ||\boldsymbol{R}_s(\boldsymbol{x}_c) - \boldsymbol{R}_f(\boldsymbol{x}_f)||, \tag{1.7}$$

where $||\circ||$ represents a norm such as l_1, l_2, or Huber [14]. Here the design parameters $\boldsymbol{x}_c$ and $\boldsymbol{x}_f$ represent the low- and high-fidelity spaces respectively.

These are usually the same value as both spaces have the same underlying, physical representation. If this is the case then the subscripts are dropped and the design variable representation reduces simply to $\boldsymbol{x}$. Once the error between the responses has been minimised, the surrogate model is passed to an optimisation routine that finds an optimal point in low-fidelity space, $\boldsymbol{x}_c^*$. This is used to evaluate the next iteration in fine model space. The flow diagram in Figure 1.2 shows how the different stages link together. Once he fine model evaluation has completed it can be compared against its goal specification, in the same way as (1.3) and will terminate if the model meets the specifications. Some general space-mapping algorithms evaluate the optimal coarse model response to see if the initial specification has been reached [4, chap. 8.3, pg. 158]. If the specifications are not met yet then the iteration count is incremented and the process continues.

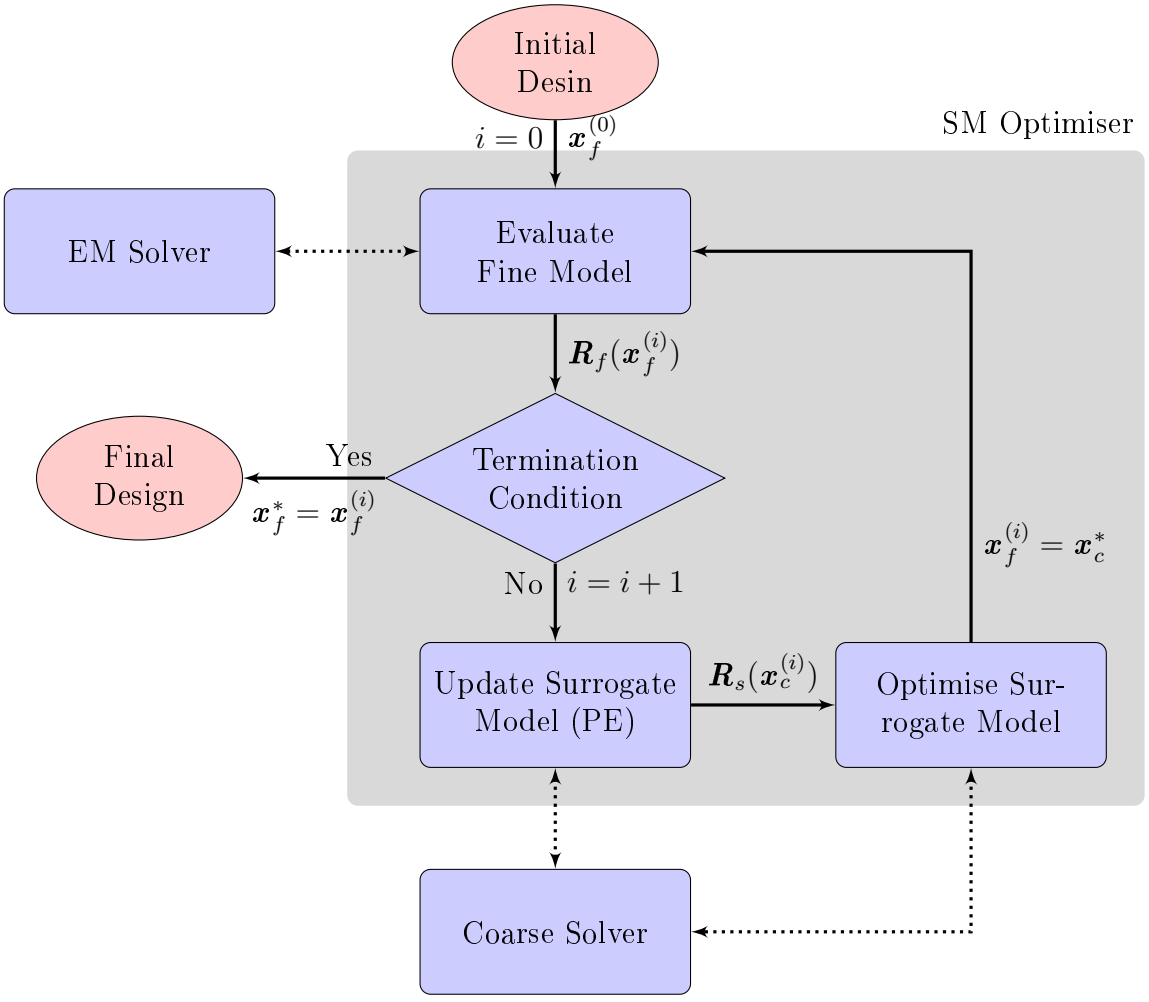


Figure 1.2: Flow diagram showing a space-mapping based optimisation

Generally, space-mapping algorithms can be summarised as consisting of

four main steps [3]. They are as follows:

1. Run a fine model simulation, $\boldsymbol{R}_f(\boldsymbol{x}_f^{(i)})$.
2. Extract the parameters of a coarse or surrogate model.
3. Update the surrogate or mapping functions.
4. Use an optimisation routine on the surrogate model.

In the more than two decades since that first paper there have been numerous improvements [3]. An aggressive space-mapping (ASM) approach uses the fine model evaluations to build the surrogate as soon as they are available [14]. The parameter extraction phase can result in a non-unique solution and the algorithm can break down [15]. To improve various approaches are suggested: step multipoint PE [15, 16], penalty based PE [17] and aggressive PE [18, 19].

The likelihood that a space-mapping optimisation routine will succeed, rests heavily on how similar the fine and surrogate models are [20]. In other engineering disciplines, surrogate models are often built up without there being any underlying physical basis between the models and instead only using fine model data [21–25]. A proper choice of coarse and surrogate models can improve convergence and even the overall performance of the space-mapping system [26, 27]. Even with a suitable model and mapping type convergence to a final design is not guaranteed and it may be necessary to manually verify the result [20, 23]. Furthermore, zero and first-order consistency conditions are not necessarily satisfied [23] between the fine and coarse models. This arises because the value and first-order derivative between the surrogate and fine models may not align at each iteration [20]. There is, therefore, no guarantee that the error will be reduced between iterations [27]. For an automated approach to attaining an optimal solution efficiently, this is of serious concern.

Incorporating a trust-region (TR) to the space-mapping algorithm can improve robustness and protect against convergence problems [27–31]. Koziel *et al.* suggest that even though applying a trust-region to the space-mapping algorithm does not rigorously ensure convergence, it instead becomes a heuristic that does indeed improve robustness [20]. A trust-region operates by restricting the design space optimisation to within a region (radius) where the fine and surrogate models have a reasonably good agreement. This is done by setting up a trial point and evaluating different metrics of how well the reduction of error between surrogate models compares with the reduction between the fine models [29]. If there is a good agreement then the radius in which the optimisation is bounded is increased as it appears that the surrogate model accurately represented the fine model at the last iteration. The agreement between the two models can change throughout the design space and thus the trust-region is used throughout the optimisation process. If an improvement between the surrogate and high-fidelity runs is not seen, then the trust-region

radius is reduced around the last iteration point (where there was good alignment). A new trial step is then set out within this new reduced region and the process continues.

Before the entire system is put together, some parts are analysed independently. In Chapter 2 the original space-mapping algorithms is outlined, a detailed example of frequency space-mapping (FSM) is carried out in isolation and the basic trust-region (BTR) algorithm is explained through the use of an example.

Once the robustness of the system is suitable to handle a fairly wide variety of problems, it can be presented to a user. The purpose of this project is to present a space-mapping framework to a user in an automated way for the intention of using it for device optimisation. Filters and radiating structures, at microwave frequencies, form a use-case of the type of device intended to be optimised using the system presented in this thesis.

A MATLAB based methodology is presented in Chapter 3, [32]. A number of space-mapping approaches are combined into a general and automated system that can be used in various different configurations. Available space-mapping options include input, output, frequency and implicit space-mapping. The automated system interfaces with a variety of external high and low-fidelity solvers including MATLAB based circuit models [32, 33], AWR-MWS [34], CST [35] and FEKO [36]. The user specifies their initial, pre-constructed fine and coarse models through an input directive and selects which form of space-mapping the system should use. Default optimisers are specified for alignment and design space optimisation, but these can be specified directly by the user.

The space-mapping framework is analysed in Chapter 5. The framework is tested using a simple mathematical model that is built up to handle a greater number of design variables. Detailed examples of each space-mapping type are explored within the context of the entire system using a simple microstrip stub example. FEKO is used as the fine model solver while a coaxial AWR-MWS example is used as the coarse model. The design of a gap wave-guide filter is carried out using CST as the high-fidelity model and MATLAB circuit components as the low-fidelity model [37].

Finally, in Chapter 6, closing remarks and recommendations for future is given.

Chapter 2

Space-Mapping Optimisation Techniques/Approaches

Space-mapping establishes a correction between high-fidelity simulation results $\boldsymbol{R}_f$ and that of low-fidelity approximate results $\boldsymbol{R}_c$. The corrected low-fidelity or coarse model is called a surrogate model and its response is denoted $\boldsymbol{R}_s$.

In this chapter, a selection of space-mapping techniques are presented that form a base for operating on a variety of different model types. The original space-mapping approach, by Bandler *et al.* [13], is introduced first in Section 2.1. This is the foundation from which the other techniques were built.

Koziel *et al.* break the space-mapping procedure up into four main groups [4, chap. 3.3.4.2, pg. 48]:

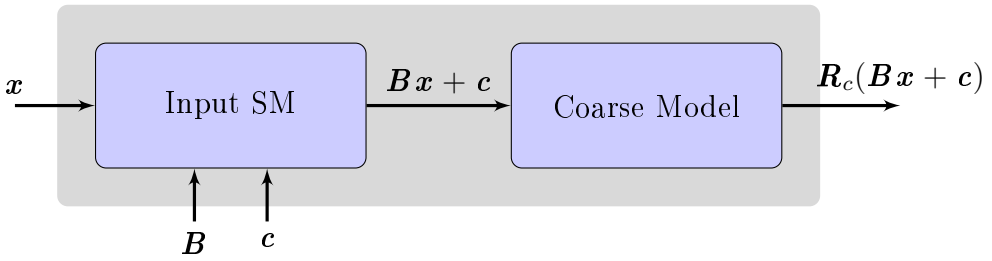
1. Input space-mapping, where a multiplicative term $\boldsymbol{B}$ and an additive term $\boldsymbol{c}$, are applied directly to the model design parameters, see Figure 2.1a. Once the design parameters have been adjusted, they are then passed through to the low-fidelity solver, which is directly the surrogate response, $\boldsymbol{R}_s = \boldsymbol{R}_c(\boldsymbol{B}\boldsymbol{x} + \boldsymbol{c})$

2. Output space-mapping (OSM) applies a correction to the response of the coarse model. Once again a multiplicative $\boldsymbol{A}$ and additive $\boldsymbol{d}$ factor can be used. The coarse model is initially evaluated and then the factors are applied, see Figure 2.1b. Here the surrogate response is given by $\boldsymbol{R}_s = \boldsymbol{A}\boldsymbol{R}_c(\boldsymbol{x}) + \boldsymbol{d}$.

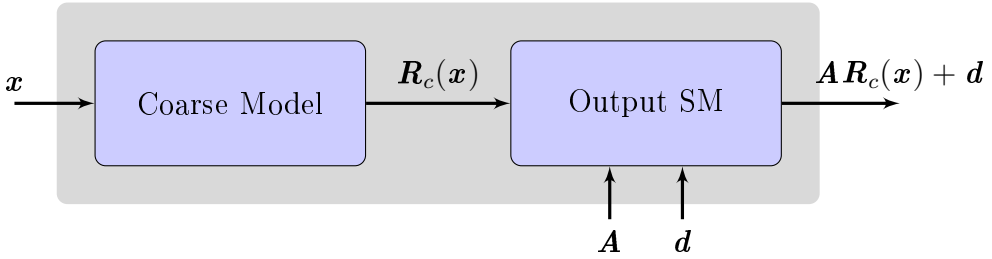
3. Implicit space-mapping (ISM) introduces extra parameters into the coarse model $\boldsymbol{x}_p$, see Figure 2.1c. This allows extra degrees of freedom through which the coarse model can be corrected to match the fine model. It is evaluated using the low-fidelity solver which gives the surrogate response $\boldsymbol{R}_s = \boldsymbol{R}_c(\boldsymbol{x}, \boldsymbol{x}_p)$. This is a powerful technique that allows insufficiencies in the coarse model to be compensated for.

4. Custom corrections which act on the independent axis. In the EM field this is often applied as frequency space-mapping (FSM). A frequency

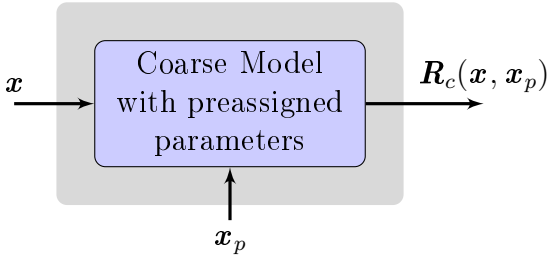
shift δ or axis scaling σ can be applied, see Figure 2.1d. The surrogate response is then calculated using the low-fidelity solver giving $\boldsymbol{R}_s = \boldsymbol{R}_c(\boldsymbol{x}, \sigma \boldsymbol{f}_c + \delta)$.



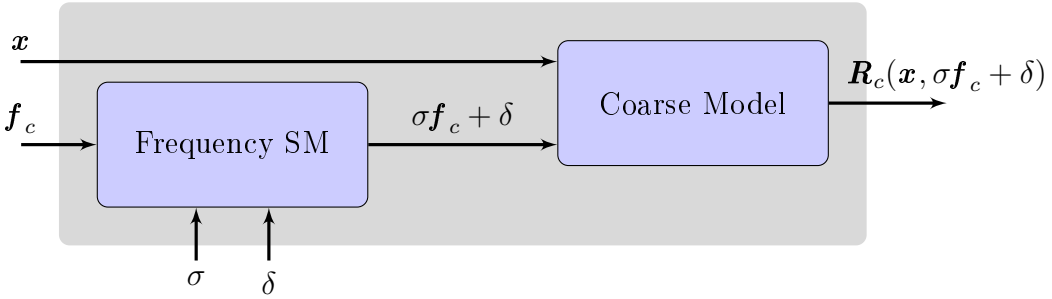
(a) Input parameter space-mapping.



(b) Output response space-mapping.



(c) Implicit space-mapping.



(d) Frequency space-mapping.

Figure 2.1: Four main space-mapping categories.

Each of these techniques can be applied to various different engineering problems. A detailed explanation of FSM in given in Section 2.2. Its origin, use in literature and some subtle implementation details are presented.

To address robustness issues when using the various space-mapping approaches [20], a detailed trust-region (TR) explanation is given in Section 2.3. The trust-region places a limit on the optimisation space of the design variables. An optimisation is carried out using an aligned surrogate model that is built up using one, or a combination of, space-mapping techniques. The improvement between the responses of the surrogate model before and after the confined optimisation is compared against the actual improvement between fine model responses. If the change in aligned surrogate model changes in the same way as the fine model, then it is trusted. However, if the way the responses change from iteration to iteration diverge, then the region is shrunk back to where it was last trusted.

2.1 Original Space Mapping Algorithm

Bandler *et al.* , combine the use of computationally cheap circuit model approximations, and that of high fidelity EM simulations, to accurately and efficiently optimise some specific microstrip problems [13]. In this paper, they develop a system where a mathematical mapping is created between optimised circuit models that correspond to EM evaluations. The fast evaluating circuit model is called the coarse model. The EM evaluation, that has a good accuracy, is the fine model.

A formal definition for the input parameters of the fine model are shown in (1.1) and that of the response is described in (1.2). Similarly, the coarse model's input/design parameters $\boldsymbol{x}_c$ are defined in (1.5). The output response $\boldsymbol{R}_c$, of the given design parameters is defined in (1.6). The dimensions of the fine and coarse model responses do not need to match but this is often the case.

In contrast to the typical direct optimisation routine, shown in (1.3), Bandler *et al.* bring together the coarse and fine parameter spaces using a transformation/mapping function $\boldsymbol{P}$. This mapping is represented by

$$\boldsymbol{x}_c = \boldsymbol{P}(\boldsymbol{x}_f)\,, \tag{2.1}$$

which must satisfy

$$\boldsymbol{R}_c(\boldsymbol{P}(\boldsymbol{x}_f)) \approx \boldsymbol{R}_f(\boldsymbol{x}_f) \tag{2.2}$$

in the region of interest. This only holds where the high-fidelity and coarse approximation models have a reasonable agreement. If they do not, then the norm of the responses will not tend to ϵ. Or said another way, an error between the coarse and fine model can be defined as

$$||\boldsymbol{R}_f(\boldsymbol{x}_f) - \boldsymbol{R}_c(\boldsymbol{P}(\boldsymbol{x}_f))|| \leq \epsilon\,, \tag{2.3}$$

where $||\circ||$ is a suitable norm and ϵ is as small a positive constant as possible [13]. Different norm functions are described later in Section 3.5.5.5.

To find an estimation for $\boldsymbol{x}_f^*$, without direct optimisation (1.3), $\overline{\boldsymbol{x}}_f$ is defined

$$\overline{\boldsymbol{x}}_f \triangleq \boldsymbol{P}^{-1}(\boldsymbol{x}_c^*)\,, \tag{2.4}$$

where $\boldsymbol{x}_c^*$ is the optimal solution of the coarse model and $\overline{\boldsymbol{x}}_f$ is the image of $\boldsymbol{x}_c^*$ subject to (2.3), [13].

Bandler *et al.* adopt an iterative process to build up $\boldsymbol{P}$ using the previous fine model evaluations [13]. Typically, $\boldsymbol{P}$ is a simplified zero or first order model. A dimension k set of evaluations is represented by

$$\boldsymbol{D}_f = \{\boldsymbol{x}_f^{(1)}, \boldsymbol{x}_f^{(2)}, \ldots, \boldsymbol{x}_f^{(k)}\}\,. \tag{2.5}$$

To build up the set, an initial point $\boldsymbol{x}_f^{(1)}$ is chosen within the parameter space. This point can be determined by finding the optimal coarse model $\boldsymbol{x}_c^*$. Figure 2.2 shows the steps that the algorithm goes through to obtain an accurate model taking coarse model parameters to the fine model space [13]. The left-hand side set of axes represent the coarse model space and the points where responses are calculated. The fine model space is on the right-hand side axes. Figure 2.2a shows the first step where the fine model is evaluated at the same point as the optimal coarse model.

Next, a further k evaluations in the neighbourhood of $\boldsymbol{x}_f^{(1)}$ are evaluated, see Figure 2.2b. k is a predefined number of solutions to be evaluated. The additional fine models are used to get a good mapping function. Bandler *et al.* use five additional points in their example, [13].

Through parameter extraction, the coarse model set

$$\boldsymbol{D}_c = \{\boldsymbol{x}_c^{(1)}, \boldsymbol{x}_c^{(2)}, \ldots, \boldsymbol{x}_c^{(k)}\}\,, \tag{2.6}$$

is found such that (2.3) holds. Each parameter pair between $\boldsymbol{D}_c^{(1)}$ and $\boldsymbol{D}_f^{(1)}$ are evaluated allowing the mapping $\boldsymbol{P}_1$ to be created. Figure 2.2c show the new coarse model evaluations and an optimal coarse model evaluation.

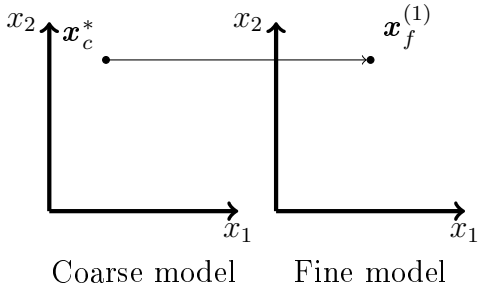
The inverse transform $\boldsymbol{P}_j^{-1}$ is applied to the optimal coarse model point to find the next jth fine model point,

$$\boldsymbol{x}_f^{(k_j+1)} = \boldsymbol{P}_j^{-1}(\boldsymbol{x}_c^*)\,. \tag{2.7}$$

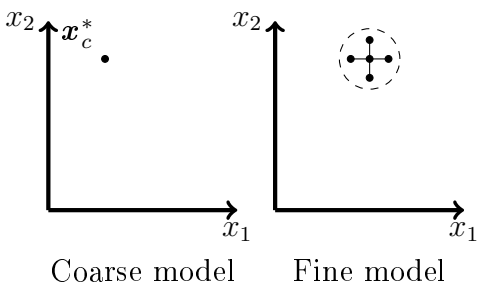
This point in fine model parameter space will be different to that of the coarse model space until there is no further mapping required to get the previous k fine and coarse model responses to align. This step is shown in Figure 2.2d. The responses $\boldsymbol{R}_c(\boldsymbol{x}_c^*)$ is compared to the new fine model response $\boldsymbol{R}_f(\boldsymbol{x}_f^{(k_j+1)})$. If the difference,

$$||\boldsymbol{R}_f(\boldsymbol{x}_f^{(k_j+1)}) - \boldsymbol{R}_c(\boldsymbol{x}_c^*)|| \leq \epsilon\,, \tag{2.8}$$

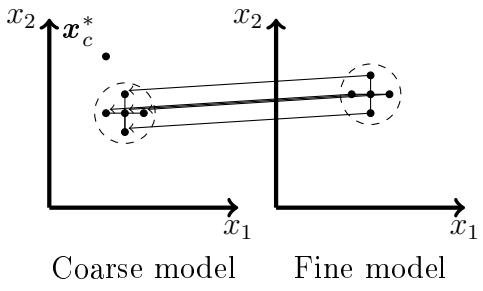
then $\boldsymbol{R}_f(\boldsymbol{x}_f^{(k_j+1)})$ is the desired fine model solution $\overline{\boldsymbol{x}}_f$. If the termination criteria is not met, then $\boldsymbol{D}_f$ is expanded to include the new fine model.



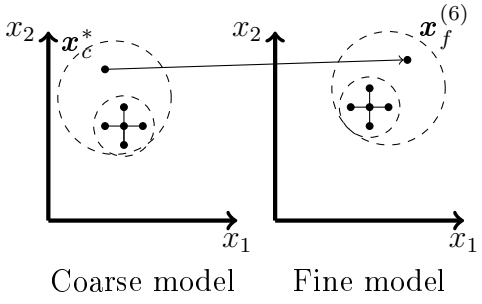
(a) An initial, optimal coarse model is found. The fine model is evaluated at this point.



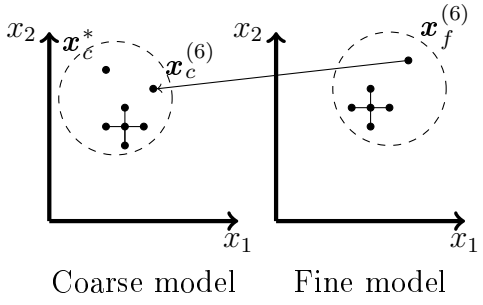
(b) Fine model design parameter is perturbed in the vicinity of the first fine model evaluation.



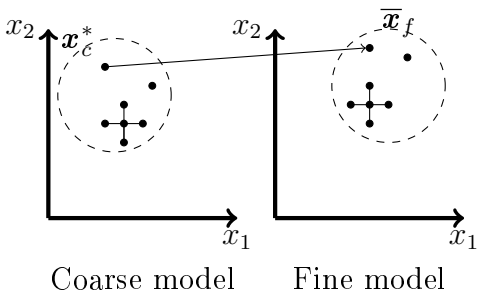
(c) Parameter extraction is used to obtain a mapping where the coarse and fine models give the same response.



(d) $\boldsymbol{P}^{-1}$ is applied to the optimal coarse model point to find the next fine model.



(e) Parameter extraction is carried out of the new fine model evaluation to obtain $\boldsymbol{x}_c^{(6)}$.



(f) Applying the updated inverse transform and meeting termination criteria.

Figure 2.2: Snapshots of how the original space-mapping algorithm moves through the different stages of the algorithm.

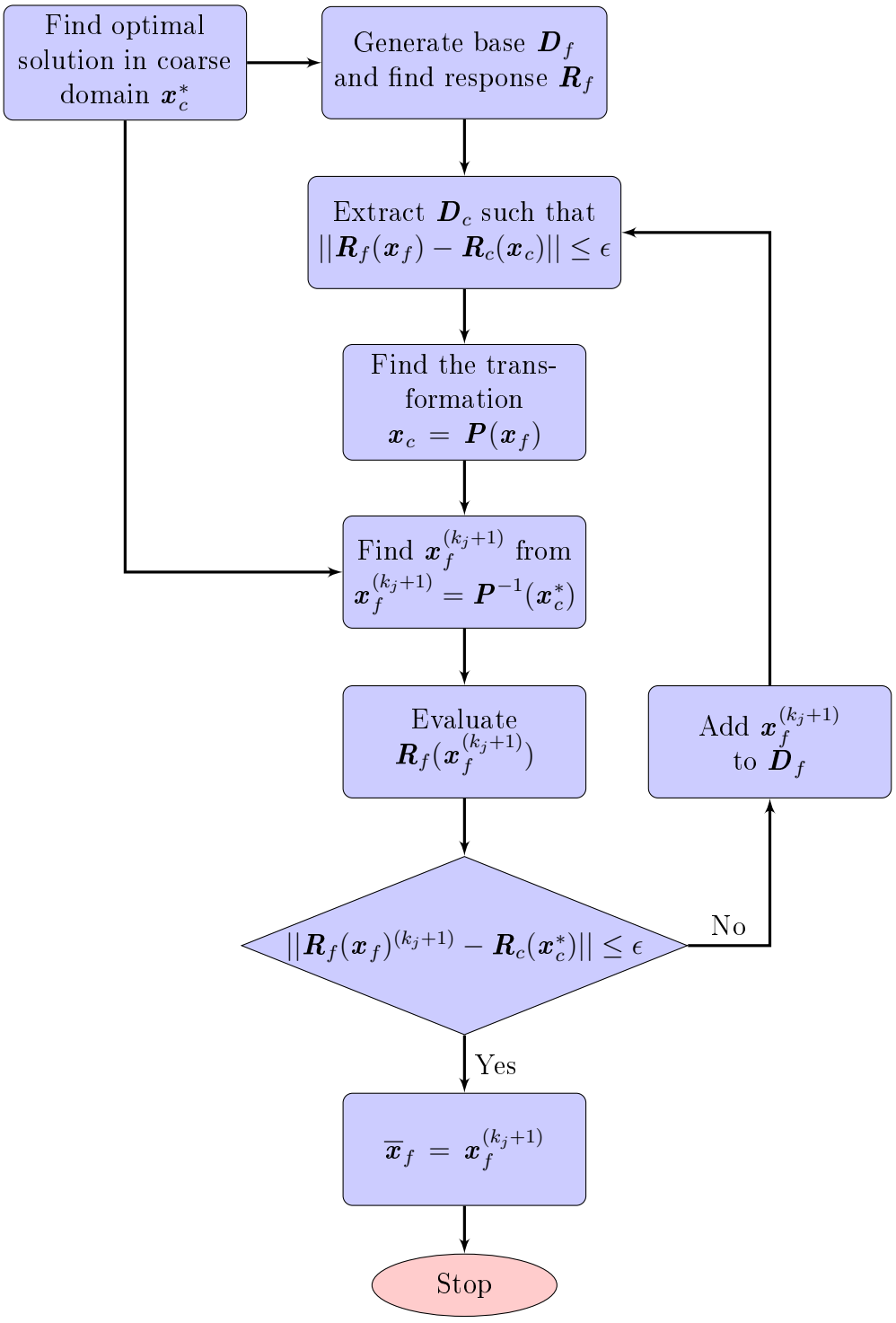


Figure 2.3: Flow diagram of the original SM algorithm

model, continues until termination criteria (2.8) is met.

The final step is shown in Figure 2.2f, where, $\overline{\boldsymbol{x}}_f$, the final image of $\boldsymbol{x}_c^*$, being found. Figure 2.3 shows a detailed flow diagram of the steps and loops [13].

This original space-mapping algorithm uses mapping functions to calculate the surrogate model. The example presented in the next section manipulates the independent variable used to evaluate the coarse model.

2.2 Frequency Space-Mapping (FSM)

Bandler *et al.* introduce a custom correction approach to space-mapping, in their 1995 paper [14], by acting on the independent variable. When evaluating models, within the EM field, the independent variable is often frequency. This method is therefore called frequency space-mapping (FSM).

Figure 2.1d shows that the frequency is fed into the coarse model evaluation. This is from an overview perspective with an initial optimisation having already been completed. The figure shows σ and δ terms being applied to the next evaluation. When the σ and δ terms are calculated, only the data from one coarse model evaluation is required. Interpolation is used on that data to get the new results for the surrogate response. This section is specifically about the way that the σ and δ points are calculated.

