Delft University Of Technology

Master of Science Thesis

Report №: 2020.045

Adaptive Structural Optimization Methods

Ioannis Pappas

Supervisors: Stijn Koppen, Matthijs Langelaar

October 29, 2020

Department of Structural Optimization and Mechanics (SOM) · Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

TU Delft Delft University of Technology

Abstract

Although the field of structural optimization has undergone substantial growth the last decades, the efficiency of the programs used to generate structurally optimal designs for industrial applications remains questionable. To make matters worse, as structural problems become more complex, e.g. by adding multi-disciplinary optimization targets to the problem formulation, the slow convergence speed associated with the inefficiency of these programs turns into a significant issue. Thus, one can argue that there is a need for improvements in the optimization methods used to address such problems.

One of the most prominent methods to tackle structural optimization problems is Sequential Approximate Optimization (SAO). This method is comprised of 2 steps: approximating locally the non-analytical functions that describe the studied phenomenon and solving the resultant nonlinear optimization problem. This report is a thorough investigation of the function approximation part on a fundamental level, and an attempt to improve the quality of the generated approximation.

We start by conducting a literature survey of the state-of-the-art function approximation methods and evaluating their performance – in terms of convergence speed – under various scenarios. Then, we focus our attention to the most widely applied family of approximations (i.e. the Method of Moving Asymptotes (MMA)) by locating and understanding their inadequacies. These inadequacies are then addressed by proposing a solution that includes the generation of novel mixed approximation schemes that can outperform their constituent members for a set of small-scale, albeit challenging, test problems.

Apart from implementing, validating and evaluating fairly different approximation schemes, we also generate approximation selection criteria, whose output is the 'optimal' approximation method for any given function at an arbitrary point. The outcome of this process is an 'Optimization Laboratory' that offers an in-depth understanding of the behaviour of any function approximation method under examination on a fundamental level. It is effectively an object-oriented software library that can be used to compare different approximation schemes on a set of problems, while offering a detailed insight on the quality of any approximation at any given point. Numerical examples that support our assertions are presented, building a strong case in favor of the possible superiority of a mixed scheme compared to its component members, as far as adaptiveness, robustness and total computational cost of SAO algorithms are concerned.

More specifically, all approximations used herein are analyzed with a benchmark profiling method. Different sets of initial points, multiple convergence criteria and several tolerances are used in order to obtain an objective comparison between the considered methods. The results of this thorough comparison are summarized in performance profile plots and discussed thereafter.

The results of the present research suggest that the generated mixed schemes are superior to their constituent members for the considered problem set. A significant reduction in the number of iterations required to converge to a local optimum was achieved and overall convergence was improved. Even in the most conservative scenario, the best mixed scheme outperformed all others for 46% of the implemented problems (compared to MMA that did so for just above 15% of them). What is more, improved robustness was also achieved by successfully solving 95% of the considered problems (compared to 72% for MMA).

However, this improved performance is accompanied by an increase in the computational cost of the approximation process. To this end, an analysis of the relative computational cost of mixed schemes compared to the state-of-the-art approximation methods – in terms of time spent per iteration on the approximation – follows the performance results and a method to estimate the viability of approximation enhancement is proposed. Finally, conclusions on the relative performance of all methods are drawn from the aforementioned comparison and recommendations for further research on improving the performance and lowering the computational cost of mixed schemes are given.

Acknowledgements

After a year of intense work, I can confidently say I am happy with the result and, most of all, grateful for the experience. To me, it was clear from my first meeting with Stijn that we would get along well. I can now reflect on the past year and feel nothing but respect, appreciation and gratitude for the time we spent together. Both within and outside the context of this thesis. Stijn was as good a supervisor any student could ask for, but more importantly a very good person. Of course, part of the credit needs to be given to Matthijs for creating a culture of hard-working researchers and leading them by example. Our meetings were insightful, efficient and fruitful. The feedback I got was always to the point and, in a very subtle and professional way, I was pushed to improve my work to the best of my abilities whenever it was needed. I was very lucky to have worked under these people. Stijn and Matthijs, thank you for this joyful experience.

I would also like to thank Dr. Munro, Dr. Groenwold and Dr. Svanberg for their help. Our correspondence was useful and insightful on several occasions where important decisions were needed to be made.

Furthermore, I would like to express my deepest gratitude to Chatu, without whom the past 2 years would have been much harder. She stood by my side whenever it was necessary and helped me grow in many ways, some of which I was completely unaware of. Chatu, thank you for everything you have done for me.

My sincere appreciation needs to go to my friends as well. A group of people who have been like brothers to me for over 15 years and have provided me with everything a friend could ask for. You are too many to mention, but you have made my life better and I will always be grateful for it.

Last but not least, a great part of the credit needs to be given to my family. My mother, my father and my two sisters, all of whom have a special role in my life. I am lucky to be able to lean on you, and it is my pleasure to support you whenever I can.

Contents

List of Figures

Acronyms

AM	Adaptively Mixed (approximation schemes)
DOF	Degree of Freedom
DSA	Dual Sequential Approximation
FA	Function Adaptive (approximation schemes)
FEA	Finite Element Analysis
FVA	Function Variable Adaptive (approximation schemes)
HY	Hybrid (approximation schemes)
IM	Initially Mixed (approximation schemes)
MMA	Method of Moving Asymptotes
NAND	Nested Analysis and Design
NM	Non Mixed (approximation schemes)
OC	Optimality Criteria
PDSA	Primal-Dual Sequential Approximation
SA	Sensitivity Analysis
SAND	Simultaneous Analysis and Design
SAO	Sequential Approximate Optimization
SIMP	Solid Isotropic Material with Penalization
SO	Structural Optimization
TO	Topology Optimization

Symbols

$L_{ij}^{(k)}$	Lower asymptotes for GMMA-based approximations
$L_i^{(k)}$	Lower asymptotes for MMA-based approximations
UB_j	Upper bound for $g_j(\mathbf{X})$ in the calculation of Φ_j
$U_{ij}^{(k)}$	Upper asymptotes for GMMA-based approximations
$U_i^{(k)}$	Upper asymptotes for MMA-based approximations
Φ_j	Approximation quality index
$\alpha_i^{(k)}$	Exponent of exponential approximations
α	Line-search step-size in the Newton direction
$\boldsymbol{\lambda}$	Lagrange multiplier vector
$\mathbf{D}$	Newton direction vector
$\mathbf{X}^{(k)}$	Current design point
$\mathbf{X}^*$	Optimal design point of $\mathcal{P}_{\text{NLP}}$ (minimizer)
$\mathbf{X}$	Design variable vector
$\boldsymbol{\Delta}$	Move limit vector
$\overline{\mathbf{X}}$	Upper bound vector
$\underline{\mathbf{X}}$	Lower bound vector
$\alpha_i^{(k)}$	Lower bound for MMA-based approximations
$\beta_i^{(k)}$	Upper bound for MMA-based approximations
$\widetilde{g}_j(\mathbf{X})$	Approximate response function
d	Direction index
e_j	Expectation index
$g_j(\mathbf{X})$	Exact response function
k	Number of iterations
m	Number of constraints
n	Number of design variables
p_j	Participation index
$p_{ij}^{(k)}$	Parameter used by MMA-based approximations
$q_{ij}^{(k)}$	Parameter used by MMA-based approximations
$r_j^{(k)}$	Parameter that collects 0-order terms for MMA-based approximations
$\boldsymbol{\lambda}^{*(k)}$	Optimal Lagrange multipliers for $\widetilde{\mathcal{P}}_{\text{NLP,P}-\text{D}}$
$\mathbf{X}^{*(k)}$	Optimal design point of $\widetilde{\mathcal{P}}_{\text{NLP}}$ (minimizer)
$\mathcal{L}(\mathbf{X}, \boldsymbol{\lambda})$	Lagrangian

$\mathcal{P}_{\text{NLP}}$	Initial nonlinear problem
$\widetilde{\mathcal{P}}_{\text{NLP}}$	Local nonlinear sub-problem
$\widetilde{\mathcal{P}}_{\text{NLP,P}-\text{D}}$	Local primal/dual nonlinear sub-problem

1 | Introduction

1.1 Structural optimization

SO is often defined as the subject of making an assemblage of materials sustain certain loads in the 'best possible way' [22]. After years of pursuing optimized structures, a class of computational methods aimed at finding the optimal design – within a design domain – with respect to certain criteria and under given load cases was formed. The definition of the word 'best' determines both the objective and the constraints of the optimization process. To give an example, for some applications the 'best' design might translate to the lightest one, while for others, the stiffest design might be desirable. There are 3 main types of SO:

- Sizing optimization
- Shape optimization
- Topology optimization

The last category – being the most general, versatile and recent – will be the ultimate goal of the present survey, though all methods and findings can be extrapolated to the other two categories as well. TO effectively means distributing material in a specific domain in the 'best possible way', while sustaining the desired loads and obeying given constraints. The interested reader is referred to [22] for some detailed definitions on structural and topology optimization. Although TO was initially developed for load carrying structures, the rapid increase in computational power has enabled scientists to use this method even for complicated multidisciplinary problems, which may involve structural, thermal, fluid mechanics and chemical finite element simulations, e.g. [23], [24]. An example of TO can be seen in Fig. 1.1, where the frame of a motorbike has been optimized with respect to its weight and compliance.

Figure 1.1: Optimized bike frame by [1]

Although computer power is increasing exponentially, so is the complexity of the problems solved by most industries. This means there is a growing need for computational methods that can tackle the aforementioned problems with a minimal use of resources or, from an optimization point of view, minimum computational cost. To put things in perspective, such a structural optimization process might take days before converging to a solution, even in powerful supercomputers. Therefore fast, robust, flexible and overall convergent optimization methods are of increasing interest.

Amongst numerous optimization algorithms that have been developed over the past century, see Fig. 1.2 for the most commonly used ones, some of them appear to be more advantageous for large-scale TO problems.

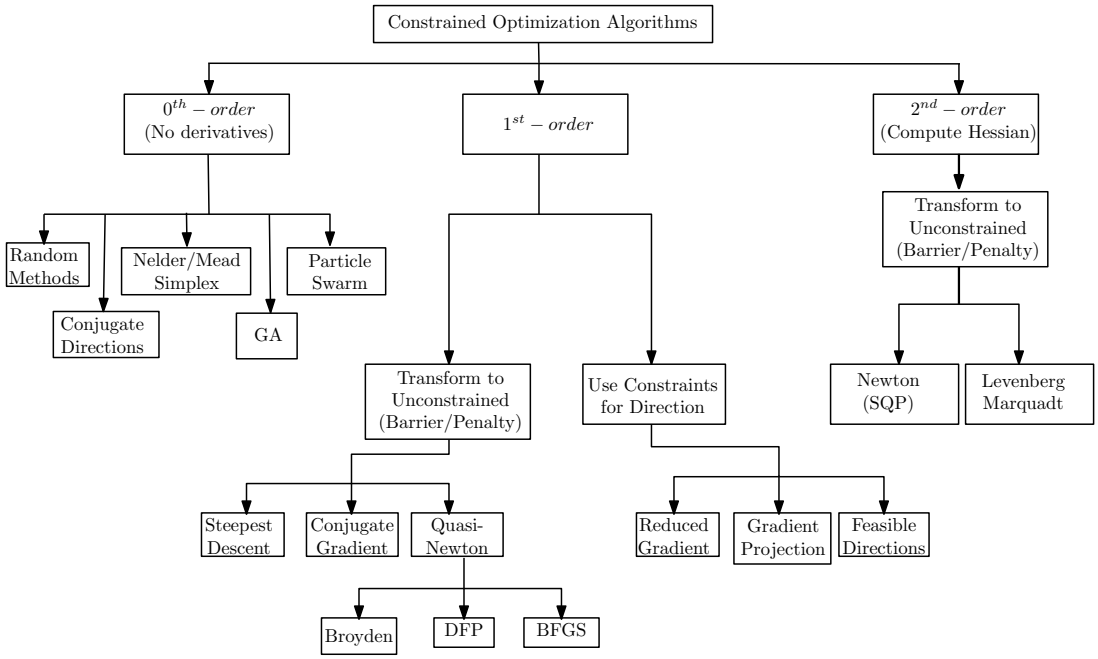


Figure 1.2: Typical constrained optimization algorithms, classified by the order of the gradient information used

Without loss of generality, since we ultimately aim to include all types of SO problems in the present research, we can use TO as an example. These problems exhibit a specific set of attributes that typically involve:

- A very large number of design variables
- Multiple constraints
- Inherently nonlinear responses (i.e. objective function and/or constraints)
- Availability of response sensitivities, stemming from the Sensitivity Analysis (SA)
- Computationally expensive function evaluations (e.g. it has been reported that for large-scale minimum compliance problems function evaluations take up to 97% of the total computational time [17])
- The number of design variables far exceeds the number of constraints for nested formulation settings, i.e. Nested Analysis and Design (NAND). See [25] for more details on the TO formulation settings

Evidently, in order to improve the efficiency of a constrained optimization algorithm used in a TO framework, one must exploit these attributes by tailoring the algorithm to the specific set of problems it addresses. For example, it is self-evident that – if handled appropriately – response sensitivity information can be used to accelerate convergence. The literature on structural optimization algorithms agrees that the 'best-fit' to the large-scale TO problems described above is the SAO algorithm introduced in Section 1.3. Consequently, the present research will revolve around this approach.

1.2 Mathematical definitions

Before discussing the most prominent SO methods found in literature, some mathematical definitions must be introduced. In this section, we will define the primal optimization problem $\mathcal{P}_{\text{NLP}}$, its local approximation $\widetilde{\mathcal{P}}_{\text{NLP}}$ and its augmented primal-dual version $\widetilde{\mathcal{P}}_{\text{NLP,P-D}}$, which is a transformation of $\widetilde{\mathcal{P}}_{\text{NLP}}$ used by several optimization algorithms. The significance of those definitions will become apparent to the reader in the coming sections.

Primal problem

We start by introducing the primal nonlinear optimization problem $\mathcal{P}_{\text{NLP}}$. In the general case, it consists of an n–dimensional design variable vector $\mathbf{X} = [x_1\ x_2 \dots x_n]^T$, a scalar-valued nonlinear objective function $g_0(\mathbf{X})$, m–nonlinear

constraints $g_j(\mathbf{X})$, and $2n$–bound constraints on the design variables x_i, see Eq. (1.1) below:

$$\mathcal{P}_{\text{NLP}} = \begin{cases} \underset{\mathbf{X}}{\text{minimize}} & g_0(\mathbf{X}) & \\ \text{s.t.} & g_j(\mathbf{X}) \leq 0, & j = 1, \ldots, m \\ & \underline{x}_i \leq x_i \leq \overline{x}_i, & i = 1, \ldots, n \end{cases} \tag{1.1}$$

This is the most general mathematical description of the non-linear, continuous and constrained problem that is subject to optimization. All the essential modelling aspects are included herein and the value of the optimization's outcome largely depends on the accuracy of this formulation with respect to the physical phenomenon under investigation. Therefore, one must aim to include the minimum number of parameters (e.g. variables, functions) of the studied phenomenon needed to capture the desired degree of representation.

Primal approximate sub-problem

We are interested in problems for which the evaluation of (some of) the responses in Eq. (1.1) are computationally demanding, due to for example one or multiple FEAs. "For problems of high dimensionality, efficient second order methods such as SQP may become impractical; not only due to the storage requirements of order n^2 associated with the Hessian matrix, but also the computational effort associated with calculating the Hessian matrix in the first place, the more so if the problem of Eq. (1.1) is indeed simulation-based. A viable alternative to the second order methods is SAO, in which the objective and constraints are separately approximated using simple analytical functions of (hopefully) reasonable quality" [26]. Under these conditions, the initial problem $\mathcal{P}_{\text{NLP}}$ is replaced by its local approximation $\widetilde{\mathcal{P}}_{\text{NLP}}$, which is computationally inexpensive to solve under certain assumptions (see Section 1.3). The main reasons for approximating Eq. (1.1) are:

- There is no analytical form for responses $g_j(\mathbf{X})$, as evaluating them at any point (i.e. $g_j(\mathbf{X}^{(k)})$) is typically the outcome of an FEA, rendering $\mathcal{P}_{\text{NLP}}$ too computationally expensive to address directly. Therefore, a minimum number of FEA is desired.
- SA can be included in the FEA and yield inexpensive computation of local response gradients, i.e. $\partial g_j / \partial x_i(\mathbf{X}^{(k)})$, whose use in an approximate function can be beneficial.

$$\widetilde{\mathcal{P}}_{\text{NLP}} = \begin{cases} \underset{\mathbf{X}}{\text{minimize}} & \widetilde{g}_0(\mathbf{X}) & \\ \text{s.t.} & \widetilde{g}_j(\mathbf{X}) \leq 0, & j = 1, \ldots, m \\ & \underline{x}_i \leq x_i \leq \overline{x}_i, & i = 1, \ldots, n \end{cases} \tag{1.2}$$

There are many ways to approximate Eq. (1.1), but the main idea is that the objective and constraint functions are approximated at design – or expansion – points $\mathbf{X}^{(k)}$ using appropriate schemes (e.g. Taylor-like expansions) to construct a local, approximate sub-problem [12], as in Eq. (1.2). This sub-problem is then solved using the methods of Fig. 1.3. The process is then iterated until certain convergence criteria (e.g. Eq. (1.4)) are met.