Adjusting the frequency of the coarse model response $\boldsymbol{R}_c$ can provide an efficient alignment to the fine model response $\boldsymbol{R}_f$. This can take the form of a shift/offset in frequency δ, and/or a scaling factor σ [14]. The frequency of the fine model $\boldsymbol{f}_f$ is kept constant while the frequency of the coarse model $\boldsymbol{f}_c$ is adjusted to get the coarse model's response data to align with that of the fine model's response. The frequencies that show the best match/alignment of the responses is given as $\boldsymbol{f}_s$. A mapping between the coarse and optimal frequency is given by

$$\boldsymbol{f}_s = \sigma \boldsymbol{f}_c + \delta \,. \tag{2.9}$$

To effectively align $\boldsymbol{R}_c$ to $\boldsymbol{R}_f$ the following minimisation takes place,

$$\arg\min_{\sigma,\delta} ||\boldsymbol{R}_c(\boldsymbol{x}_c, \sigma \boldsymbol{f}_c + \delta) - \boldsymbol{R}_f(\boldsymbol{x}_f)|| \,, \tag{2.10}$$

where $||\circ||$ is a suitable norm. Norm functions discussed in Section 3.5.5.5. $\boldsymbol{x}_c$ and $\boldsymbol{x}_f$ remain constant through this optimisation [14]. The surrogate response is defined as

$$\boldsymbol{R}_s = \boldsymbol{R}_c(\boldsymbol{x}_c, \sigma \boldsymbol{f}_c + \delta) \,. \tag{2.11}$$

To demonstrate the effect of δ and σ, a frequency shift and scaling examples are presented below. Simple inverse tangent mathematical models are used in both examples. Even though mathematical examples evaluate extremely quickly the coarse model is expressed as a simpler function of the fine model.

The fine model has some extra shift or scaling applied to it that the aligning phase must reduce when building the surrogate model, see (2.10).

2.2.1 Frequency Shift Example

The coarse model response $\boldsymbol{R}_c$ uses an inverse tangent functions with an inflection point around 25 Hz, see Table 2.1. The inflection point is chosen so that only positive frequency points need to be evaluated. The function is calculated between $0 - 50$ Hz, with 13 samples. This frequency range is chosen so that there is sufficient space around the inflection point and that the function can tend to a constant at the start and end, even when shifted. The number of samples is small so that the graphs do not appear cluttered and the shifts can be seen clearly, typically this should have more samples for accurate results.

The fine model response $\boldsymbol{R}_f$ is the same as the coarse model, except it is shifted by a further 10 Hz, see Table 2.1.

Just one input parameter x_1 is used for these functions. It starts off at a value of one and does not change in this example because only the alignment phase of FSM is described. As with most SM problems, the input parameter is the same for both the fine and coarse model ($\boldsymbol{x}_c = \boldsymbol{x}_f = x_1$).

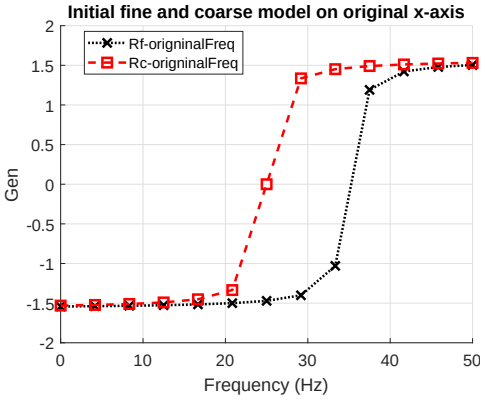
Table 2.1: Fine and coarse model equations for FSM frequency shift example

Coarse model response	Fine model response
$\boldsymbol{R}_c = \tan^{-1}(x_1(f) - 25)$	$\boldsymbol{R}_f = \tan^{-1}(x_1(f - 10) - 25)$

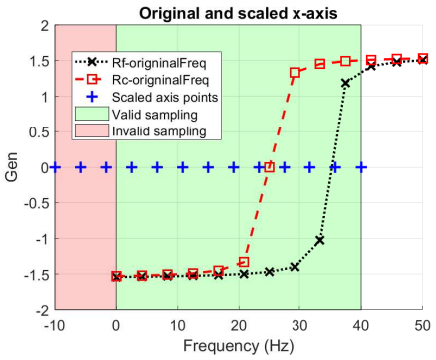
Figure 2.4a shows the initial fine and coarse models. The black crosses represent the calculated points of the fine model. It crosses zero at about 35 Hz. The coarse model's inflection point can be seen 10 Hz earlier, (red squares). For easy comparison lines are plotted through the fine and coarse models for all the figures in Figure 2.4, (dotted line for fine and dashed line for coarse).

Both models have the same underlying function and are just offset from one another. A frequency shift can be applied to the coarse model to sufficiently align it with the fine model. To do this (2.9) is used. For this example case, the offset is known so it is easy to determine the multiplicative and additive constants (σ and δ). No scaling is required so $\sigma = 1$. The offset total offset between the graphs is used for the additive constant, $\delta = -10$. These values are not typically known and this is solved as an optimisation problem using (2.10).

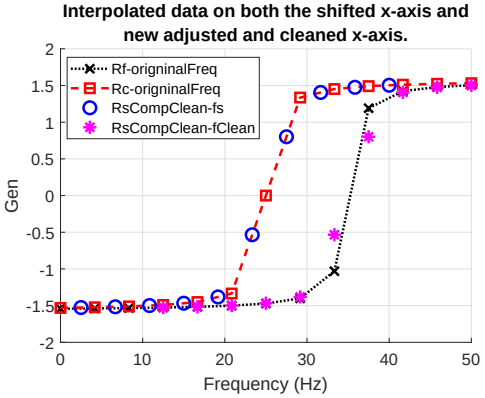
Figure 2.4b shows these delta and sigma values applies to the frequency axis $\boldsymbol{f}_s$. A 10 Hz shift is clearly seen with the blue pluses. The actual axis is shifted and new response data is required. The green region of Figure 2.4b shows points where the coarse model response can be used to acquire surrogate data. There is no data for the surrogate between -10 – 0 Hz and is highlighted



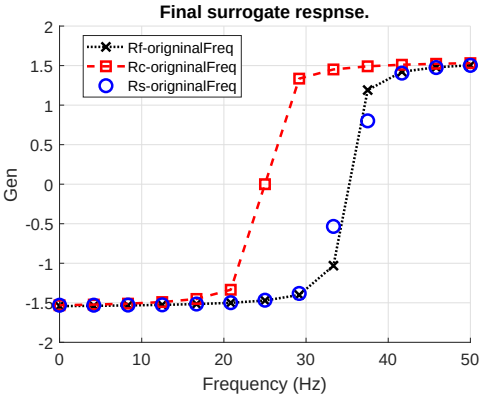
(a) Initial fine and coarse model responses for frequency shift example.



(b) Independent axis adjustment to allow alignment. Sampling sections highlighted to show valid data points.



(c) The interpolated coarse model data with invalid points removed. The data is plotted on both the original frequency axis and the cleaned new axis.



(d) The final surrogate model response interpolated using existing data and end points.

Figure 2.4: The different stages of FSM for a frequency shift example.

by a red region. The region between $40 - 50\,\text{Hz}$ is not relevant to the new frequency axis and is thus not highlighted red nor green.

Now that there is an axis to work from, the values of the surrogate response $\boldsymbol{R}_s$ can be calculated. This is done through interpolation in the region where the surrogate axis $\boldsymbol{f}_s$ and the coarse model's axis $\boldsymbol{f}_c$ overlap (the green region in Figure 2.4b). Values outside of the common region are set to NaN since no data is available.

The MATLAB function `interp1` is used to do the interpolation [38]. This is a 1-D table lookup type of interpolation with various interpolation methods. Three of the available methods are considered for this problem: a shape-preserving piecewise cubic method called `pchip`, `spline` and `linear`. See Figure 2.5 for a comparison between these three interpolation methods. By

default, the `pchip` and `spline` functions extrapolate and values outside the overlapping region must be set to NaN and cleaned up manually. For examples like this, with relatively small samples sets, the `spline` gives a less-smooth transition as the inflection starts, these regions of difference are highlighted in Figure 2.5. Even though both the `pchip` and `spline` method appear to be better candidates in this example `linear` is used for this stage of the alignment.

It is important to note that the input data need to be free of invalid values, i.e. values at $\pm$ infinity or NaN. The `interp1` returns invalid results or gives an error under these circumstances. Furthermore, values from an extrapolation region with the `interp1` function are set to NaN. To clear invalid values, the result data points and their associated frequency points should be removed. The result of the coarse model interpolation, with invalid points removed, is shown in Figure 2.4c - blue circles. Note that there are no blue circle values between $40 - 50\,\text{Hz}$. The clean data is now projected back onto the original frequency axis, see Figure 2.4c - magenta stars. There are no magenta star values from $0 - 10\,\text{Hz}$.

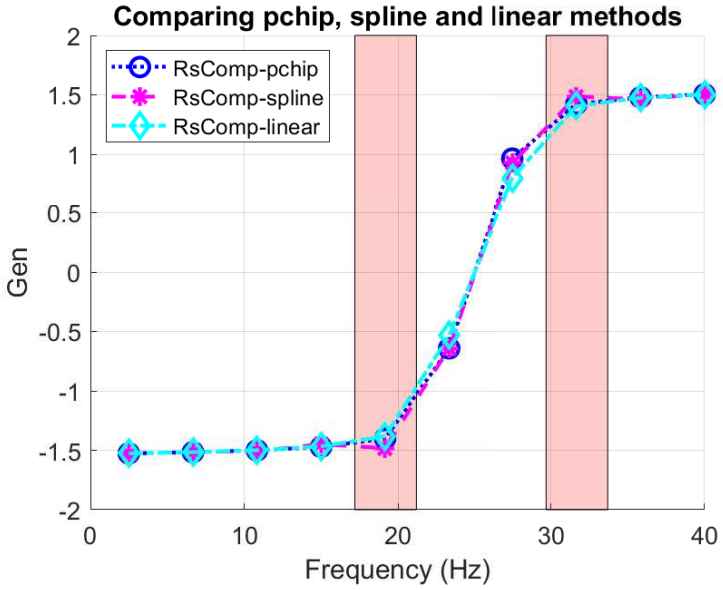


Figure 2.5: Plot of the difference between MATLAB interpolation (`interp1`) techniques: `pchip`, `spline` and `linear`

The last step is to ensure that the surrogate data matches up with the fine model's. The validation of data in the previous steps removed values leaving an inconstancy in the number of points. Furthermore, the endpoints could have been removed resulting in a data step at the ends. If the new frequency axis starts higher or ends lower than the original frequency axis, the first/last original axis value is prepended/appended to the new frequency axis values. The associated data value is also taken from the original coarse model. In the

same way that this example has a flat start and end, so too is it expected for most examples. This is a zero order extrapolation. If derivative information is known, and it is non-zero, then a first order extrapolation can be applied. This is not done here in these illustrative examples because they do not have enough samples.

Once this is done, an additional interpolation of the surrogate data from the new frequency axis to the original frequency axis is performed. This ensures that the new values line-up correctly for future comparisons with the fine model response. Now that the boundary values are correct and all irregularities have been removed from the frequency points the `pchip` shape-preserving method can be used instead of linear. This validation step is useful, but it is still necessary to ensure that *good* responses are generated that are sufficiently sampled and contain critical data at the centre of the response. It is also possible to only evaluate a specific, narrower, section of the response, this allows the system to ignore edge anomalies. These regions are typically defined around the goal region.

The final result is shown in Figure 2.4d - blue circles. There is a good agreement between the fine and the surrogate results. In the next example, a scaling of the independent axis is required instead of a shift.

2.2.2 Frequency Scaling Example

As in the previous example, an inverse tangent function is used. It is evaluated between $0-50\,\text{Hz}$, with 21 samples. The inflection point is once again around 25 Hz. Here the fine model is scaled up by a factor of 3.5 instead of being shifted, see Table 2.2. The coarse model is scaled down by 0.7 to further emphasise the difference between the two models.

A single input parameter x_1 is given a value of one. This example is just looking at the alignment phase so the parameter does not change ($\boldsymbol{x}_c = \boldsymbol{x}_f = x_1$).

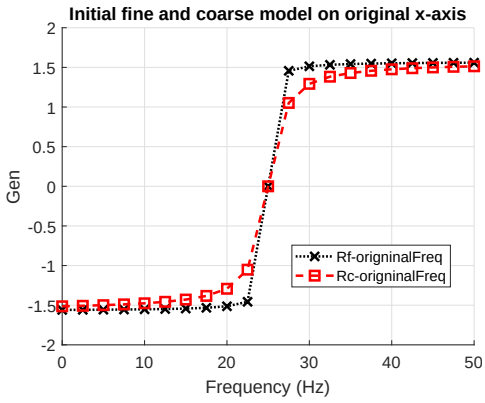
Table 2.2: Fine and coarse model equations for FSM frequency scaling example

Coarse model response	Fine model response
$\boldsymbol{R}_c = \tan^{-1}(3.5x_1(f-25))$	$\boldsymbol{R}_f = \tan^{-1}(0.7x_1(f-25))$

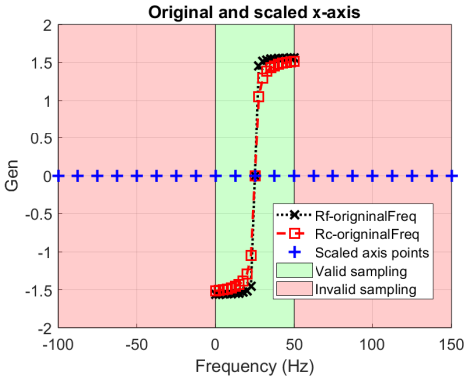
Figure 2.6a shows the initial fine (black crosses) and coarse models (red squares) responses.

To change the coarse model so that it aligns to the fine model, the gradient needs to increase. This is done by manipulating the frequency axis so that samples are pushed closer together at the centre. The axis is then stretched out (using a factor of $3.5/0.7 = 5.0$) to exist between -100$-$150 Hz, see Figure 2.6b.

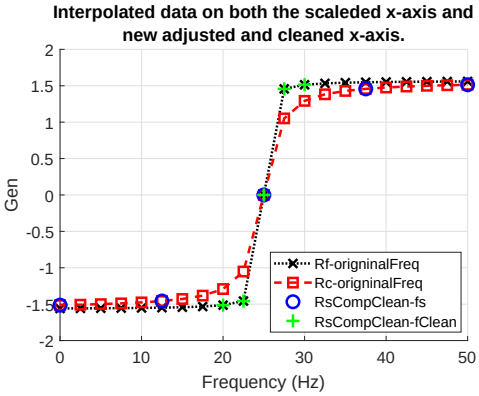
Only five points of this new axis lie on the original axis. These points will not necessarily line up with any of the original points. MATLAB's `interp1`



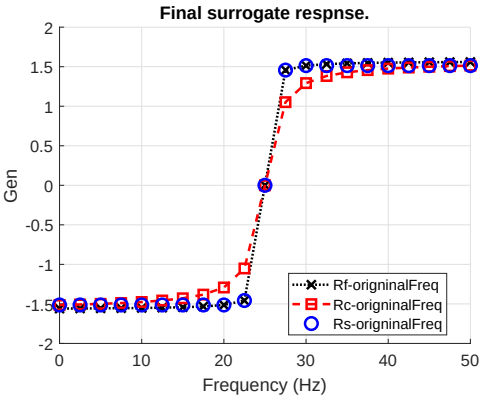
(a) Initial fine and coarse model responses for frequency scaling example.



(b) Independent axis adjustment to allow alignment. Sampling sections highlighted to show valid data points.



(c) The interpolated coarse model data with invalid points removed. The data is plotted on both the original frequency axis and the cleaned new axis.



(d) The final surrogate model response interpolated using existing data and end points.

Figure 2.6: The different stages of FSM for a frequency scaling example.

function is used to acquire points from the coarse model using the new and the original axis. It is very important to note that only interpolation from existing points is possible. Using extrapolation methods on this function are unlikely to succeed. It is however still possible to use an interpolation routine like `pchip` when acquiring points from the coarse model. By default, the `pchip` and `spline` routines use extrapolation, thus values would need to be forced to NaN outside the valid original axis. In this example the `linear` option for `interp1` is used. The newly interpolated points are shown as blue circles in Figure 2.6c. These five points line up successfully with the coarse model and are now moved onto the original frequency axis, see the green crosses in Figure 2.6c. The values line up with the fine model.

Values are still required from $0 - 20\,\text{Hz}$ and from $30 - 50\,\text{Hz}$, where NaN

values where forced. The first and last coarse model values are used here. In general, if the first frequency from the original axis is lower than that of the cleaned axis then the first data point from the coarse model is assigned to the first element for the surrogate's data. Similarly, if there is a higher frequency on the original axis, that the cleaned axis, then the final point is added to the end of the surrogate's data. Once these points have been inserted, then the final interpolation phase can be carried out. MATLAB's `interp1` function using `pchip`, is used for here. Only valid data exists and the end-point values have been inserted. Interpolated points at the start and the end could just be filled linearly but points along the rest of the results benefit from the curve fitting interpolation. The final result of this surrogate is shown in blue circles on Figure 2.6d. Typically, only having five points of useful data would result in erogenous results but in this trivial case, it is sufficient.

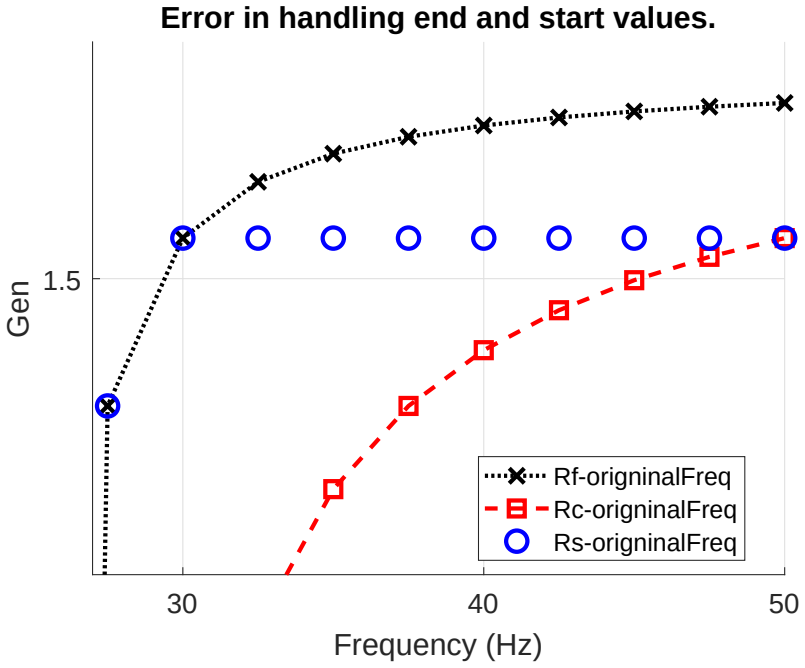


Figure 2.7: Error when using the last/first value from the coarse model to replace NaN values

The usage of the coarse model as the final/starting point for the surrogate can result in errors. An enlarged version of the top-right part of Figure 2.6d is shown in Figure 2.7. Here there is a clear discrepancy between the surrogate and the fine model. In general, this error is still significantly less than using an extrapolation routine. If a first order extrapolation method were to be used, this error could be reduced.

2.2.3 Overview

Steps from the two examples above can be used to summarise the approach for FSM alignment. Figure 2.8 shows a flow diagram of the various stages. In the bigger scheme of things, the FSM alignment phase runs as part of the broader optimisation.

An initial fine and coarse model are received. If the response has complex values then pre-formatting takes place by taking the absolute value or converting the response to decibels. This is different for frequency space-mapping in comparison to other space-mapping techniques. The phase information is useful for alignment and matching both the real and imaginary part is typically done.

New σ and δ terms are calculated and applied to the original frequency values. The new values are used by an interpolated routine to obtain useful response data in the overlapping region. Invalid values are then removed and any necessary boundary values are appended. The response data, corresponding to new frequency values, is pushed through another interpolation routine using the original frequency values. This is effectively the response of the surrogate.

The fine model and surrogate responses can now be compared. The norm of the differences between them is calculated and used as the error that is sent to the optimiser. This stage forms part of the parameter-extraction/updating the surrogate model phase seen in Figure 1.2. An inner loop iterates until a suitable surrogate is found. Once the error/difference has been sufficiently reduced, the σ and δ values are returned to the main algorithm as the chosen FSM parameters for this alignment. The surrogate model is now known and can be used in the next optimisation phase where the overall surrogate response $\boldsymbol{R}_s$ goals are minimised to find the next optimal design space parameter $\boldsymbol{x}_c^*$.

In the next section, a technique for improving the robustness of the outer optimisation loop is discussed. This is used in combination with a space-mapping technique such as FSM.

2.3 Trust-Region Convergence Safeguard

The space-mapping framework allows for optimisation problems to be solved quickly and with less computational resources. It is important to ensure that the optimal coarse model solution is feasible in the fine model space. If the coarse model and fine models do not give similar results then the optimisation is likely to fail. This falls into a broader topic of convergence and robustness of the space-mapping algorithm. Divergence is observed when the error between iterations is not reduced or there is a reduction in either the coarse/surrogate model or the fine model. The first step to achieving convergence is to ensure there are underlying physical similarities between the fine and coarse model

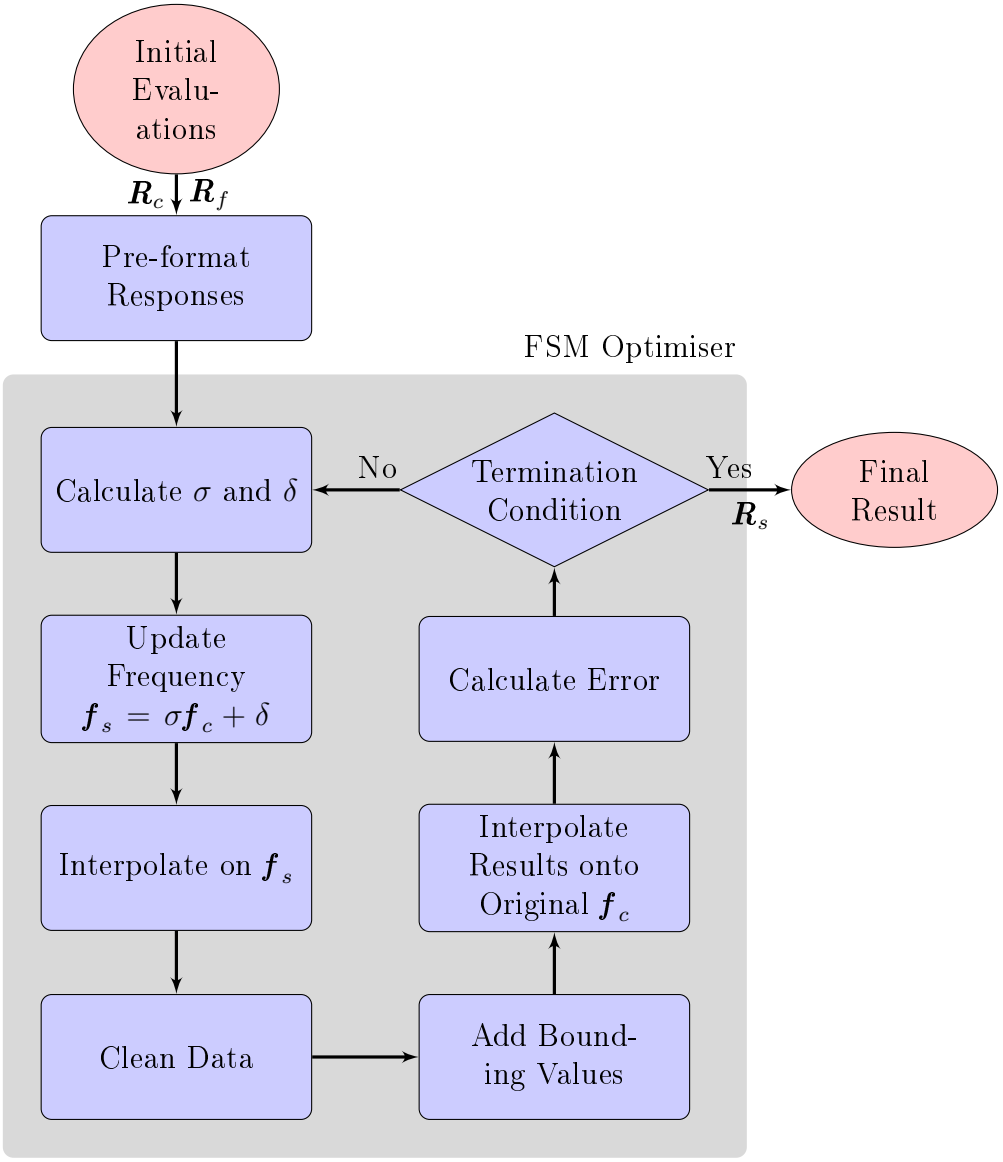


Figure 2.8: Flow diagram of FSM.

[20, 26]. Even if this is the case, convergence is not ensured. Another step is to ensure that the limits are set up correctly for both models. For example, if in the fine model geometric quantity changes result in overlaps the simulation may become invalid. Some EM solvers require geometry that is overlapping to be unioned together. For low-fidelity solvers, there are often approximations that only hold within specific regions of operation. If limits are not set up to ensure these regions are avoided, invalid results may be obtained.

For convergence to be guaranteed first- and second-order consistency conditions would need to be satisfied between the surrogate and fine models [23]. Since this is not the case, other convergence safeguards are required [20, 27, 39]. This is especially important for an automated system.

These safeguards restrict the optimisation space to ensure that convergence is achieved. A trust-region is a form of safeguard that restricts the optimiser for taking steps that are too big to be correct in both the surrogate and fine models. Typically, an optimiser chooses the step-size and the direction is specified as "downhill". Here, with a trust-region, the step-size is limited and the direction is left up to the optimisation routine. Koziel *et al.* note that the trust-region method is not formally justified dues to first-order constancy mismatch [20]. However, they still find it to be a useful heuristic to apply [20].
.

For SM, the trust-region is applied in the main optimisation loop where the coarse model evaluations explore the parameter space. The basic algorithm is described below.

2.3.1 Basic Trust-Region Algorithm (B.T.R.)

Conn *et al.* provides a detailed explanation of how to implement a Basic Trust-Region algorithm (B.T.R.) [29, chap. 6.1, pg. 115]. A simplified flow diagram of their algorithm is shown in Figure 2.9, this highlights five key stages to the algorithm [29, chap. 6.1, pg. 116].

1. *Initialisation*, boundaries of parameters, starting point, initial trust-region radius and constant definitions.

2. *Model definition*, where a surrogate model is built up.

3. *Trial point calculation*, running an optimisation on the surrogate model.

4. *Evaluation of trial point*, where the change in the fine model is run at the trial point and the success of the iteration is evaluated.

5. *Trust-region radius update*, that updates the region for the optimisation in the next iteration. The radius is updated differently depending on of whether the evaluated trial point was considered a success or not.

A mathematical example is used to illustrate the different stages of the algorithm and the choices made along the way. As before, the set of parameters is represented by $\boldsymbol{x}$ and now a superscript k represents the iteration number. For example, $\boldsymbol{x}^{(0)}$ is a vector of the model parameters at the start of the first iteration. The example presented is an adaptation from [29, chap. 1, pg. 1]. Consider the fine model, with two parameters, given by

$$\boldsymbol{R}_f(x_1, x_2) = \text{-}15x_1^2 + 15x_2^2 - 6\sin(x_1 x_2) + 2x_1 + 2x_1^4 . \tag{2.12}$$

In a practical problem it would be far too expensive to evaluate the entire model in high fidelity space, but for this simple mathematical example it is done to give the reader an overview. A contour plot of the fine model is shown

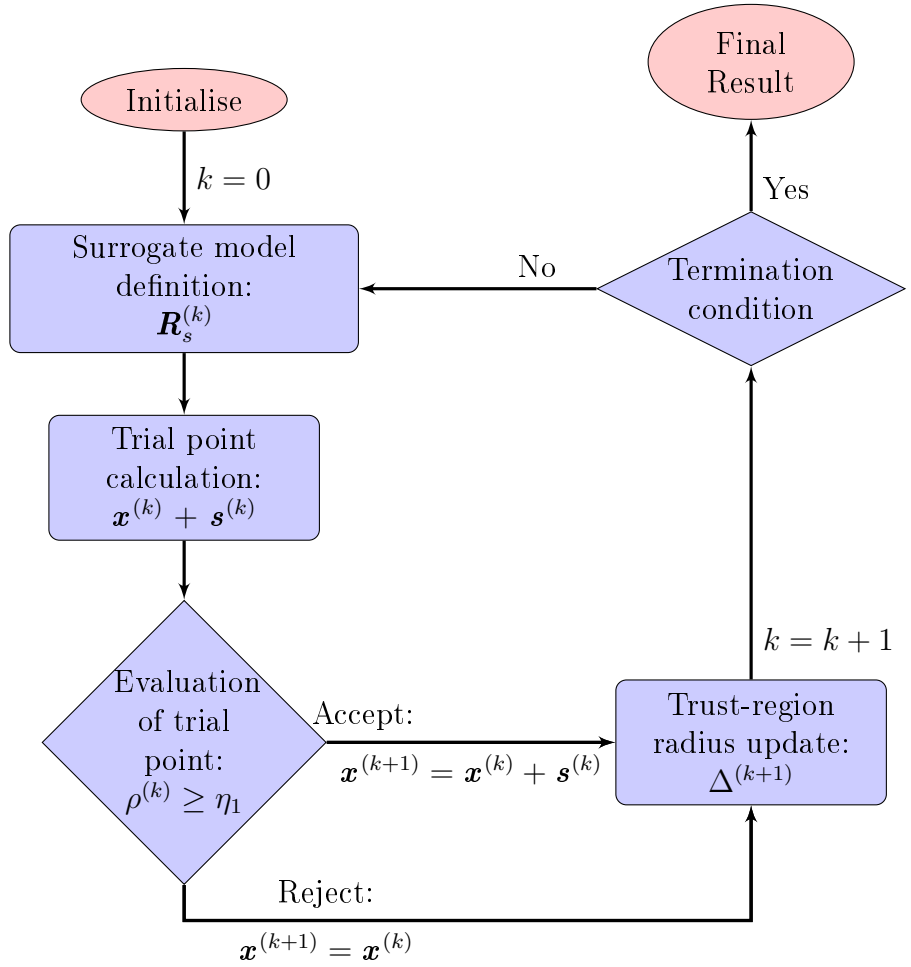


Figure 2.9: B.T.R. flow diagram

in Figure 2.10. Equation (2.12) is evaluated from -3.5−3.5 along both parameters. There are steep walls around the edge of the model with two minima near the centre, one slightly lower than the other. For this mathematical function it is also possible to compute the gradient at each point, this is also shown in Figure 2.10 with arrows. If the entire fine model space is known, then it is trivial to choose the optimal point $\boldsymbol{x}^*$ that gives a fine model minimum at $\boldsymbol{R}_f(\boldsymbol{x}^*)$. In practice a computationally cheaper operation would be desirable, for example employing this B.T.R. approach.