Primal-dual approximate sub-problem

There are numerous ways to solve $\widetilde{\mathcal{P}}_{\text{NLP}}$. One of the most widely applied approaches in the field of SO is to transform $\widetilde{\mathcal{P}}_{\text{NLP}}$ to $\widetilde{\mathcal{P}}_{\text{NLP,P-D}}$. To do so, the Lagrangian of Eq. (1.3) needs to be introduced. Although this augments the working variable space dimension (i.e. the number of unknowns in the equations to be solved) from $\mathbf{X} \in \mathbb{R}^n$ to $\{\mathbf{X}, \boldsymbol{\lambda}\} \in \mathbb{R}^{n+m}$, under certain assumptions (e.g. convexity and separability of functions $\widetilde{g}_j(\mathbf{X})$, see [12]) it is beneficial to do so in order to solve Eq. (1.2) efficiently. The aforementioned Lagrangian reads as:

$$\mathcal{L}(\mathbf{X}, \boldsymbol{\lambda}) = \widetilde{g}_0(\mathbf{X}) + \sum_{j=1}^{m} \lambda_j \widetilde{g}_j(\mathbf{X}) \tag{1.3}$$

where, $\boldsymbol{\lambda} = [\lambda_1 \ \lambda_2 \ \ldots \lambda_m]^T$ is the Lagrange multiplier vector or, in a Primal-Dual Sequential Approximation (PDSA) framework, a vector containing the dual variables. The efficiency of this method largely depends on the convexity and separability assumptions made when defining $\widetilde{\mathcal{P}}_{\text{NLP}}$, which guarantees that the Lagrangian is also separable and

convex. Consequently, the Lagrangian can be minimized by many algorithms that are driven by the 1^{st}-order KKT optimality conditions below:

$$\left.\frac{\partial \widetilde{g}_0}{\partial x_i}\right|_{\mathbf{X}^{(k)}} + \sum_{j=1}^{m} \lambda_j \left.\frac{\partial \widetilde{g}_j}{\partial x_i}\right|_{\mathbf{X}^{(k)}} = 0 \ , \qquad \forall i \tag{1.4a}$$

$$\widetilde{g}_j(\mathbf{X}) \leq 0 \ , \qquad \forall j \tag{1.4b}$$

$$\lambda_j \geq 0 \ , \qquad \forall j \tag{1.4c}$$

$$\lambda_j \widetilde{g}_j(\mathbf{X}) = 0 \ , \qquad \forall j \tag{1.4d}$$

The quasi-unconstrained (only low-level non-negativity constraints are present, i.e. $\lambda_j \geq 0, \ \forall j$) primal-dual approximate sub-problem can be formulated as:

$$\widetilde{\mathcal{P}}_{\text{NLP,P-D}} = \begin{cases} \min\limits_{\mathbf{X}} \max\limits_{\boldsymbol{\lambda}} & \mathcal{L}(\mathbf{X}, \boldsymbol{\lambda}) \\ \text{s.t.} & \lambda_j \geq 0, \qquad j = 1, \ldots, m \end{cases} \tag{1.5}$$

which can be solved efficiently by any gradient-based optimization method that can handle unilateral constraints, see Fig. 1.2. Then, the optimal set of primal ($\mathbf{X}^{*(k)}$) and dual ($\boldsymbol{\lambda}^{*(k)}$) variables is substituted in Eq. (1.4) to obtain the residuals and assess the procedure's convergence. Finally, move limits ($\boldsymbol{\Delta} = [\delta_1 \ \delta_2 \ldots \delta_n]^T$) and bound constraints ($\underline{\mathbf{X}} \leq \mathbf{X} \leq \overline{\mathbf{X}}$) must be applied to ensure stable convergence and feasibility of the optimal design point:

$$\mathbf{X}^{(k+1)} = \mathbf{X}^{*(k)} \tag{1.6a}$$

$$\mathbf{X}^{(k+1)} \geq \max(\mathbf{X}^{(k)} - \boldsymbol{\Delta}, \ \underline{\mathbf{X}}) \tag{1.6b}$$

$$\mathbf{X}^{(k+1)} \leq \min(\mathbf{X}^{(k)} + \boldsymbol{\Delta}, \ \overline{\mathbf{X}}) \tag{1.6c}$$

The assumption made here is that the feasible domain defined by Eq. (1.6) has a hyper-rectangular shape. Since the move limit strategy is a research topic by itself, in this study we consider only this simple – albeit widely applied – approach.

1.3 Sequential approximate optimization

Probably the most successful optimization algorithms for solving large-scale TO problems, see Fig. 1.3, include the transformation from discrete to a continuous design variable space in order to make use of the local gradient information, i.e. SA. In practice, TO algorithms that include the procedures of FEA and SA are often implemented in the

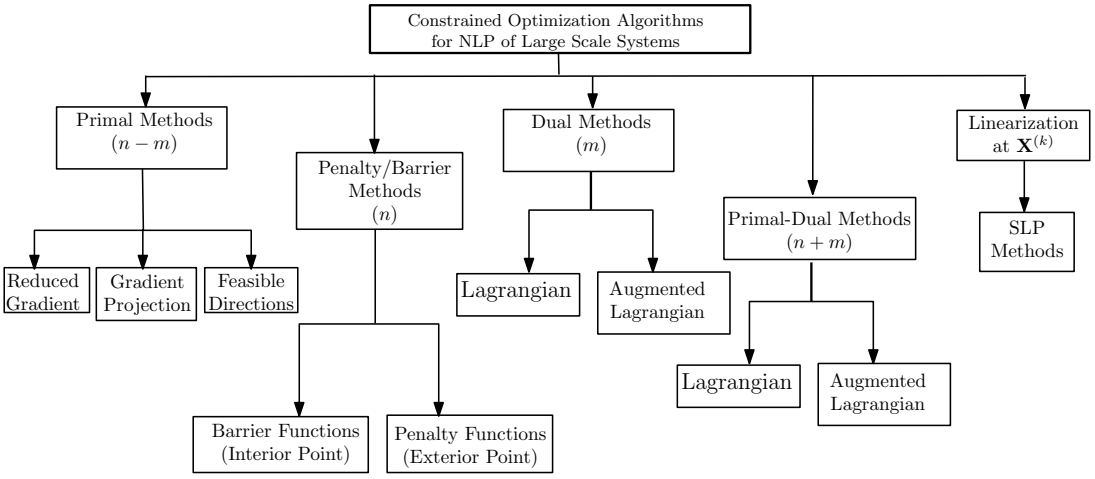


Figure 1.3: Prominent constrained optimization algorithms that applicable to large-scale TO problems [2], [3]

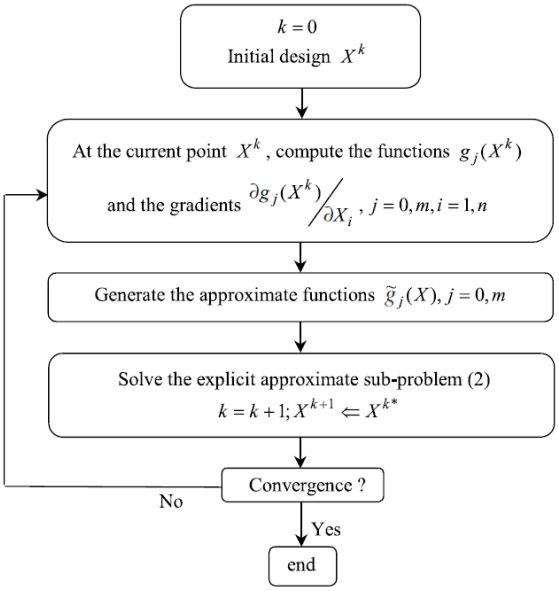
context of SAO. This algorithm, see Fig. 1.4, approximates the primary optimization problem $\mathcal{P}_{\text{NLP}}$ with a sequence of simpler explicit sub-problems $\widetilde{\mathcal{P}}_{\text{NLP}}$ at a given design point $\mathbf{X}^{(k)}$, such that the initial $\mathcal{P}_{\text{NLP}}$ is solved by solving a sequence of approximate sub-problems $\widetilde{\mathcal{P}}_{\text{NLP}}$. This solution strategy for $\mathcal{P}_{\text{NLP}}$ can be seen in Fig. 1.4a, and its graphical representation for the case of 2 design variables in Fig. 1.4b.

The procedure should lead to a stationary solution within a minimum number of iterations and without constraint violations. Important to note is that evaluating the functions $g_j(\mathbf{X}^{(k)})$ and their derivatives $\partial g_j/\partial x_i(\mathbf{X}^{(k)})$ costs the vast majority of the overall computational time. Although such algorithms can tackle robustly problems with a large number of design variables, there has not been any significant change in the methods commonly used during the last decades (i.e. the MMA [27]), despite the fact that several ideas for improvements have arisen [4], [28]. In an attempt to identify the reasons why this occurs, further understanding of the SAO algorithm is necessary. To this end, in the present thesis SAO methods will be studied extensively and on a fundamental level.

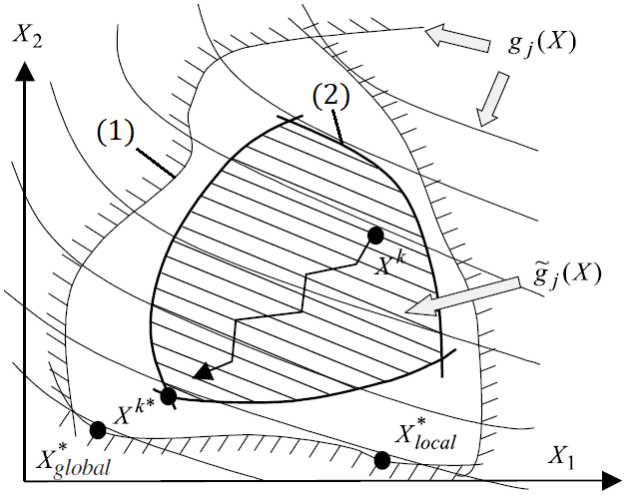
As mentioned above, SAO algorithms comprise of 2 subsequent parts:

(a) Approximation of responses $g_j(\mathbf{X})$ at a design point $\mathbf{X}^{(k)}$, leading to $\widetilde{\mathcal{P}}_{\text{NLP}}$ formulation

(b) Solution of $\widetilde{\mathcal{P}}_{\text{NLP}}$, using an algorithm from Fig. 1.3



(a) Iterative solution scheme



(b) Original optimization problem (1) and its approximated form (2) around the current design point $\mathbf{X}^{(k)}$

Figure 1.4: SAO procedure [4]: Algorithm flowchart (a) and a graphical representation of the solution space at the current iteration (b)

As far as the second part is concerned, in Fig. 1.3 the solution algorithms have been classified with respect to the variable space they operate in (i.e. the number of unknowns involved in the equations whose solution we seek). Primal methods use the constraints to reduce the solution space (i.e. $n \rightarrow n-m$) and essentially look for feasible directions in the primal space ($\mathbf{X} \in \mathbb{R}^n$) that can keep constraints satisfied while minimizing the objective function. Penalty/Barrier methods transform the constrained problem $\mathcal{P}_{\text{NLP}}$ to an unconstrained one (e.g. $\widetilde{\mathcal{P}}_{\text{NLP,P-D}}$) and solutions are found in the primal space ($\mathbf{X} \in \mathbb{R}^n$) too. Primal-Dual methods construct a Lagrangian (or an augmented version of it) and perform an unconstrained minimization with various methods (see [2]) in the augmented variable space ($\{\mathbf{X}, \boldsymbol{\lambda}\} \in \mathbb{R}^{n+m}$). Although dual methods work similarly, they require one more assumption (i.e. closed form of primal-dual relationships, see below) which leads to a simplified problem formulation that allows one to work in the reduced dual space (i.e. $\boldsymbol{\lambda} \in \mathbb{R}^m$). The major advantage of these methods, and the reason of their success, is that the number of inner iterations to solve $\widetilde{\mathcal{P}}_{\text{NLP}}$ depends only on the number of constraints (m) and is decoupled from the number of design variables (n) that is typically large. In combination with techniques for reducing (m), dual methods can become very cost-effective. Of course, the simplest SAO scheme would be to (a) linearize the problem (e.g. with a 1^{st}-order Taylor expansion) and (b) use Linear Programming methods to solve it. However, the convergence of this method would depend on the degree of non-linearity of $\mathcal{P}_{\text{NLP}}$, which is undesirable.

The vast majority of the SAO community seems to implicitly agree on the fact that dual solvers are the favorable option amongst other methods for large scale SO problems in a NAND framework (see Fig. 1.3) by directing most of the ongoing research on them. In order for dual methods to be efficient, the approximate primal approximate sub-problem $\widetilde{\mathcal{P}}_{\text{NLP}}$ must abide by the following requirements:

1. **Separability**
   The primal approximate sub-problem $\widetilde{\mathcal{P}}_{\text{NLP}}$ is separable if its constituent functions (i.e. $\widetilde{g}_j(\mathbf{X}),\ \forall j$) are separable, see Eq. (1.2). The consequences of such a requirement are listed below:
   - A separable function has one fundamental property that can be exploited to improve the efficiency of iterative solution schemes: it has a diagonal Hessian. This makes its computational cost viable, in contrast with the prohibitive cost of the non-diagonal case. Thus, $\widetilde{\mathcal{P}}_{\text{NLP}}$ can be solved very efficiently using approximate 2^{nd}-order information (e.g. Quasi-Newton schemes).
   - A separable Lagrangian means one can solve separately n one dimensional minimization problems [13], instead of solving one $n-$dimensional problem. This can be exploited using parallel computing and thus decrease the duration of the solution process.
   - Bound constraints (i.e. $\underline{x}_i \leq x_i \leq \overline{x}_i,\quad i = 1,\ldots,n$) can be handled very efficiently, as one can rewrite $\mathbf{X} = \mathbf{X}(\boldsymbol{\lambda})$ as $x_i = x_i(\boldsymbol{\lambda}),\ \forall i$. This enables the explicit application of each bound constraint on the respective design variable [13]. Otherwise, one would have to consider $2n$ extra constraints, which would render the method inefficient, as the working variable space would become $\{\mathbf{X}, \boldsymbol{\lambda}\} \in \mathbb{R}^{2n+m}$.
2. **Convexity**
   $\widetilde{\mathcal{P}}_{\text{NLP}}$ is convex if its constituent functions (i.e. $\widetilde{g}_j(\mathbf{X}),\ \forall j$) are convex. Although convexity is a sufficient – but not a necessary – condition in order to invoke Falk's dual formulation [29], most research papers avoid its strict mathematical requirements and simply assume convexity of all functions $\widetilde{g}_j(\mathbf{X})$. However, the actual prerequisite is uniqueness of minimizer $\mathbf{X}^*$ for the Lagrangian $L(\mathbf{X}, \boldsymbol{\lambda})$ within its bound constraints (i.e. $x_i \in [\underline{x}_i, \overline{x}_i],\quad i = 1,\ldots,n$) for any arbitrary $\boldsymbol{\lambda}$ [30]. The benefits of such assumptions are:
   - Guarantees the sufficiency – apart from necessity – of KKT conditions to attain convergence, see Eq. (1.4)
   - The unique solution of convex problems allows the use of efficient gradient-based algorithms that converge fast
   - The solution of the primal problem coincides with the solution of its dual (i.e. no duality gap [13])
3. **Closed form of primal-dual relationships $\mathbf{X} = \mathbf{X}(\boldsymbol{\lambda})$**
   For the effectiveness of dual solvers, it is crucial for $\widetilde{\mathcal{P}}_{\text{NLP}}$ to be formulated in a way that the KKT conditions of Eq. (1.4) can be solved efficiently and yield $x_i = x_i(\boldsymbol{\lambda}),\ \forall i$ within each iteration. The repercussions of this requirement are:
   - This $(n \times n)$ system can be solved analytically during every iteration (k), without the need of an iterative process (e.g. a Newton-Raphson scheme) which would result in an increased computational cost. This is the reason why some more advanced approximation schemes that do not allow an analytical solution (e.g. GCMMA [27], TANA [31]) have not been used extensively in SAO [12]. Typically, explicit relationships of $x_i = x_i(\boldsymbol{\lambda}),\ \forall i$ that can be solved analytically are possible if the approximation functions $\widetilde{g}_j(\mathbf{X})$ are simple enough. Unfortunately, no strict definition for which approximations yield explicit primal-dual relationships that can be solved analytically has been found in the literature. However, researchers (see [32], [33]) have reported methods that are guaranteed to yield such explicit relationships.