2.3.1.1 Initialisation

The bounds in which the fine model can be evaluated are known and the parameter values make physical sense. For a mathematical model, like this example, that translates to checking that there should be no singularities or undefined points. In physically based examples other check can be carried out like checking that geometry/meshes do not overlap or that the limits of

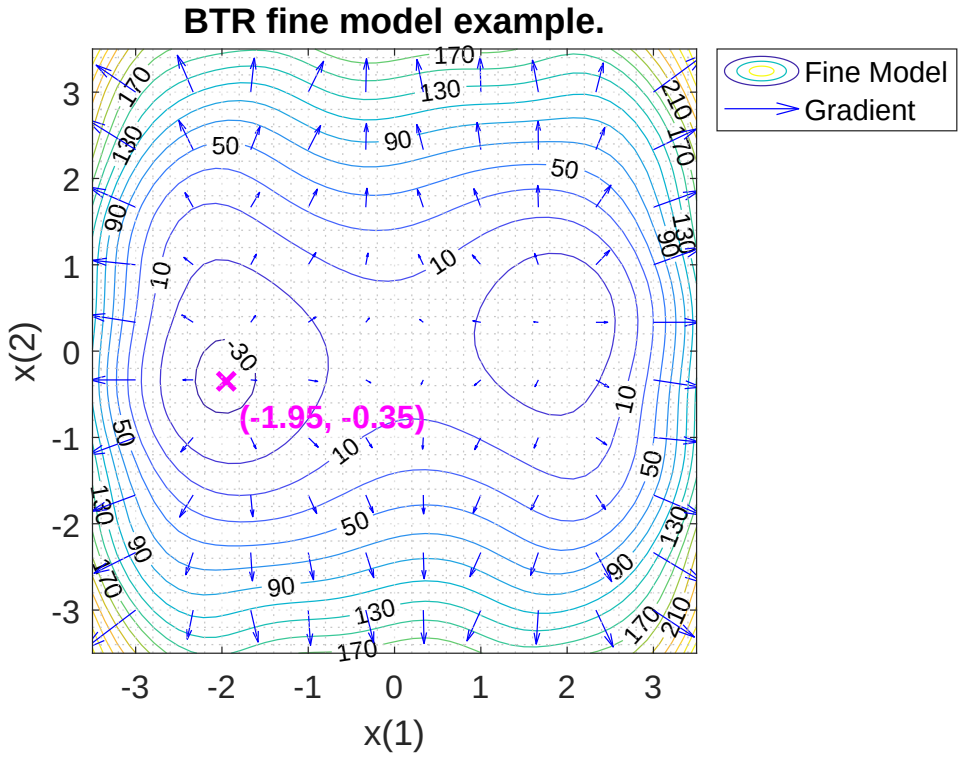


Figure 2.10: Fine model contour plot for B.T.R. example

solution methods are not exceeded. Once the valid region is known, a starting point can be chosen. Although fairly arbitrary, this is a just a point where the fine model can be evaluated. In this example the initial point (iteration 0) and the fine model evaluation is given by

$$\boldsymbol{x}^{(0)} = (0.25, \text{-}1.92) \qquad \boldsymbol{R}_f^{(0)} = \boldsymbol{R}_f(\boldsymbol{x}^{(0)}) = 57.64\,. \tag{2.13}$$

The B.T.R. algorithm is iterative and uses the previous evaluation in future iterations, therefore the value of each parameter and the response is stored. The trust-region is the set of all parameters

$$\mathcal{B}^{(k)} = \{x \,\epsilon\, \Re^n \mid ||\boldsymbol{x} - \boldsymbol{x}^{(k)}|| \leq \Delta^{(k)}\}\,, \tag{2.14}$$

where $\Delta^{(k)}$ is the trust-region radius and $||\circ||^{(k)}$ is an iteration-dependent norm [29, chap. 6.1, pg. 115]. Various norms can be applied, this is discussed later in Section 3.5.5.5. For this example a Euclidean vector norm is used [29, chap. 6.7, pg. 162]. The size of the trust-region radius is worked out as a factor of the entire boundary size. The initial trust-region size is set to

$$\Delta^{(0)} = 1.75\,,$$

centred around point $\boldsymbol{x}^{(0)}$. Conn *et al.* suggest using an iterative approach to determine the initial trust-region radius using the model's Hessian and ensuring that the Cauchy point lies on the boundary [29, chap. 17.2, pg. 784].

2.3.1.2 Model Definition

At this stage only a single point, within the bounded region, is actually known (in a real problem). Various approximations can be made from the fine model that are valid as long as they are evaluated within a *suitably small neighbourhood* around the fine model point. One of the functions of the trust-region is to ensure that the *neighbourhood* of the low fidelity model remains valid, more on this follow in Section 2.3.1.6. A transformation can be applied to the coarse model so that it aligns better to the fine model. If this is done, then the transformed model it is called the surrogate model. The previous section discusses in detail the use of an alignment mechanism. For the simplicity of this example no alignment phase is actually carried out, however, the surrogate term is still used because it is typically the case.

To evaluate the surrogate model at a point away from $\boldsymbol{R}_f^{(0)}$ (the point $\boldsymbol{x}^{(0)}$ where the fine model was evaluated), a trial point $\boldsymbol{s}^{(k)}$ is introduced. Once again, k represents the iteration number. This point is the distance, in each parameter, that is moved away from the original point. The trial step must remain within the trust-region radius. Typically, a local optimiser controls where this step is taken and is discussed in Section 2.3.1.3. For this example steps are taken throughout the entire region to show what the surrogate space looks like and to illustrate how dangerous it can be to allow the *trusted* region to grow. Conn *et al.* build up a surrogate *model* by adopting a quadratic form that uses the last fine model response, its derivative at that point, and if available the second derivative too [29, chap. 6.1, pg. 117]. The quadratic form is used because it is better than a linear approximation and still relatively cheap and easy to optimise. Formally this surrogate model response $\boldsymbol{R}_s$ is defined as

$$\boldsymbol{R}_s(\boldsymbol{x}^{(k)} + \boldsymbol{s}^{(k)}) = \boldsymbol{R}_f(\boldsymbol{x}^{(k)}) + \langle g_k, \boldsymbol{s}^{(k)} \rangle + \frac{1}{2}\langle \boldsymbol{s}, \mathbf{H}_k\, \boldsymbol{s}^{(k)} \rangle\,, \tag{2.15}$$

where $\boldsymbol{R}_f(\boldsymbol{x}^{(k)})$ is the result of the previous fine model evaluation, $g_k = \nabla_{\boldsymbol{x}} \boldsymbol{R}_f(\boldsymbol{x}^{(k)})$ the Jacobian and $\mathbf{H}_k$ the symmetric approximation or Hessian $\nabla_{\boldsymbol{x}\boldsymbol{x}} \boldsymbol{R}_f(\boldsymbol{x}^{(k)})$[29, chap. 6.1, pg. 117]. The angle brackets $\langle \circ \rangle$ represent the Euclidean inner product (dot product)

$$\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{n} a_i\, b_i\,. \tag{2.16}$$

Using this approximation as the surrogate model, and extending trial steps to test every point in the sample space, a contour plot of the model can be developed, see Figure 2.11. The trust-region is shown by a dashed line, the original point where the last fine model was evaluated is shown by a circle and finally, a cross shows the trial point. The selection of a trial point is discussed in Section 2.3.1.3.

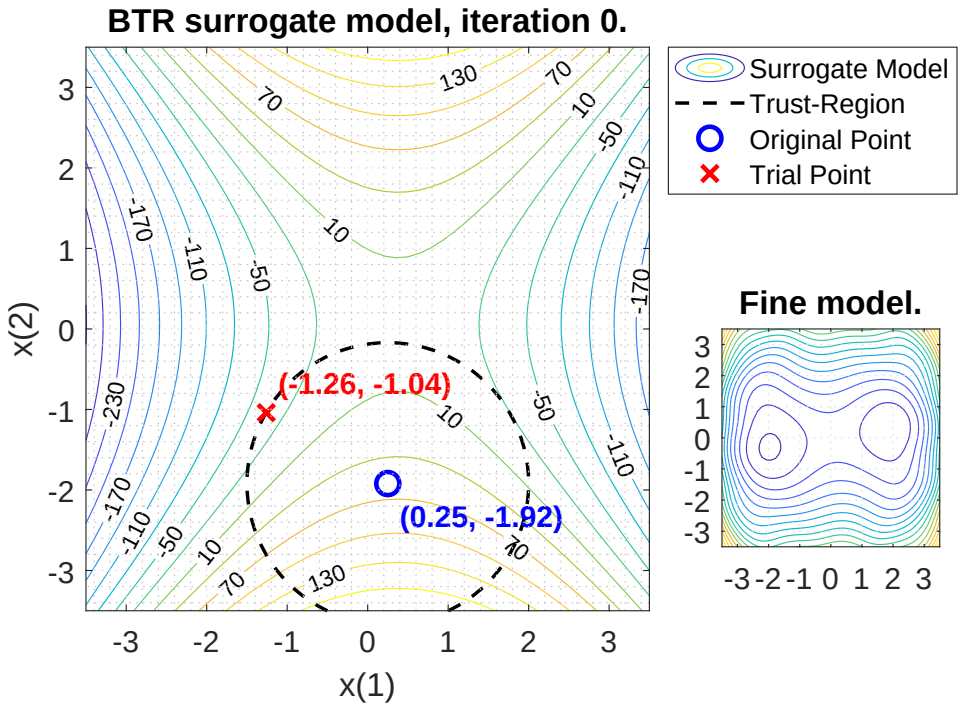


Figure 2.11: Surrogate model $\boldsymbol{R}_s^{(0)}$, based on $\boldsymbol{R}_f^{(0)}$. Fine model mini-map shown on the bottom right.

There are a number of characteristics to note while analysing the surrogate model (2.15), and when comparing the graphs of the surrogate and fine models, Figure 2.11 and Figure 2.10 respectively. At the original point, both the fine and the surrogate model response have the same values. Clearly, this happens because the step size is zero leaving the only non-zero term being the fine model response. This may be trivial to note, but it is an important check to carry out when developing different models. Furthermore, from the graphs, it can be seen that within the neighbourhood of this original point the landscape appears similar between the two. As the step moves further away from this point the surrogate model starts losing accuracy. Towards the east and west it runs steeply downhill whereas in the fine model those edges build up again. This surrogate model is a good approximation, but not sufficient to be used throughout the space/domain. In a real problem, the accuracy of the surrogate model would not be known since there is only one fine model evaluation. It can now be seen how a balance needs to be struck between remaining in a valid region and not wasting time re-evaluating the fine model when calculating a new trial point.

2.3.1.3 Trial Point Calculation

Using the surrogate model space, a local optimisation can be run, minimising $\boldsymbol{R}_s$,

$$\boldsymbol{x}_s^{*(k)} = \boldsymbol{x}^{(k)} + \boldsymbol{s}^{(k)} = \arg \min_{\boldsymbol{x}^{(k)}+\boldsymbol{s}^{(k)}} U\left(\boldsymbol{R}_s(\boldsymbol{x}^{(k)} + \boldsymbol{s}^{(k)})\right), \tag{2.17}$$

where $\boldsymbol{x}_s^*$ or $\boldsymbol{x}^{(k)} + \boldsymbol{s}^{(k)}$ is the *trial point* in surrogate space. The trial point is made up of the original point $\boldsymbol{x}^{(k)}$, that is kept the constant, and the trial step $\boldsymbol{s}^{(k)}$, that is adjusted. The trial step $\boldsymbol{s}^{(k)}$ is restricted to the size of the trust-region radius $||\boldsymbol{s}^{(k)}||_k \leq \Delta^{(k)}$ and is an element of $\mathcal{B}^{(k)}$. This restricts the optimiser and quantifies (albeit fairly arbitrary at the first iteration) the assumption of trust. A local optimiser is likely to follow the slope to the lowest point within the trust-region radius. The trial point, at iteration zero, is shown by a red cross in Figure 2.11. The value of the point and the surrogate result are given below:

$$\boldsymbol{x}_s^{*(0)} = \boldsymbol{x}^{(0)} + \boldsymbol{s}^{(0)} = (\text{-}1.26, \text{-}1.04) \qquad \boldsymbol{R}_s(\boldsymbol{x}^{(0)} + \boldsymbol{s}^{(0)}) = \text{-}34.12$$

This is a significant decrease compared to $\boldsymbol{R}_f^{(0)} = 57.64$, see (2.13). There is no guarantee that the fine model will have changed in the same way as the surrogate though. Deciding on whether or not the trial step is valid is described next.

2.3.1.4 Accepting a Trial Point

Once this optimal trial point has been found, the fine model is run at the point too, $\boldsymbol{R}_f(\boldsymbol{x}^{(k)} + \boldsymbol{s}^{(k)})$. The change in response, from $\boldsymbol{x}^{(k)} \rightarrow \boldsymbol{x}^{(k)} + \boldsymbol{s}^{(k)}$, is compared for both the fine and surrogate models. The surrogate model should have improved, else the optimisation failed. However, the fine model could very well have deteriorated. For this example the fine model response is

$$\boldsymbol{R}_f(\boldsymbol{x}^{(0)} + \boldsymbol{s}^{(0)}) = \text{-}10.87\,. \tag{2.18}$$

This is a reasonable improvement to the original fine model evaluation ($\boldsymbol{R}_f^{(0)} = 57.64$), see (2.13). To decide if the trial point should be *accepted or rejected*, a ratio of these changes can be determined. The ratio ρ is defined as

$$\rho^{(k)} = \frac{U(\boldsymbol{R}_f(\boldsymbol{x}^{(k)})) - U(\boldsymbol{R}_f(\boldsymbol{x}^{(k)}) + \boldsymbol{s}^{(k)})}{U(\boldsymbol{R}_s(\boldsymbol{x}^{(k)})) - U(\boldsymbol{R}_s(\boldsymbol{x}^{(k)} + \boldsymbol{s}^{(k)}))}\,. \tag{2.19}$$

Note here that $U(\circ)$ is used to show that it is actually the cost of the response that is used. In this example a simple response is used and a single value is returned, therefore the reference to the cost function is omitted. It is however important to take this into account for instances where the response is itself a function of something (e.g. frequency), then a cost function must be determined, see Section 3.6.1.

In iteration zero of this example, the ratio is calculated as

$$\rho^{(0)} = \frac{\boldsymbol{R}_f(\boldsymbol{x}^{(0)}) - \boldsymbol{R}_f(\boldsymbol{x}^{(0)} + \boldsymbol{s}^{(0)})}{\boldsymbol{R}_s(\boldsymbol{x}^{(0)}) - \boldsymbol{R}_s(\boldsymbol{x}^{(0)} + \boldsymbol{s}^{(0)})} = \frac{57.64 - (\text{-}10.87)}{57.64 - (\text{-}34.12)} = 0.75\,.$$

A constant η_1 is used to decide if the trial point should be accepted or not,

$$\eta_1 = 0.05\,. \tag{2.20}$$

Conn *et al.* suggest that the only way to find a value for η_1 is through experimentation [29, chap. 17.1, pg. 781]. If the change in ρ is greater than η_1 it is considered an improvement and then the step is accepted. Therefore, a criterion for *accepting a trial point* for iteration k, is defined by the following condition,

$$\rho^{(k)} \geq \eta_1\,. \tag{2.21}$$

If this criterion is met, then, the trial point becomes the starting/original point for the next iteration $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \boldsymbol{s}^{(k)}$.

The trial step, in iteration zero, results in a relative improvement (between the fine and surrogate models) that is good enough to be classified as successful, ($\rho^{(0)} > \eta_1$). The surrogate model approximation is sufficient to be used instead of the fine model within the trust-region. It is not known if the radius could have been larger or not. It is possible to increase the radius and try to reuse the existing surrogate model again. However, since there is another fine model evaluation, a new surrogate model can be constructed. This is typically not more costly an operation than evaluating the surrogate model again. Therefore, the iteration count is incremented $k = k+1$ and the algorithm goes back to finding a new *surrogate model definition*.

2.3.1.5 Rejecting a Trial Point

Now that a successful step has been taken a surrogate model is built and iteration one ($k = 1$) begins. The original point in this iteration is taken from the combination of the previous parameter position and the trial step change $\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(0)} + \boldsymbol{s}^{(0)}$. The fine model has already been evaluated at this point ($\boldsymbol{R}_f^{(1)} = \boldsymbol{R}_f(\boldsymbol{x}^{(1)}) = \boldsymbol{R}_f(\boldsymbol{x}^{(0)} + \boldsymbol{s}^{(0)})$) and this becomes the basis for the surrogate model for iteration one, using (2.15). A contour plot of the response of this surrogate model $\boldsymbol{R}_s^{(1)}$ is shown in Figure 2.12, where once again the blue circle represents the original point for this iteration $\boldsymbol{x}^{(1)}$, the dotted circle is the trust-region that is the same size as iteration zero,

$$\Delta^{(1)} = \Delta^{(0)} = 1.75\,,$$

and the red cross represents the optimal trial point. The trial point is evaluated using the same minimisation shown in (2.17).

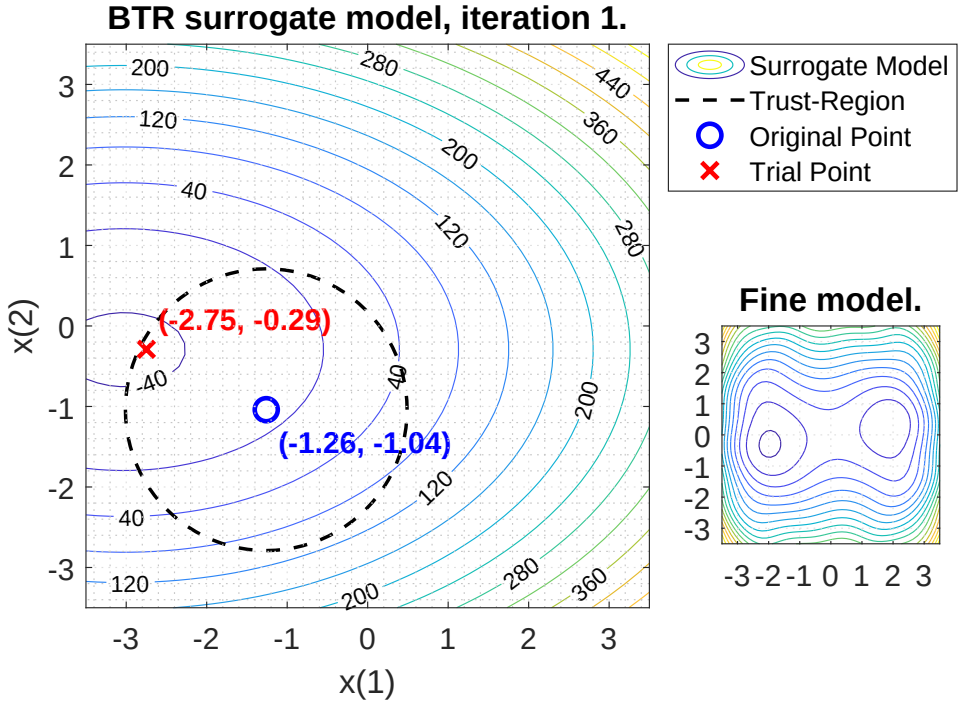


Figure 2.12: Surrogate model $\boldsymbol{R}_s^{(1)}$, based on $\boldsymbol{R}_f^{(1)}$ with the same sized trust-region as in iteration zero. Fine model mini-map shown on the bottom right.

The optimal trial point in the surrogate space, and the response at that point is found to be

$$\boldsymbol{x}_s^{*(1)} = \boldsymbol{x}^{(1)} + \boldsymbol{s}^{(1)} = (\text{-}2.75, \text{-}0.29) \qquad \boldsymbol{R}_s(\boldsymbol{x}^{(1)} + \boldsymbol{s}^{(1)}) = \text{-}43.68\,.$$

This is once again at the edge of the trust-region and appears to be a significant improvement. Now the trial point needs to be evaluated to see if the improvement is seen in fine model space. Evaluation the fine model response gives

$$\boldsymbol{R}_f(\boldsymbol{x}^{(1)} + \boldsymbol{s}^{(1)}) = \text{-}7.59\,. \tag{2.22}$$

This does not appear to be an improvement from the original point for this iteration one, see (2.18). $\rho^{(1)}$ is calculated, using (2.19), to formally decide if this is a successful step,

$$\rho^{(1)} = \frac{\boldsymbol{R}_f(\boldsymbol{x}^{(1)}) - \boldsymbol{R}_f(\boldsymbol{x}^{(1)} + \boldsymbol{s}^{(1)})}{\boldsymbol{R}_s(\boldsymbol{x}^{(1)}) - \boldsymbol{R}_s(\boldsymbol{x}^{(1)} + \boldsymbol{s}^{(1)})} = \frac{(\text{-}10.87) - (\text{-}7.59)}{(\text{-}10.87) - (\text{-}43.68)} = \text{-}0.1\,.$$

Comparing this to the success criteria in (2.21) it is seen that this is a failing trial point, $\rho^{(1)} < \eta_1$. The criteria for *rejecting a trial point* for iteration k can be defined as

$$\rho^{(k)} < \eta_1\,. \tag{2.23}$$

When a trial point is rejected, the algorithm returns to last point where the surrogate model and the fine model were trusted, i.e. the original point from

the iteration. This point is used, once again, as the initial point in the next iteration $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)}$. The region in which the surrogate was trusted needs to be reduced so that the next iteration of the optimiser is confined to a search area that is more reliable.

2.3.1.6 Trust-Region Radius Update

The trust-region radius is updated after the success of the last iteration has been evaluated, see Figure 2.9. Success is measured using ρ which is a ratio of the change in fine model and surrogate model over the last iteration, see (2.19). If the iteration is unsuccessful the trust-region radius is reduced, if the result is moderately successful then the radius is kept the same, and if there is a significant improvement then the radius is increased.

An unsuccessful trial step occurs if (2.21) is satisfied. In this case the trust-region radius is reduced to restrict the optimiser to only evaluate a region in which the surrogate model correctly approximated the fine model. In Figure 2.13, the original point is kept from iteration one, and the radius is shrunk by a factor

$$\gamma_1 = 0.5\,, \tag{2.24}$$

where this constant is proposed in [29, chap. 6.1, pg. 117]. This means that, for iteration two, the trust-region radius is reduced by half

$$\Delta^{(2)} = \gamma_1 \Delta^{(1)} = 0.875\,.$$

The same surrogate model, from iteration one, is evaluated within the new trust-region. The optimal point is found to be

$$\boldsymbol{x}_s^{*(2)} = \boldsymbol{x}^{(2)} + \boldsymbol{s}^{(2)} = (\text{-}1.99, \text{-}0.59) \qquad \boldsymbol{R}_s(\boldsymbol{x}^{(2)} + \boldsymbol{s}^{(2)}) = \text{-}34.78\,.$$

Evaluating the fine model at this point gives

$$\boldsymbol{R}_f(\boldsymbol{x}^{(2)} + \boldsymbol{s}^{(2)}) = \text{-}32.33\,. \tag{2.25}$$

Using the success ratio from (2.19), the ratio for iteration two is

$$\rho^{(2)} = \frac{\boldsymbol{R}_f(\boldsymbol{x}^{(2)}) - \boldsymbol{R}_f(\boldsymbol{x}^{(2)} + \boldsymbol{s}^{(2)})}{\boldsymbol{R}_s(\boldsymbol{x}^{(2)}) - \boldsymbol{R}_s(\boldsymbol{x}^{(2)} + \boldsymbol{s}^{(2)})} = \frac{(\text{-}10.87) - (\text{-}32.33)}{(\text{-}10.87) - (\text{-}34.78)} = 0.89\,,$$

which satisfies $\eta_1 < \rho^{(2)}$, suggesting a successful step. The fine model improvement is similar to that of the surrogate model and can be trusted. Therefore, the trust-region does not need to be shrunk again for the next iteration. With this updated the next iteration $k = 3$ can be started.

A new surrogate model is established and is shown in Figure 2.14. A small step is taken to the optimal point within this trust-region radius,

$$\boldsymbol{x}_s^{*(3)} = \boldsymbol{x}^{(3)} + \boldsymbol{s}^{(3)} = (\text{-}1.95, \text{-}0.35) \qquad \boldsymbol{R}_s(\boldsymbol{x}^{(3)} + \boldsymbol{s}^{(3)}) = \text{-}33.92\,.$$

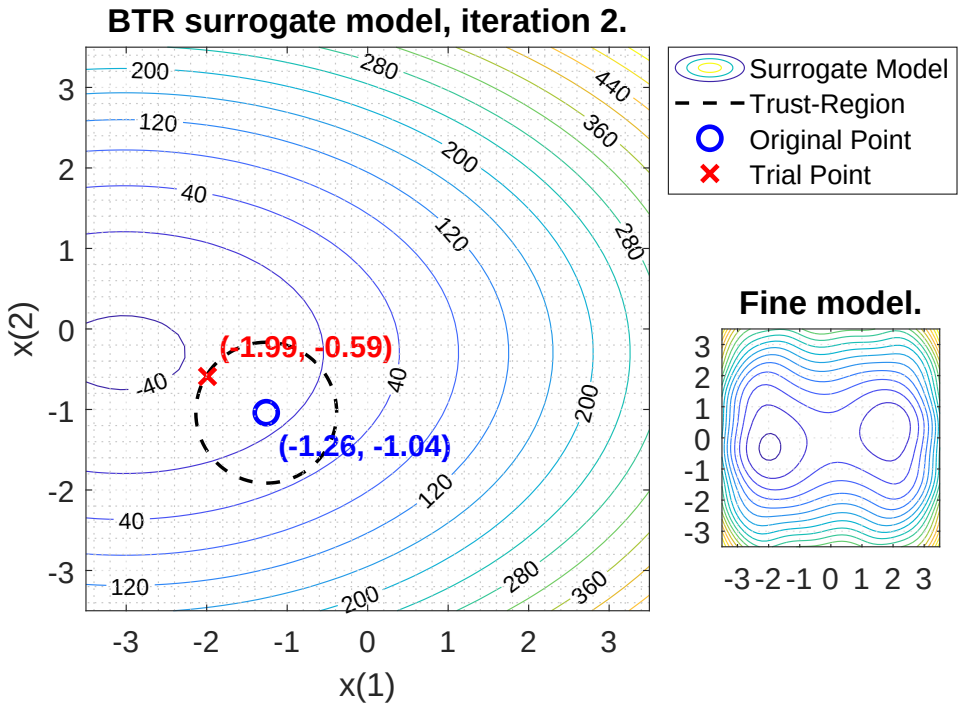


Figure 2.13: Surrogate model $\boldsymbol{R}_s^{(2)}$, based on $\boldsymbol{R}_f^{(2)}$. A reduced trust-region due to previously trial step failing. Fine model mini-map shown on the bottom right.