Furthermore, keeping in mind that the influence of the solver is not the focus of the present research and that it has been reported that the non-explicit character of primal-dual relationships is manageable [13], we attempt to keep the generated method (see Chapter 4) as general as possible by implementing a primal-dual interior point solver instead of a purely dual one.

Primal-dual Sequential Approximate Optimization

Being a sub-field of SAO, the Dual Sequential Approximation (DSA) algorithm [12] is perhaps the most widely applied algorithm for TO problems, considering the popularity of the NAND formulation and the poor performance of Optimality Criteria (OC) methods when multiple constraints are present [34]. One can implement several different

solvers in a DSA framework, depending on the problem type at hand. In the present report, we will be using a variation of DSA with a primal-dual interior point solver [35]. To depict this difference, we will refer to the DSA variation used herein as PDSA. In this section, we will briefly describe the general steps one needs to take in order to apply PDSA to any problem:

- Formulate $\mathcal{P}_{\text{NLP}}$: Modelling of physical phenomena in the form of Eq. (1.1)
- Calculate function values $g_j(\mathbf{X}^{(k)})$ and their sensitivities $\partial g_j/\partial x_i(\mathbf{X}^{(k)})$ by performing FEA and SA respectively at a design point $\mathbf{X}^{(k)}$
- Formulate $\widetilde{\mathcal{P}}_{\text{NLP}}$: Approximate $\mathcal{P}_{\text{NLP}}$ locally with Taylor-like expansions around the current point $\mathbf{X}^{(k)}$, see Eq. (1.2)
- Formulate $\widetilde{\mathcal{P}}_{\text{NLP,P}-\text{D}}$: Convert the constrained $\widetilde{\mathcal{P}}_{\text{NLP}}$ to an unconstrained problem that can be solved more efficiently under certain assumptions, see Eq. (1.5)
- Solve $\widetilde{\mathcal{P}}_{\text{NLP,P}-\text{D}}$: Find the design variable vector (i.e. minimizer $\mathbf{X}^{*(k)}$) that minimizes $\widetilde{\mathcal{P}}_{\text{NLP,P}-\text{D}}$ and therefore $\widetilde{\mathcal{P}}_{\text{NLP}}$
- Update the current design point ($\mathbf{X}^{(k+1)} = \mathbf{X}^{*(k)}$) and repeat the process from the 2^{nd} step until convergence

An overview of the the PDSA algorithm can be seen in Fig. 1.5, from which one can clearly see the similarities with Fig. 1.4a of the SAO scheme.

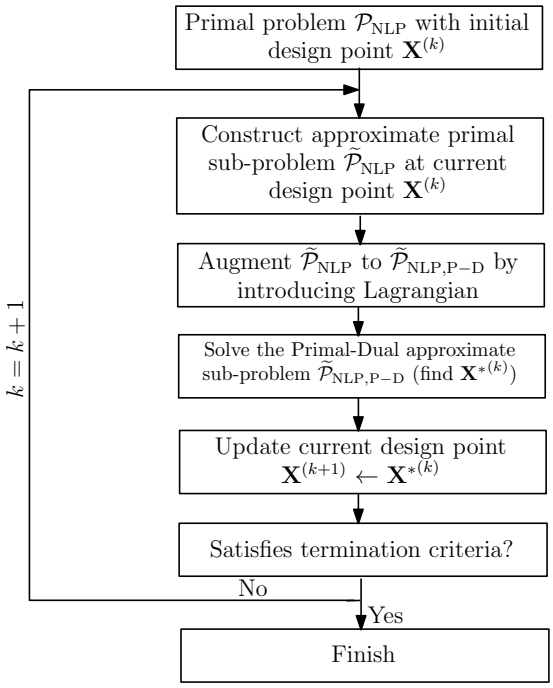


Figure 1.5: PDSA algorithm flowchart

1.4 Examples of inadequate performance of existing methods

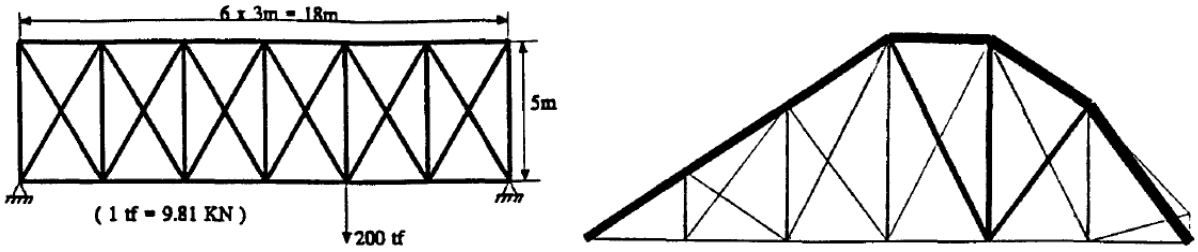
There are several problems in the SAO literature that are known to cause approximations of low quality. For example, researchers have reported poor convergence properties of optimization problems concerning certain truss configurations [5], composite material layering [4] and self-weight problems [8]. In this section we will briefly describe some of those issues, as well as the reasons for the inadequate performance of approximations. A list of some of the SO problems that are known to cause these issues can be found below:

- Dissimilar behaviour of design variables
- Aggregated response functions

- Design-dependent loads
- Design-dependent constraints
- Dissimilar behaviour of response functions

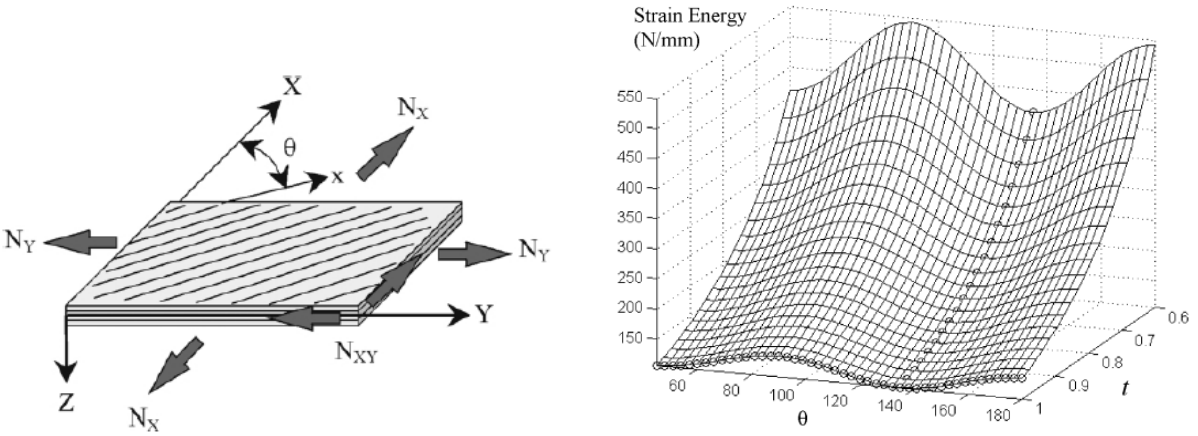
Dissimilar behaviour of design variables

Already from 1998, researchers [5] had observed that the diversity in design variable types of SO problems could have serious implications on the performance of an optimization algorithm. To the author's knowledge, the first paper that addressed this issue referenced truss configuration problems with both cross-sectional areas and nodal coordinates as design variables, see Fig. 1.6. Along the same lines, composite material layering typically includes finding the optimal fiber orientation (θ^*) as well as the optimal ply thickness (t^*). In optimization terms, this effectively means there are at least two inherently different design variable types that exhibit monotonous and non-monotonous behaviour respectively, see Fig. 1.7. Under these conditions, and taking into account that the 'structure' of an approximating function of Section 2.1 must be similar to the 'nature' of the function it approximates (e.g. Fig. 1.7b) for an efficient SAO algorithm, one can clearly conclude that a single type of approximation will be sub-optimal for this type of problems. In both cases, the proposed solution was to introduce a mixed approximation scheme (i.e. a combination of different local analytical approximation functions) able to address inherently dissimilar design variable types.



(a) Initial truss configuration **(b)** Optimal truss configuration

Figure 1.6: Truss configuration problem [5]: Nodal coordinates and cross-sectional areas are used as design variables



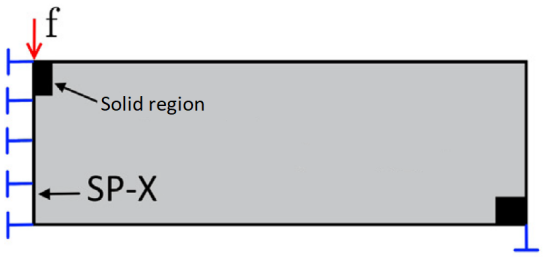
(a) In plane loading of ply laminates **(b)** Strain energy density of ply laminate

Figure 1.7: Composite layering optimization problem [6]: Shear and torsional stress of a planar laminate

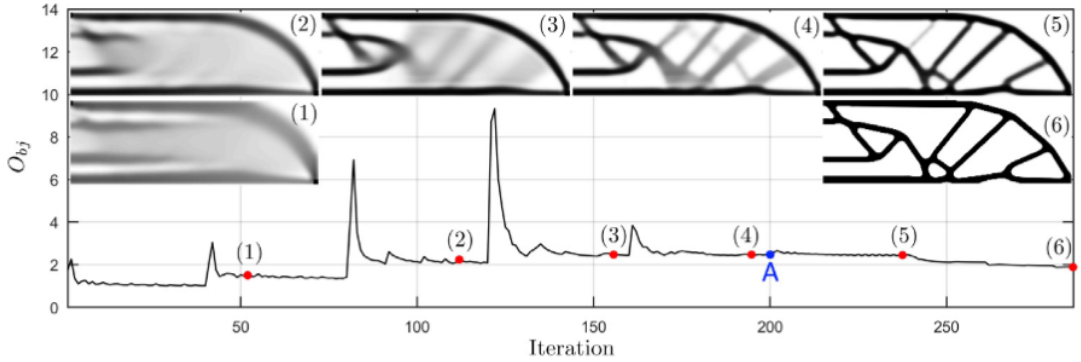
Aggregated response functions

A fabrication method that is attracting increasing interest in topologically optimized designs is additive manufacturing. The design of a structure was initially decoupled from its manufacturing process, resulting in large discrepancies

between the expected (from FEA simulations) and the observed (from experiments on the fabricated product) performance. However, researchers [7] were able to include manufacturability issues in the problem formulation of $\mathcal{P}_{\text{NLP}}$. Minimum and maximum feature sizes were imposed on the design as constraints to increase the robustness of the fabricated product's performance against manufacturing defects. These local geometric constraints are implemented in an aggregated formulation in order to preserve the efficiency of dual solvers in a NAND setting of TO problems when $m << n$. Nonetheless, including these aggregated response functions in a TO problem formulation has significant consequences on its convergence curve, see Fig. 1.8b.



(a) 2D-MBB beam of size $3L \times L$: The blue lines represent the DoFs that are fixed and the black areas are the non-optimizable zones defined as solid material. SP-X represents a symmetry plane with respect to the X-axis



(b) Convergence curve and design evolution throughout the optimization

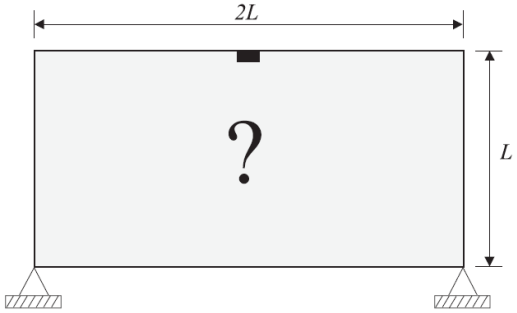
Figure 1.8: Density-based TO of a 2D-MBB beam with aggregated geometric constraints on minimum and maximum feature size for robust performance against manufacturing defects [7]

Design-dependent loads

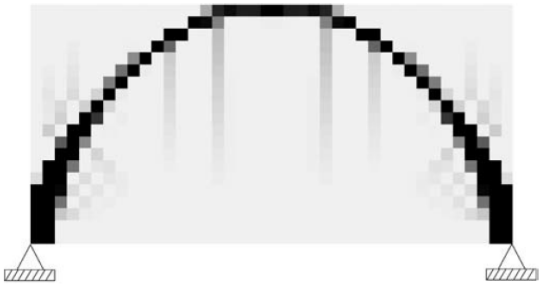
This class of problems is a generalized formulation of the most popular forms of TO problems (e.g. compliance minimization with constant loads, i.e. $\mathbf{F} = const$). As their name suggests, they involve problem formulations for which the applied loads on the structure under design depend on the design itself, i.e. $\mathbf{F} = \mathbf{F}(\mathbf{X})$. The interested reader is referred to [36] for a detailed review on the topic. As an example, we can think of a bridge design (see Fig. 1.9b) whose applied load is its own weight. Clearly, the applied load depends on the design itself, adding a degree of complexity to the problem as the response functions involved change – along with the design – from one iteration to another. For example, the self-weight problems have proven to be particularly hard to solve. Researchers [8] have reported that a direct extension from minimum compliance problems leads to poor algorithm convergence, due to:

- A non-monotonous compliance behaviour, see Fig. 1.9c
- A possibly unbounded solution
- Parasitic effects for densities close to zero when the Solid Isotropic Material with Penalization (SIMP) law is used, see Fig. 1.9b

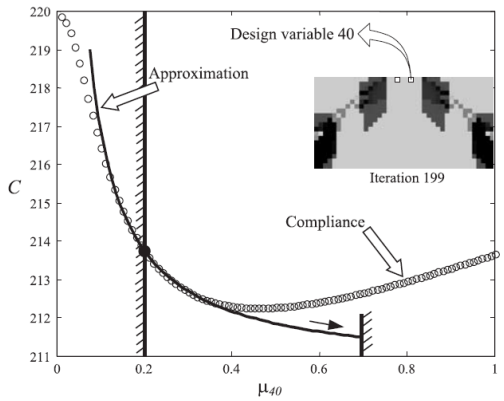
Various workarounds have been suggested for self-weight problems, see [8]. Those may include a combination of modifying the SIMP law for close to zero densities in order to clear out artifacts, as well as the use of mixed approximation schemes to tackle the aforementioned complicated response function behaviour.



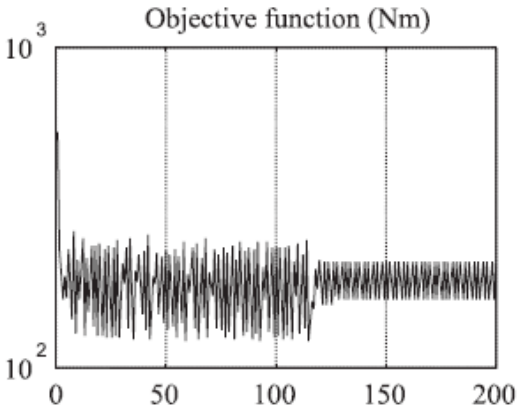
(a) Self-weight arch problem



(b) Artifacts of SIMP model for low densities



(c) Non-monononous behaviour of compliance

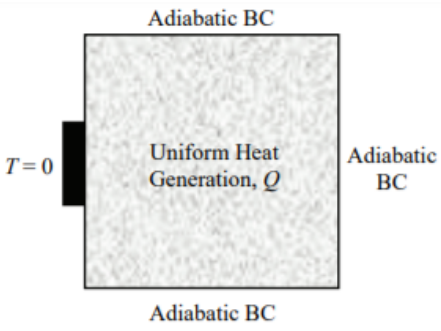


(d) Convergence oscillations

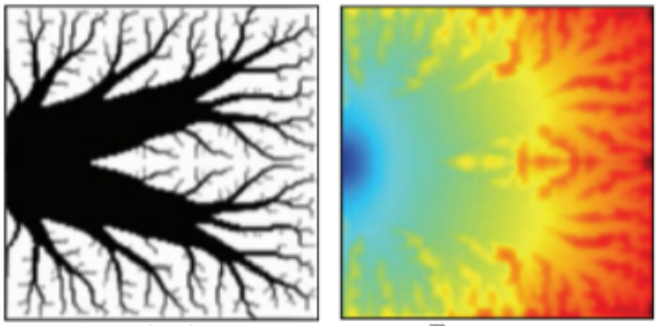
Figure 1.9: Intricacies of self-weight TO problems [8] illustrated by a 2D arch problem

Design-dependent constraints

Another category of problems that are prone to undesired convergence behaviour are design-dependent constraint problems, see [10]. The main reason for this undesired behaviour is the fact that when material is absent from a certain region, so is the respective constraint. Just like stress constraints in truss configurations are present only for members that have a cross-sectional area larger than zero (see Fig. 1.11), heat constraints are only present when the density of a region is positive (see Fig. 1.10). Therefore, as the design changes form between iterations, apart from the applied loads, the applied constraints must also change accordingly.



(a) A 2-D heat transfer TO problem

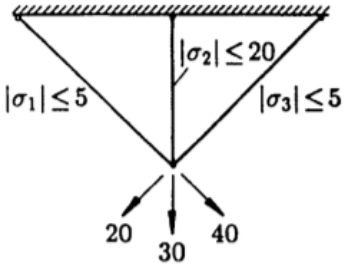


(b) Material distribution (left) and Temperature field (right)

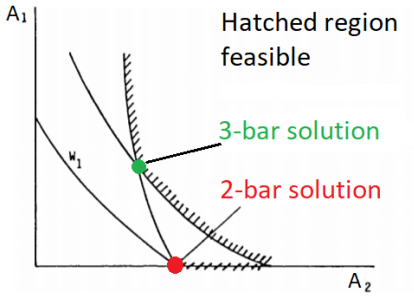
Figure 1.10: A heat distribution TO problem [9]

The implications of design-dependent constraints are well known and have been described in detail. To give an example, [10] investigated the 3-bar truss configuration of Fig. 1.11a and concluded that without the modifications

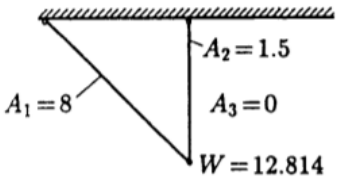
proposed therein, only the 3-bar solution of Fig. 1.11d can be obtained. The 2-bar solution of Fig. 1.11c is unreachable from any starting point of the solution space shown in Fig. 1.11b.



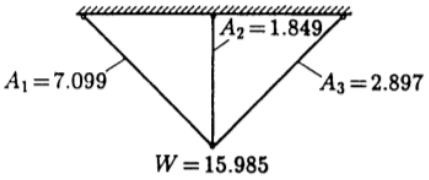
(a) 3-bar truss problem



(b) Solution space for cross-sectional areas



(c) 2-bar solution



(d) 3-bar solution

Figure 1.11: Intricacies of design-dependent constraint problems [10]: The 3-bar truss problem

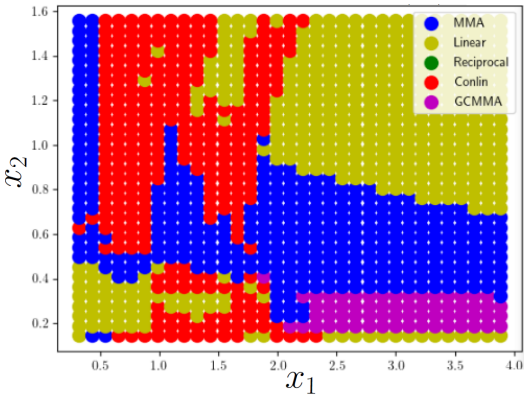
Dissimilar behaviour of response functions

From the previous examples one can conclude that the increasing complexity of optimization problems will likely render most current approximation methods of SAO insufficient in the near future. In addition, the need for intricate optimization problems is expected to keep rising as the available computational power grows and multi-disciplinary problem formulations become common practice, see [23], [24] and [9] for some examples. Inevitably, an increase in the number and diversity of the design variables used to model these problems will follow.

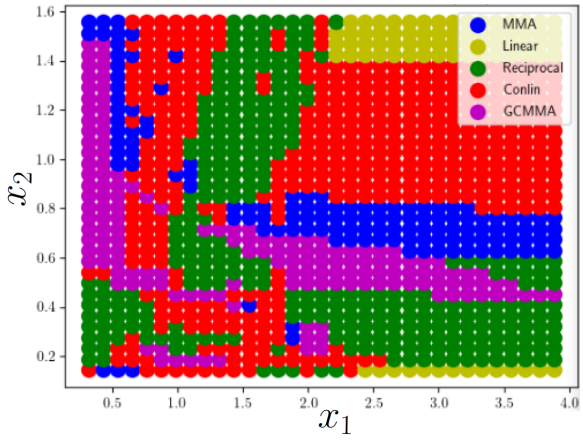
Reasonably, one can estimate that only an equally diverse and adaptable approximation scheme will be able to address the aforementioned complex problems robustly and efficiently. As an example, one can clearly see in the novel analysis of Fig. 1.12 that, even for the simple problem of Fig. 1.12a, different functions $g_j(\mathbf{X})$ have different optimal approximations $\widetilde{g}_j(\mathbf{X})$ at different approximation points $\mathbf{X}^{(k)}$ (the colors in the plots indicate the best-fitting local approximation). Taking the above into consideration, advances in the SAO community must follow the requirements of the state-of-the-art problems by upgrading the algorithms used in order to address the most intricate problems.

$$
\begin{aligned}
\underset{\mathbf{X}}{\text{minimize}} \quad & g_0(\mathbf{X}) = c_1 x_1 \sqrt{1+x_2^2} \\
\text{s.t.} \quad & g_1(\mathbf{X}) = c_2 \sqrt{1+x_2^2} \left(\frac{8}{x_1} + \frac{1}{x_1 x_2} \right) - 1 \leq 0 \\
& g_2(\mathbf{X}) = c_2 \sqrt{1+x_2^2} \left(\frac{8}{x_1} - \frac{1}{x_1 x_2} \right) - 1 \leq 0 \\
& 0.2 \leq x_1 \leq 4 \\
& 0.1 \leq x_2 \leq 1.6
\end{aligned}
$$

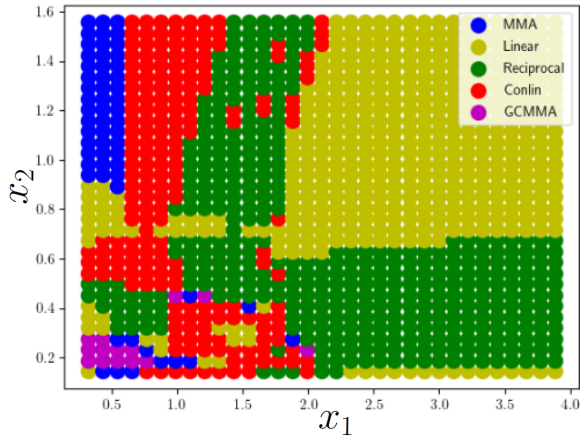
(a) 2-bar truss problem [12]



(b) Best approximation for $g_0(\mathbf{X})$



(c) Best approximation for $g_1(\mathbf{X})$



(d) Best approximation for $g_2(\mathbf{X})$

Figure 1.12: A novel representation of the optimal approximation functions $\widetilde{g}_j(\mathbf{X})$ for all possible approximation points of the feasible domain of (a), i.e. $\forall \mathbf{X}^{(k)} \in [\underline{\mathbf{X}}, \overline{\mathbf{X}}]$

1.5 Thesis aim and approach

The aim of the present research is to improve the performance of SAO algorithms. To do so, the number of iterations (k) and/or the cost per iteration must be minimized. Since the computational cost largely depends on the number of FEA – and therefore the number of iterations – in the present work we focus on reducing the number iterations by improving the approximation $\widetilde{g}_j(\mathbf{X})$ of response $g_j(\mathbf{X})$ at each design point $\mathbf{X}^{(k)}$, using information that is readily available (e.g. from previous iteration points). This is possible because of the high computational cost of redundant function evaluations that are associated with the oscillatory behaviour and/or the insufficient change in the design variables when approximations of low quality are generated. Moreover, since we ultimately aim to address the most intricate SO problems (e.g. multi-FEA TO formulations) and preserve the general character of our method, our approach will include the implementation of a primal-dual solver. By using the same solver for all the implemented approximations, we can test the effects of a new method. Consequently, caution is needed to preserve the prerequisites of such a solver (i.e. convexity and separability, see Section 1.3).

It should be clear by now that constructing an accurate approximation for any arbitrary response $g_j(\mathbf{X})$ at any design point $\mathbf{X}^{(k)}$ is not trivial. Moreover, another important consideration is to what extent enriching the approximation process with more information is cost effective, due to additional computational cost of the accompanying parameter estimation (e.g. applying Eq. (2.17)). Consequently, approximation enrichment must be realized with caution.

In order to improve the performance of SAO algorithms, one must first understand their behaviour on a fundamental level. To this end, applying these algorithms to a set of complicated and large-scale problems is neither insightful nor efficient. At this stage, a set of small-scale – albeit challenging – problems that can simulate the characteristics of the large-scale problems we ultimately aim to address can offer significantly more insights. This can be explained

by the fact that one can perform a detailed, graphical and computationally expensive analysis (e.g. plotting pairs of $\{x_i, g_j\}$ and $\{x_i, \widetilde{g}_j\}$ at every iteration) on a problem with a small number of variables and functions, whereas for large-scale problems this is no longer possible. Only after an in-depth understanding of the algorithms under investigation is achieved and strong indications for improved performance on smaller problems are present, can one proceed to including complicated, large-scale problems. Consequently, the focus of the present thesis is to understand the behaviour and improve the performance of SAO methods on small-scale problems. This effectively means that the product of the number of design variables and response functions $n \cdot (m + 1)$ is relatively small for all the considered cases. This product is referred to as the 'characteristic size' of a problem in the present thesis.

Our approach to this topic will include the following aspects:

- An extensive literature review of the state-of-the-art SAO methods used in SO.
- Implementing the most popular methods (i.e. MMA family of approximations, see [4] for a detailed review thereof) and verifying results with literature by applying them to benchmark problems. It is important however for this implementation have a modular structure, capable of being enriched with additional features (e.g. approximation methods, problems, performance metrics, etc.) as the research progresses. Therefore, our goal is to create a modular software library that will facilitate the generation of new approximation schemes. We will herein refer to this library as 'Optimization Lab'.
- Locating and understanding the inadequacies of the state-of-the-art optimization methods (e.g. oscillating or slow convergence when dissimilar design variable types are present [6]). Our endeavour addresses challenging problems, wherein the aforementioned methods perform poorly, see [12], [37], [14] and [38]. This step is quite involved, since patterns of detrimental situations for the implemented algorithms need to be detected in order to avoid them later on.
- Subsequently, a class of optimization problems wherein the state-of-the-art methods perform poorly will be defined and a method will be generated, specifically tailored for this class of problems.
- The method generated will be:
  - Using the appropriate approximation method for each response $g_j(\mathbf{X})$, instead of using the same one for responses that may be inherently dissimilar. To the author's knowledge, the most prominent method to achieve this systematically is the GMMA family of approximations, see [39]. Therefore, our approach will investigate the performance of this method and use it as a reference for benchmarking.
  - Choosing consistently the most suitable approximation method $\widetilde{g}_j(\mathbf{X})$ at each design point $\mathbf{X}^{(k)}$ and avoiding error propagation, in the sense of convergence complications that arise when approximations do not 'fit' the respective responses $g_j(\mathbf{X})$ accurately enough. This effectively means that based on inexpensive information stored from previous iteration points a criterion is formed, the output of which is the selection of an appropriate approximation method for the current iteration (k). Therefore, this step addresses method selection depending on the situation encountered. This will lead to an adaptive scheme that has the ability to deal with challenging problems robustly and efficiently. Researchers have already addressed this possibility, see [40], [13] and [12], but there is still plenty of room for further research.
- Last but not least, the performance evaluation of the generated method needs to be taken into account. Apart from choosing a diverse set of benchmark problems that would simulate the variety of complications an SAO algorithm may encounter, the influence of a response's 'structure' around an arbitrary expansion point $\mathbf{X}^{(k)}$ needs to be addressed as well, as the local behaviour of a function would affect the results and might compromise the objectivity of the derived conclusions.

2 | Review of existing methods

2.1 Approximation methods

The use of approximation functions in SO problems is not a new approach. The first implemented approximations already appeared in the mid-70s [41]. The most accepted classification of approximation methods for SO [11] distinguishes between local, medium-range and global approximations (with respect to their range of application) and covers function (i.e. $g_j(\mathbf{X})$), as well as problem (i.e. $\mathcal{P}_{\text{NLP}}$) simplifications (with respect to the approximated quantity), see Fig. 2.1 and Fig. 2.2. An important note here is that function and problem approximations are independent of one another. In the present report we focus our attention on function approximations, assuming that $\mathcal{P}_{\text{NLP}}$ has already been simplified as much as possible in the modelling phase.

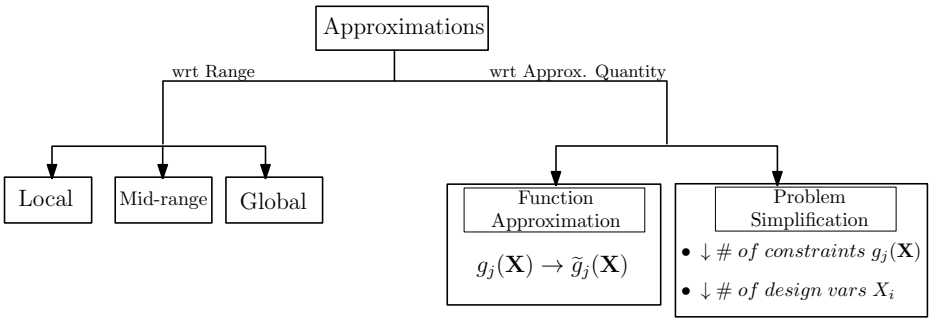


Figure 2.1: Classification of approximation methods depending on their range and the approximated quantity [11]

Moreover, a summary of the optimization process for an SO problem can be seen in Fig. 2.3, wherein one can better understand the role of a function approximation in such a framework. Since we aim to minimize the number of function evaluations, which is the dominant computational cost, it makes sense to focus our research on local and mid-range function approximations rather than global ones, as they require a minimum amount of FEA (i.e. 1 per iteration, see Fig. 2.3). In the coming sections, these methods will be described in detail.