The fine model is evaluated at this point,

$$\boldsymbol{R}_f(\boldsymbol{x}^{(3)} + \boldsymbol{s}^{(3)}) = \text{-}33.97\,, \tag{2.26}$$

and the success ratio is calculated,

$$\rho^{(3)} = \frac{\boldsymbol{R}_f(\boldsymbol{x}^{(3)}) - \boldsymbol{R}_f(\boldsymbol{x}^{(3)} + \boldsymbol{s}^{(3)})}{\boldsymbol{R}_s(\boldsymbol{x}^{(3)}) - \boldsymbol{R}_s(\boldsymbol{x}^{(3)} + \boldsymbol{s}^{(3)})} = \frac{(\text{-}32.33) - (\text{-}33.97)}{(\text{-}32.33) - (\text{-}33.92)} = 1.03\,.$$

Although the trial step in iteration three is not very large, and the change in fine model response is small, this is considered a *very* successful step. Another success constant η_2 is introduced to describe *how* successful a trial step is:

$$\eta_2 = 0.9\,. \tag{2.27}$$

If the success ratio, (2.19), results in

$$\rho^{(k)} \geq \eta_2\,, \tag{2.28}$$

then the trust-region radius can be increased. There is such a good correlation between the fine model and the surrogate that it is possible that the surrogate model can be evaluated further away from the original point. This is the case

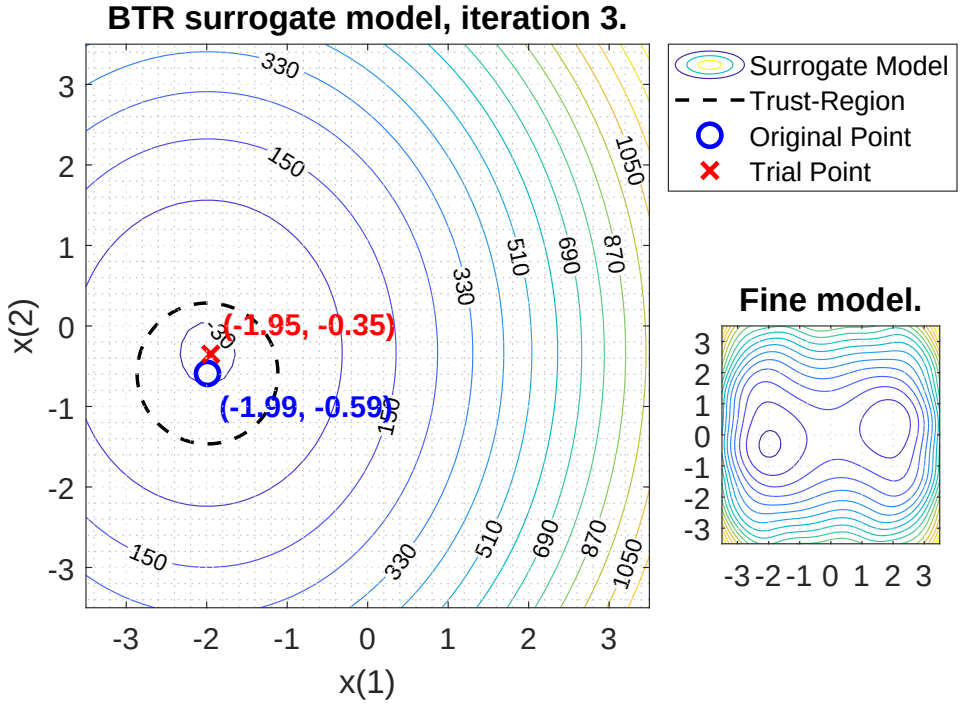


Figure 2.14: Surrogate model $\boldsymbol{R}_s^{(3)}$, based on $\boldsymbol{R}_f^{(3)}$ with the same sized trust-region as in iteration two. Fine model mini-map shown on the bottom right.

for iteration three and the radius is expanded by the same amount that it was shrunk in iteration two. Therefore,

$$\Delta^{(3)} = 2.0 \times \Delta^{(2)} = 1.75\,,$$

and the iteration number is incremented to $k = 4$.

Iteration four is shown in Figure 2.15 with the dotted black line showing the increased trust-region. A new trial step is calculated, but for all intents and purposes, the optimal point has already been reached. This is one way that the algorithm can terminate, see Section 2.3.2 for more on termination criteria. Even though the trust-region radius increased, there is still no better solution.

This is the end of the B.T.R. example. In general the trust-region radius, for the next iteration, is updated as follows:

$$\Delta^{(k+1)} \; \epsilon \; \begin{cases} [\Delta^{(k)}, \infty) & \text{if } \rho^{(k)} \geq \eta_2 \\ [\gamma_2 \Delta^{(k)}, \Delta^{(k)}] & \text{if } \eta_1 \leq \rho^{(k)} < \eta_2 \\ [\gamma_1 \Delta^{(k)}, \gamma_2 \Delta^{(k)}] & \text{if } \rho^{(k)} < \eta_1 \end{cases} . \tag{2.29}$$

2.3.2 Termination criteria

There are several reasons that the trust-region loop should terminate. One mentioned in the section above is that the 'optimal' solution is obtained within

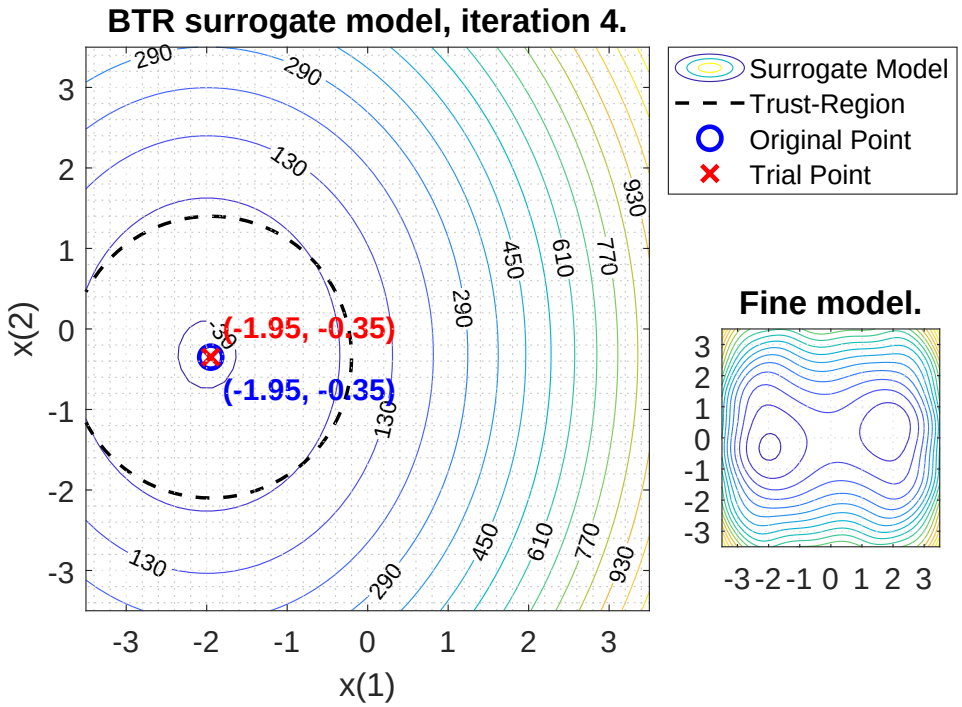


Figure 2.15: Surrogate model $\boldsymbol{R}_s^{(4)}$, based on $\boldsymbol{R}_f^{(4)}$. Trust-region radius increasing due to previous models aligning well. Fine model mini-map shown on the bottom right.

some predefined tolerance.

Another typical termination condition that is important to consider is a maximum number of iterations. If this is not in place and the solution does not succeed, then the operation would continue indefinitely.

Another safeguard is to define a minimum size for the trust-region or the step size that can be taken. If the optimiser has reached a point where meaningful steps are not being made, then the 'best' solution has probably been found.

2.4 Overview

Two space-mapping techniques are descried in great detail. They show the implementation details in isolation without being complicated by other design decisions. The original approach places shows how the space-mapping approach started and through the coming chapter is seen how it has developed.

The trust-region is a crucial development that can provide robustness through various device optimisations. The detailed, example lead approach allows the reader to become familiar with the topic before it is interwoven into the broader automated space-mapping optimisation system.

Chapter 3

Methodology

To achieve an automated and robust system space-mapping techniques and convergence safeguards need to be combined and presented to the user in an encapsulated form. The system needs to be flexible enough to handle a variety of problems and different solvers but simple enough to be picked up by a new user. Extendibility and maintainability of the system must also be kept in mind.

The generalised implicit space-mapping (GISM) framework that Koziel *et al.* present in their 2006 paper [39] is a starting point for how to achieve such a system. MATLAB is also used as the basis for their space-mapping framework (SMF) using number of solvers/drivers including Sonnet's **em**, MEFiSTo, Aligilent's ADS and FEKO [39]. There are several steps that a user must follow to set up a model in the SMF. To set up the problem arguments are passed to the SMF to configure it, a starting point and design variables are given, frequencies of interest are specified and the type of space-mapping is chosen [39]. Once the fine and coarse models have been specified a user-interface is available to adjust built-in trust-region specific options and initiating executions [39]. Concepts outlined in the Koziel *et al.* SMF are used as a basis for the custom implementation introduced below.

Within this chapter a custom implementation that pulls together four different space-mapping techniques and wraps them within a trust-region enhanced optimisation routine. The space-mapping techniques include those presented in Figure 2.1, input (ISM), output (OSM), implicit (ISM) and frequency space-mapping (FSM). The basic trust-region (BTR) method presented in Section 2.3.1 forms the basis for the safeguard enhancement. The sections of this chapter are presented in the order of operation of the algorithm. This allows the reader to have the base that is required to understand each following step. An overview of the algorithm is shown in Figure 3.1. Initialisation defaults and normalisation of design parameters is discussed first in Section 3.1 and 3.2 respectively. Ways to find a suitable starting point within the design space are discussed in Section 3.3. An option to pre-populate the workspace is also presented in this section. It is useful for stopping and starting the

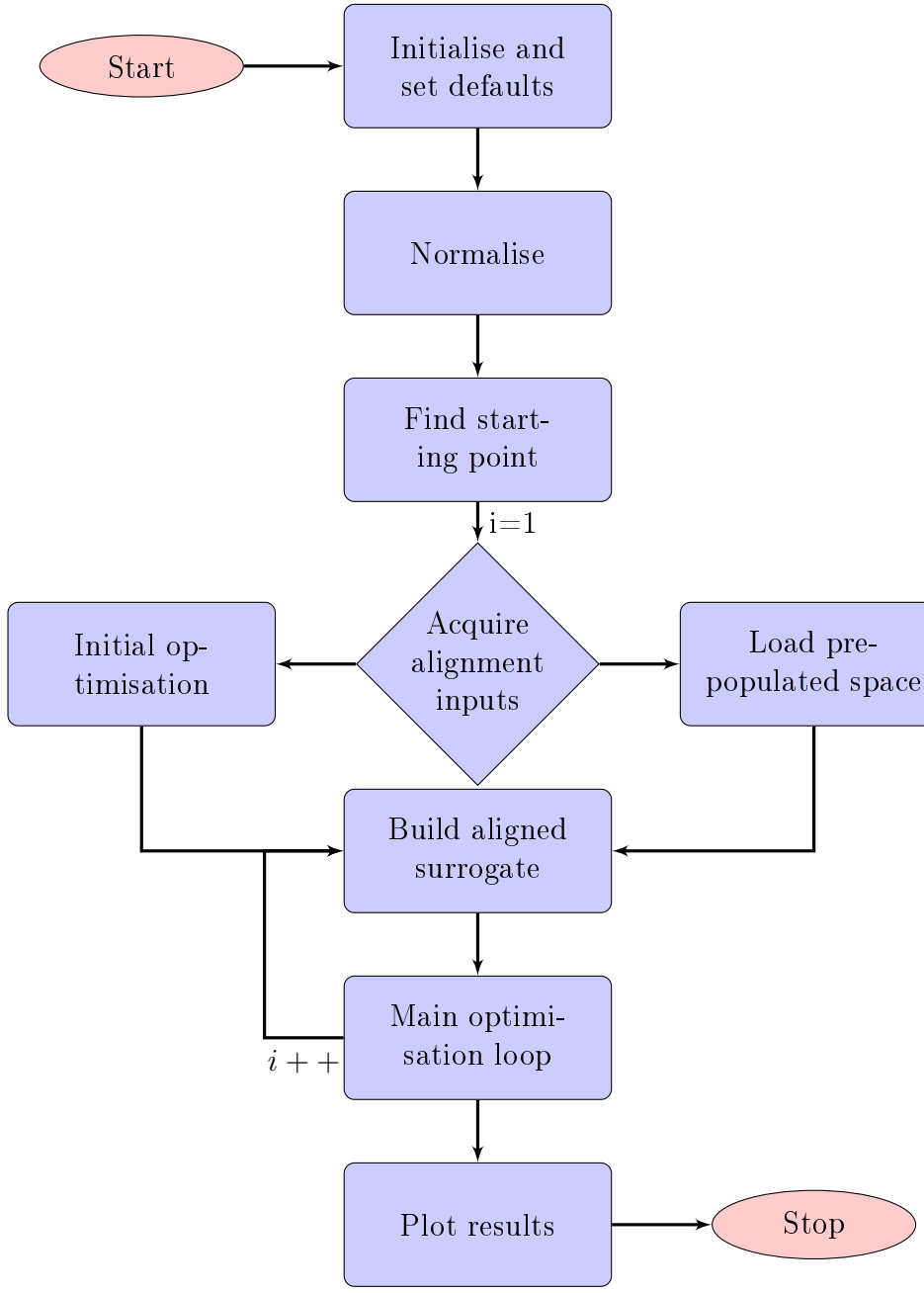


Figure 3.1: Overview of custom space-mapping algorithm.

system or to change the SM approach initially chosen. The initial fine model simulation run is discussed in Section 3.4.1 following which the process for building a surrogate model and carrying out alignment is detailed. Different space-mapping techniques and how they interface with various MATLAB optimisation routines is presented. The main optimisation loop and the integration of the trust-region is discussed in Section 3.6. Finally, results are plotted.

Following the formulation of the algorithm an additional three sections

are presented. An interface between the different coarse and fine solvers is described in Section 4.2 and the user-to-framework interface is explained in Section 4.1.

3.1 Initialisation and Defaults

A number of variables/constants are set with defaults so that the user does not need to set them ever time. They are all however configurable.

A tolerance `TolX` is specified as a termination condition. If none of the design variable change more than the tolerance, then the algorithms terminates. The default tolerance is set to

$$TolX = 10^{-2}\,. \tag{3.1}$$

The number of main loop optimisation iterations is capped to `Ni`. If the count `i` meets this value, then algorithm terminates. By default,

$$Ni = 10\,. \tag{3.2}$$

The trust-region loop count `TRNi` is also capped. By default, it is also set to

$$TRNi = 10\,. \tag{3.3}$$

The trust-region default parameters are given the following default values,

$$\begin{aligned} \eta_1 &= 0.05 && (3.4)\\ \eta_2 &= 0.9 && (3.5)\\ \alpha_1 &= 2.5 && (3.6)\\ \alpha_2 &= 0.25 && (3.7)\\ \Delta_{init} &= 0.25\,. && (3.8) \end{aligned}$$

η_1 and η_2 are defined in (2.20) and (2.27) respectively. α_1 and α_2 are constants that govern the rate at which the trust-region changes size [29, chap. 17.1, pg. 782]. Δ_{init} is the initial trust-region radius.

3.2 Normalise Design Parameters

Normalisation allows the optimiser to successfully operate on design parameters that differ by orders of magnitude. A simple normalisation is applied to all design parameters. Design parameters are denoted $\boldsymbol{x}$. The user specified a maximum and minimum values for each of the design variables, these are denoted $\boldsymbol{x}_{max}$ and $\boldsymbol{x}_{min}$ respectively. The normalised design parameters are given by $\boldsymbol{x}_n$ where

$$\boldsymbol{x}_n = \frac{\boldsymbol{x} - \boldsymbol{x}_{min}}{\boldsymbol{x}_{max} - \boldsymbol{x}_{min}}\,. \tag{3.9}$$

This effectively bring the values into a range of zero to one. Clearly the minimum and maximum then become

$$\boldsymbol{x}_{n\,min} = 0 \tag{3.10}$$
$$\boldsymbol{x}_{n\,max} = 1\,. \tag{3.11}$$

3.3 Design Parameter Starting Point

Using only the coarse model space, see Section 4.2.1, a suitable starting point is found. This can improve the initial alignment of the first surrogate model. A global optimisation routine can be run first to evaluate the entire space. A local optimisation is then initiated to find the optimal point $\boldsymbol{x}_c^*$.

Although similar to the main optimisation loop, this is done first as a stand-alone step.

3.3.1 Global Optimisation for Initial Point

A `globOpt` flag is set to control when the global optimiser is used.

- 0: This is the default option and means that no global optimisation is conducted.
- 1: Specifically only the first iteration has a global optimisation run.
- 2: Each iteration goes through global optimisation phase.

Thus, for the initial design starting point run, option one or two results in a global optimisation run. Approach for setting up the global optimiser is discussed in Section 4.1.3. The normalised design parameter upper and lower bounds are passed through to the optimiser and no inequality or equality values are set.

After the global optimisation routine has completed, a local optimisation step is conducted using the design parameters found.

3.3.2 Direct Local Optimisation for Initial Point

The local optimisation step either accepts a starting point set up by the user or from a global optimisation routine. Configuring the local optimiser is discussed in Section 4.1.3. The normalised design parameter upper and lower bounds are passed through to the optimiser and no inequality or equality values are set. The design parameters found from the local optimisation are passed through to the alignment phase.

3.3.3 Initial Point from Externally Specified Surrogate

Here the initial design space optimisation phase is skipped. The surrogate is specified from the outside and no computation is required. This is useful when designing circuits, such as filters, where the optimal response is known. The `useScAsOpt` boolean flag is set to use this feature.

3.3.4 No Optimisation - Align from Specified point

This option mainly used for testing as no initial optimisation is carried out, instead the algorithm starts at the initial point specified by the user. The variable *startWithIterationZero* is used to turn this feature on or off. Once an initial point has been decided on, then the models for alignment can be acquired.

3.4 Acquire Model for Alignment

The coarse model is quick to evaluate but not very accurate. The space-mapping builds up a surrogate model that takes the coarse model and aligns it to a high-fidelity model. At this point in the algorithm there are no evaluations in fine model space. A fine model evaluation is either run now or taken from a save point in a previous space-mapping framework run.

3.4.1 Calculate Fine Model Response at Initial Point

The point chosen in Section 3.3 is where fine model response $\boldsymbol{R}_f$ is calculated. Details of which solvers are available and how they are initialised are given in Section 4.2.2.

3.4.2 Pre-Populate Space

A pre-populated space is a way to continue from a save point from previous run. At the end of each successful alignment phase save point is created. If the iteration is not successful, then the save is not trigged. For a successful iteration a MATLAB log is created containing various useful properties that can be used to carry on where that iteration left off. The iteration count and design variables are streamed. The fine, coarse, surrogate and aligned surrogate responses as well as all the trust-region details are also written out. These variables are also useful when analysing how a particular iteration went and what path the system took to get to a final result. There is enough data available to re-run plotting routines making it easy to change the way graphs are presented without redoing the optimisation runs.

The files are written out to the location where the current MATLAB system run is taking place. The name of the file begins with *SMLog* followed by the

make of the fine model and the date/time. A Once the file has been written out, it can be read in to a new system run using the `prepopulatedSpaceFile` variable. This variable is set to the name of the file to use.

The save point can also be used to change the space-mapping options or optimiser without losing the runs that have already been carried out.

3.5 Building a Surrogate and Alignment Phase

In this phase the optimisation variables revolve around building the surrogate model. This is not to be confused with the main optimisation loop where the *design variables* are the optimisation focus.

Here the space-mapping parameters are established that build up a surrogate model that takes a coarse model response and adapts it to closely resemble the fine model response. This stage is both a building and an alignment step. All four of the space-mapping techniques described in Figure 2.1 are available in this framework. These techniques are output, input, implicit and frequency space-mapping and are explained in detail in the following sections. Their form as well as the way they are initialised, their constraints and how they are used in the alignment process are detailed.

Constraining the optimiser is necessary to ensure that a valid surrogate model is obtained. A maximum and minimum value is known for each design parameter, if a value outside this bound is given as a result that iteration must be excluded. This is in addition to the lower and upper bounds on the space-mapping parameters. The optimisers in use only accept less than or equal to constraints therefore greater than constraints are multiplied with negatives. A general optimisation problem is built up that can be used for both global and local constrained optimisers from the MATLAB Optimisation Toolbox.

After the different space-mapping techniques have been described they are pulled together and used in the parameter-extraction or alignment phase. Here the optimiser is run and the best alignments values are found within the bounds described. The way that the new surrogate model is evaluated is by using error function, options for this step is also outlined in the coming section.

3.5.1 Input Space-Mapping (B & c)

Input parameter space-mapping works directly with the input parameters or design variables, see Figure 2.1a. The multiplicative $\boldsymbol{B}$ and additive $\boldsymbol{c}$ components are applied in the following way

$$\boldsymbol{B}\boldsymbol{x} + \boldsymbol{c}\,, \tag{3.12}$$

where the values in $\boldsymbol{x}$ are the design parameters. $\boldsymbol{B}$ is an $(N_n \times N_n)$ matrix and $\boldsymbol{c}$ is a column vector of dimension (N_n). The resulting parameters are fed

into the coarse model. This finally results in a surrogate response

$$\boldsymbol{R}_s = \boldsymbol{R}_c(\boldsymbol{B}\,\boldsymbol{x} + \boldsymbol{c})\,. \tag{3.13}$$

Within this framework the `getB` flag is used to enable the multiplicative aspect of this technique and `getc` for the additive.

3.5.1.1 Initialisation and Bounds

The multiplicative input space-mapping can be used in three ways:

1. The entire matrix is set to be used. The initial values for the off-diagonal or cross terms are set to -0.5 and the main diagonal is 0.5.

$$\boldsymbol{B}_{init} = \begin{bmatrix} 0.5 & -0.5 & -0.5 \\ -0.5 & \ddots & \vdots \\ -0.5 & \dots & 0.5 \end{bmatrix} \tag{3.14}$$

   The maximum bounds for the $\boldsymbol{B}$ matrix cross terms is 0.5 and on the main diagonal it is 2.0.

$$\boldsymbol{B}_{max} = \begin{bmatrix} 2.0 & 0.5 & 0.5 \\ 0.5 & \ddots & \vdots \\ 0.5 & \dots & 2.0 \end{bmatrix} \tag{3.15}$$

   The minimum bounds matrix is the same as the initial matrix.

$$\boldsymbol{B}_{min} = \begin{bmatrix} 0.5 & -0.5 & -0.5 \\ -0.5 & \ddots & \vdots \\ -0.5 & \dots & 0.5 \end{bmatrix} \tag{3.16}$$

2. Using only the main diagonal, which keeps the off-diagonal terms zero. This results in fewer terms to evaluate. An initial value of 0.5 is also used on the main diagonal,

$$\boldsymbol{B}_{init} = \begin{bmatrix} 0.5 & 0.0 & 0.0 \\ 0.0 & \ddots & \vdots \\ 0.0 & \dots & 0.5\,. \end{bmatrix} \tag{3.17}$$

   This is once again the minimum value and 2.0 for the upper bound

$$\boldsymbol{B}_{max} = \begin{bmatrix} 2.0 & 0.0 & 0.0 \\ 0.0 & \ddots & \vdots \\ 0.0 & \dots & 2.0 \end{bmatrix} \tag{3.18}$$

$$\boldsymbol{B}_{min} = \begin{bmatrix} 0.5 & 0.0 & 0.0 \\ 0.0 & \ddots & \vdots \\ 0.0 & \dots & 0.5\,. \end{bmatrix} \tag{3.19}$$

3. Specific entries in the matrix can be included/excluded. If they are excluded they are simply zero.

The additive input space-mapping term has an initial value of zero,

$$\boldsymbol{c}_{init} = \begin{bmatrix} 0 & \dots & 0 \end{bmatrix}^T \tag{3.20}$$

and bounds directly related to that of the design variables and the multiplicative matrix

$$\boldsymbol{c}_{min} = \boldsymbol{x}_{min} - \boldsymbol{B}_{max}\, \boldsymbol{x}_{max} \tag{3.21}$$

$$\boldsymbol{c}_{max} = \boldsymbol{x}_{max} - \boldsymbol{B}_{min}\, \boldsymbol{x}_{min}\,. \tag{3.22}$$

3.5.1.2 Constraints

For a surrogate to be valid it needs to translate into a valid model in coarse and fine model space. Therefore, the bounds for those spaces must be taken into account here. To do that inequality constraints are applied.

For input space-mapping the following equation form constraints for the space-mapping parameters. The maximum

$$\boldsymbol{B}\boldsymbol{x}^{(i)} + \boldsymbol{c} \leq \boldsymbol{x}_{max} \tag{3.23}$$

and minimum

$$\boldsymbol{B}\boldsymbol{x}^{(i)} + \boldsymbol{c} \geq \boldsymbol{x}_{min}\,.$$

The minimum then becomes

$$-\boldsymbol{B}\boldsymbol{x}^{(i)} - \boldsymbol{c} \leq -\boldsymbol{x}_{min}\,, \tag{3.24}$$

where $\boldsymbol{x}^{(i)}$ are the design parameters from the last fine model evaluation. Equations (3.23-3.24), for input space-mapping, are reorganised and reshaped into the following form

$$\begin{bmatrix} -x_1 & 0 & 0 & \dots & -x_{N_n} & 0 & 0 & -1 & 0 & 0 \\ 0 & \ddots & \vdots & \dots & 0 & \ddots & \vdots & 0 & \ddots & \vdots \\ 0 & \dots & -x_1 & \dots & 0 & \dots & -x_{N_n} & 0 & \dots & -1 \\ x_1 & 0 & 0 & \dots & x_{N_n} & 0 & 0 & 1 & 0 & 0 \\ 0 & \ddots & \vdots & \dots & 0 & \ddots & \vdots & 0 & \ddots & \vdots \\ 0 & \dots & x_1 & \dots & 0 & \dots & x_{N_n} & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} B_{1,1} \\ \vdots \\ B_{N_n,1} \\ \vdots \\ B_{1,N_n} \\ \vdots \\ B_{N_n,N_n} \\ c_1 \\ \vdots \\ c_{N_n} \end{bmatrix} \leq \begin{bmatrix} -x_{1min} \\ \vdots \\ -x_{Nnmin} \\ x_{1max} \\ \vdots \\ x_{Nnmax} \end{bmatrix}. \tag{3.25}$$

The left-hand-side matrix has dimensions $((2N_n) \times (N_n^2 + N_n))$ and is denoted by $\mathbf{LHS_{input}}$. The left-hand-side column vector has dimension $(N_n^2 + N_n)$ and denoted by $\mathbf{vec_{input}}$. Finally, the right-hand-side column vector has dimension $(2N_n)$ and is represented by $\mathbf{rhs_{input}}$. Substituting the new terms into (3.25) gives

$$\mathbf{LHS_{input}}\ \mathbf{vec_{input}} \leq \mathbf{rhs_{input}}\,. \tag{3.26}$$

This form of the constraints is accepted by the optimiser containers described in Section 3.5.5.2.

3.5.2 Output Space-Mapping (A & d)

Output space-mapping applies a correction to the response of the coarse model. Figure 2.1b show a flow diagram that describes this process. The coarse model $\boldsymbol{R}_c$ is evaluated at the specified design parameter position $\boldsymbol{x}$. A multiplicative $\boldsymbol{A}$ and additive $\boldsymbol{d}$ term are then applied to give the surrogate response

$$\boldsymbol{R}_s = \boldsymbol{A}\boldsymbol{R}_c(\boldsymbol{x}) + \boldsymbol{d}\,. \tag{3.27}$$

The `getA` flag is used to select the multiplicative term while `getd` is used for selecting the additive term.

This method can be used in conjunction with other space-mapping techniques. It is applied last, after the effects of the other methods have been applied.

3.5.2.1 Initialisation and Bounds

The multiplicative value $\boldsymbol{A}$ can be applied to the system in two ways:

1. As a single value that is applied to each response point. In this case the $\boldsymbol{A}$ matrix resolves to a single value. It is initialised to

$$\boldsymbol{A}_{init} = 1\,. \tag{3.28}$$

The lower and upper bounds are set to

$$\boldsymbol{A}_{min} = 0.5 \tag{3.29}$$
$$\boldsymbol{A}_{max} = 2.0\,. \tag{3.30}$$

2. A different value for each response point. $\boldsymbol{A}$ then becomes a diagonal matrix of dimension $(N_m \times N_m)$. It is initialised to

$$\boldsymbol{A}_{init} = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & \ddots & \vdots \\ 0.0 & \dots & 1.0 \end{bmatrix}\,. \tag{3.31}$$

The maximum is set to

$$\boldsymbol{A}_{max} = \begin{bmatrix} 2.0 & 0.0 & 0.0 \\ 0.0 & \ddots & \vdots \\ 0.0 & \dots & 2.0 \end{bmatrix} \tag{3.32}$$

$$\tag{3.33}$$

and the minimum to

$$\boldsymbol{A}_{min} = \begin{bmatrix} 0.5 & 0.0 & 0.0 \\ 0.0 & \ddots & \vdots \\ 0.0 & \dots & 0.5 \end{bmatrix} . \tag{3.34}$$

The additive term $\boldsymbol{d}$ is an N_n column vector and is initialised to zeros.

3.5.2.2 Constraints

Output space-mapping parameters do not directly operate on the other space-mapping parameters and a place-holder zero vector,

$$\begin{bmatrix} 0 & \dots & 0 \end{bmatrix} \begin{bmatrix} A_1 \\ \vdots \\ A_{N_m} \end{bmatrix} . \tag{3.35}$$

The left-hand-side row vector has dimension $(1 \times N_m)$, or one, and is denoted $\mathbf{lhs_{output}}$. The left-hand-side column vector has dimension $(N_m \times 1)$, or one, and is denoted $\mathbf{vec_{output}}$. There is no right-hand-side constraint vector added since this is simply a place-holder. This gives

$$\mathbf{lhs_{output}} \ \mathbf{vec_{output}} . \tag{3.36}$$

The additive term is applied right at the end and does not form part of the alignment optimisation. Therefore, there are no associated constraints.

3.5.3 Implicit Space-Mapping (G & $\boldsymbol{x}_p$)

Implicit space-mapping introduces new parameters into the coarse model which gives extra degrees of freedom to align the coarse model to high-fidelity models. As with the previous two space-mapping options, there is an additive and a multiplicative part. Within this framework the `getxp` flag is used to enable the additive part of this technique and `getG` for the multiplicative.

The additive part or pre-assigned parameters $\boldsymbol{x}_p$ introduces extra variables alongside the design variables $\boldsymbol{x}$, see Figure 2.1c. These variables are not within the main design space and are only used when evaluating the coarse model. $\boldsymbol{x}_p$ is a column vector of dimension (N_q).

The pre-assigned parameters can be dependent on design variables, in which case a multiplicative term $\boldsymbol{G}$ is applied to the input design variables. This takes the form

$$\boldsymbol{G}\boldsymbol{x} + \boldsymbol{x}_p \,, \tag{3.37}$$

where $\boldsymbol{G}$ is a matrix of dimension $(N_q \times N_n)$.