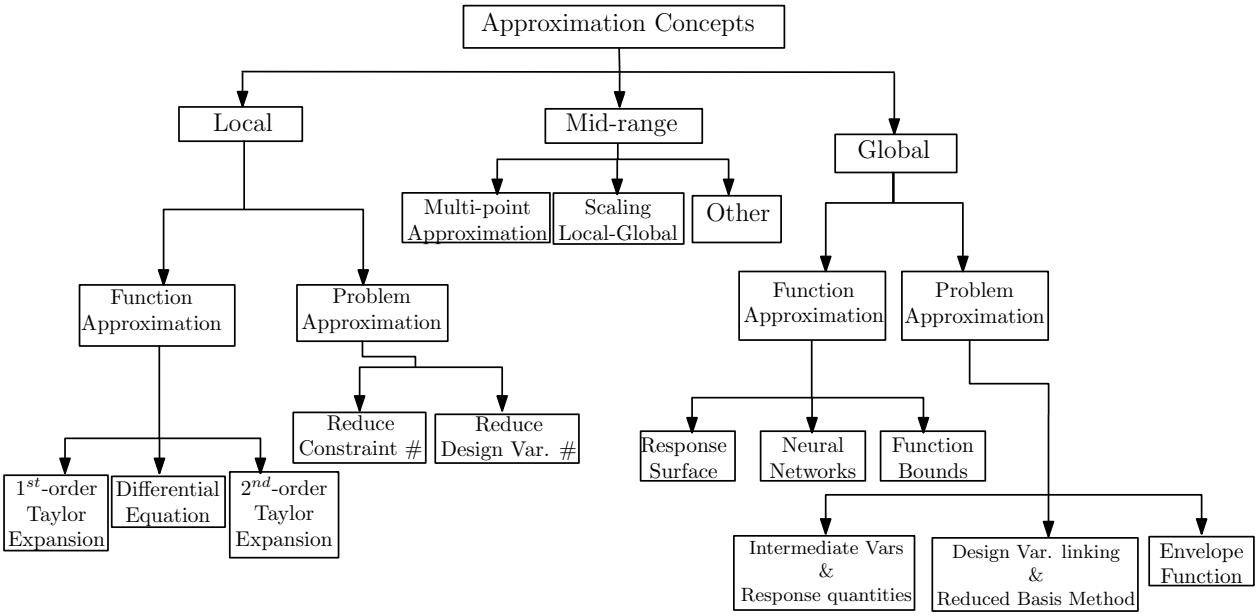


Figure 2.2: Detailed classification of approximation methods with respect to their range [11]

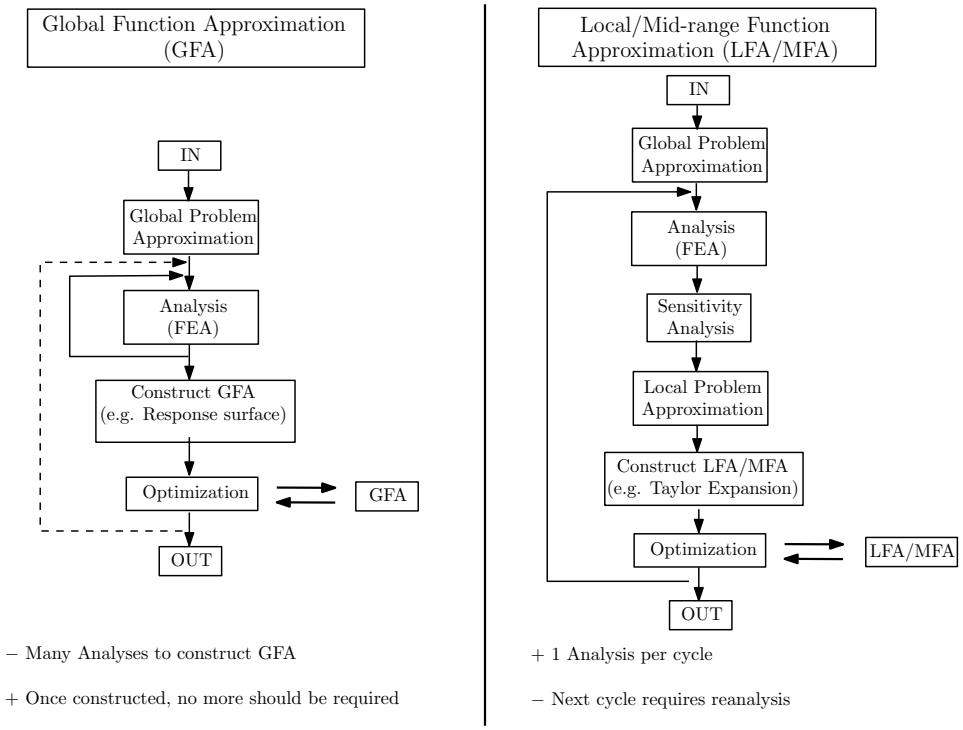
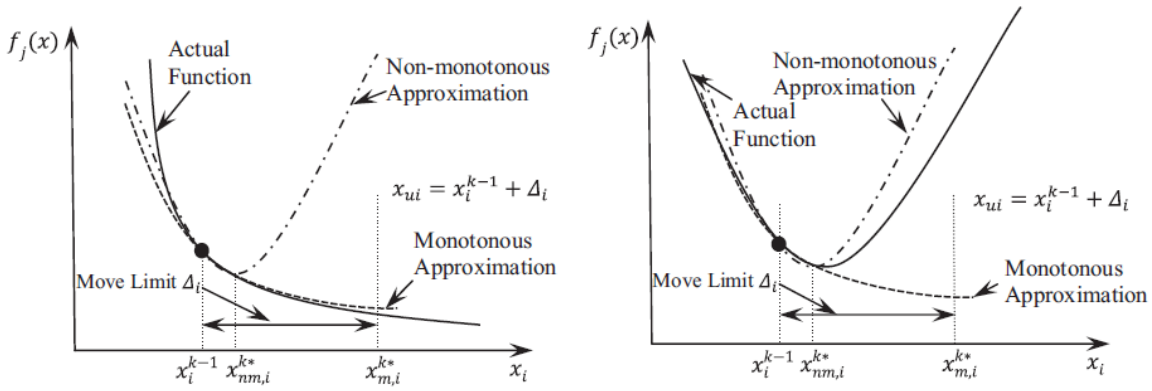


Figure 2.3: Flowcharts of the optimization process of global vs. local/mid-range function approximations in SO [11]

Another way to classify approximations is with respect to their monotonicity. In calculus, a function f_j is defined as monotonous within a specified domain if – and only if – it is either entirely non-increasing, or entirely non-decreasing. On the contrary, a non-monotonous function f_j must change its monotonicity (and therefore the sign of at least one of its gradient components) within the specified domain. This is an important classification factor, as it plays an crucial role in the quality of the generated approximation. On the one hand, if the gradient components of the actual function at hand f_j change sign in the vicinity of the expansion point $\mathbf{X}^{(k)}$, one would prefer a non-monotonous approximation for convergence stability, see Fig. 2.4b. On the other hand, if f_j shows monotonous behaviour around $\mathbf{X}^{(k)}$, faster convergence is achieved with an approximation that shares a similar – monotonous – behaviour, see Fig. 2.4a.



(a) Monotonous function

(b) Nonmonotonous function

Figure 2.4: Monotonous, nonmonotonous functions and their local approximations [12]

While studying the influence of approximation methods on the performance of an optimization algorithm, one can clearly conclude that there is no 'free lunch'. This means that there is no function $\widetilde{f}_A$ that can outperform all others (e.g. $\widetilde{f}_B$) for a large set of $\{\mathbf{X}^{(k)}, f\}$. This can be explained by considering that the best approximation $\widetilde{f}$ for a function

f at a given point $\mathbf{X}^{(k)}$ is the one that has the most similar 'structure' or 'nature' to f [42]. Consequently, one should not look for a single approximation function that is superior to all others, but for a flexible family of approximations and a selection criterion for them, depending on the characteristics of the case at hand. In the following sections, the most notable local and mid-range function approximations are described.

2.2 Local function approximations

Local approximations are valid in the vicinity of their expansion point $\mathbf{X}^{(k)}$. Typically, these approximations are variations of the truncated Taylor series expansion with respect to direct, see Eq. (2.1), or intermediate (i.e. $y_i = y_i(x_i)$) variables, see Eq. (2.2).

Linear Approximation

This is the simplest form of local approximation and the basis of the forthcoming approaches. It is based on the 1^{st}-order Taylor expansion around the current design point $\mathbf{X}^{(k)}$:

$$\widetilde{g}_j(\mathbf{X}) = g_j(\mathbf{X}^{(k)}) + \sum_{i=1}^{n} \left[\left.\frac{\partial g_j}{\partial x_i}\right|_{\mathbf{X}^{(k)}} \left(x_i - x_i^{(k)}\right) \right] \tag{2.1}$$

Reciprocal Approximation

Attempts to improve the accuracy of approximations without increasing their computational cost (e.g. by adding 2^{nd}-order terms) were first introduced in the 80s. Inarguably, the most popular choice of intermediate variables is the use of reciprocal ones (i.e. $y_i = y_i(x_i) = 1/x_i$), see Eq. (2.2). Although there is no rigorous mathematical proof of their superiority over other options, their success in SO can be justified by the following facts:

- Stresses and displacements are exact linear functions of $y_i = 1/x_i$ for sizing problems. Therefore, since the responses $g_j(\mathbf{X}) \leq 0, \quad j = 1, \ldots, m$ typically include stress and displacement constraints in SO problems, their approximation in the reciprocal variable space (i.e. Taylor expansion with respect to $y_i = 1/x_i$) improves the approximation accuracy without any additional computational cost (e.g. 2^{nd}-order information).
- Reciprocal expansion convexifies the approximation when $\frac{\partial g_j(\mathbf{X})}{\partial X_i} < 0$.

$$\begin{aligned} \widetilde{g}_j(\mathbf{X}) &= g_j(\mathbf{X}^{(k)}) + \sum_{i=1}^{n} \left[\left.\frac{\partial g_j}{\partial x_i}\right|_{\mathbf{X}^{(k)}} \left.\frac{\partial x_i}{\partial y_i}\right|_{\mathbf{Y}^{(k)}} \left(y_i - y_i^{(k)}\right) \right] \overset{1}{=} g_j(\mathbf{X}^{(k)}) + \sum_{i=1}^{n} \left[-\left(x_i^{(k)}\right)^2 \left.\frac{\partial g_j}{\partial x_i}\right|_{\mathbf{X}^{(k)}} \left(\frac{1}{x_i} - \frac{1}{x_i^{(k)}}\right) \right] \\ &= g_j(\mathbf{X}^{(k)}) + \sum_{i=1}^{n} \left[\left.\frac{\partial g_j}{\partial x_i}\right|_{\mathbf{X}^{(k)}} \frac{x_i^{(k)}}{x_i} \left(x_i - x_i^{(k)}\right) \right] \end{aligned} \tag{2.2}$$

However, when $\left.\frac{\partial g_j}{\partial x_i}\right|_{\mathbf{X}^{(k)}} > 0$ the reciprocal approximation $\widetilde{g}_j$ becomes nonconvex, which is undesirable if one wishes to use a primal-dual (or a purely dual) solver. This complication is circumvented by the next approximation scheme.

CONLIN Approximation

Introduced by [43], CONvex LINearization combines the benefits of both Linear and Reciprocal approximation schemes. It works by opting for one of them, depending on the following conditions:

1. If $\left.\frac{\partial g_j}{\partial x_i}\right|_{\mathbf{X}^{(k)}} \geq 0$, then $\sum_+$ in Eq. (2.3) is used for variable x_i
2. If $\left.\frac{\partial g_j}{\partial x_i}\right|_{\mathbf{X}^{(k)}} < 0$, then $\sum_-$ in Eq. (2.3) is used for variable x_i

That way, the approximation is unconditionally convex and separable. Moreover, it is worth noting that being the most conservative 1^{st}-order approach, the probability of oscillating convergence behaviour is quite low. This means that CONLIN is locally conservative, as it has an inherent tendency to generate a sequence of steadily improving feasible

[1] By substituting $y_i = 1/x_i$

designs $\mathbf{X}^{(k)}$ by overestimating the values of actual functions [43], i.e. $\widetilde{g}_j(\mathbf{X}^{(k)}) \geq g_j(\mathbf{X}^{(k)})$. The approximation scheme is given by:

$$\widetilde{g}_j(\mathbf{X}) = g_j(\mathbf{X}^{(k)}) + \sum_{+} \left[\frac{\partial g_j}{\partial x_i}\bigg|_{\mathbf{X}^{(k)}} \left(x_i - x_i^{(k)}\right)\right] + \sum_{-} \left[-\left(x_i^{(k)}\right)^2 \frac{\partial g_j}{\partial x_i}\bigg|_{\mathbf{X}^{(k)}} \left(\frac{1}{x_i} - \frac{1}{x_i^{(k)}}\right)\right] \tag{2.3}$$

The most significant drawback of this method is its fixed curvature, i.e. there is no parameter in Eq. (2.3) to adjust the convexity of $\widetilde{g}_j(\mathbf{X})$ based on the available information. This can lead to bad fitting of the approximation on the actual function, rendering convergence either too conservative (i.e. slow) or unstable [13].

MMA

Curvature adjustment became possible with the Method of Moving Asymptotes [27]. This monotonous scheme is a generalization of CONLIN, as it is essentially still a 1^{st}-order Taylor expansion with respect to the intermediate variables $y_i = \frac{1}{U_i - x_i}$ or $y_i = \frac{1}{x_i - L_i}$ depending on the sign of the function's partial derivatives $\frac{\partial g_j}{\partial x_i}\big|_{\mathbf{X}^{(k)}}$. The resulting formula is given by:

$$\widetilde{g}_j(\mathbf{X}) = r_j^{(k)} + \sum_{i=1}^{n} \frac{p_{ij}^{(k)}}{U_i^{(k)} - x_i} + \sum_{i=1}^{n} \frac{q_{ij}^{(k)}}{x_i - L_i^{(k)}} \tag{2.4}$$

where,

$$\begin{aligned} p_{ij}^{(k)} &= \max\left\{0,\ \left(U_i^{(k)} - x_i^{(k)}\right)^2 \frac{\partial g_j}{\partial x_i}\bigg|_{\mathbf{X}^{(k)}}\right\} \\ q_{ij}^{(k)} &= \max\left\{0,\ -\left(x_i^{(k)} - L_i^{(k)}\right)^2 \frac{\partial g_j}{\partial x_i}\bigg|_{\mathbf{X}^{(k)}}\right\} \end{aligned} \tag{2.5}$$

with $r_j^{(k)}$ collecting all 0^{th}-order terms and $L_i^{(k)}, U_i^{(k)}$ acting as lower and upper 'asymptotes' of the current approximation, such that $L_i^{(k)} < x_i < U_i^{(k)}$ [13]. These asymptotes play the role of move limits, restricting the allowable step in the design space. The behaviour of the MMA approximation can be adjusted by fine-tuning their values. More specifically, as the asymptotes approach the expansion point, the approximation's convexity is increased and the step-size is reduced. What is more, by substituting $L_i = 0$ and $U_i \to \infty$ in Eq. (2.4), one obtains the CONLIN approximation of Eq. (2.3). Initial values and update schemes for the asymptotes are given by [27] as heuristic rules:

- for $k = 1, 2$:

$$\begin{aligned} L_i^{(k)} &= x_i^{(k)} - s_0(\overline{x}_i - \underline{x}_i) \\ U_i^{(k)} &= x_i^{(k)} + s_0(\overline{x}_i - \underline{x}_i) \end{aligned} \tag{2.6}$$

  where, $s_0 = 0.5$

- for $k > 2$:

$$\begin{aligned} L_i^{(k)} &= x_i^{(k)} - s^{(k)}\left(x_i^{(k-1)} - L_i^{(k-1)}\right) \\ U_i^{(k)} &= x_i^{(k)} + s^{(k)}\left(U_i^{(k-1)} - x_i^{(k-1)}\right) \end{aligned} \tag{2.7}$$

  where,

$$s^{(k)} = \begin{cases} 0.7, & if\ \left(x_i^{(k-2)} - x_i^{(k-1)}\right) \cdot \left(x_i^{(k-1)} - x_i^{(k)}\right) < 0 \\ 1.2, & if\ \left(x_i^{(k-2)} - x_i^{(k-1)}\right) \cdot \left(x_i^{(k-1)} - x_i^{(k)}\right) > 0 \\ 1, & if\ \left(x_i^{(k-2)} - x_i^{(k-1)}\right) \cdot \left(x_i^{(k-1)} - x_i^{(k)}\right) = 0 \end{cases} \tag{2.8}$$

  In Eq. (2.8), the 1^{st} case applies when oscillating behaviour is detected. Then, step-size should be reduced by increasing the approximation's convexity as the asymptotes get shifted towards the expansion point $\mathbf{X}^{(k)}$. The 2^{nd} case applies when convergence is smooth and one wants to increase the step-size by relaxing the approximation's convexity, as the asymptotes are moved away from the expansion point to accelerate convergence. In the 3^{rd} case, the asymptotes (and therefore the step-size) are kept constant because of their adequacy.

Furthermore, MMA – and all of its variations found in the present report – use the following bounds to limit allowable variable change every iteration:

$$\alpha_i^{(k)} = \min\{\underline{x}_i \ , \ \ 0.9L_i^{(k)} + 0.1x_i^{(k)}\} \ , \qquad \forall i \tag{2.9a}$$

$$\beta_i^{(k)} = \min\{\overline{x}_i \ , \ \ 0.9U_i^{(k)} + 0.1x_i^{(k)}\} \ , \qquad \forall i \tag{2.9b}$$

GMMA

MMA uses the same set of asymptotes for all responses $g_j(\mathbf{X}), \ \ j = 0, \ldots, m$. Although this is quite efficient in terms of computational cost, considering the diversity of responses in Eq. (1.1) for a SO problem, it can compromise the quality of the approximation. GMMA is an attempt to add some flexibility and adjust the approximation of each response according to its own 'structure', see [39]. In practice, this means adding a set of moving asymptotes to each response, i.e. use $L_{ij}^{(k)}, U_{ij}^{(k)}$ instead of $L_i^{(k)}, U_i^{(k)}$. This is accomplished by using information from previous iteration points to determine the value of parameter $s^{(k)}$, by enforcing the current approximation to satisfy:

$$\widetilde{g}_j(\mathbf{X}^{(k-1)}) = g_j(\mathbf{X}^{(k-1)}) \tag{2.10}$$

which requires however an additional iterative numerical scheme in order to solve Eq. (2.10) [13]. The resulting formula is similar to Eq. (2.4) and reads as:

$$\widetilde{g}_j(\mathbf{x}) = r_j^{(k)} + \sum_{i=1}^{n} \frac{p_{ij}^{(k)}}{U_{ij}^{(k)} - x_i} + \sum_{i=1}^{n} \frac{q_{ij}^{(k)}}{x_i - L_{ij}^{(k)}} \tag{2.11}$$

where, $p_{ij}^{(k)}, q_{ij}^{(k)}$ are – similarly to Eq. (2.5) – given by:

$$\begin{aligned} p_{ij}^{(k)} &= \max\left\{0, \ \left(U_{ij}^{(k)} - x_i^{(k)}\right)^2 \frac{\partial g_j}{\partial x_i}\bigg|_{\mathbf{X}^{(k)}}\right\} \\ q_{ij}^{(k)} &= \max\left\{0, \ -\left(x_i^{(k)} - L_{ij}^{(k)}\right)^2 \frac{\partial g_j}{\partial x_i}\bigg|_{\mathbf{X}^{(k)}}\right\} \end{aligned} \tag{2.12}$$

Similarly to Eq. (2.7), the asymptotes are updated by:

$$\begin{aligned} L_{ij}^{(k)} &= x_i^{(k)} - s_j^{(k)}\left(x_i^{(k-1)} - L_{ij}^{(k-1)}\right) \\ U_{ij}^{(k)} &= x_i^{(k)} + s_j^{(k)}\left(U_{ij}^{(k-1)} - x_i^{(k-1)}\right) \end{aligned} \tag{2.13}$$

only this time, a systematic method to calculate an appropriate value for the parameter $s_j^{(k)}$ is required. The most promising ways of doing so – in order to tailor the asymptotes to each response's 'structure' – are by using either the approximated diagonal elements of the Hessian [44], i.e. $\partial^2 g_j / \partial x_i^2$, or by enforcing the approximation to satisfy the previous iteration point, i.e. Eq. (2.10). However, both methods require a one-dimensional line search with an iterative scheme (e.g. Newton-Raphson) to solve the implicit Lagrangian problems, see [39].

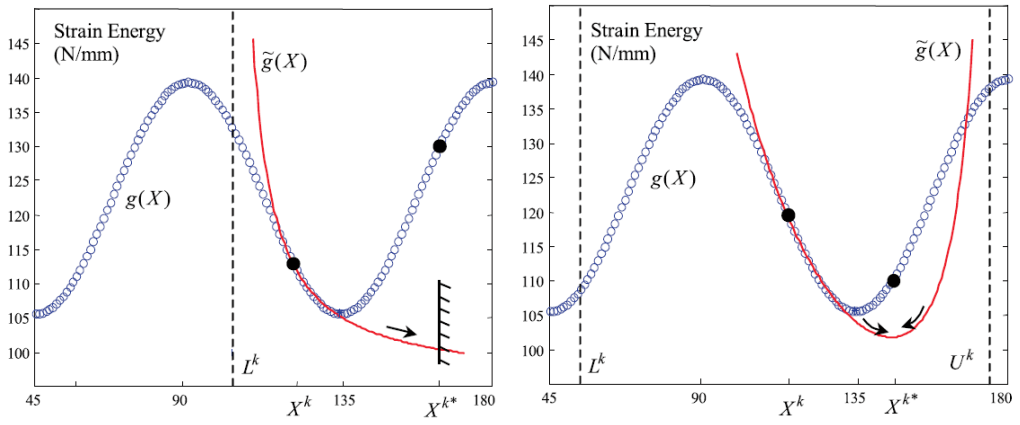
2.3 Mid-range function approximations

In this section, methods that make use of multi-point information are described. Typically, information from previous iteration points that is readily available – if stored – is used. What is more, various methods of the previous section can be transformed to mid-range approximations by incorporating multi-point information in parameter estimation, such as enforcing the condition of Eq. (2.10).

GCMMA

Generally, the actual function we aim to approximate is nonmonotonous. Thus, using monotonous approximations will probably lead to oscillations as we approach a local minimum, even when a move limit strategy is applied. As one

can see in Fig. 2.5a, the current optimization step results in an optimum that has a higher objective function value than its starting point, i.e. $g_j(\mathbf{X}^{(k)}) < g_j(\mathbf{X}^{*(k)})$. An efficient way to restrict the motion of the MMA algorithm is to use both lower and upper asymptotes simultaneously, as in GCMMA [45]. This way, the iteration steps become more robust due to better local fitting of the approximation to the actual function, see Fig. 2.5b. However, researchers have reported [13] that for problems where MMA converges, GCMMA is usually slower because the approximation becomes increasingly conservative, resulting unnecessarily small steps and thus more iterations.



(a) MMA with move limits **(b)** GCMMA

Figure 2.5: Oscillating approximation behaviour of nonmonotonous functions [13]

As suggested by Svanberg (i.e. the founder of MMA) in his improved version with diagonal 2^{nd}-order information [4], the resulting approximation formula is given by:

$$\widetilde{g}_j(\mathbf{X}) = g_j(\mathbf{X}^{(k)}) + \sum_{i=1}^{n} p_{ij}^{(k)} \left(\frac{1}{U_i^{(k)} - x_i} - \frac{1}{U_i^{(k)} - x_i^{(k)}} \right) + \sum_{i=1}^{n} q_{ij}^{(k)} \left(\frac{1}{x_i - L_i^{(k)}} - \frac{1}{x_i^{(k)} - L_i^{(k)}} \right) \tag{2.14}$$

where $p_{ij}^{(k)}, q_{ij}^{(k)}$ can be simultaneously nonzero, i.e. both $L_i^{(k)}, U_i^{(k)}$ are used in Eq. (2.14), and read as [45]:

$$\begin{aligned} p_{ij}^{(k)} &= \frac{\left(U_i^{(k)} - x_i^{(k)}\right)^3}{2\left(U_i^{(k)} - L_i^{(k)}\right)} \cdot \left[2\frac{\partial g_j}{\partial x_i}\bigg|_{\mathbf{X}^{(k)}} + \left(x_i^{(k)} - L_i^{(k)}\right) \frac{\partial^2 g_j}{\partial x_i^2}\bigg|_{\mathbf{X}^{(k)}} \right] \\ q_{ij}^{(k)} &= \frac{\left(x_i^{(k)} - L_i^{(k)}\right)^3}{2\left(U_i^{(k)} - L_i^{(k)}\right)} \cdot \left[-2\frac{\partial g_j}{\partial x_i}\bigg|_{\mathbf{X}^{(k)}} + \left(U_i^{(k)} - x_i^{(k)}\right) \frac{\partial^2 g_j}{\partial x_i^2}\bigg|_{\mathbf{X}^{(k)}} \right]. \end{aligned} \tag{2.15}$$

The most important drawback of the GCMMA method is the lack of an explicit primal-dual relationship [12], i.e. $\mathbf{X} = \mathbf{X}(\boldsymbol{\lambda})$. Consequently, an iterative solution scheme (e.g. Newton-Raphson) is required at each iteration to solve the above system numerically. What is more, in Eq. (2.15), only diagonal terms of the Hessian matrix $\partial^2 g_j / \partial x_i x_j$ are used, which are often approximated by finite differences [4]. This only makes sense close to a local optimum where the step-size is small and the approximations are accurate.

There are multiple variations of the MMA family of approximations of which we only mention some examples hereunder:

- **GCMMA1** — The original version of GCMMA that uses nonmonotonic parameters $\rho_j^{(k)}$ to ensure convexity
- **GCMMA2** — The version described above. It is an improved version of the original that replaces the parameter $\rho_j^{(k)}$ with the use of diagonal 2^{nd}-order information, as in Eq. (2.15)
- **GBMMA1** — A version of GCMMA that matches the response gradients at the current and previous iteration points, i.e. $\partial \widetilde{g}_j(\mathbf{X}^{(k-1)})/\partial x_i = \partial g_j(\mathbf{X}^{(k-1)})/\partial x_i$ and $\partial \widetilde{g}_j(\mathbf{X}^{(k)})/\partial x_i = \partial g_j(\mathbf{X}^{(k)})/\partial x_i$
- **GBMMA2** — Here, a backward finite difference scheme for $\partial^2 \widetilde{g}_j(\mathbf{X}^{(k)})/\partial x_i^2$ in Eq. (2.15) is used

- **GBMMA3** — This is a combination of Eq. (2.14) and Eq. (2.13)
- **GBMMA4** — Eq. (2.14) and Eq. (2.10) are combined
- **TGMMA** — This is a modified mid-range version of MMA with previous point information, tailor made for TO problems of compliant mechanisms [37]

For more information on the above methods and a detailed review of the MMA family up to 2002, the interested reader is referred to [4].

Exponential Approximation

Another way to adjust the curvature of the approximation to the needs of each response's local structure, is the exponential approximation [46]. This monotonous method can be seen as a generalization of some of the aforementioned methods, as it is essentially a 1^{st}-order Taylor expansion with respect to the intermediate variable $y_i = x_i^{\alpha_i^{(k)}}$. Here, $\alpha_i^{(k)}$ is the exponent parameter for each design variable x_i at the current iteration (k). One can easily see that for $\alpha_i^{(k)} = 1$ we obtain the linear approximation of Section 2.2 and for $\alpha_i^{(k)} = -1$, we get the reciprocal approximation of Section 2.2. The resulting approximation is given by:

$$\widetilde{g}_j(\mathbf{X}) = g_j(\mathbf{X}^{(k)}) + \sum_{i=1}^{n} \left[\frac{\partial g_j}{\partial x_i}\bigg|_{\mathbf{X}^{(k)}} \left(\frac{x_i^{(k)}}{\alpha_i^{(k)}} \right) \left(\left(\frac{x_i}{x_i^{(k)}} \right)^{\alpha_i^{(k)}} - 1 \right) \right] \tag{2.16}$$

Convexity of Eq. (2.16) is guaranteed by choosing the appropriate exponents as [12]:

- If $\frac{\partial g_j}{\partial x_i}\big|_{\mathbf{X}^{(k)}} < 0$, then $\alpha_i^{(k)} < 1$
- If $\frac{\partial g_j}{\partial x_i}\big|_{\mathbf{X}^{(k)}} > 0$, then $\alpha_i^{(k)} > 1$

In practice, exponential approximation is applied by making use of the previous iteration point, as given by [46]:

$$\frac{\partial \widetilde{g}_j}{\partial x_i}\bigg|_{\mathbf{X}^{(k-1)}} = \frac{\partial g_j}{\partial x_i}\bigg|_{\mathbf{X}^{(k-1)}} \implies \alpha_i^{(k)} = 1 + \ln\left[\frac{\frac{\partial g_j}{\partial x_i}\big|_{\mathbf{X}^{(k-1)}}}{\frac{\partial g_j}{\partial x_i}\big|_{\mathbf{X}^{(k)}}} \right] \Bigg/ \ln\left[\frac{x_i^{(k-1)}}{x_i^{(k)}} \right], \qquad i = 1, ..., n \tag{2.17}$$

Numerical complications in Eq. (2.17), e.g. negative values within logarithms, are dealt with manual assignment of values for $\alpha_i^{(k)}$ [32]:

- If $\frac{\partial g_j}{\partial x_i}\big|_{\mathbf{X}^{(k)}} < 0$, then $\alpha_i^{(k)} = \min\left\{ \alpha_i^{(k)},\ 1 \right\}$
- If $\frac{\partial g_j}{\partial x_i}\big|_{\mathbf{X}^{(k)}} > 0$, then $\alpha_i^{(k)} = \max\left\{ \alpha_i^{(k)},\ 1 \right\}$
- If $\alpha_i^{(k)} \in \text{Im}$, then $\alpha_i^{(k)} = -1$

Although the exponential approximation is usually superior to the reciprocal – because parameter $\alpha_i^{(k)}$ can be used to tune the conservatism of the approximation $\widetilde{g}_j$ – when different exponents $\alpha_i^{(k)}$ are used for different responses g_j in a dual framework, it is often impossible to find a simple primal-dual relationship that can be solved analytically [32]. Consequently, one must either use a single value for $\alpha_i^{(k)}$ for all responses or allocate some extra computational resources to obtain these relationships in order to exploit the capabilities of this method to the fullest.

TANA family

The Two-point Adaptive Nonlinear Approximation is a modification of the exponential (i.e. $y_i = x_i^{\alpha_i^{(k)}}$), where a correction term is added to Eq. (2.16) in order to compensate for the approximation error. As with the exponential approximation, in this method each design variable x_i has a corresponding exponent $\alpha_i^{(k)}$ at each iteration (k). Therefore, the $(n+1)$ parameters $\alpha_i^{(k)}$, $\forall i$ and $\varepsilon(\mathbf{X})$ are found by enforcing $(n+1)$ coupled equations: $\widetilde{g}_j(\mathbf{X}^{(k-1)}) = g_j(\mathbf{X}^{(k-1)})$ and $\boldsymbol{\nabla}\widetilde{g}_j(\mathbf{X}^{(k-1)}) = \boldsymbol{\nabla} g_j(\mathbf{X}^{(k-1)})$. Furthermore, the 1^{st}-order Taylor expansion of Eq. (2.16) is enriched

with 2^{nd}-order diagonal information, as seen in Eq. (2.18), which describes the most recent TANA version, i.e. TANA-3 [31]:

$$\widetilde{g}_j(\mathbf{X}) = g_j(\mathbf{X}^{(k)}) + \sum_{i=1}^{n} \left[\left. \frac{\partial g_j}{\partial x_i} \right|_{\mathbf{X}^{(k)}} \frac{\left(x_i^{(k)}\right)^{1-\alpha_i^{(k)}}}{\alpha_i^{(k)}} \left((x_i)^{\alpha_i^{(k)}} - \left(x_i^{(k)}\right)^{\alpha_i^{(k)}} \right) \right] + \frac{1}{2}\varepsilon(\mathbf{X}) \sum_{i=1}^{n} \left[(x_i)^{\alpha_i^{(k)}} - \left(x_i^{(k)}\right)^{\alpha_i^{(k)}} \right]^2 \tag{2.18}$$

where,

$$\varepsilon(\mathbf{X}) = \frac{2\left\{ g_j(\mathbf{X}^{(k-1)}) - g_j(\mathbf{X}^{(k)}) - \sum_{i=1}^{n} \left[\left. \frac{\partial g_j}{\partial x_i} \right|_{\mathbf{X}^{(k)}} \left(\left(\frac{x_i^{(k-1)}}{x_i^{(k)}} \right)^{\alpha_i^{(k)}} - 1 \right) \left(\frac{x_i^{(k)}}{\alpha_i^{(k)}} \right) \right] \right\}}{\sum_{i=1}^{n} \left[(x_i)^{\alpha_i^{(k)}} - \left(x_i^{(k-1)}\right)^{\alpha_i^{(k)}} \right]^2 + \sum_{i=1}^{n} \left[(x_i)^{\alpha_i^{(k)}} - \left(x_i^{(k-2)}\right)^{\alpha_i^{(k)}} \right]^2} \tag{2.19}$$

Although this method has proved to be efficient in many cases, it cannot yield an explicit primal-dual relationship of the form $\mathbf{X} = \mathbf{X}(\boldsymbol{\lambda})$. Similarly to GMMA, an iterative solution scheme (e.g. Newton-Raphson) is required at each iteration to solve the above system numerically, which might lead to prohibitive computational cost. There are various versions of this method (e.g. TPEA, TANA-1, TANA-2, TANA-3, TDQA). The main difference lies in the calculation of their parameters (e.g. $\alpha_i^{(k)}$, $\varepsilon(\mathbf{X})$). For a thorough review and a detailed comparison of these methods, the reader is referred to [31], [47] and [48].