3.5.3.1 Initialisation and Bounds

The pre-assigned parameter initial values $\boldsymbol{x}_{p\,init}$ must be set by the user, these values cannot be inferred. If there is no coupling to the design parameters then the bounds, $\boldsymbol{x}_{p\,min}$ and $\boldsymbol{x}_{p\,max}$, are also specified by the user. A new variable $\boldsymbol{p}$ is defined for the bounds because it is possible that they are not simply the limits on the pre-assigned parameters and these concepts should not be confused. This resolves simply to

$$\boldsymbol{p}_{min} = \boldsymbol{x}_{p\,min} \tag{3.38}$$

$$\boldsymbol{p}_{max} = \boldsymbol{x}_{p\,max} \,. \tag{3.39}$$

If, however, there is a dependency on design variables then the bounds include the multiplicative term,

$$\boldsymbol{p}_{min} = \boldsymbol{x}_{p\,min} - \boldsymbol{G}_{max}\ \boldsymbol{x}_{max} \tag{3.40}$$

$$\boldsymbol{p}_{max} = \boldsymbol{x}_{p\,max} - \boldsymbol{G}_{min}\ \boldsymbol{x}_{min} \,. \tag{3.41}$$

There is not typically a significant dependency on existing design variables and so

$$\boldsymbol{G}_{init} = \begin{bmatrix} 0.0 & 0.0 & 0.0 \\ 0.0 & \ddots & \vdots \\ 0.0 & \dots & 0.0 \end{bmatrix} \,. \tag{3.42}$$

The minimum and maximum $\boldsymbol{G}$ terms are defined as follows

$$\boldsymbol{G}_{max} = \begin{bmatrix} 2.0 & 2.0 & 2.0 \\ 2.0 & \ddots & \vdots \\ 2.0 & \dots & 2.0 \end{bmatrix} \tag{3.43}$$

$$\boldsymbol{G}_{min} = \begin{bmatrix} -2.0 & -2.0 & -2.0 \\ -2.0 & \ddots & \vdots \\ -2.0 & \dots & -2.0 \end{bmatrix} \tag{3.44}$$

This is if `getG` equals one. Specifying it as a vector allows specific design variables to be included of excluded. Each values in the N_n vector that is one includes the corresponding variable while zeros exclude it. The entire matrix can be set manually by making `getG` is an $N_q \times N_n$ matrix.

3.5.3.2 Constraints

For implicit space-mapping the constraints are built up from the maximum

$$\boldsymbol{G}\boldsymbol{x} + \boldsymbol{x}_p \leq \boldsymbol{p}_{max} \tag{3.45}$$

and minimum

$$\boldsymbol{G}\boldsymbol{x} + \boldsymbol{x}_p \geq \boldsymbol{p}_{min}\,.$$

The minimum becomes

$$-\boldsymbol{G}\boldsymbol{x} - \boldsymbol{x}_p \leq -\boldsymbol{p}_{min}\,. \tag{3.46}$$

Once again equations (3.45-3.46) are broken up and reshaped to form

$$\begin{bmatrix} -x_1 & 0 & 0 & \dots & -x_{N_n} & 0 & 0 & -1 & 0 & 0 \\ 0 & \ddots & \vdots & \dots & 0 & \ddots & \vdots & 0 & \ddots & \vdots \\ 0 & \dots & -x_1 & \dots & 0 & \dots & -x_{N_n} & 0 & \dots & -1 \\ x_1 & 0 & 0 & \dots & x_{N_n} & 0 & 0 & 1 & 0 & 0 \\ 0 & \ddots & \vdots & \dots & 0 & \ddots & \vdots & 0 & \ddots & \vdots \\ 0 & \dots & x_1 & \dots & 0 & \dots & x_{N_n} & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} G_{1,1} \\ \vdots \\ G_{N_q,1} \\ \vdots \\ G_{1,N_q} \\ \vdots \\ G_{N_q,N_q} \\ x_{p1} \\ \vdots \\ x_{pN_q} \end{bmatrix} \leq \begin{bmatrix} -p_{1\,min} \\ \vdots \\ -p_{Nq\,min} \\ p_{1\,max} \\ \vdots \\ p_{Nq\,max} \end{bmatrix}. \tag{3.47}$$

that is also used in Section 3.5.5.2. Here the left-hand-side matrix, denoted $\mathbf{LHS_{implicit}}$ has dimensions $((2N_q) \times (N_qN_n + N_q))$, the left-hand-side column vector $\mathbf{vec_{implicit}}$ has dimension $(N_qN_n + N_q)$ and the right-hand-side column vector $\mathbf{rhs_{implicit}}$ has dimension $(2N_q)$.

$$\mathbf{LHS_{implicit}}\ \mathbf{vec_{implicit}} \leq \mathbf{rhs_{implicit}}\,. \tag{3.48}$$

3.5.4 Frequency Space-Mapping ($\boldsymbol{F}$)

Alignment is achieved by changing the frequency axis though either applying a scaling factor σ, or shifting it by a factor δ, or a combination of both, see Figure2.1d. Within this framework the `getF` flag is used to enable this technique. In Section 2.2 a detailed example is given using frequency space-mapping for alignment. $\boldsymbol{F}$ is defined as a column vector containing these two factors

$$\boldsymbol{F} = \begin{bmatrix} \sigma \\ \delta \end{bmatrix}. \tag{3.49}$$

3.5.4.1 Initialisation and Bounds

An initial value, where no scaling and no shift is applied,

$$\boldsymbol{F}_{init} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} . \tag{3.50}$$

The scaling term is limited between 0.5 and 2.0,

$$\sigma_{min} = 0.5 \tag{3.51}$$
$$\sigma_{max} = 2.0 \tag{3.52}$$

while this shift bounds are linked to the maximum and minimum frequencies in combination with the scaling terms,

$$\delta_{min} = 0.9 f_{min} - \sigma_{max} f_{max} \tag{3.53}$$
$$\delta_{max} = 1.1 f_{max} - \sigma_{min} f_{min} . \tag{3.54}$$

The maximum and minimum shift includes the 'worst case' scaling so that a suitable shift can still be achieved when scaling is used. These form the maximum and minimum terms for the $\boldsymbol{F}$ vector

$$\boldsymbol{F}_{min} = \begin{bmatrix} \sigma_{min} \\ \delta_{min} \end{bmatrix} \tag{3.55}$$
$$\boldsymbol{F}_{max} = \begin{bmatrix} \sigma_{max} \\ \delta_{max} \end{bmatrix} . \tag{3.56}$$

3.5.4.2 Constraints

Frequency space-mapping constraints relate to the absolute minimum and maximum frequencies

$$[f_{max} \ \ +1]\, \boldsymbol{F} \leq 1.1 f_{max} \tag{3.57}$$
$$[f_{min} \ \ +1]\, \boldsymbol{F} \geq 0.9 f_{min}$$
$$[-f_{min} \ \text{-1}]\, \boldsymbol{F} \leq -0.9 f_{min} , \tag{3.58}$$

where f_{min} and f_{max} are the minimum and maximum frequencies for the given problem. The $\boldsymbol{F}$ column vectors have dimension two. Reshaping and moving equations (3.57-3.58) gives

$$\begin{bmatrix} -f_{min} & -1 \\ f_{max} & 1 \end{bmatrix} \begin{bmatrix} \sigma \\ \delta \end{bmatrix} \leq \begin{bmatrix} -0.9 f_{min} \\ 1.1 f_{max} \end{bmatrix} , \tag{3.59}$$

where the left-hand-side matrix $\mathbf{LHS_{freq}}$ has dimensions (2×2), the column vector $\mathbf{vec_{freq}}$ dimension two and the right-hand-side vector $\mathbf{rhs_{freq}}$ also has dimension two.

$$\mathbf{LHS_{freq}}\ \mathbf{vec_{freq}} \leq \mathbf{rhs_{freq}} . \tag{3.60}$$

3.5.5 Parameter Extraction - Optimisation of Space-Mapping Parameters

Now that all the available space-mapping techniques, their initial values and constraints have been outlined the parameter extraction phase can be discussed. The coarse model must be changed in such a way that the resulting surrogate model response matches that of the fine model response. An optimisation process is used to determine the space-mapping values that are required to achieve this. In Section 3.5.5.5 an error function is constructed to be used in the optimiser. The optimiser minimises the error and this resolves simply to

$$SM^* = \arg\min_{SM} ||\boldsymbol{R}_f - \boldsymbol{R}_s|| \,, \tag{3.61}$$

where $||\circ||$ is a norm and SM are the space-mapping variables for the particular space-mapping techniques chosen. Global or local optimisation routines can be used to achieve efficient and accurate alignment variables.

The optimiser options are discussed next in Section 3.5.5.1. MATLAB accepts the problem in a specific format. This preparation of data for the bounds, constraints and specific options are discussed in Section 3.5.5.2. An argument for normalisation of the space-mapping parameters and for removing fixed parameters are presented in Section 3.5.5.3 and Section 3.5.5.4 respectively. Once all the improvements have been made, the error/objective function is described in detail. The error function operates on complex values to ensure phase information is retained. Different response types are useful to define goals and are converted from the complex values. The supported response types are defined in Section 3.5.5.6. Finally, the optimisation routine is run and special considerations are mentioned.

3.5.5.1 Choice of Optimiser

For all the space-mapping techniques, except for output space-mapping, the coarse model must be run using the new variables. The space-mapping parameters changes to the design parameters can result in the coarse model being evaluated outside of its bounds. To ensure that this is not the case constrains are set for each of the space-mapping parameters and applied to the optimiser used. The constrains are outlined in the previous sections for the different space-mapping types. This limits the optimisers that can be used from the MATLAB optimisation toolbox. The default local optimiser that is used is called `fmincon`.

Before the local optimisation is run, a global routine can be used. Koziel *et al.* recommend always running a global optimisation routine for the very first alignment stage for all surrogate based optimisers [4, chap. 3.3.4, pg. 45]. This gives a good starting point for the local optimiser to work from that is not within a local minimum. A genetic algorithm is used by default for the global optimiser.

MATLAB allows the definition of a general problem that can be used by a number of different optimisers, both global and local. The following section outlines the use of this general problem format for parameter extraction.

3.5.5.2 Prepare the Base Problem for Optimiser

The general MATLAB optimisation *problem* has several parts.

- The *objective function* is the error function that must be minimised. This is discussed in Section 3.5.5.5.
- The initial values of the space-mapping parameters are defined in Sections 3.5.2.1, 3.5.1.1, 3.5.3.1 and 3.5.4.1. They are combined into a column vector,

$$\mathbf{x_0} = \mathbf{init} = \begin{bmatrix} \boldsymbol{A}_{init} \\ \boldsymbol{B}_{init} \\ \boldsymbol{c}_{init} \\ \boldsymbol{G}_{init} \\ \boldsymbol{p}_{init} \\ \boldsymbol{F}_{init} \end{bmatrix} . \tag{3.62}$$

  MATLAB uses $\mathbf{x_0}$ to define the initial values but to avoid confusion with the design parameters **init** is rather used. If a global optimiser is used then the optimal values from that optimisation run are passed through to the local optimiser as its starting point.

- Upper and lower bounds for the space-mapping are specified using **ub** and **lb** respectively. The lower bounds column vector is made up of the minimum values for the various space-mapping techniques,

$$\mathbf{lb} = \begin{bmatrix} \boldsymbol{A}_{min} \\ \boldsymbol{B}_{min} \\ \boldsymbol{c}_{min} \\ \boldsymbol{G}_{min} \\ \boldsymbol{p}_{min} \\ \boldsymbol{F}_{min} \end{bmatrix} . \tag{3.63}$$

  The upper bounds vector is built up in the same way,

$$\mathbf{ub} = \begin{bmatrix} \boldsymbol{A}_{max} \\ \boldsymbol{B}_{max} \\ \boldsymbol{c}_{max} \\ \boldsymbol{G}_{max} \\ \boldsymbol{p}_{max} \\ \boldsymbol{F}_{max} \end{bmatrix} . \tag{3.64}$$

  See Sections 3.5.1.1, 3.5.2.1, 3.5.3.1 and 3.5.4.1 for the maximum and minimum definitions.

- For inequality constrained systems, such as this parameter extraction, $\mathbf{A_{ineq}}$ defines the matrix of linear inequality constant and $\mathbf{b_{ineq}}$ the inequality constraint vector forming

$$\mathbf{A_{ineq}}\ \mathbf{x} \leq \mathbf{b_{ineq}}\,. \tag{3.65}$$

  To avoid confusion with the $\boldsymbol{A}$ space-mapping parameter and input parameters $\boldsymbol{x}$, this equation changes to the following form

$$\mathbf{LHS\ vec} \leq \mathbf{rhs}\,. \tag{3.66}$$

  Here the left-hand-side matrix is made up of all inequality constraint matrices. The matrices are placed consecutively effectively on a big diagonal and offset with zeros. The vectors are appended one after the other. Combining all the constraints from (3.36, 3.26, 3.48, 3.60) and reshaping into a single system a full constraint problem is obtained,

$$\begin{bmatrix} \mathbf{lhs_{output}} & & & \\ \vdots & \mathbf{LHS_{input}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{LHS_{implicit}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{LHS_{freq}} \end{bmatrix} \begin{bmatrix} \mathbf{vec_{output}} \\ \mathbf{vec_{input}} \\ \mathbf{vec_{implicit}} \\ \mathbf{vec_{freq}} \end{bmatrix} \leq \begin{bmatrix} \mathbf{rhs_{input}} \\ \mathbf{rhs_{implicit}} \\ \mathbf{rhs_{freq}} \end{bmatrix}. \tag{3.67}$$

  Even though the constraints are stipulated, the optimiser can evaluate the problem at points outside this range. It should, however, discard these results evaluated outside the constrained space. This means that low fidelity models must be robust enough to handle potentially erroneous evaluations. Additional hard limits can be placed within the model evaluation stage to give warnings through to the user if invalid parameters are requested, see Section 4.2.1.

- Depending on the optimiser being used, *options* can be specified to enable settings or limits. `Display` and `diagnostic` options are useful for debugging evaluating the success of this stage. A value `DiffMinChange` can be set to force a change in variables to be greater than zero for finite-difference gradients [32]. This is useful when the parameters start around zero and initial step sizes are not known.

3.5.5.3 Normalise Alignment Parameters

It is desirable for all the parameters to be in a similar range compared to one another, as well and not being too large or small. If there is an outlier, the step size within the optimiser may be distorted. It is feasible that one length in system may change by a factor of 0.1 while another in the range of 10.0. This is already two orders of magnitude different. Furthermore, even if all the parameters are of a similar order, values that are very large 10^3 or

very small 10^{-3} can also result in difficulties finding a suitable step size. The same step size is applied to all degrees of freedom and if one requires very large changes then it is not possible to make small changes required by others or vice versa. For this alignment/parameter extraction phase this is very important especially because some parameters, like $\boldsymbol{c}$ and $\boldsymbol{x}_p$, are directly linked to the design variables.

The lower bounds **lb** and upper bound **ub** of each of the space-mapping parameters can be used to scale the parameters to a consistent range. The optimisation routine can use these parameters and a conversion back to the actual values can be done when they are applied to the model and the errors are calculated.

A general approach for normalisation, for each parameter v, is undertaken:

1. Calculate a delta Δ_v term,

$$\Delta_v = v_{max} - v_{min}\,. \tag{3.68}$$

2. Subtract minimum from value and divide by delta,

$$v_n = \frac{v - v_{min}}{\Delta_v}\,. \tag{3.69}$$

3. The lower bound is set to zero,

$$\mathrm{lb}_v = 0\,. \tag{3.70}$$

4. The upper bound is set to one,

$$\mathrm{ub}_v = 1\,. \tag{3.71}$$

The different space-mapping parameters are normalised as follows:

- $\boldsymbol{A}$ is an output space-mapping parameter and applied to the model separately at the end, so it is not normalised. Values of $\boldsymbol{B}$ typically reside in the range of negative two and positive two which is a reasonably good slope and it, therefore, not normalised either. $\boldsymbol{G}$, like $\boldsymbol{B}$ is a slope but is often very close to zero, therefore, it is multiplied by 10.
- $\boldsymbol{c}$ and $\boldsymbol{x}_p$ are normalised directly against the bounds of $\boldsymbol{x}$ and $\boldsymbol{x}_p$ respectively.
- For frequency space-mapping the scaling factor σ remains unchanged, but the additive δ term is normalised using the minimum and maximum frequencies ($\Delta = f_{max} - f_{min}$).

Inequality constraints matrix **LHS** values that correspond to $\boldsymbol{c}$, $\boldsymbol{x}_p$ and δ also needs to be normalised. Values on the diagonals are set to Δ_x, Δ_{x_p} and Δ_F respectively. The inequality vector **rhs** also needs to be adjusted. A vector of the minimum values is subtracted from the vector,

$$\mathbf{rhs} - \begin{bmatrix} \text{-}\boldsymbol{x}_{min} \\ \boldsymbol{x}_{min} \\ \text{-}\boldsymbol{x}_{pmin} \\ \boldsymbol{x}_{pmin} \\ \text{-}f_{min} \\ f_{min} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{\Delta_x} \\ \mathbf{0} \\ \mathbf{\Delta_{x_p}} \\ \mathbf{0} \\ \mathbf{\Delta_F} \end{bmatrix}, \tag{3.72}$$

where **rhs** is expanded in (3.67).

Once normalisation has been completed, the problem could be passed through to the optimiser. De-normalisation takes place when the model is run and the error function is evaluated, see Section 3.5.5.5.

3.5.5.4 Removed Fixed Parameters

Unnecessary parameters add extra complexity to the alignment optimisation problem. This is done by removing parameters that have the same upper and lower bounds. These bounds can be set by the user which allows overriding default behaviour and removing particular parameters.

When bounds are set equal, the bounds themselves are removed from the problem. The corresponding inequality constraints are also removed from the **LHS** and **rhs**. Finally, the parameter is removed from **vec**. Reconstruction of the problem is required before the model is run and errors are calculated. Even though a parameter may not be optimised it is still required for the surrogate model to be built up correctly. The initial values are used and reshaping of the space-mapping parameter matrices requires that all the parameters are in place. The parameters cannot be excluded right from the beginning because the full problem is required at the evaluation stage.

Now that the space-mapping parameters have been reduced and normalised the optimiser can evaluate the error/objective function.

3.5.5.5 Error Function Definition

The objective of the optimiser is to minimise the error function. It is the difference between the fine and surrogate model responses. The closer the surrogate response is to that of the fine model the smaller the error. The surrogate is evaluated by taking the coarse model response and applying the chosen space-mapping parameters. Space-mapping parameters are varied through the optimisation until the lowest possible error is found.

First of all the optimisation problem must be converted back into the correct form. De-normalisation and reconstruction of fixed parameters needs to take place, see Section 3.5.5.3 and 3.5.5.4 respectively.

The response of the evaluated surrogate is defined as $\boldsymbol{R}_s$, the fine model response $\boldsymbol{R}_f$ and the coarse model as $\boldsymbol{R}_c$. The responses can be made up of a number output parameters for example S-parameters and gain. Alignment must be carried out over all the output parameters. A weighting can be specified to allow one output parameter to have higher priority or for one to have a lower priority. The weighted errors are added together to give a total error of this surrogate iteration which is passed back to the optimiser.

Specific frequency bands within an output parameter can be highlighted. This is typically done over goal regions where the rest of the response is not as important. This mask/weighting `errW` is applied over the difference between the fine and the surrogate models,

$$\mathbf{diffR} = \mathbf{errW} \cdot (\boldsymbol{R}_f - \boldsymbol{R}_s)\,, \tag{3.73}$$

where this is done individually for each output parameter. The weighting does not have to be binary and specific bands can have a higher importance than others.

A norm is then applied to the differences at each weighted frequency point **diffR**. The type of norm used can be specified by the user but typically a 1-norm (L1 norm) is used here on the result differences.

A 1-norm, for an arbitrary vector $\mathbf{y}$, is defined as

$$||\mathbf{y}||_1 = \sum_{m=1}^{N_m} |\mathrm{y}_m|\,, \tag{3.74}$$

where N_m is the number of points in the vector. A 2-norm, or Euclidean distance, for a vector is defined as

$$||\mathbf{y}||_2 = \sqrt{\sum_{m=1}^{N_m} |\mathrm{y}_m|^2}\,. \tag{3.75}$$

A p-norm, for a vector is defined as

$$||\mathbf{y}||_p = \left(\sum_{m=1}^{N_m} |\mathrm{y}_m|^p\right)^{\frac{1}{p}}\,, \tag{3.76}$$

where p is an integer greater than one.

The norm for each output parameter is multiplied by the weighting scale factor and added together ready for the next stage.

Better results are found when calculating the error using complex response values rather than converting them to dB or taking the absolute values. A smoother graph has fewer outliers and is less likely to skew the norm. The 1-norm is also more robust at handling outliers. For a 2-norm, outliers may end

up dominating the error contributions. In the same way very steep/quickly varying results may be difficult to align because even just a small change will result in a large error. If a graph has a small steep section, for example an edged or a deep null, then that small section may result in a large error that dominates the alignment criteria. It is often useful to use the `errW` to skip or reduce the effect of such regions.

The framework allows the error to be calculated from multiple fine models. This allows a surrogate to be built that is valid over a large region. Extra fine model evaluation points can referred to as infill points and the process of using many points for alignment can be referred to as multi-point alignment [4] [30, chap. 3.4, pg. 55]. N_c is the number of input point cells/fine models available. To achieve this the error of all, or some (depending on the options chosen), of the fine model responses are evaluated against the current surrogate. That is to say the current surrogate is evaluated at a previous design parameter point and compared to that corresponding fine model. Once again a weighting can be applied to error from previous models, for example having the last fine model counting more than the rest. A better surrogate model can be obtained by including the extra accurate fine models available. However, including all of them can make the evaluation slow and give skewed results. The number of space-mapping unknowns can help decide how many fine models to use because that is how many unknowns there are for the alignment optimisation to solve. The SM framework option to set this is `wk`. Its options are:

- `wk` - empty: Only use most recent fine model.
- `wk` - length one: Define `NSMUnknowns`.
  - `wk` $= 0$: Use a maximum of `NSMUnknowns` fine models of N_c. Each is weighed as one. `NSMUnknowns` starts off as zero below.

```
if getA == 1
  NSMUnknowns = NSMUnknowns + 1;
elseif getA == 2
% Diagonal
  NSMUnknowns = NSMUnknowns + 1*Nm;
end
if getd == 1
  NSMUnknowns = NSMUnknowns + Nm*1;
end
if getB == 1 % Full
  NSMUnknowns = NSMUnknowns + Nn*Nn;
elseif getB == 2 % Diagonal
  NSMUnknowns = NSMUnknowns + Nn*1;
else % Custom diagonal
  NSMUnknowns = NSMUnknowns + sum(getB);
end
if getc == 1
  NSMUnknowns = NSMUnknowns + Nn*1;
end
if getG == 1 % Full
  NSMUnknowns = NSMUnknowns + Nq*Nn;
else % Custom diagonal
  NSMUnknowns = NSMUnknowns + (sum(getB)*Nq);
end
if getxp == 1
  NSMUnknowns = NSMUnknowns + Nq;
end
if getF == 1
  NSMUnknowns = NSMUnknowns + 2;
end
```

  - `wk` $\neq 0$: Use all N_c models available but weight them as specified. The weighting is applied in a power form:

```
wk = wk.^[1:Nc];
```

- `wk` - length greater than one: The vector is taken as is an applied to the associated fine model to surrogate differences.
- Default - Assign a weighting of one to all N_c fine models available and use them all.

The weighted errors from the different fine model comparisons are also summed together and this is then returned to the optimiser. As mentioned before, if possible, complex values are used to calculate the error. The user can however specify a response type that is only single values. This same error

calculation is applied regardless of the response type. The response types are discussed next.

3.5.5.6 Response Types

Although complex values are used internally wherever possible, the requests viewed by the user can be specified. The `goalResType` variable used to switch between values. It takes the following form for S-parameters

```
Sb,a_unit
```

where the unit is extracted and used for conversions. Supported unit types are:

- `complex`
- `dB`
- `deg`
- `real`
- `abs`
- `imag`
- `angle`

A `Gen` option is also available that can be used for field values.

3.5.5.7 Do Optimisation

All the tools are in place to run the optimiser and retrieve the space-mapping parameters that bring the surrogate model response as close to the fine model response as possible. The base problem outlined in Section 3.5.5.2 is passed through to a general method that switches on the `problem.solver` variable. Local options that are set up include

- `fmincon`: A MATLAB routine available in the Optimisation Toolbox that attempts to minimise constrained nonlinear multivariable problems [38]. As a constrained optimiser this routine is useful to use for parameter extraction, but can also be used as the main optimisation loop optimiser.
- `fminsearch`: A MATLAB routine that uses a simplex search method used for unconstrained multivariable problems. This makes it useful for the main optimisation loop, but not for parameter extraction.
- `fminsearchcon`: Based on `fminsearch`, but with bound constrained functionality [40]. With the addition of constraints, this is useful for parameter extraction.

The global optimisers include

- `ga`: A Genetic Algorithm that is part of the MATLAB Global Optimisation Toolbox [32].

- `patternsearch`: Pattern Search is a routine that looks for the minimum using an adaptive mesh technique. It is also part of the MATLAB Global Optimisation Toolbox [32].

Both of these global optimisers accept constrains which makes then useful for parameter extraction phase and within the main optimisation loop.

If normalisation is used, then the space-mapping parameter returned must be de-normalised and if fixed parameters were removed then they must be set.

The additive output space-mapping value is not applied during the optimisation and if selected is applied right at the end. The direct difference between the fine and aligned surrogate thus far $\boldsymbol{R}'_s$ is calculated and becomes the value for $\boldsymbol{d}$.

$$\boldsymbol{d} = \boldsymbol{R}_f - \boldsymbol{R}'_s \tag{3.77}$$

The full final surrogate model $\boldsymbol{R}_s$ is built and returned to the main optimisation loop.

3.5.5.8 Special Considerations for Frequency Space-Mapping

Phase is not taken into account for the cost/error function of frequency space-mapping evaluations. The coarse and surrogate responses are therefore converted to decibels. For values at deep nulls this will push them even deeper and infinity values may be encountered. Many optimisers are not able to handle this therefore, an interpolation to get rid of these undesired values is required.

To do this the position of infinity values (or other NaN values potentially) are found. An array of *clean* values is then established ignoring the incompatible values. The results at these locations are then extracted along with the actual frequencies at those positions. A *griddedInterpolant* is constructed and given the clean result along with the frequency vector. Finally, the interpolant is used to overwrite the incompatible values in the original response.

3.6 Main Optimisation Loop

The surrogate model described in the previous section is used to evaluate the system at different points in the design space. The system is evaluated through the use of two different loops. Figure 3.2 gives an overview of how they interact. The decision diamonds represent points where the algorithm can terminate but also other success/failure criteria that is discussed later in this section.

Termination criteria can be met within the main optimisation loop or the trust-region loop. The main loop iterates on variable i and the trust-region uses k. If at either point the number of iterated exceeds N_i or N_k respectively, then current outputs are returned. Other termination scenarios and criteria are discussed in Section 3.6.3.

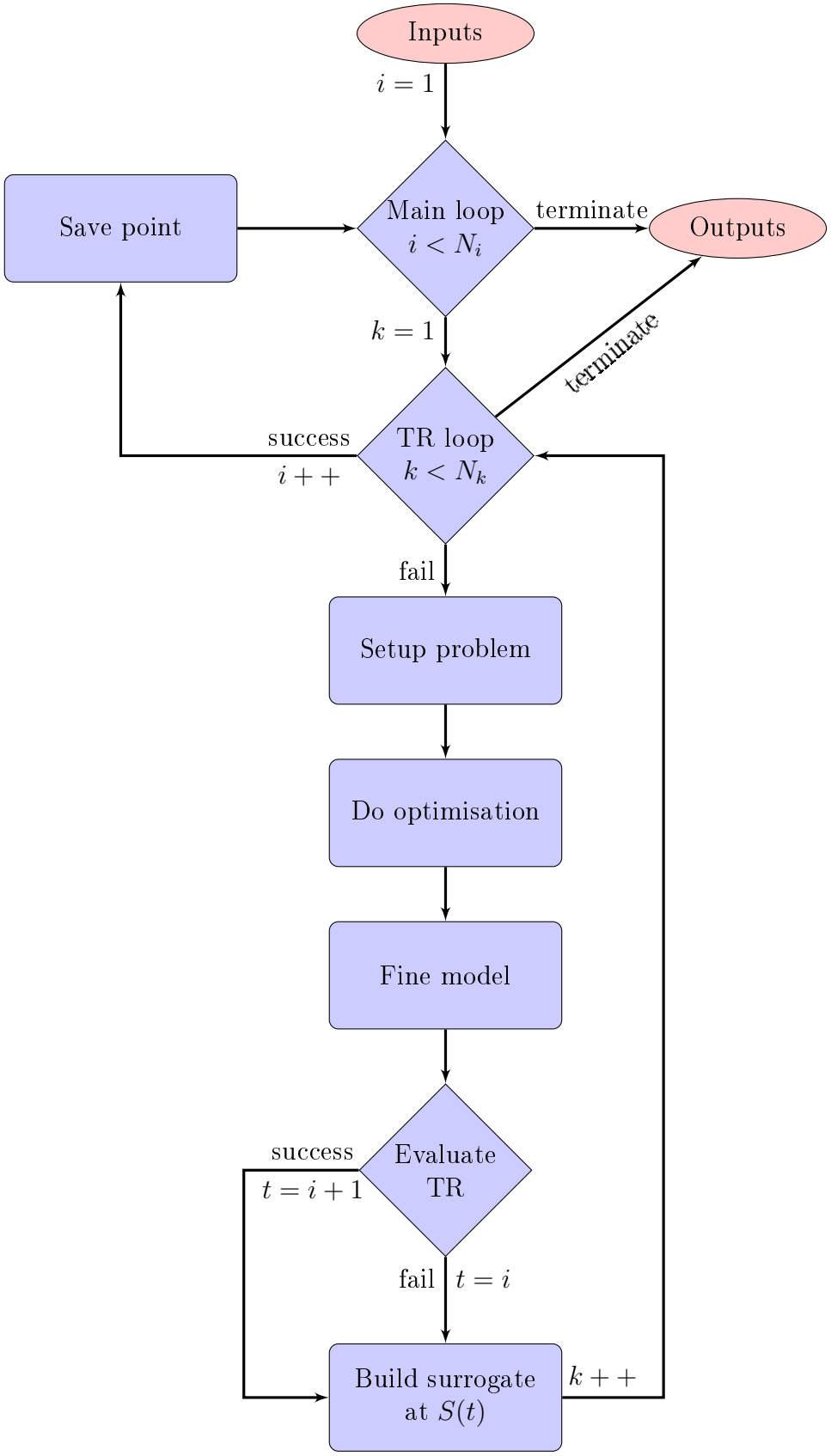


Figure 3.2: Detailed main optimisation loop flow diagram. Inputs from initialisation of defaults and building up an initial aligned surrogate. Outputs go to plotting and post processing.

An initial starting point and surrogate model are required as inputs to the main loop. These and other inputs are discussed in Section 3.6.2 below. Once the main optimisation and trust-region loops have been initialised the optimisation problem is set up. A different optimiser can be used to that of the alignment described in Section 3.5.5.2. Section 3.6.4 below outlines building the design space optimisation problem and running it. The objective/error function and goal types that the user can specify are discussed in Section 3.6.1.

The optimiser specifies a new position which is evaluated running a coarse model and the latest surrogate model. The trust-region framework limits the parameter space ensuring that the surrogate model does not drift out of sync with the fine model. This is done by specifying bounds on the optimiser. Once a new optimum point is found, the surrogate model is updated and the next iteration commences unless under termination conditions. This phase is discussed in Section 3.6.6. Lastly the outputs are discussed and an overview is given.

3.6.1 Goals Types and Error/Objective Function

The objective of an optimiser is typically to minimise a particular quantity (reduce its cost). Within the EM context a user is likely to, for example, want to acquire a particular S-parameter response for a subset of the frequency range, or perhaps a specific gain. The response types that are catered for are discussed in Section 3.5.5.6. Different `goalTypes` can be used to define a particular goal, the different type that this system is configured to accept is discussed later in this section. Multiple goals can be specified for the same response type forming an N_g row vector of the goal types, where $_g$ represents the number of goals. The options for the goals are also specified as N_g row vectors where each row corresponds to the goal type specified. Goal members include

- `goalResType`, the response type that this goal applies to .
- `goalWeight` which scales the particular goal. The default it one.
- `goalValue` is value of the goal on the response. For example for an `S1,1_dB` result type, -20 may be a goal value for a less or greater than response type.
- `goalStart` and `goalStop` are the start and end frequencies over which the goal applies.
- `goalCent` is the centre point if a bandwidth goal type is used.
- `errNorm` a particular error norm can be specified for each goal type if need be. Integers or `inf` are accepted.

The objective/error function used by the optimiser is based on the cost of the total surrogate model responses having the relevant goals applied. Firstly, the design variables are received in normalised form and are denormalised, see Section 3.2. The response type for each response (of N_r responses) is queried and the surrogate is evaluated.

Once the responses have been determined, the cost of the function is calculated. Here a cost is calculated for each goal N_g. The goal has an associated response type that is then used to retrieve the correct response for the collection. Response points falling between the start and end or centre goal frequencies are defined as $\boldsymbol{R}_{\text{valid}}$ and are filtered out from the rest of the points by setting the rest to zero. The cost of each goal $\mathbf{c}_0$ is calculated though a couple of steps. Firstly, the difference, at each point, between the goal value `goalValue` and the valid response points must be calculated, let this be defined as $\mathbf{y}$. Secondly, the norm of the differences is taken to reduce the cost from each goal to a single value. `norm` is a MATLAB function and is described in Section 3.5.5.5. The available `goalTypes` and their calculations are defined below:

- `lt` - less than:

```
y = Rvalid - goalValue;
y(y < 0) = 0;
c0 = norm(y, errNorm);
```

- `gt` - greater than:

```
y = Rvalid - goalValue;
y(y > 0) = 0;
c0 = norm(y, errNorm);
```

- `eq` - equal to:

```
if length(goalValue) == length(Ri.r)
  y = Rvalid - goalValue(iStart:iStop);
else
  y = Rvalid - goalValue;
end
c0 = norm(y, errNorm);
```

- `eqPhaseTune` - softer than `eq`, hard tunes the first frequency phase to be equal:

```
pDiff = angle(Rvalid(iCent)) - angle(goalValue(iCent));
Rvalid = Rvalid.*exp(-1i.*pDiff);   % Force first frequency
    phases equal at least...
if length(goalValue) == length(Ri.r)
  y = Rvalid - goalValue(iStart:iStop);
else
  y = Rvalid - goalValue;
end
c0 = norm(y, errNorm);
```

- `minimax` - the maximum of the response:

```
c0 = max(Rvalid);
```

- `bw` - bandwidth:

```
y = Rvalid - goalValue;
iValid = find(y < 0);
if isempty(iValid)
  c0 = Nm + min(y);
else
  % Get the lowest in band index
  i1 = iValid(1);
  % Get the highest in band index
  i2 = iValid(end);
  % Get number of indexes to estimate a sensible penalty
    factor
  Ni = i2-i1;
  % New valid region response
  yi = y(i1:i2);
  yi(yi < 0) = 0;
  if max(yi) == 0
    b = 0;
  else
    b = 10*Ni./max(yi).^2;
  end
  iCent = round(Nm/2);
  c0 = -min(i2-iCent,iCent-i1) + b*norm(yi,2);
end
```

where N_m is the number of output responses.

- `nPeaks` - number of peaks in the response:

```
pks = findpeaks(Rvalid);
c0 = (goalValue - length(pks))./goalValue;
```

- `peakVal` - equal peak value:

```
pks = findpeaks(Rvalid);
y = pks - goalValue;
c0 = norm(y, errNorm);
```

This gives a cost c_0 for one goal. The cost is scaled by the goal weight and added to the total cost

$$c_{sum} = \texttt{goalWeight} \times c_0 \,.$$

Once the total cost of all the goals has been determined, the averaged is calculated. This is done by summing all the weights w_{sum} and dividing the total cost by this value

$$\text{cost} = \frac{c_{sum}}{w_{sum}} \,.$$

The final cost can be returned to the optimisation loop.

3.6.2 Inputs to the Main Optimisation Loop

Figure 3.2 start at the top with inputs into the system. These are values set up before the loop starts depending on the different options that the user specifies.

In Section 3.4.1 the process for acquiring an initial fine model is outlined, given the initial stating point set in Section 3.3. The fine model evaluation and a coarse model has been used to create an aligned surrogate model as discussed in Section 3.5. This initial surrogate model along with the goals specified by the user (Section 3.6.1) and the error function are passed through to the main loops where they are used for the design space optimisation.

Default values for tolerances, termination criteria and trust-region constants are introduced in Section 3.1. Termination condition are discussed next.

3.6.3 Termination Criteria

The main optimisation loop terminates if any one of the following criteria are met at the start of an iteration.

- Main loop iteration count i reaches a predefined maximum N_i, i.e.

$$i \leq N_i \,. \tag{3.78}$$

- Trust-region iteration count k reaches a predefined maximum N_k, i.e.

$$k \leq N_k \,. \tag{3.79}$$

- The cost of the fine model is reduced to zero i.e. the specification is reached for all the specified goals, see Section 3.6.1.

- A predefined tolerance in $\boldsymbol{x}$ is met. If $\boldsymbol{x}$ does not move a meaningful amount, then the optimisation runs should stop. The tolerance is defined as the L_2 norm of the current normalised design parameter $\boldsymbol{x}^{(i)}$ subtracted from the new position x^{i+1}. This norm is then compared a tolerance (default 10^{-2}).

3.6.4 Do Design Space Optimisation

All the building blocks are in place to run the optimiser. This MATLAB optimiser is set up in the same way as the in the alignment phase with a general *problem*, see Section 3.5.5.2. The design variables are directly on hand and so no equality nor inequality constraints are required. The lower bound **lb** and upper bound **ub** are directly the normalised bounds of the design variables scaled by the trust-region radius Δ. This translates to

$$\mathbf{lb} = \left[\max(\boldsymbol{x}^{(i)} - \Delta^{(i)}, \boldsymbol{x}_{n\,min})\right] \,, \tag{3.80}$$

and

$$\mathbf{ub} = \left[\min(\boldsymbol{x}^{(i)} - \Delta^{(i)}, \boldsymbol{x}_{n\,max})\right] \,. \tag{3.81}$$

Here the minimum and maximum are taken between the trust-region adjusted values and the absolute normalised bounds to ensure that the values do not exceed their bounds even if the trust-region were to allow it. Each iteration the trust-region radius is updated and set to corresponding iteration count i.

The starting point is simply the previous normalised design parameter

$$\mathbf{x}_0 = \boldsymbol{x}_n^{(0)} \,. \tag{3.82}$$

Normalisation of design parameters and bounds is discussed in Section 3.2.

The *objective function* is set to the cost function described in Section 3.6.1.

There are a couple of different options the user can choose between for running the optimisation.

- If the `globOpt` flag is set to two then the global optimiser routine will be run on each and every simulation. Recall that if the flag is set to one then only the initial iteration runs through a global optimiser, see Section 3.3.1.

- If the global optimiser has been run, then its optimal point is assigned to $\mathbf{x}_0$. The local optimiser then runs through and select the next trial point.

- If the `useScAsOpt` flag is set then no optimisation is carried out, see Section 3.3.3. Instead, the next trial point is taken from the surrogate specified by the user.

Once a trial point has been found, the trust-region evaluation takes place.

3.6.5 Trust-Region Evaluation

A basic trust-region (B.T.R.) algorithm is detailed in Section 2.3.1. The same approach is used here with a couple of minor changes. In the previous section an optimisation routine is used to find the trial point $\boldsymbol{x}^{(i+1)}$. It is made up of the current design space position $\boldsymbol{x}^{(i)}$ and a trial step away from it $\boldsymbol{s}^{(i)}$. In the B.T.R. described before the k iterator is used but here $i+1$ is used. In practice, it is not necessary to keep track of unsuccessful trial steps. Instead, the $i+1$ term is overwritten and only successful trial points are kept and used in the following main iteration.

At the start of each main loop iteration the k iterator is set to one. This allows the trust region loop to keep track of how far along it is and when it should terminate. To evaluate how successful the trial point is, the change in fine model response against change in surrogate is compared. This marker of success is defined as $\rho^{(k)}$ described in (2.19) from Section 2.3.1.4. Within the context of the main optimisation loop this equation can be rewritten as

$$\rho^{(k)} = \frac{U(\boldsymbol{R}_f(\boldsymbol{x}^{(i)})) - U(\boldsymbol{R}_f(\boldsymbol{x}^{(i)} + \boldsymbol{s}^{(i)}))}{U(\boldsymbol{R}_s(\boldsymbol{x}^{(i)})) - U(\boldsymbol{R}_s(\boldsymbol{x}^{(i)} + \boldsymbol{s}^{(i)}))} \,. \tag{3.83}$$

If $\rho^{(k)}$ is great that or equal to η_1 or η_2, see (2.20) and (2.27) respectively, the trial step is accepted. The η_1 criteria is the minimum requirement and is used in the flow diagram shown in Figure 2.9. Including η_2 allows for three possible outcomes from the trial point.

- For $\rho^{(k)} < \eta_1$ then the trial step is considered a failure and a new trial point needs to be calculated. The trust-region is shrunk so that the surrogate is only evaluated in a region where it is a good approximation to the fine model. The existing trust-region is overwritten to be reused when setting up the next optimisation problem. Here i is not incremented because the trial point failed. The trust-region becomes

$$\Delta^{(i)} = \alpha_2 ||\boldsymbol{s}^{(i)}|| \,, \tag{3.84}$$

  where $||\circ||$ is the Euclidean norm (or 2-norm) and α_2 is defined in (3.7).

- For $\eta_1 \leq \rho^{(k)} < \eta_2$ the trial point is considered a success. The same trust-region is carried through to the next iteration

$$\Delta^{(i+1)} = \Delta^{(i)} \,. \tag{3.85}$$

- If $\rho^{(k)} \geq \eta_2$, the trial point is considered *very* successful. The trust-region radius is expanded as there is a strong agreement between the changes in surrogate and fine model. The new trust-region becomes

$$\Delta^{(i+1)} = \max(\alpha_1 ||\boldsymbol{s}^{(i)}||, \Delta^{(i)}) \,, \tag{3.86}$$

  where once again $||\circ||$ is the Euclidean norm and α_1 is defined in (3.6).

For a successful run, the iteration count i is incremented and the once the new surrogate is built the algorithm will move through a save point before starting the next main loop iteration, as seen in Figure 3.2. With an unsuccessful trial-step the trust-region radius is reduced and i is not incremented, i.e. the main loop will not continue unless an improvement is seen. Even if the trial-step is not successful, the fine model is used to achieve a better surrogate model. The only difference is that the target count t is different. If the trial point is successful, $t = i + 1$. However, if the step is unsuccessful then the target count becomes $t = i$. This is also reflected in the main loop flow diagram seen in Figure 3.2. Building of the surrogate within a design space optimisation loop context is discussed next.

3.6.6 Build Surrogate and Alignment in Main Optimisation Loop

Once the trust-region evaluation has concluded, the surrogate model can be updated using the extra fine model evaluation that is available. As mentioned before, if the trial point is successful then the surrogate model is set to be used in the next main iteration i.e. $t = i+1$. If however the trial point is unsuccessful, then the target count becomes $t = i$ and overwrites the current surrogate model and is used in the next trust-region cycle. Any extra information about the fine model space can be useful when building up the surrogate model, this is discussed further in Section 3.5.5.5. If the `useAllFine` flag is true, then the new fine model response $\boldsymbol{R}_f^{(i+1)}$ (corresponding to $\boldsymbol{x}^{(i+1)}$) is appended to a vector of all the other fine model evaluations and passed through to the surrogate building and alignment phase, see Section 3.5. If `useAllFine` is false, then only the last latest fine model response is passed to the surrogate building and alignment phase.

3.6.7 Output and Overview

With a new surrogate model available the next iteration can commence. If the trial point is unsuccessful the trust-region criteria then the trust region iteration count increments $k++$ and the optimisation operates with a reduced trust-region radius. If the trial point is successful then the MATLAB space is written out as a save point that can be used by a different system run if desired, see Section 3.4.2. The main iteration loop count is incremented $i++$

and the process starts again. The termination conditions are checked and if any one of the conditions are met, the output is prepared and the loops are exited. Besides the save/log file that is written out there are a number of variables that are returned from the main loop `SMmain`.

All the fine model responses are returned along with the corresponding design parameters. The last design parameter is the optimal point, given the users goals. The space-mapping parameters that make up the surrogate model are also returned, but only if they resulted in a successful trial step. Debug output is also returned. This included the cost at each iteration, the optimiser output, the limits at each iteration and the trust-region information from each iteration (successful or not), radii, step sizes, and each ρ. Plots that are generated from the output are shown in Chapter 5.

Chapter 4

Framework Interface

To run an optimisation the user must specify at least three items:

1. An input configuration file that calls the SM framework.
2. A high-fidelity model.
3. A low-fidelity model.

In this chapter these three inputs to the framework are defined and discussed.

4.1 User to Framework Interface

There are a number of configuration option available to the user. These are explained as needed throughout Chapter 2 and 3. There are some input that are required, these are outlined below.

4.1.1 Specify Fine Model

The high-fidelity model name and path must be specified. The solvers available and how they operate is discussed in Section 4.2.2. A `mf` variable is used to contain all the fine model options. The type of solver is specified as text, for example

```
Mf.solver = 'FEKO';
```

The path to the fine model is required. This can be determined using the following commands:

```
filename = mfilename('.m');
fullpath = mfilename('fullpath');
currentPath = replace(fullpath, filename, '')
```

Once the `currentPath` is known, the relative path of the model can be specified. Typically, each solver file is placed in its own folder so that all the files generated by that program are self-contained. The path and the name of the file is specified, for example

```
Mf.path = [currentPath,'FEKO\']
Mf.name = 'DoubleFoldedStub_base';
```

With the model specified the design space parameters can be specified along with their upper and lower bounds. An extract from the double folded stub filter example, shown in Section 5.2, input file is used.

```
Mf.name = 'DoubleFoldedStub_base';
Mf.solver = 'FEKO';
Mf.params = {'l1', 'l2', 's'};
Mf.ximin = [ 35.0, 35.0, 1.0 ]';
Mf.ximax = [ 90.0, 90.0, 15.0 ]';
```

The parameters are listed in cells with corresponding minimum and maximum values.

The last item that must be specified is the frequency to be used.

```
fmin = 5e9;
fmax = 20e9;
Nm = 151; % Number of frequencies
Mf.freq = reshape(linspace(fmin,fmax,Nm), Nm, 1);
```

4.1.2 Specify Coarse Model

The coarse model is set up in a similar way:

```
Mc.path = [currentPath,'AWR\'];
Mc.name = 'DoubleFoldedStub_base';
Mc.solver = 'AWR';
Mc.params = {'l1', 'l2', 's'};
Mc.ximin = Mf.ximin;
Mc.ximax = Mf.ximax;
Mc.freq = reshape(linspace(fmin,fmax,Nm), Nm, 1);
```

The upper and lower bounds are reused from the fine model bounds.

Extra information that can be specified in the coarse model are the implicit parameters. This is once again taken from the example in Section 5.2.

```
Mc.Iparams = {'cm',    'l3'    'eps_r' };
Mc.xpmin =   [15.0,    20.0    08.0    ]';
Mc.xpmax =   [90.0,    40.0    14.0    ]';
```

4.1.3 Specify Optimiser options

The maximum number of main loop optimisations is set using

```
OPTopts.Ni = 5;
```

The maximum number of trust-region iterations

```
OPTopts.TRNi = OPTopts.Ni*2;
```

The type of response and the associated goals

```
OPTopts.Rtype = {'S2,1'};

OPTopts.goalType =      {'gt',       'lt',       'gt'};
OPTopts.goalResType =   {'S2,1_dB', 'S2,1_dB', 'S2,1_dB'};
OPTopts.goalVal =       {-3,         -30,        -3};
OPTopts.goalWeight =    {1,          1,          0.1};
OPTopts.goalStart =     {5.0e9,      12.0e9,     16.5e9};
OPTopts.goalStop =      {9.5e9,      14.0e9,     20.0e9};
OPTopts.errNorm =       {1,          1,          1};
```

Although this is not used in the example mentioned this is how the global

optimiser is set

```
OPTopts.globOpt = 1;
OPTopts.globalSolver = 'ga';
```

The global optimisers that the space-mapping system handles are

- `ga`
- `patternsearch`

Similarly for the local optimiser

```
OPTopts.localSolver = 'fminsearchcon';
```

There are defaults for these options, therefore the user does not need to set them.

The supported local optimisers are

- `fmincon`
- `fminsearch`
- `fminsearchcon`

4.1.4 Specify Space-Mapping options

Space-mapping options are selected as follows

```
SMopts.getA = 0;
SMopts.getB = 0;
SMopts.getc = 1;
SMopts.getG = 0;
SMopts.getxp = 1;
SMopts.getF = 0;
SMopts.getE = 0;
SMopts.getd = 0;
```

This automated approach allows the user to easily switch between the different options with the implementation details abstracted away in the framework.

When implicit parameters are used, initial values must be specified

```
xpinit = [  44.5, 30, 9.9]';
```

The space-mapping framework optimiser is set as follows

```
SMopts.globalSolver = 'ga';
SMopts.optsGlobalOptim = optimoptions('ga');

SMopts.localSolver = 'fminsearchcon';
```

Once again this is not required as it is set-up with default.

4.1.5 Run Main Space-Mapping routine

Finally, the command to actually run the space-mapping framework optimiser is:

```
SMmain(xinit, Sinit, SMopts, Mf, Mc, OPTopts, plotOpts);
```

Here the initial design values are given as `xinit`, for example

```
xinit = [   66.727, 60.228, 9.592]';
```

4.2 External Model/Solver Interface

Although some solvers can be used as both fine or coarse model solvers, they are broken up based on their typical use-case. Details about the implementation of each solver is given.

Validation and hard Limit must be placed on the inputs to ensure that they are always obeyed. The optimisers can evaluate outside their given bounds to try to work out gradient information. They will not suggest an optimal solution outside of the range but the evaluation may still happen and this could cause the solver to report an error. The output from the different solves is caught and redirected to log files.

Each solver has to implement the specific repose types and give an error if an unsupported type is encountered.

4.2.1 Coarse Model Evaluation - Low Fidelity Modelling

4.2.1.1 AWR-MWS

To interface with AWR-MWS a built in MATLAB system `actxserver` is used [32]. This is set up as follows

```
awr = actxserver('AWR.MWOffice');

awr.invoke('Open', [M.path,M.name,'.emp']);
```

Once there is a handle on the solver a subset of the standard API is available [34]. For example accessing the global equations is done in the following way

```
eqns = proj.GlobalDefinitionDocuments.Item(1).Equations;
```

Some notes about using AWR-MWS in this way:

- Writing frequencies is very slow and can cause delays when using frequency space-mapping.
- Not all items are available in the API using this method for example accessing `Item` on some collections is not possible.

4.2.1.2 MATLAB

Custom MATLAB solvers can be build up. They need to handle a vector of input and implicit parameters. The frequencies requested are is passed in and the response is output.

4.2.2 Fine Model Evaluation - High Fidelity Modelling

4.2.2.1 FEKO

FEKO operated using MATLAB`system` commands. The parameters are adjusted and a new mesh is built using

```
% Build parameter string
parStr = [];
for nn = 1:Nn
parStr = [parStr,' -#',M.params{nn},'=',num2str(xi(nn))];
end
% Remesh the structure with the new parameters
FEKOmesh = ['cadfeko_batch ',[M.path,M.name,'.cfx'],parStr];
[statusMesh, cmdoutMesh] = system(FEKOmesh);
```

The solver is then run

```
FEKOrun = ['runfeko ', [M.name,'.cfx']];
[statusRun, cmdoutRun] = system(FEKOrun);
```

S-parameter are the only results supported at this stage. The touchstone file export must be set when setting up the requesting in CADFEKO itself.

4.2.2.2 CST

CST also uses the built in MATLAB system `actxserver` command.

```
cst = actxserver('CSTSTUDIO.Application');
persistent mws
mws = invoke(cst,'OpenFile',[M.path,M.name,'.cst']);
```

A `persistent` variable is used to ensure that instances of the application are managed correctly.

Parameters are adjusted in the following way

```
% Update parameters
for nn = 1:Nn
invoke(mws,'StoreParameter',M.params{nn},xi(nn));
end
% invoke(mws,'Rebuild');
invoke(mws,'RebuildOnParametricChange',true,true);
```

Currently only S-parameters is implemented for CST.

Chapter 5

Analysis

In this chapter examples of how the space-mapping framework is applied to different example is examined. In each case an introduction to the example is given and the methods used to construct and initialise the models, both fine and coarse. Graphs are generated from the final save points and output from the space-mapping framework, see Section 3.6.7. Goals, coarse and fine model response as well as surrogate responses are shown on the graphs. Finally, the results and the space-mapping technique chosen for the particular example are evaluated.

To examine the use-cases for different space-mapping techniques, a basic stub example is presented, see Section 5.1. A greater than and a less than goal are specified on an $S_{1,1}$ request type. FEKO is used as the high fidelity model and AWR-MWS is used as the coarse model. Different coarse models are built to highlight the usage of the space-mapping techniques. Input, implicit and frequency space-mapping are all described in detail.

A microstrip double folded stub filter example is presented in Section 5.2. This example is taken from literature and requires three design parameters to be optimised. Three goals are specified for the $S_{2,1}$ request of this bandstop filter. FEKO is used as the high-fidelity solver. A single-layered infinite substrate is used with P.E.C. lines. A simple mesh convergence study is done to ensure that the complexities of the model are sufficiently handled. An AWR-MWS coarse model is used. Microstrip line, tee and bend components are used with interlinked capacitor.

5.1 Electromagnetic Stub Examples

The same fine model is used for the next three example to demonstrate the different ways that a problem can be solved using space-mapping. A microstrip stub example is set up in FEKO as the high-fidelity model, see Figure 5.1. The lines are modelled as P.E.C. (perfect electric conductor) on an infinite, single layered substrate [36]. Microstrip ports are used for both the input and

output ports with an impedance of 50 Ω. The design variable ls is the length of the stub.

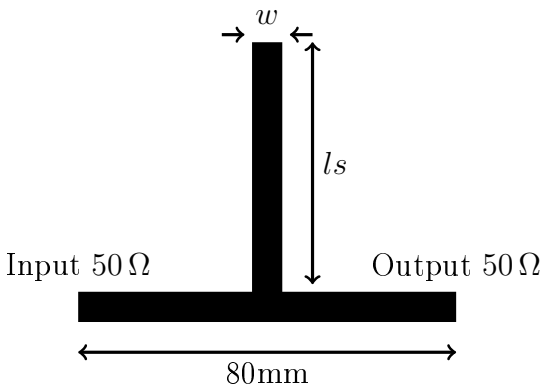


Figure 5.1: Stub microstrip fine model example. The length of the stub ls is the design parameter. The width of lines w are 5 mm, and the length of the feed line is 80 mm. A height h of 1.5 mm is used for the substrate with a permittivity ϵ_r of 2.1.

A single $S_{1,1}$ response request is used for model evaluation, see Section 3.5.5.6 for details on the different response types. The response itself is treated as a complex quantity and both the real and imaginary components come into play with building an aligned surrogate, see Section 3.5.5.5 for details.

Goal criteria for an S-parameter request is typically expressed in decibels. Two goals are specified for this example:

- $|S_{1,1\,dB}| <$ -20 dB over the frequency range 1.30 GHz to 1.45 GHz weighting 1.0.
- $|S_{1,1\,dB}| >$ -10 dB over the frequency range 1.60 GHz to 2.00 GHz weighting 0.1.

The less-than goal is the more important of the two and given a weighting of one while the greater-than weighting is reduced to 0.1. The goals are plotted in Figure 5.2 where the cyan horizontal line represents the less-than criteria and the magenta greater-than. Black represents the fine model response and the coarse model is shown in red. Section 3.6.1 details the different goal types and their combinations.

The coarse model is set up using an AWR-MWSmodel. The diagram of this coarse is shown in Figure 5.3. Feed lines and the stub are modelled using transmission lines with the same effective permittivity as the substrate in the fine model ($\epsilon_r = 2.1$). The is not the most accurate approach both i.t.o the transmission line and the permittivity choice, but they are still used for this example for illustrative purposes. The coarse model response is offset from that of the fine model and the different alignment effects are employed

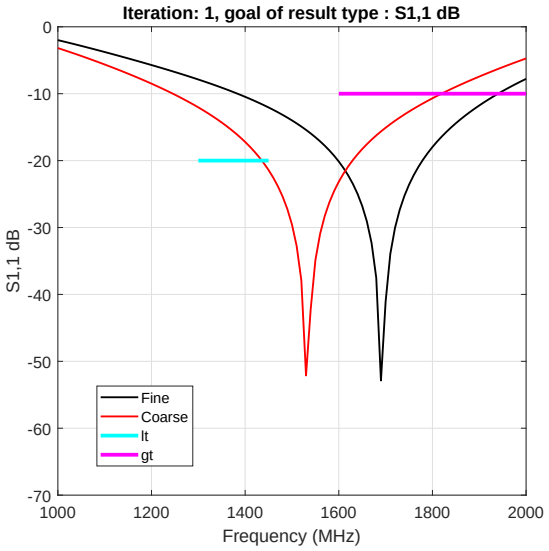


Figure 5.2: A base plot showing the difference between the fine FEKO and coarse AWR-MWS model responses. A less- and greater-than goal are shown but no alignment takes place.

to demonstrated how they are used. `fminsearchcon` is used as the optimiser type for both the alignment phases and the main design space optimisation.

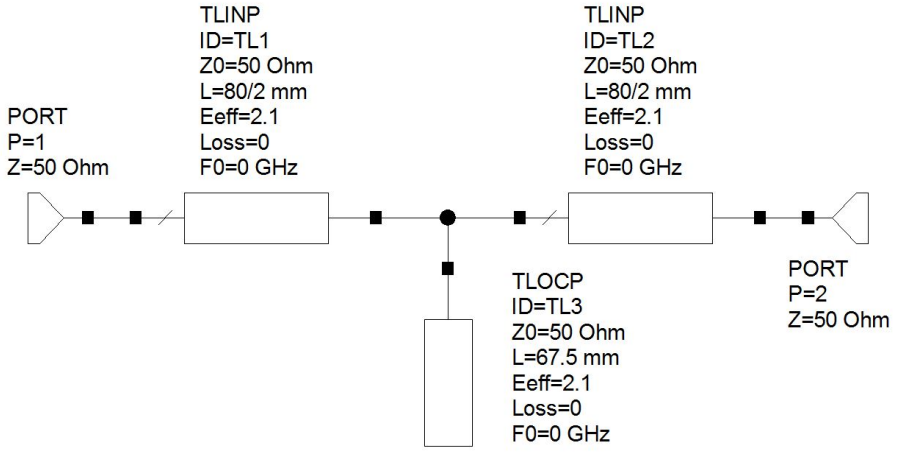


Figure 5.3: Base AWR-MWS coarse model equivalent to fine model.