DQA

The Diagonal Quadratic Approximation applies a 2^{nd}-order Taylor expansion to a response function, while only considering the diagonal terms of its Hessian matrix, i.e. $\partial^2 \widetilde{g}_j / \partial x_i^2$. These terms can be easily approximated by efficient Quasi-Newton methods, e.g. BFGS. Within the SO literature, this method is typically applied to a function approximation (e.g. MMA, reciprocal, exponential, etc.) instead of the exact function itself. The reasons for that counter-intuitive choice will become apparent later on. The resulting formula is given below [12], where the symbol $\widetilde{\widetilde{g}}_j$ is to emphasise that this is a (quadratic) approximation of an approximation function:

$$\widetilde{\widetilde{g}}_j(\mathbf{X}) = \widetilde{g}_j(\mathbf{X}^{(k)}) + \sum_{i=1}^{n} \left[\left. \frac{\partial \widetilde{g}_j}{\partial x_i} \right|_{\mathbf{X}^{(k)}} \left(x_i - x_i^{(k)} \right) \right] + \frac{1}{2} \sum_{i=1}^{n} \left[c_{ij}^{(k)} \left(x_i - x_i^{(k)} \right)^2 \right] \tag{2.20}$$

Depending on the selection of intermediate variables $y_i = y_i(x_i)$, the curvatures $c_{ij}^{(k)}$ of the approximation $\widetilde{\widetilde{g}}_j(\mathbf{X})$ in Eq. (2.20) are calculated as follows:

- **For all design variables**: $\forall y_i$

$$\left. \frac{\partial \widetilde{g}_j}{\partial x_i} \right|_{\mathbf{X}^{(k)}} = \left. \frac{\partial g_j}{\partial x_i} \right|_{\mathbf{X}^{(k)}} \tag{2.21}$$

- **Direct variables**: $y_i = x_i$
  This is effectively a diagonal quadratic approximation of the actual response $g_j(\mathbf{X})$. No approximated approximation is used since there are no intermediate variables. The curvatures are approximated by:

$$c_{ij}^{(k)} = \left. \frac{\partial^2 \widetilde{g}_j}{\partial x_i^2} \right|_{\mathbf{X}^{(k)}} = \left. \frac{\partial^2 g_j}{\partial x_i^2} \right|_{\mathbf{X}^{(k)}} \tag{2.22}$$

- **Reciprocal intermediate variables**: $y_i = 1/x_i$
  In this case, the response $g_j(\mathbf{X})$ is approximated by $\widetilde{g}_j(\mathbf{X})$ given in Eq. (2.2). Subsequently, the function $\widetilde{g}_j(\mathbf{X})$ is again approximated by its 2^{nd}-order diagonal Taylor expansion of Eq. (2.20). The resulting approximation is denoted by (T2:R) and its curvatures are estimated by:

$$c_{ij}^{(k)} = \left. \frac{\partial^2 \widetilde{g}_j}{\partial x_i^2} \right|_{\mathbf{X}^{(k)}} = -\frac{2}{x_i^{(k)}} \left. \frac{\partial g_j}{\partial x_i} \right|_{\mathbf{X}^{(k)}} \tag{2.23}$$

- **Exponential intermediate variables**: $y_i = x_i^{\alpha_i^{(k)}}$
  Here the response $g_j(\mathbf{X})$ is approximated by $\widetilde{g}_j(\mathbf{X})$ given in Eq. (2.16). Then, the function $\widetilde{g}_j(\mathbf{X})$ is again approximated by its 2^{nd}-order diagonal Taylor expansion of Eq. (2.20). The resulting approximation is denoted by (T2:E) and its curvatures are estimated by

$$c_i{}_j^{(k)} = \frac{\partial^2 \widetilde{g}_j}{\partial x_i^2}\bigg|_{\mathbf{X}^{(k)}} = \frac{\alpha_i^{(k)} - 1}{x_i^{(k)}} \frac{\partial g_j}{\partial x_i}\bigg|_{\mathbf{X}^{(k)}} \tag{2.24}$$

Additional examples of various intermediate variables can be found in [42], [32], [47]. Another popular way to handle the curvatures $c_i{}_j^{(k)}$ is to make the so-called **spherical approximation**, developed by [49] and made popular by [50]. This method can be used in conjunction with the above intermediate variables and selects $c_i{}_j^{(k)} = c_j{}^{(k)}\ \forall i$, which effectively means only a single parameter needs to be determined for each approximated approximation $\widetilde{\widetilde{g}}_j$. For example, enforcing condition Eq. (2.25) is probably the simplest form of a two-point approximation one can encounter, see [28].

$$\widetilde{g}_j(\mathbf{X}^{(k-1)}) = g_j(\mathbf{X}^{(k-1)}) \implies c_j{}^{(k)} = \frac{2\left[g_j(\mathbf{X}^{(k-1)}) - g_j(\mathbf{X}^{(k)}) - \boldsymbol{\nabla}^T g_j(\mathbf{X}^{(k)})\left(\mathbf{X}^{(k-1)} - \mathbf{X}^{(k)}\right)\right]}{\left\|\mathbf{X}^{(k-1)} - \mathbf{X}^{(k)}\right\|^2} \tag{2.25}$$

Other ideas for curvature manipulation exist as well. The most recent one [12], checks certain conditions – based on information stored from previous iteration points – and adjusts the curvature accordingly. That way a type of mixed approximation scheme is formulated that can exploit the benefits of various approximations (e.g. linear, reciprocal, exponential, etc.) and avoid the complications of using a different approximation for each response (e.g. non-analytical primal-dual relationships). While the idea of approximated approximations may initially seem counter-intuitive, researchers [32] have shown it can become very powerful in combination with purely dual methods for an SAO framework. In addition to the approximations considered above, many others may be subjected to similar treatment. The main benefits of a diagonal quadratic approximation of an approximation are listed below [32]:

- Different approximation methods can be used for different responses g_j, thus allowing each response to be approximated by the most suitable method, while having only one general dual statement. Otherwise, the introduction of new intermediate variables for a response requires the formulation of a new dual.
- The sub-problems generated yield simple analytical primal-dual relationships, i.e. $\mathbf{X} = \mathbf{X}(\boldsymbol{\lambda})$, which means there is no need for a time-consuming iterative solution scheme for them, as is often the case for enriched approximation methods (e.g. GMMA, GCMMA, TANA) for cases where the computational cost of the optimizer and the total cost are of the same magnitude.
- Conservatism can be enforced by increasing the diagonal curvatures, i.e. $\partial^2 \widetilde{g}_j / \partial x_i^2$, which is beneficial as far as global convergence is concerned [2]. This can be done by adding an inner loop (l) – within each iteration (k) – which will increase the curvatures $c_i{}_j^{(k)}$ of non-conservative approximations until conservatism is achieved, i.e. $\widetilde{g}_j(\mathbf{X}^{*(k)}) \geq g_j(\mathbf{X}^{*(k)})$.
- Although quadratic approximations of the approximation functions may differ significantly from their original counterparts (see Fig. 2.6), they are similar enough within the trust region they are supposed to operate in, considering that the original approximations (e.g. reciprocal, exponential, MMA, etc.) have their own limited region of validity.

What is more: researchers [28] have shown that the DQA method can also be used should one wish to apply an efficient SQP (primal) method in case dual solvers are computationally inadvantageous, as is the case in a Simultaneous Analysis and Design (SAND) setting, wherein $m \approx n$ or $m > n$.

The foregoing methods are by no means exhaustive and the interested reader is referred to [11] for a thorough survey up to 1993, to [42] for an overview of the Incomplete Series Expansion methods, as well as [4] for a detailed review on the MMA family of approximations. Although the variations of approximation methods are numerous, the ones described in Section 2.2 and Section 2.3 seem to prevail over the field of SAO.

[2] Global convergence in the sense of converging to a local minimum from any arbitrary starting point

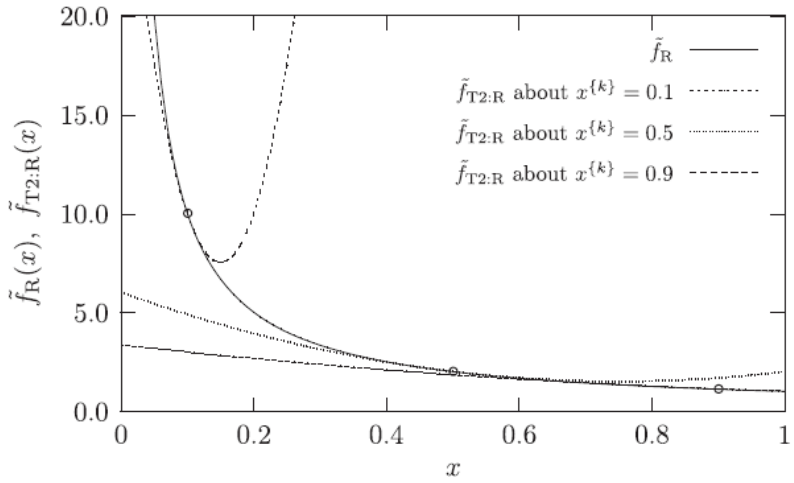


Figure 2.6: Diagonal 2^{nd}-order Taylor approximation $\widetilde{f}_{T2:R}(\mathbf{X})$ of the reciprocal approximation $\widetilde{f}_{R}(\mathbf{X})$ [14]

2.4 Mixed schemes

As mentioned in Section 1.4, researchers have revealed the benefits of mixed approximation schemes in an SAO framework for problems that include a diverse set of design variables and response functions. However, choosing the most appropriate function at any arbitrary point $\mathbf{X}^{(k)}$ becomes crucial. In this step, every scenario the optimizer might encounter must be predicted and accounted for, in order for a mixed scheme to have the desired behaviour. This section briefly discusses some of the most popular mixed schemes found in literature in order to better understand the functionality of a robust and adaptive scheme we aim to generate. At this point, an subtle – but important – distinction must be made between the following terms:

- **Approximation functions/methods**
  These are essentially the local Taylor-like expansions $\widetilde{g}_j(\mathbf{X})$ described in Section 2.2 that approximate $g_j(\mathbf{X})$ at an arbitrary point $\mathbf{X}^{(k)}$, e.g. Linear, MMA, GCMMA, GMMA, etc.
- **Approximation scheme**
  An approximation scheme comprises of one or more approximation functions/methods. Depending on how these functions are combined, a novel categorization method is introduced in Section 4.1.

2.4.1 GBMMA family mixed scheme

Based on the popular MMA approximation function, researchers introduced an enriched version of it with previous point information and created a family of approximations, see Section 2.3. As one can see from Fig. 2.7, initially the non-monotonous GCMMA scheme is used to avoid approximating a non-monotonous function with a monotonous one. Then, if the design variable change is small (e.g. in the vicinity of the optimum), approximate second order information is used to accelerate convergence by opting for GBMMA2. Otherwise, the previous point gradients are matched by using GBMMA1. If the resulting approximation is non-convex, the scheme preserves convexity by falling back to the more conservative GCMMA approximation. That way, accelerated convergence is achieved by using stored information without compromising robustness.

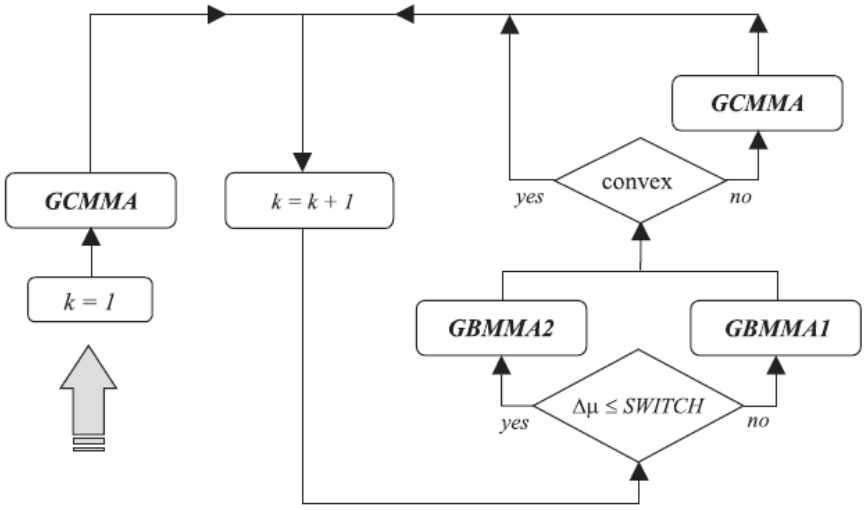


Figure 2.7: GBMMA family mixed scheme [13]

2.4.2 Adaptive quadratic approximation scheme

More recently, researchers [12] used a diagonal quadratic approximation algorithm (DQA) and analyzed all the possible cases for an arbitrary set of $\{f_j, x_i, \partial f_j/\partial x_i\}$ used in Eq. (2.26), see Fig. 2.8. Then, using the values of certain ratios defined by Eq. (2.27), they opt for an appropriate value for the quadratic term of the approximation (i.e. $c_{ij}^{(k)}$).

$$\widetilde{f}_j(\mathbf{X}) = f_j(\mathbf{X}^{(k)}) + \sum_{i=1}^{n} \left[\left. \frac{\partial f_j}{\partial x_i} \right|_{\mathbf{X}^{(k)}} \left(x_i - x_i^{(k)} \right) \right] + \frac{1}{2} \sum_{i=1}^{n} \left[c_{ij}^{(k)} \left(x_i - x_i^{(k)} \right)^2 \right] \tag{2.26}$$

$$\begin{aligned} N_i &\triangleq \frac{\frac{\partial f_j}{\partial x_i}(\mathbf{X}^{(k-1)})}{\frac{\partial f_j}{\partial x_i}(\mathbf{X}^{(k)})} \quad , \qquad i = 1, ..., n \\ D_i &\triangleq \frac{x_i^{(k-1)}}{x_i^{(k)}} \qquad\qquad , \qquad i = 1, ..., n \end{aligned} \tag{2.27}$$

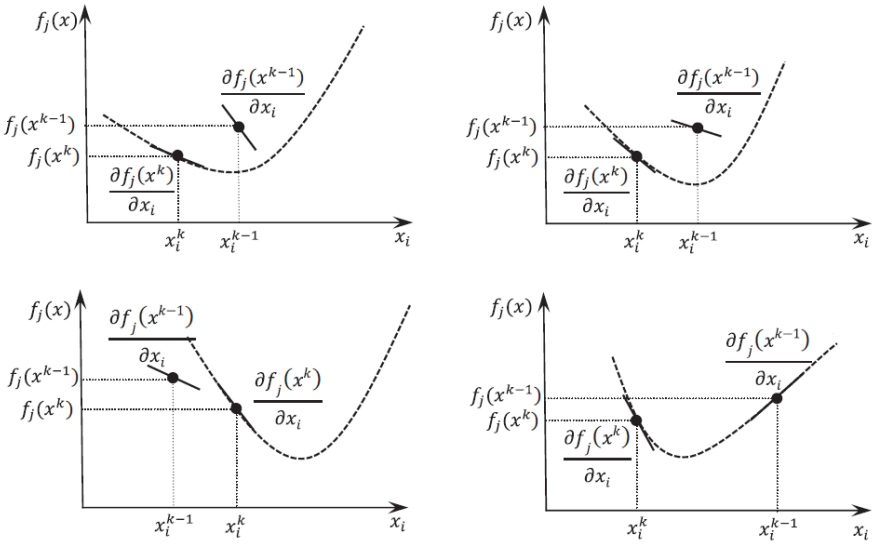


Figure 2.8: Examples of possible scenarios for the set of $\{f_j, x_i, \frac{\partial f_j}{\partial x_i}\}$ [12]

Further details on this method, as well as some numerical examples that show the benefits of using such an adaptive scheme for truss optimization problems can be found in the aforementioned publication.