First off an input space-mapping example is shown. Only the additive `getc` term is used. The following example shows implicit space-mapping where a capacitor is introduced between the end of the transmission line stub and ground. Only an additive implicit space-mapping configuration is used, `getxp`. The final illustrative example used frequency space-mapping, `getF`. A copy of

the coarse AWR-MWS model used for the input space-mapping example is used here too.

5.1.1 Input Space-Mapping for Stub

The FEKO fine model stub shown in Figure 5.1 is used for this example. An AWR-MWS model, shown in Figure 5.4, is used at the low-fidelity model. 50 Ω input and output ports are use for both the fine and coarse models. The length of the feed is specified as $lf = 80\,\text{mm}$ for the coarse model. This is the same as in the fine model. A single design variable ls is used. This is the length of the stub in both models. The same permittivity is used for both the fine model substrate and in the transmission line. The coarse model transmission line impedance is set to 50 Ω.

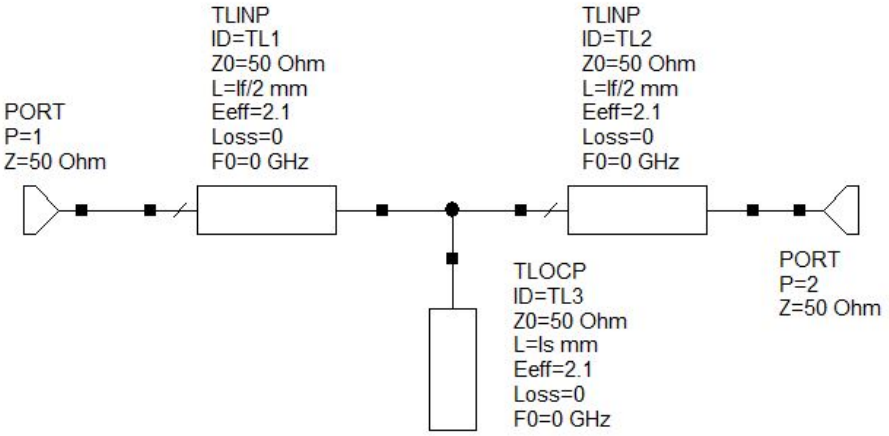
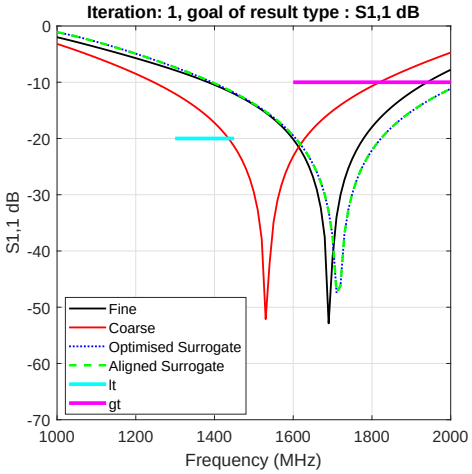


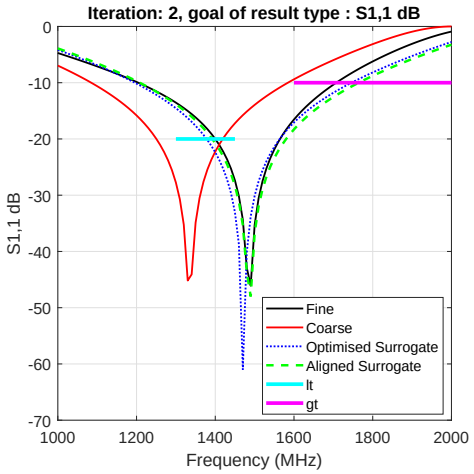
Figure 5.4: AWR-MWS coarse model for input space-mapping stub example.

If the less-than condition is met, the rest of the response less than 1.3 GHz is not important given the goal requirements. In the same way, if the first part of the greater-than goal is met, then the rest is not important. The most useful part of the graph for alignment is therefore between the start of the first goal and the start of the second goal, i.e. from 1.3 GHz to 1.6 GHz. The `errW` vector is therefore set to this range, see Section 3.5.5.5 for details of how this option is used. Alignment within this specified region is prioritised.

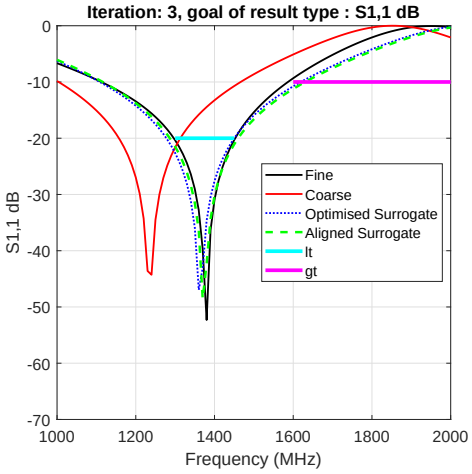
The first iteration is shown in Figure 5.5a. The coarse model response $\boldsymbol{R}_c^{(1)}$ is about 200 MHz left of the fine model response $\boldsymbol{R}_f^{(1)}$. $\boldsymbol{R}_f^{(1)}$ starts off about 10 dB above the cyan less-than goal and only meets the greater-than goal at about 1.9 GHz. The aligned surrogate, shown in dashed green, is in good agreement with the fine model, within the specified range. The dotted blue line represents the optimised surrogate which is the previous surrogate model evaluated at the new point. For the first iteration there is not a previous surrogate and therefore this lies on top of the aligned surrogate response.



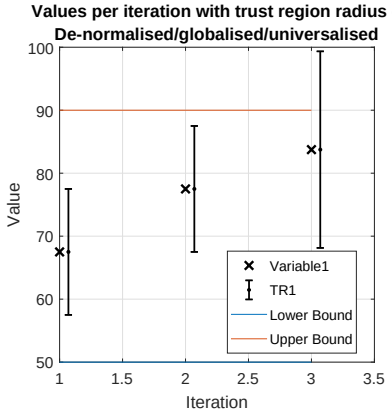
(a) First iteration results of the input space-mapping stub example.



(b) Second iteration results of the input space-mapping stub example.



(c) Third and final iteration for the input space-mapping stub example.



(d) Plot showing the design parameter iterations for an input space-mapping stub example. The values are the actual values and not normalised.

Figure 5.5: Plots showing responses, per iteration, for the stub example using input space-mapping.

Figure 5.5d shows the iteration steps and the trust-region at each point. These are the fine model evaluation points. The initial input parameter is set to $ls = 67.5\,\text{mm}$ and remains at that point for the first iteration because the `startWithIterationZero` flag is set. This means that no optimisation is carried out before the first iteration starts, see Section 3.3.4. The second iteration lines up with the upper edge of the trust-region.

The response at the second iteration is shown in Figure 5.5b. An improvement has been made to achieve the goals but from Figure 5.5d it can be seen the trust-region limited the change. The aligned surrogate response $\boldsymbol{R}_s^{(2)}$ once again has good agreement with the fine model $\boldsymbol{R}_f^{(2)}$. The optimised surrogate

is not exactly the same as the aligned surrogate suggesting that the first surrogate model is accurate within this range. No trust-region radius change in seen for iteration two (Figure 5.5d) which means that the η_1 condition was met but not η_2, refer to Section 3.6.5.

Figure 5.5c shows the final iteration. The surrogate models still have good agreement with the fine model over the frequencies of interest. Both the less-than and greater-than goals are satisfied. In Figure 5.5d the final iteration is well within the trust-region radius and has not reached a boundary. It can also be seen that the trust-region radius has grown in the last iteration. Here the η_2 condition has been met and the radius is expanded. An optimal solution has already been found and no further iterations are seen.

5.1.2 Implicit Space-Mapping for Stub

Once again, the FEKO fine model in Figure 5.1 is used. The AWR-MWS model, shown in Figure 5.6. A capacitor is place between the end of the stub and ground to model edge effects. The capacitor is specified in fF.

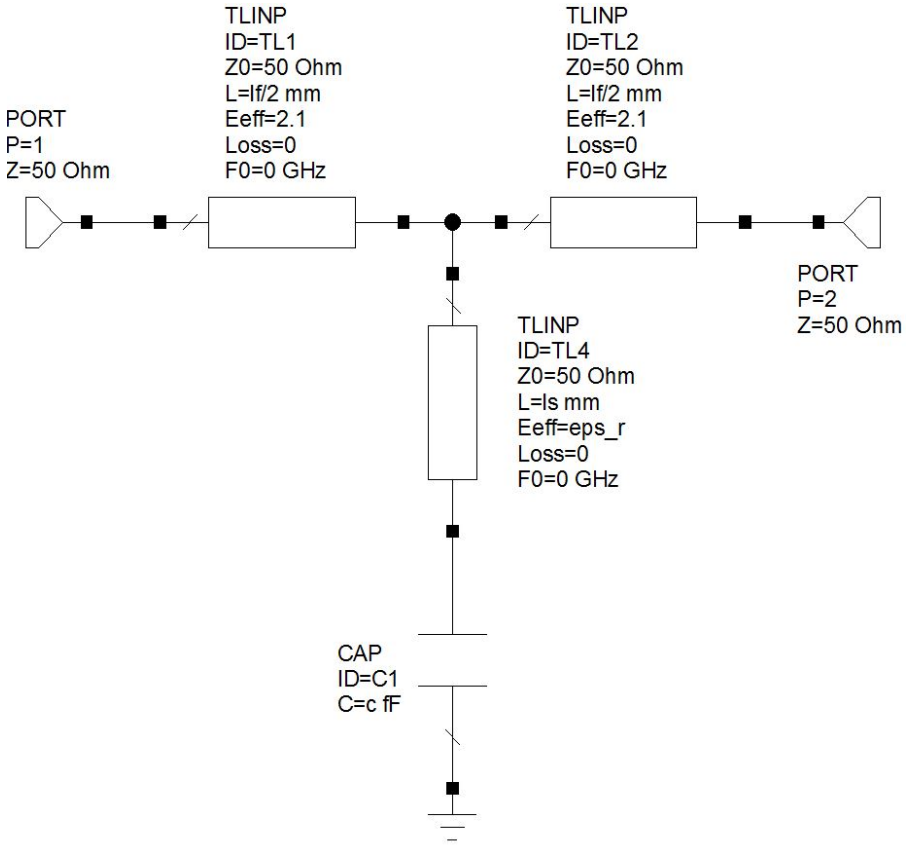
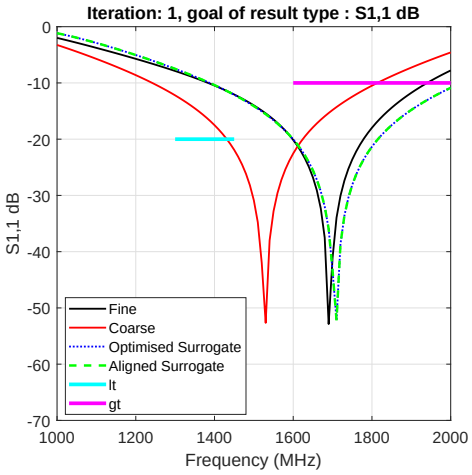
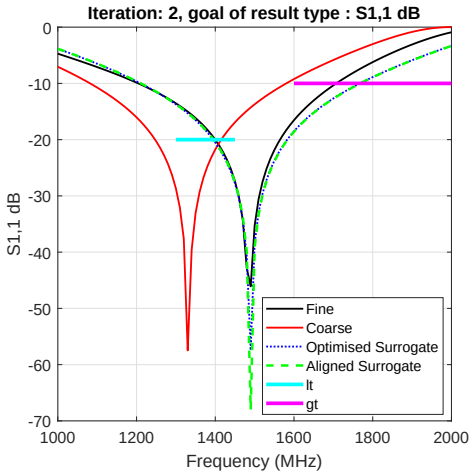


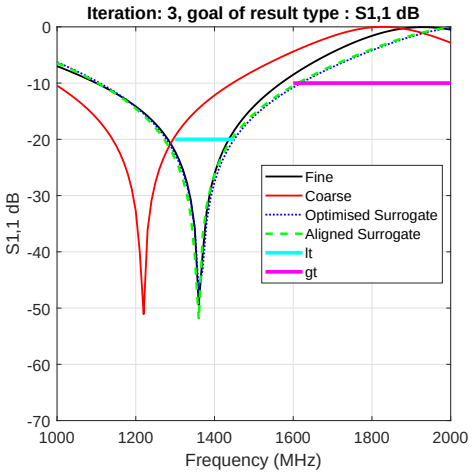
Figure 5.6: AWR-MWS coarse model for implicit space-mapping stub example. Only the capacitor is used for alignment.



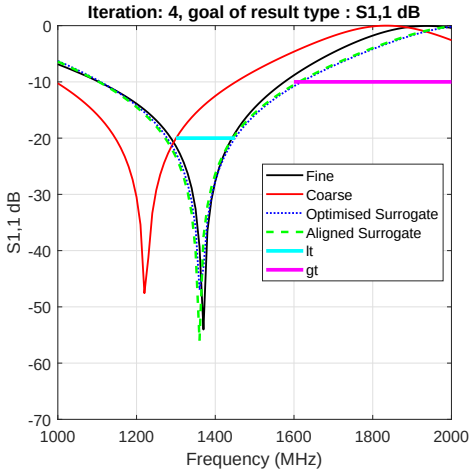
(a) First iteration results of the implicit space-mapping stub example.



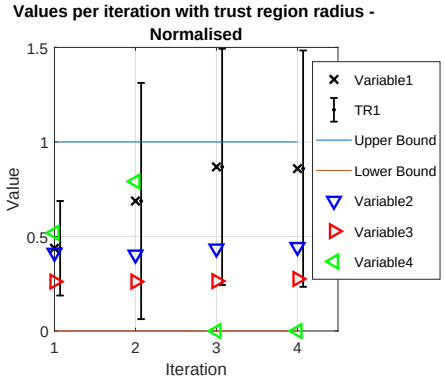
(b) Second iteration results of the implicit space-mapping stub example.



(c) Third iteration results of the implicit space-mapping stub example.



(d) Fourth and final iteration results of the implicit space-mapping stub example.



(e) Normalised plot showing the design parameter iterations for the implicit space-mapping stub example.

Figure 5.7: Plots showing responses, per iteration, for the stub example using implicit space-mapping.

The same goal are used as specified before. Only implicit space-mapping `getxp` is used for this example. The permittivity of the substrate `eps_r` and the length of the input/output coaxial line `lf` are also added as implicit parameters to aid in the alignment process. Their limits are shown below

```
Mc.Iparams = {'eps_r', 'lf', 'c'};
Mc.xpmin =   [1.0      , 70,     0.1]';
Mc.xpmax =   [2.6      , 90,     100]';
```

The results of the first iteration is shown in Figure 5.7a. Good agreement between the aligned surrogate and the fine model responses is seen. Figure 5.7e show the normalised parameter values for each iteration. The normalised values are shown this time because the implicit values are an order of magnitude different. The trust-region around the design parameters is shown and once again it is seen within the middle of the design space. In iteration two the trust-region expands which means there is a very good agreement between the change in responses of the fine and surrogate models in the first iteration. The iteration is however limited by the trust-region.

Figure 5.7b and Figure 5.7c show the results of the second and third iteration respectively. In both cases good progress is made reducing the design space goal errors. The aligned and optimised surrogate models line up very well with the fine models within the specified region 1.30 GHz to 1.60 GHz. Figure 5.7c still shows an overlap on the less-than goal.

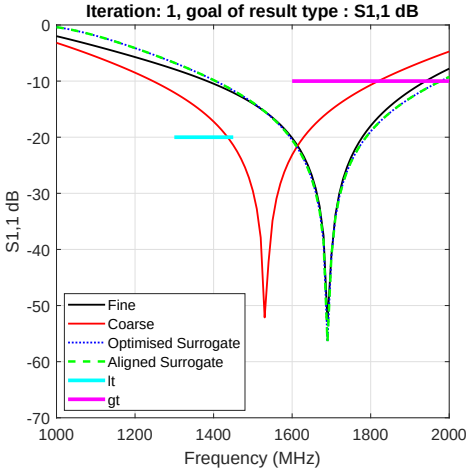
The overlap is reduced in the final iteration, see Figure 5.7d. Figure 5.7e shows that the capacitor implicit variable goes down to the bottom it its lower bound. This could show that the bounds are too small but in this case they are sufficient.

5.1.3 Frequency Space-Mapping for Stub

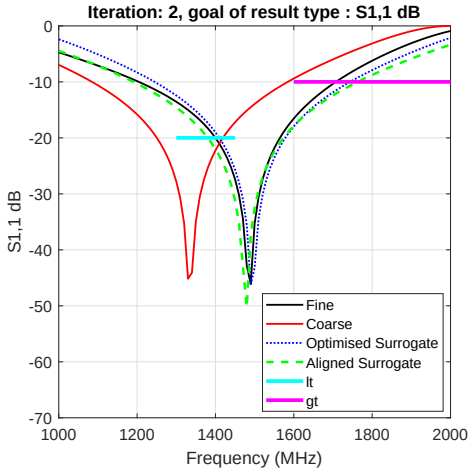
A detailed, isolated example of frequency space-mapping is shown in Section 2.2. Here this technique is applied to the microstrip stub filter example. Once again, the same fine model is used, see Figure 5.1. The AWR-MWS model from Section 5.1.1 is also reused here, see Figure 5.4.

Figure 5.8a shows the first iteration. The coarse model is successfully aligned to the fine model within the specified 1.30 GHz to 1.60 GHz range. The frequency space-mapping values are not shown on Figure 5.8d because they are not directly linked to the design space. In the second iteration it can be seen that once again the model is evaluated right at the edge of the trust-region.

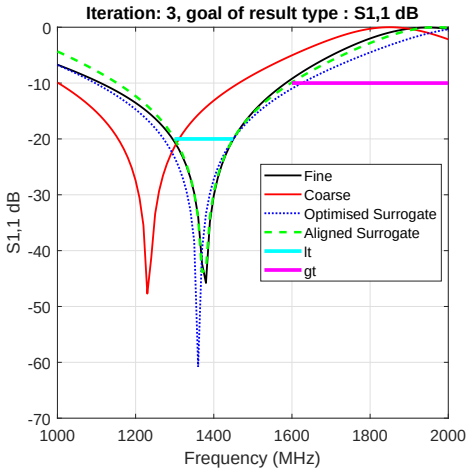
Figure 5.8b shows the response at iteration two. There is a very good agreement between the fine and surrogate models. The improvement matched so well that the trust-region expands to almost fill the design space, see Fig-



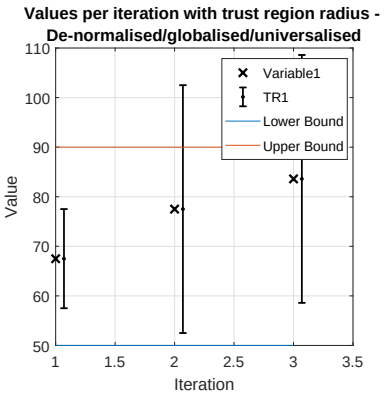
(a) Response to frequency space-mapping stub example, iteration one.



(b) Response to frequency space-mapping stub example, iteration two.



(c) Third and final response for the frequency space-mapping stub example.



(d) Plot showing the design parameter iterations for the frequency space-mapping stub example. The values are the actual values and not normalised.

Figure 5.8: Plots showing responses, per iteration, for the stub example using frequency space-mapping.

ure 5.8d. A step is then taken in the third and final iteration right to an optimal solution. Figure 5.8c shows this result.

5.2 Double Folded Stub

The double folded stub (DFS) filter in as an example case used in a number of space-mapping papers [3, 13, 18]. Figure 5.9 show a fine model representation of the DFS bandstop filter. FEKO is used as the high fidelity solver. A 5 mil substrate with relative permittivity 9.9. Microstrip lines for the fine model are modelled as P.E.C. on an infinite, single layered substrate [36].

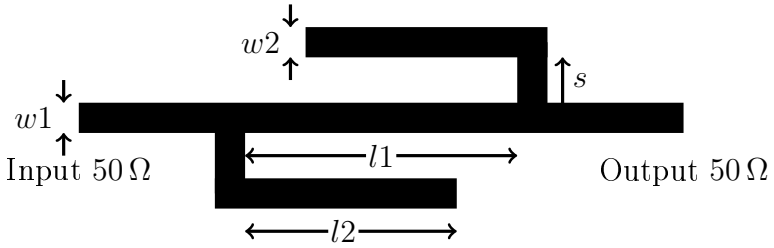


Figure 5.9: Double folded stub microstrip fine model example.

Three design variables are used. $l1$ is the length between the two stubs, $l2$ is the width of the horizontal part of the arms and s is the height of the vertical part of the arms. The width of the microstrip lines $w1$ and $w2$ are fixed 4.8 mil. The input and output ports are used at 50 Ω. Model evaluation takes place from 5 GHz to 20 GHz.

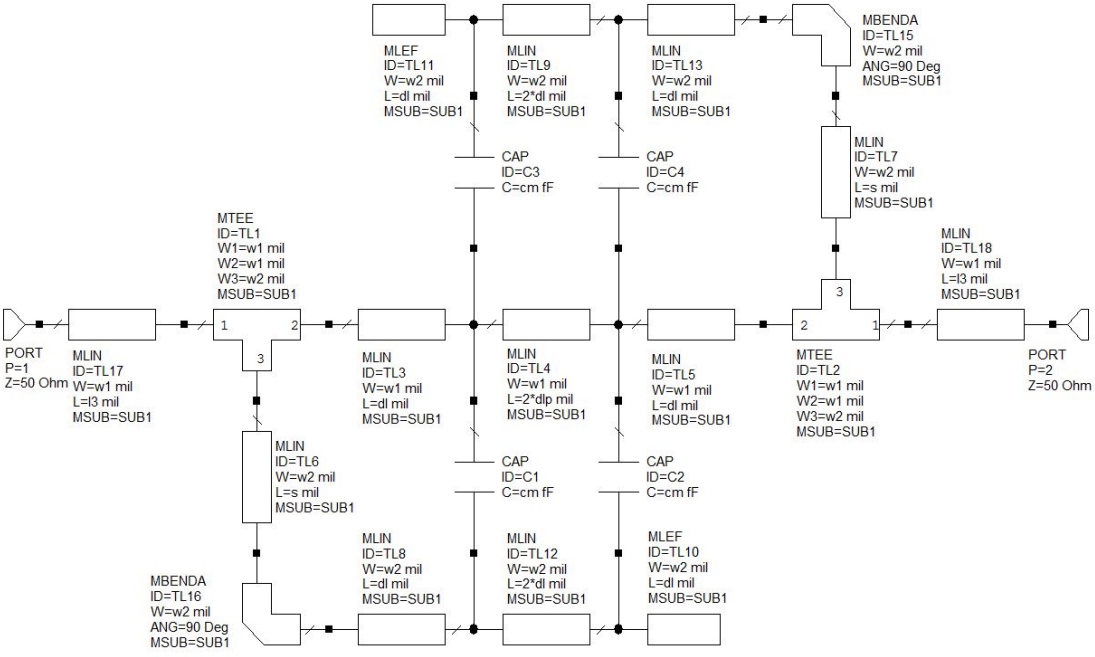


Figure 5.10: AWR-MWS coarse double folded stub model.

AWR-MWS is used for the low fidelity model. It is shown in Figure 5.10. Microstrip line, bend and tee components are used. The same line width used in the fine model are used here. The strip lengths are broken up into deltas and four capacitors are distributed between the main line and the arms [3]. Capacitors are used to approximate coupling between the arms and each is represented by an implicit variable *cm*. Arm lengths are broken up into

$$l2 = 4\Delta_l\,, \tag{5.1}$$

and the centre length

$$l1 = 2\Delta_l + 2\Delta_l'\,. \tag{5.2}$$

$l1$ and $l2$ are still used as the design variables. s represents the height of the microstrip between the centre line and the horizontal arm. An extra length is $l3$ is introduced to the coarse model between the microstrip tees and the input and output ports. This is introduced as an implicit variable to allow the phase of the surrogate model response to change. The third final implicit variable that is introduces is the relative permittivity of the substrate ϵ_r.

An extract of the input file for the DFS is shown below to summarise the initial values and the bounds used:

```
%            l1       l2       s
xinit = [   66.727, 60.228, 9.592]';
%           cm     l3     eps_r
xpinit = [  44.5, 30.0, 9.9]';

Mf.solver = 'FEKO';
Mf.params = {'l1',  'l2',   's' };
Mf.ximin = [ 35.0,   35.0,   1.0]';
Mf.ximax = [ 90.0,   90.0,  15.0]';

Mc.solver = 'AWR';
Mc.params = {'l1',  'l2',   's'};
Mc.ximin = Mf.ximin;
Mc.ximax = Mf.ximax;
Mc.Iparams = {'cm',      'l3'     'eps_r' };
Mc.xpmin =   [15.0,      20.0     08.0    ]';
Mc.xpmax =   [90.0,      40.0     14.0    ]';
```

All lengths are given in mil and the capacitance is in fF.

This is a bandstop filter therefore, the request type `Rtype` is set to `S2,1`. There are three goals specified this request and each are given dB. Therefore, the `goalResType` is set to `S2,1_dB`. The goals are

- $|S_{2,1}| \geq$ -3 dB for 5.0 GHz $\leq f \leq$ 9.5 GHz,
- $|S_{2,1}| \leq$ -30 dB for 12.0 GHz $\leq f \leq$ 14.0 GHz, and
- $|S_{2,1}| \geq$ -3 dB for 16.5 GHz $\leq f \leq$ 20 GHz.

The goal weighting of the last goal is dropped to 0.1 to give the other two goals preference. This choice is discussed later in Section 5.2.2.

Before the results are presented, a meshing analysis/refinement is conducted to check that the FEKO fine model is set up accurately without introducing excessive solver runtime. After the meshing analysis, the results of this approach is shown.

5.2.1 Mesh Analysis/Refinement

Four sets of meshing options are tested using FEKO on the DFS fine model seen in Figure 5.9. A continuous interpolation frequency range is set and the design parameters are set to the optimal model $\boldsymbol{x}_f^*$ found in [3]. That is $l1 = 78.964\,\text{mil}$, $l2 = 81.210\,\text{mil}$ and $s = 7.901\,\text{mil}$.

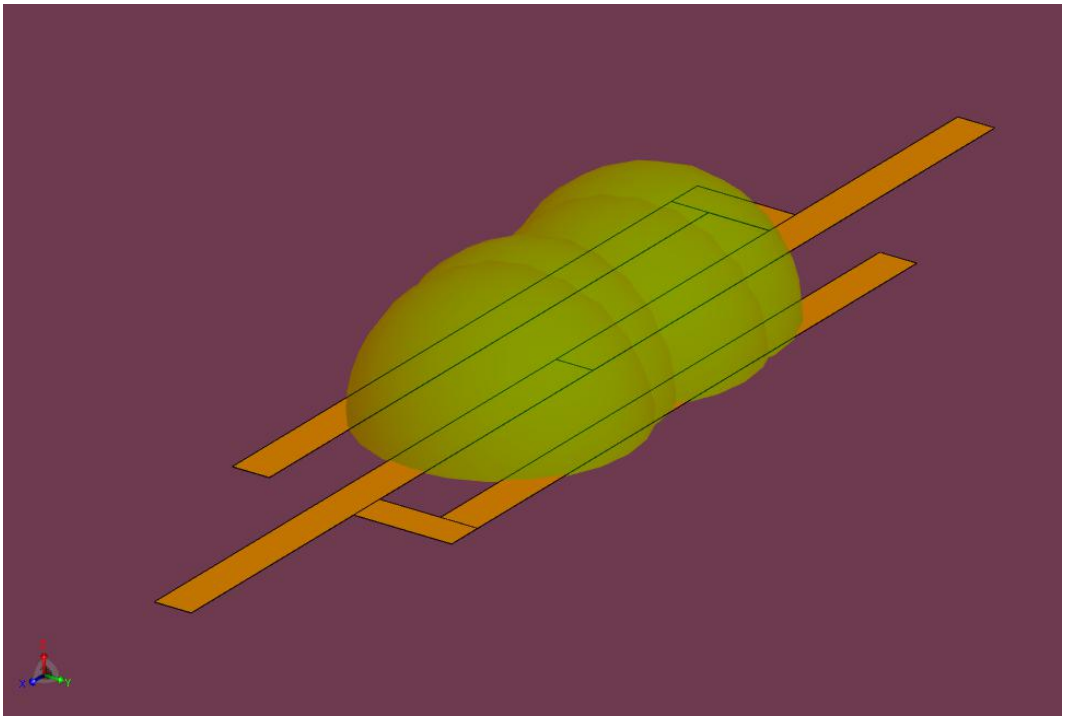
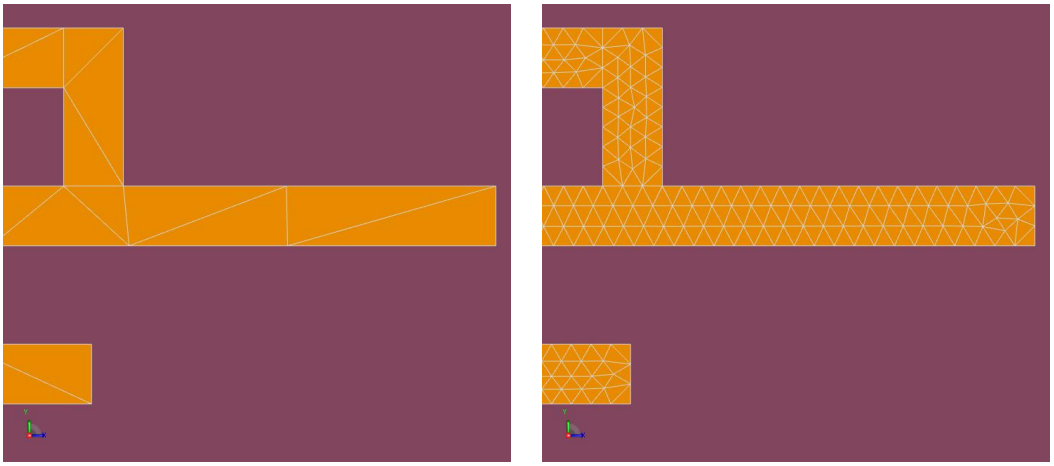


Figure 5.11: Meshing refinements resulting from a FEKO mesh refinement and error estimation routine.

Four different meshing configurations are used to determine mesh convergence. The configurations are:

1. Standard.
2. Standard meshing with meshing refinement.
3. Custom $w1/3$.



(a) Double folded stub standard meshing.

(b) Double folded stub custom $w1/3$ meshing.

Figure 5.12: Visual comparison between standard and custom meshes.

4. Custom $w1/3$ with refinement.

The meshing refinement options are available when an error estimation request is specified and an initial run has taken place. The meshing zones are automatically set up in the regions where errors may be present [36]. Figure 5.11 shows the meshing refinement zones over the model. The custom mesh size of $w1/3$ ensures that there at least three triangles along the width of the microstrip, see Figure 5.12b. Using a standard mesh a single triangle is seen along the width of the microstrip line, see Figure 5.12a.

The S-parameter results for these four runs are shown in Figure 5.13. The dotted lines represent the response without meshing refinement and solid lines with the refinement. Blue and green lines are the standard mesh while the orange and red are custom mesh. There is very little difference between the respective base meshes and their corresponding refined meshes. There is however a difference between the standard and the custom mesh.

A frequency shift from 13.976 GHz to 14.160 GHz is seen at the minimum point, i.e. a shift of 0.184 GHz or 1.2 % of the frequency space. The actual value at this null is not relevant. The value at the local maximum is however important.

The standard mesh has a value of -30.728 dB at the local maximum. -29.108 dB is seen on the custom mesh with refinement. These two responses are compared because it is expected that the finest mesh would give the best results and the coarsest mesh would give the poorest results. This is only a 1.620 dB difference. The difference in results are not significant enough to justify the extra runtime and the standard mesh is used for the base approach seen next in Section 5.2.2.

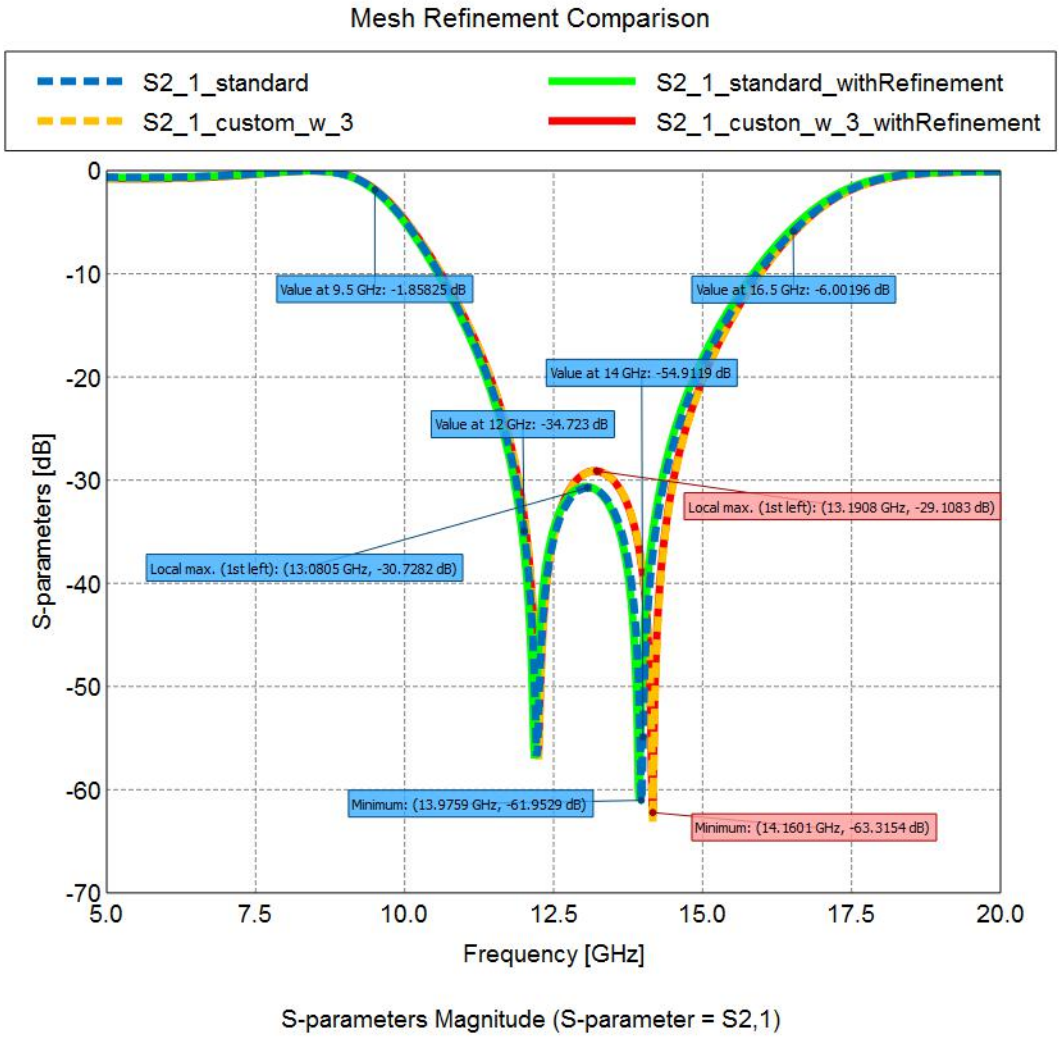


Figure 5.13: A FEKO fine model mesh convergence comparison.

Other points to note from the exercise is to inspect the values at the goal points.

- At 9.5 GHz the response is meant to be greater-than -3 dB. In this case a value of -1.858 dB is seen. The responses successfully meet the criteria.
- Within the range from 12.0 GHz to 14.0 GHz $|S_{2,1}|$ must be less-than -30 dB. The end points are clear for all the responses. However, for the custom mesh simulations the local maximum breaks the criteria by 0.728 dB.
- Finally, at 16.5 GHz the response must be greater-than -3 dB. Here none of the responses make the specification. At -6 dB an extra 3 dB is required.

With these criteria in mind it is expected that the optimisation routine may struggle to meet the specification because using the same results published do not meet specifications.

5.2.2 Base Approach

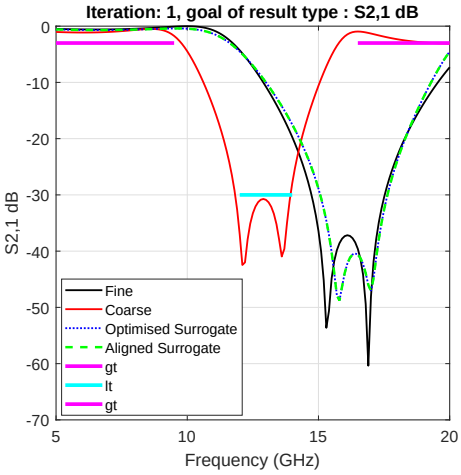
To solve this problem a combination of additive input space-mapping `getc` and additive implicit `getxp` is used. The `wk` option is set to zero meaning that the number of fine models used for alignment is restricted, see Section 3.5.5.5. `fminsearchcon` is used as the local solver and no global optimisers are used. Unlike the illustrative example, `startWithIterationZero` is not set. This means that a local optimisation will be used before the first fine model is evaluated. No special alignment evaluation zones are set.

The first iteration is shown in Figure 5.14a. The coarse model is offset from the fine model by about -3 GHz and does not meet the second nor the third goal specification.

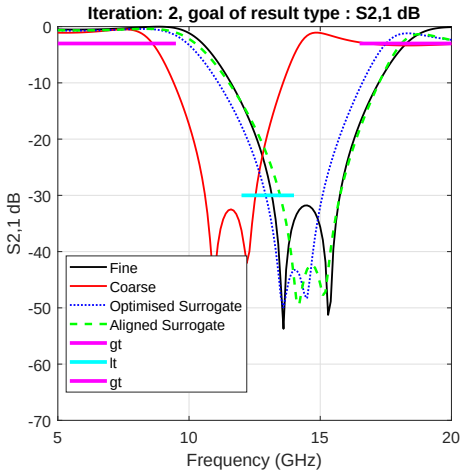
As mentioned before, the last goal weighting is dropped to 0.1. In this first iteration of optimisation, the reduction in weight gives preference to the other two goals. The second and last goals start with no successful fine model points, while the first one has only successful points. If all the goals are equally weighted, the shift of the main nulls could either be to higher or lower in frequency without a significant difference in cost. The reduction on weight of the last goal helps the correct step to be made.

An aligned surrogate model bridges the frequency difference and improves on the form matching. It is not a perfect match but sufficient for optimisation. In Figure 5.14e the normalised design variable analysis is given for each iteration. The normalised version is shown due to the differences between the actual values of the design and implicit parameters. The design variables are depicted by an x, square and circle. Implicit parameters are shown by triangles in different orientations, given in the order of definition. The design parameters are roughly in the centre of the design space and their trust-regions are shown using error-bars. After the initial optimisation, the implicit variables drift to the top and bottom of the available space to achieve alignment. This could signify that the bounds are too strict, but in this case the bounds are valid and ensure that the does not degrade to evaluations that no longer make sense.

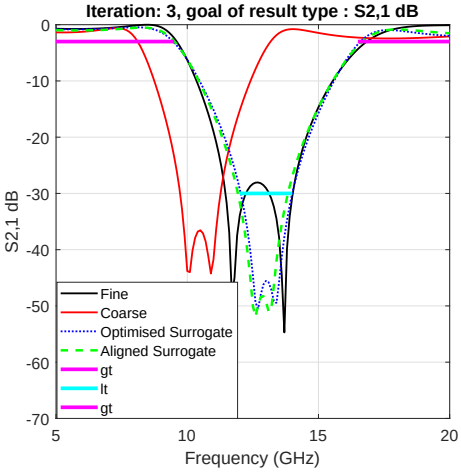
In the second iteration the design variables move to the edge of the trust-regions. Results of this iteration can be seen in Figure 5.14b. The fine model shifts to the left reducing the error for the less-than goal and the second greater-than goal. The optimised surrogate is offset down from the fine model by about 1 GHz. This is the surrogate from the first iteration evaluated at the new position. This offset reduced significantly by the aligned surrogate generated using the new fine model data. There is good agreement between the aligned surrogate and the fine model along the sides of the graph but there is a noticeable



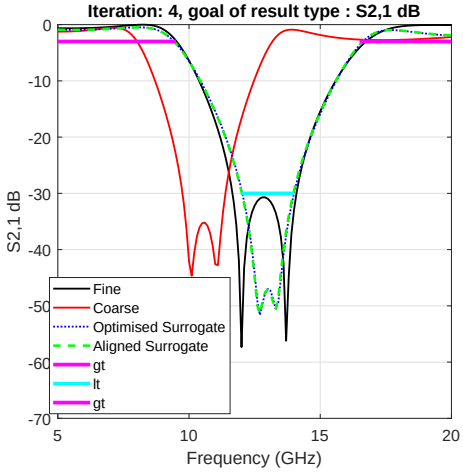
(a) Double folded stub, first iteration.



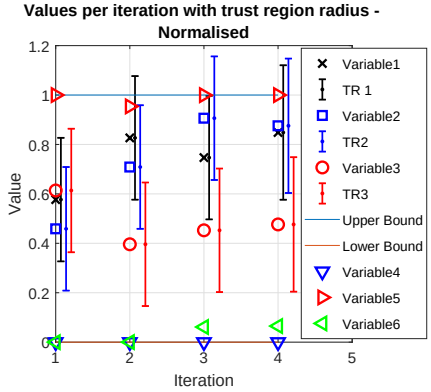
(b) Double folded stub, iteration two.



(c) Double folded stub, iteration three.



(d) Double folded stub, final iteration.



(e) A normalised representation of the design parameter at each iteration for double folded stub example.

Figure 5.14: Double folded stub example using input and implicit space-mapping.

difference at the central local maximum. A drop of about 10 dB is seen. The values at -30 dB are small but this discrepancy could lead to problems meeting the less than specification. A final observation to note for this step is that the trust-region has remained the same.

Figure 5.14c shows the results of the third iteration. Once again, an improvement is seen in the fine model results. The second greater-than goal is almost met but the local maximum in the centre pushes above the -30 dB level. Figure 5.14e shows that the values are no longer at the edge of the previous trust-region and are moving around within the radius limits. The optimised and aligned surrogates have a small deviation but it appears that they are converging. With three design parameters and three implicit parameters it is expected that another three fine models are required to have enough fine model data cater for all the degrees of freedom.

The forth and final iteration is seen in Figure 5.14d. Here the optimised and aligned surrogate meet the first greater-than and the less-than goals. There is not much room along the edges of the goals but there is a gap of about 16 dB between local maximum (between the nulls) and the less than goal. This extra room in not however seen in the fine model. It only just makes the -30 dB specification. The second greater-than goals sees a small overlap from both the surrogate and fine models.

Table 5.1: Optimal solution to double folded stuff filter

Design Variable	$\boldsymbol{x}_f^*$ [mil]
$l1$	81.662
$l2$	83.156
ls	7.6694

Although there is a marginal overlap for the third goal, this is considered a successful solution to double folded stub problem. The combination of implicit and input space-mapping allowed an accurate surrogate model to be built and the problem two be solved with only four fine model evaluations. The optimal design parameters are listed in Table 5.1.

Chapter 6

Conclusion

6.1 Summary and Conclusions

An automated space-mapping framework is presented that can be configured in various ways. Additive and multiplicative options are available for input, output and implicit space-mapping techniques, as well as scaling and shifting for the frequency space-mapping option. The user can specify multiple goals across various solution request types. High fidelity solvers such as FEKO and CST are available, as well as low fidelity solves such as AWR-MWS and MATLAB. The framework allows introducing new solver and supports various levels of error validation. Multiple local and global optimisers are available and are configurable by the user. The framework also allows new solvers to be easily added through the use of MATLAB `problem` definition structure. Results are saved at each successful iteration and the system is able to pick up from one of the previous save points with new options.

Implementation details and explanations of the framework are presented. Relevant defaults and configuration options are discussed for each stage of the algorithm. Two separate and configurable optimisation stages are used. The space-mapping optimiser is used for extracting the space-mapping parameters that achieve the best alignment between the surrogate and fine model results. Constraints are required for this optimiser because the bounds on the design variables must carry through to the surrogate model being built. An option to normalise the space-mapping parameters is provided and allows the optimiser to make fine adjustments with any degree of freedom. The error function used for the parameter extraction phase compares the difference in results of the surrogate and fine model responses. Complex-valued responses give the best alignment results because phase is considered. Options for different norms are discussed for the parameter extraction error function (the L1 norm is the default option). Multiple fine model evaluations can be used to achieve an aligned surrogate that is valid over a large portion of the parameter space.

The second optimiser uses the surrogate model to find an optimal solution

to the given device specification. Available goals to be used for device specification include less-than, greater-than, and bandwidth operations. These are used to set an error/objective function for the design space optimiser. The design space available to the optimiser is limited by a trust-region safeguard. This ensures that the surrogate model is not evaluated in a region that does not accurately represent the fine model. When an optimal design space position is proposed by the optimiser (a trial point), that point is also evaluated in fine model space. A trust-region evaluation phase then takes place. The change in surrogate model response, between this trial point and the point before, is compared against the fine model response at the same two points. If the changes do not match up, then the trial point is rejected and the trust-region shrinks. This means that the bounds given to the optimiser are closer to the previous point, where the fine and surrogate models were aligned. If, however, the change in responses match, then the evaluation concludes that a successful trial step was made.

With the extra fine model information, a new surrogate model is built. This happens regardless of whether the trial point is accepted or not. In both instances, more information is available for the parameter extraction process to use. The framework allows for various different configurations for the use of the fine models. One method is to use the same number of fine model evaluations as there are space-mapping parameters. If there are more fine models than this, the earlier ones are discarded. The user is provided with options for changing this behaviour.

An illustrative microstrip stub example is presented for input, implicit and frequency space-mapping. This example shows how to apply the different techniques to EM devices. The objective given for these examples is to optimise the $S_{1,1}$ response to conform to a less-than and greater-than goal. FEKO is used as the high-fidelity solver and AWR-MWS as the low-fidelity one. The fine model uses an infinite single-layered substrate with P.E.C. lines and Transmission line are used in the coarse model. Each step is analysed and details are given about how the trust-region operates.

Finally, a microstrip double folded stub filter is demonstrated. The example is taken from literature and pushed through the system. FEKO and AWR-MWS are used once again. Three design variables are used in the example: the horizontal spacing between the two arms, the horizontal lengths of the arms themselves, as well as the height of the arms. The coarse model uses microstrip lines, tees and bends. Capacitors are included as implicit variables to model coupling. The permittivity of the substrate and the length of the input/output feed are also included as implicit variables. Additive input and implicit space-mapping techniques are used to solve this bandstop optimisation problem. Three goals are specified for the $S_{2,1}$ response. The system allows a custom weighting of the goals to be specified which ensures that the first optimisation step is in the correct direction. This shows that there is still room for further improvements in robustness. With this adjustment a successful optimisation

is observed with four fine model evaluations.

6.2 Future Work

It is always useful to have additional solvers, optimisers and more response types available to the user, but it is more important to ensure that when new items are added correctly and fulfil API contracts that are exposed. Using an object-oriented architecture using polymorphism can improve the extendibility and maintainability of the framework. Interfaces for the solvers and the optimisers would force a clear and repeatable solution. Pairing this an API level test suite or even unit-testing would improve the maintainability and ability for developers to confidently change internal workings.

Any type of testing would require example models that are very quick to evaluate and can handle various different configurations. A mathematical, series resonant circuit incorporated into a Rosenbrock function would satisfy such criteria. The mathematical mature allows for quick evaluations and the Rosenbrock function testing as the dimensions of the system grow. Implicit variables need to also be incorporated to ensure the entire system is evaluated.

A polymorphic, object-oriented approach also lends itself re-usability and expandability of the space-mapping options. Currently, the framework does not use any derivative information for alignment improvements. Although derivative information is not available from fine model evaluations, a Broyden update approximation could be used.

In terms of validating the current work presented, the trust-region should be compared against optimisation runs that do not have the option enabled to see where improvements are made. Improvements could be made to how the initial trust-region radius is calculated and allow automatic retries if the system converges without meeting specifications.

An aspect that is not automated is that of which space-mapping options to use. A wizard to guide a new user to which options to use for a particular problem would help.

A further advancement would be to extend this system to handle interconnected, multi-disciplinary systems. For example, linking thermal or mechanical systems to the existing EM solution. The space-mapping paradigm is already used through various engineering disciplines and the framework could support this extension.

List of References

[1] Bandler, J.W.: Optimization methods for computer-aided design. *IEEE Transactions on Microwave Theory and Techniques*, vol. 17, no. 8, pp. 533–552, August 1969. ISSN 0018-9480.

[2] Steer, M.B., Bandler, J.W. and Snowden, C.M.: Computer-aided design of rf and microwave circuits and systems. *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 3, pp. 996–1005, March 2002. ISSN 0018-9480.

[3] Bandler, J.W., Cheng, Q.S., Dakroury, S.A., Mohamed, A.S., Bakr, M.H., Madsen, K. and Sondergaard, J.: Space mapping: the state of the art. *Microwave Theory and Techniques, IEEE Transactions on*, vol. 52, no. 1, pp. 337–361, 2004. ISSN 0018-9480.

[4] Koziel, S. and Yang, X.-S.: *Computational Optimization, Methods and Algorithms*. 1st edn. Springer-Verlag Berlin Heidelberg, 2011. ISBN 978-3-642-20858-4.

[5] Koziel, S., Cheng, Q. and Bandler, J.: Space mapping. *IEEE Microwave Magazine*, vol. 9, no. 6, pp. 105–122, 2008. ISSN 1527-3342.

[6] Temes, G.C. and Calahan, D.A.: Computer-aided network optimization the state-of-the-art. *Proceedings of the IEEE*, vol. 55, no. 11, pp. 1832–1863, Nov 1967. ISSN 0018-9219.

[7] Bandler, J.W. and Chen, S.H.: Circuit optimization: the state of the art. *IEEE Transactions on Microwave Theory and Techniques*, vol. 36, no. 2, pp. 424–443, Feb 1988. ISSN 0018-9480.

[8] Bandler, J.W., Kellermann, W. and Madsen, K.: A superlinearly convergent minimax algorithm for microwave circuit design. *IEEE Transactions on Microwave Theory and Techniques*, vol. 33, no. 12, pp. 1519–1530, Dec 1985. ISSN 0018-9480.

[9] Meijer, P.B.L.: Fast and smooth highly nonlinear multidimensional table models for device modeling. *IEEE Transactions on Circuits and Systems*, vol. 37, no. 3, pp. 335–346, March 1990. ISSN 0098-4094.

[10] Bandler, J.W., Biernacki, R.M., Chen, S.H., Grobelny, P.A. and Ye, S.: Yield-driven electromagnetic optimization via multilevel multidimensional models.

IEEE Transactions on Microwave Theory and Techniques, vol. 41, no. 12, pp. 2269–2278, Dec 1993. ISSN 0018-9480.

[11] Dounavis, A., Gad, E., Achar, R. and Nakhla, M.S.: Passive model reduction of multiport distributed interconnects. *IEEE Transactions on Microwave Theory and Techniques*, vol. 48, no. 12, pp. 2325–2334, Dec 2000. ISSN 0018-9480.

[12] Zaabab, A.H., Zhang, Q.-J. and Nakhla, M.: A neural network modeling approach to circuit optimization and statistical design. *IEEE Transactions on Microwave Theory and Techniques*, vol. 43, no. 6, pp. 1349–1358, June 1995. ISSN 0018-9480.

[13] Bandler, J.W., Biernacki, R.M., Chen, S.H., Grobelny, P.A. and Hemmers, R.H.: Space Mapping Technique for Electromagnetic Optimization. *IEEE Transactions on Microwave Theory and Techniques*, vol. 42, no. 12, pp. 2536–2544, 1994. ISSN 15579670.

[14] Bandler, J.W., Biernacki, R.M., Chen, S.H., Hemmers, R.H. and Madsen, K.: Electromagnetic Optimization Exploiting Aggressive Space Mapping. *IEEE Transactions on Microwave Theory and Techniques*, vol. 43, no. 12, pp. 2874–2882, 1995. ISSN 15579670.

[15] Bandler, J.W., Biernacki, R.M. and Chen, S.H.: Fully automated space mapping optimization of 3d structures. In: *1996 IEEE MTT-S International Microwave Symposium Digest*, vol. 2, pp. 753–756. June 1996. ISSN 0149-645X.

[16] Bandler, J.W., Biernacki, R.M., Chen, S.H. and Omeragic, D.: Space mapping optimization of waveguide filters using finite element and mode-matching electromagnetic simulators. In: *1997 IEEE MTT-S International Microwave Symposium Digest*, vol. 2, pp. 635–638. June 1997. ISSN 0149-645X.

[17] Bandler, J.W., Biernacki, R.M., Chen, S.H. and Huang, Y.F.: Design optimization of interdigital filters using aggressive space mapping and decomposition. *IEEE Transactions on Microwave Theory and Techniques*, vol. 45, no. 5, pp. 761–769, May 1997. ISSN 0018-9480.

[18] Bakr, M.H., Bandler, J.W. and Georgieva, N.: An aggressive approach to parameter extraction. In: *1999 IEEE MTT-S International Microwave Symposium Digest (Cat. No.99CH36282)*, vol. 1, pp. 261–264. June 1999.

[19] Bakr, M.H., Bandler, J.W. and Georgieva, N.: Corrections to an aggressive approach to parameter extraction. *IEEE Transactions on Microwave Theory and Techniques*, vol. 48, no. 9, pp. 1596–1596, Sept 2000. ISSN 0018-9480.

[20] Koziel, S., Bandler, J.W. and Cheng, Q.S.: Robust trust-region space-mapping algorithms for microwave design optimization. *IEEE Transactions on Microwave Theory and Techniques*, vol. 58, no. 8, pp. 2166–2174, Aug 2010. ISSN 0018-9480.

[21] Queipo, N.V., Haftka, R.T., Shyy, W., Goel, T., Vaidyanathan, R. and Tucker, P.K.: Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, vol. 41, no. 1, pp. 1 – 28, 2005. ISSN 0376-0421.
Available at: `https://doi.org/10.1016/j.paerosci.2005.02.001`

[22] Booker, A.J., Dennis, J.E., Frank, P.D., Serafini, D.B., Torczon, V. and Trosset, M.W.: A rigorous framework for optimization of expensive functions by surrogates. *Structural optimization*, vol. 17, no. 1, pp. 1–13, Feb 1999. ISSN 1615-1488.
Available at: `https://doi.org/10.1007/BF01197708`

[23] Alexandrov, N.M. and Lewis, R.M.: An overview of first-order model management for engineering optimization. *Optimisation and Engineering*, vol. 2, no. 4, pp. 413–430, Dec 2001.

[24] Leary, S.J., Bhaskar, A. and Keane, A.J.: A constraint mapping approach to the structural optimization of an expensive model using surrogates. *Optimization and Engineering*, vol. 2, no. 4, pp. 385–398, Dec 2001. ISSN 1573-2924.
Available at: `https://doi.org/10.1023/A:1016038305014`

[25] Choi, H.-S., Kim, D.-H., Park, I.-H. and Hahn, S.-Y.: A new design technique of magnetic systems using space mapping algorithm. *IEEE Transactions on Magnetics*, vol. 37, no. 5, pp. 3627–3630, Sept 2001. ISSN 0018-9464.

[26] Koziel, S., Bandler, J.W. and Madsen, K.: Quality assessment of coarse models and surrogates for space mapping optimization. *Optimization and Engineering*, vol. 9, no. 4, pp. 375–391, Dec 2008. ISSN 1573-2924.
Available at: `https://doi.org/10.1007/s11081-007-9032-0`

[27] Koziel, S., Bandler, J.W. and Cheng, Q.S.: Trust-region-based convergence safeguards for space mapping design optimization of microwave circuits. In: *2009 IEEE MTT-S International Microwave Symposium Digest*, pp. 1261–1264. June 2009. ISSN 0149-645X.

[28] Koziel, S. and Bandler, J.W.: Space-mapping optimization with adaptive surrogate model. *IEEE Transactions on Microwave Theory and Techniques*, vol. 55, no. 3, pp. 541–547, March 2007. ISSN 0018-9480.

[29] Conn, A., Gould, N. and Toint, P.: *Trust Region Methods*. Society for Industrial and Applied Mathematics, 2000.

[30] Koziel, S., Bandler, J.W. and Madsen, K.: Towards a rigorous formulation of the space mapping technique for engineering design. In: *2005 IEEE International Symposium on Circuits and Systems*, pp. 5605–5608 Vol. 6. May 2005. ISSN 0271-4302.

[31] Bakr, M.H., Bandler, J.W., Biernacki, R.M., Chen, S.H. and Madsen, K.: A trust region aggressive space mapping algorithm for em optimization. In: *1998 IEEE MTT-S International Microwave Symposium Digest (Cat. No.98CH36192)*, vol. 3, pp. 1759–1762. June 1998. ISSN 0149-645X.

[32] MATLAB R2018a. The MathWorks, Inc., Natick, MA, USA, February 2018.

[33] *Microstrip and Stripline Design.* Analog Devices, Norwood, MA, USA. Online `http://www.analog.com/media/en/training-seminars/tutorials/MT-094.pdf`; accessed February 19, 2019.

[34] AWR 12.03r. National Instruments - AWR Group, El Segundo, CA, USA, 2016.

[35] CST STUDIO SUITE 2018. A Dassault Systems company, Velizy-Villacoublay, France, October 2017.

[36] Altair HyperWorks FEKO 2017.2. Altair Engineering Inc., Troy, MI, USA, November 2017.

[37] Sibanda, C.: *Design and Optimisation of Gap Waveguide Components through Space Mapping.* Thesis, Stellenbosch University. Faculty of Engineering. Dept. of Electrical and Electronic Engineering, March 2018.
Available at: `http://hdl.handle.net/10019.1/103495`

[38] MATLAB *"Function Reference R2018a".* The MathWorks, Inc., Natick, MA, USA, February 2018. Online `https://www.mathworks.com/`; accessed February 19, 2019.

[39] Koziel, S., Bandler, J.W. and Madsen, K.: A space-mapping framework for engineering optimization - theory and implementation. *IEEE Transactions on Microwave Theory and Techniques*, vol. 54, no. 10, pp. 3721–3730, Oct 2006. ISSN 0018-9480.

[40] D'Errico, J.: *fminsearchbnd, fminsearchcon.* MATLAB Central File Exchange, September 2006. Online `https://uk.mathworks.com/matlabcentral/fileexchange/8277-fminsearchbnd-fminsearchcon`; Retrieved August 16, 2016.