2.4.3 Numerical implementation considerations

Along the same lines, other mixed schemes can be found in literature, the description of which is out of the scope of the present report. As an example, the interested reader is referred to [40] for another mixed scheme of the MMA family and to [5] for a combination of GMMA and DQA.

It should be clear by now that – from an implementation point of view – approximation functions should be compatible with one another for the resulting mixed scheme to be both versatile and computationally efficient. To the author's knowledge, most strategies found in literature attempt to use a versatile approximation method as a base (typically the popular MMA method) and fine-tune its convexity based on the available information (e.g. previous points). This is attributed to the fact that fine-tuning the parameters of an approximating function (e.g. parameter $c_{ij}^{(k)}$ of Eq. (2.26)) must be done automatically and efficiently to better capture the true behaviour of the functions involved in the optimization problem at hand. Thus, one must conclude that the versatility required from the adaptive scheme we aim to generate will be based on an existing approximation family with the ability of adjustment to a variety of different conditions.

Taking the above into consideration, in order to generate a robust and adaptive mixed approximation scheme, one must create a modular toolbox that facilitates the generation, analysis and evaluation of all candidate schemes. To this end, an object-oriented modular software library is implemented, see Appendix A, that is herein referred to as 'Optimization Lab'. This environment is effectively a framework, within which different approximation schemes can be tested and compared. Among other modules, it includes several problems, approximations and solvers, whose influence can be investigated by using certain performance metrics.

3 | Optimizer evaluation

3.1 Preliminary definitions

This chapter discusses the evaluation of different approximation schemes. It is important to define a set of performance measures for two reasons: to compare different schemes fairly and to obtain a clear image of each approximation's fit on the respective exact function at every iteration. First, we define a template optimization problem, which will be used as an example by applying on it all the performance measures used herein. This problem was chosen considering the following argumentation:

- Being a 2-bar truss problem, it shares similar response functions to some of the SO problems we target our method to (i.e. weight minimization subject to stress constraints)
- It is widely used by the SO community to evaluate the performance of approximation functions
- It is 2D, which facilitates graphical representation

It goes without saying that by no means do we claim that this problem is able to represent the entirety of SO problems. Nonetheless, it is a simple – and relative to SO – example that can facilitate the comprehension of the forthcoming performance measures. The interested reader is referred to [12] for further details on the aforementioned problem.

$$\mathcal{P}_{\text{NLP}} = \begin{cases} \underset{\mathbf{X}}{\text{minimize}} & g_0(\mathbf{X}) = c_1 x_1 \sqrt{1+x_2^2} \\ \text{s.t.} & g_1(\mathbf{X}) = c_2 \sqrt{1+x_2^2}\left(\dfrac{8}{x_1} + \dfrac{1}{x_1 x_2}\right) - 1 \leq 0 \\ & g_2(\mathbf{X}) = c_2 \sqrt{1+x_2^2}\left(\dfrac{8}{x_1} - \dfrac{1}{x_1 x_2}\right) - 1 \leq 0 \\ & 0.2 \leq x_1 \leq 4 \\ & 0.1 \leq x_2 \leq 1.6 \end{cases} \tag{3.1}$$

A summarizing list of the performance measures used in the Optimization Lab of Appendix A can be found hereunder. They are classified with respect to their field of application into two main categories: the global convergence plots and the local approximation quality plots. On the one hand, the former indicate how well an optimizer (and therefore its approximation scheme) performs from a macroscopic point of view (e.g. redundant iterations, oscillatory behaviour, function value comparison). On the other hand, the latter show a more detailed picture of the approximation quality at an arbitrary iteration point $\mathbf{X}^{(k)}$ of the optimization cycle (e.g. how $\widetilde{\mathcal{P}}_{\text{NLP}}$ changes between iterations, the error propagation in the direction indicated by the solver, and the contribution of each design variable x_i to each function g_j). More information on each performance measure can be found in the coming sections, wherein the template problem of Eq. (3.1) will be used.

- Post-optimization performance measures
  - Global convergence plots
    - Function value plots: $g_j(\mathbf{X}^{(k)})$ – iterations
    - Design variable plots: $x_i^{(k)}$ – iterations
    - KKT norm plot: $||KKT||_2$ – iterations
    - Norm of variable change plot: $||\Delta\mathbf{X}^{(k)}||_2$ – iterations
    - Approximation quality index: Φ_j – iterations
    - Exact problem contour plots (only for 2D problems): $\mathcal{P}_{\text{NLP}}$ – iterations
  - Local approximation quality plots (at any iteration)
    - Approximate problem contour plots (only for 2D problems): $\widetilde{\mathcal{P}}_{\text{NLP}}$ – iteration

    - Pair analysis plots: $g_j - x_i$
    - Approximation error plots: $e_j - \alpha$
- Real-time behaviour indicators
  - Any index derived from $\{x_i,\ g_j,\ \partial g_j/\partial x_i\}^{(l)}\ ,\quad l = k-2,\ k-1,\ k$
- Additional indicators
  - Direction index
  - Expectation index
  - Participation index
  - Switching frequency array
- Performance profiles [51]

The strategy here is to run an arbitrary structural, nonlinear, bounded and continuously differentiable optimization problem, e.g. Eq. (3.1), with multiple approximation schemes and analyze their performance using the above post-optimization performance measures. Cases of undesired behaviour (e.g. oscillations, small steps, etc.) are located by the global convergence plots, and subsequently further investigation on those cases is conducted using the local approximation quality plots (e.g. pair analysis). This analysis is then used to construct real-time indicators from the available information, i.e. $\{x_i,\ g_j,\ \partial g_j/\partial x_i\}$, aiming at eliminating the undesired behaviour detected by the post-optimization performance measures. This strategy is only applicable to optimization problems whose 'characteristic size' $n \cdot (m+1)$ is relatively small. When problems of high dimensionality are to be included, aggregating functions must be applied to (some of) the aforementioned measures. Finally, after several problems are solved, an elaborate and objective approximation scheme comparison is conducted by generating the performance profiles thereof.

3.2 Post-optimization performance measures

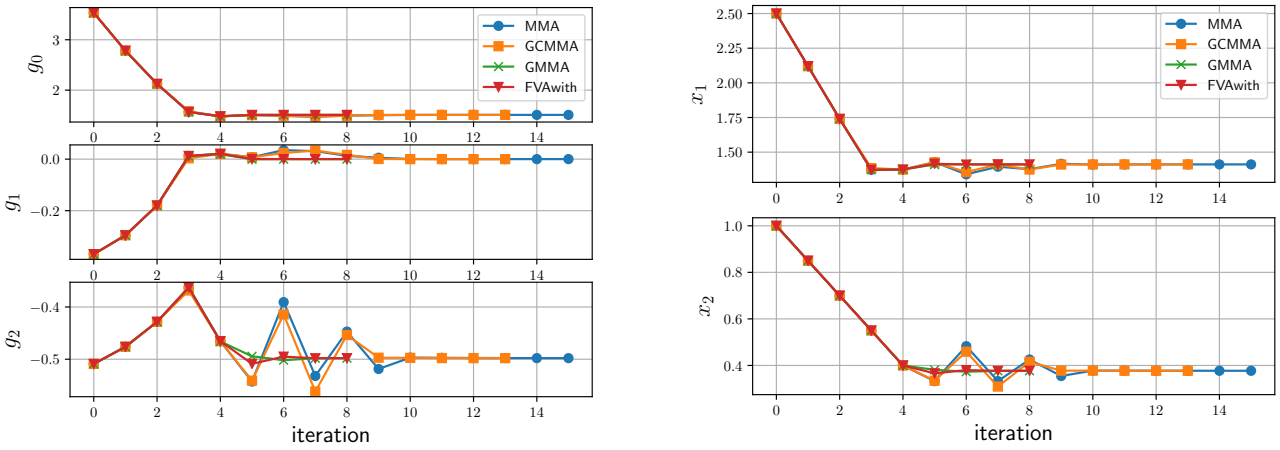
This section covers all the performance measures involved in post-processing. After an optimization algorithm has converged to a stationary point, several plots are generated in order to gain further insight on its performance. Therefore, the measures described herein cannot be used to assess the performance of a scheme in 'real-time' and make decisions (e.g. choose which approximation method to use on a function $g_j(\mathbf{X})$) while the optimization is still in progress.

3.2.1 Global convergence plots

As mentioned in Section 3.1, this is the first level of analysis for the approximation schemes under investigation. Using the template problem of Eq. (3.1), we obtain the following plots, wherein 3 of the most prominent optimizers of the SO field are compared with one of the generated mixed schemes called 'FVAwith', see Section 4.4.

Response and design variable history

By using these performance measures in combination, the information one can infer is maximized because they complement each other. The performance of every design can be seen from the function values of Fig. 3.1a, whereas areas where a $g_j(\mathbf{X})$ plateaus can only be investigated through the design variables of Fig. 3.1b, since the function value does not change. Furthermore, oscillatory behaviour – which leads to redundant iterations – is easily detected when variable values $x_i^{(k)}$ and function values $g_j(\mathbf{X}^{(k)})$ are inspected simultaneously. More information can be derived from plotting several other performance measures with respect to iterations (e.g. $\partial g_j/\partial x_i(\mathbf{X}^{(k)})$, $||\nabla \mathbf{g}||$) but the numerical experiments conducted thus far have shown that they do not add much value to the search for a robust and adaptive mixed scheme.



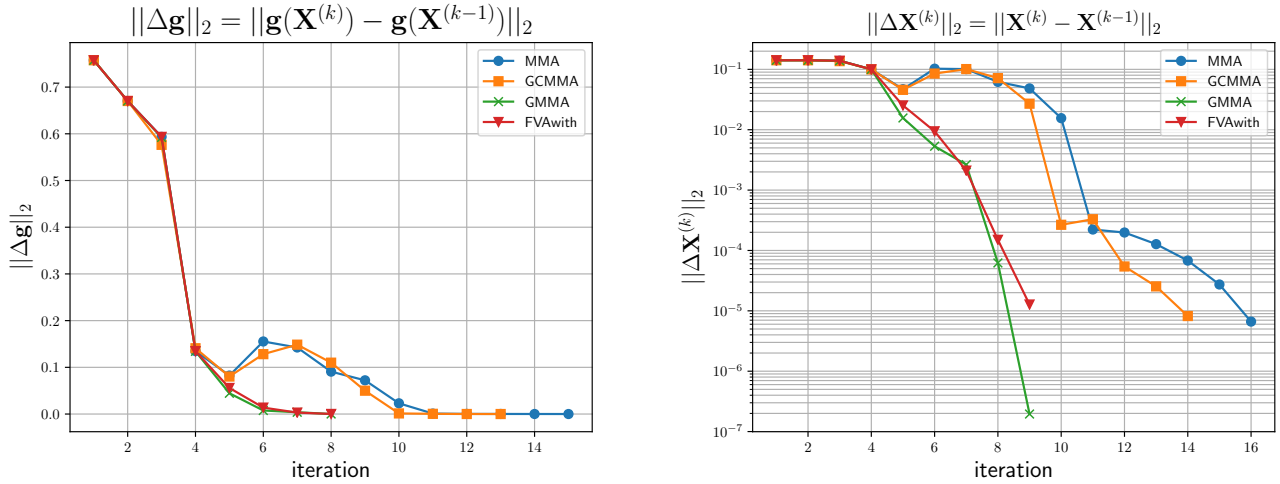
(a) Convergence plot of $g_j(\mathbf{X}^{(k)})$

(b) Convergence plot of $x_i^{(k)}$

Figure 3.1: Function values (a) and design variables (b) for Eq. (3.1)

Aggregated function and variable plots

For problems with a large number of design variables n (and/or response functions m), using Fig. 3.1b (or Fig. 3.1a) becomes impractical. Thus, aggregated versions of those figures are only applicable, as in Fig. 3.2. However, it should be noted that useful information may be lost when considering the Eucledian norm of a vector (i.e. $||\Delta\mathbf{X}^{(k)}||_2$ or $||\Delta\mathbf{g}(\mathbf{X}^{(\mathbf{k})})||_2$ respectively) instead of its constituent elements.



(a) Response vector change

(b) Design variable vector change

Figure 3.2: Euclidean norm of response vector change (a) and design variable change (b) for Eq. (3.1)

KKT norm

The KKT conditions, see Eq. (1.4), are a widely applied convergence criterion in the field of optimization. They are used to confirm that the converged point $\mathbf{X}^{*(k)}$ is indeed a local optimum. Although the norm of the KKT residual is a strong indication for a local optimum, mathematically Eq. (1.4) are necessary optimality conditions and not sufficient. Therefore, one should keep in mind that there is still a possibility these 1^{st}-order conditions can be misleading (e.g. inflection points). Nevertheless, Fig. 3.3 is one of the most widely applied convergence plots that can be found in the optimization literature.

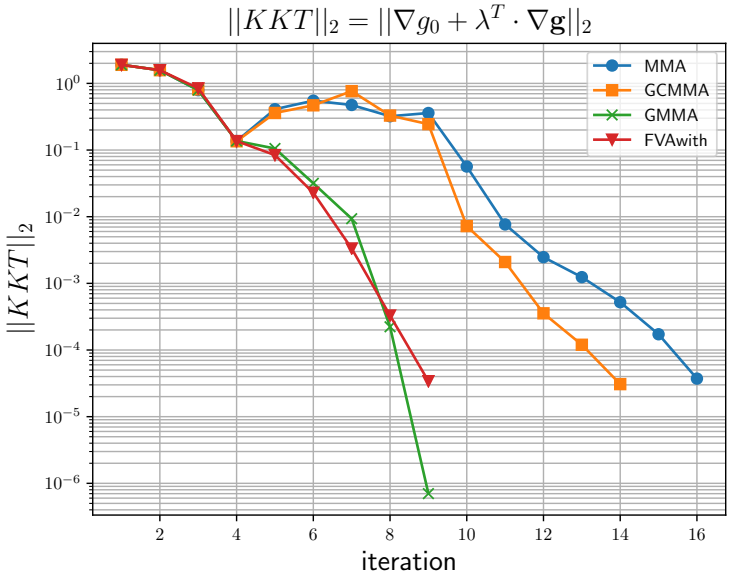


Figure 3.3: KKT modulus of Eq. (1.4a)

Approximation quality index

This performance measure aims to compare different approximation functions $\widetilde{g}_j(\mathbf{X})$ at an arbitrary point $\mathbf{X}^{(k)}$ consistently and equitably. To achieve this, the same sequence of points $\mathbf{X}^{(k)}$ must be used for the approximations under comparison. Since the reference approximation method used for TO problems is MMA, its solution path (i.e. the sequence of $\mathbf{X}^{(k)}$ produced by MMA in an optimization run, see Fig. 3.5a) was used for all approximations. At every point, each approximation scheme is used to calculate its constituent functions $\widetilde{g}_j(\mathbf{X})$, resulting to the formulation of its respective local approximate sub-problem $\widetilde{\mathcal{P}}_{\text{NLP}}$ shown in Eq. (1.2). Then, the implemented primal-dual solver (see [35]) finds the Newton step direction and the following function is evaluated to assess the error propagation of each function $\widetilde{g}_j(\mathbf{X})$ in that direction:

$$e_j(\alpha) = f[g_j(\alpha), \widetilde{g}_j(\alpha)] \ , \qquad \forall j \tag{3.2}$$

where,

α	Line-search step-size in the Newton direction ($\mathbf{X} = \mathbf{X}^{(k)} + \alpha \mathbf{D}^{(k)}$)
$\mathbf{D}^{(k)}$	Newton direction indicated by the solver [35]
UB_j	Upper bound for the step-size used when investigating the error propagation of $\widetilde{g}_j(\mathbf{X})$, i.e. $0 \leq \alpha \leq UB_j$, defined by the most conservative terms (i.e. closer to the expansion point $\mathbf{X}^{(k)}$) between the move limits, the asymptotes and the domain bounds, see Eq. (3.4)
$f[g_j(\alpha), \widetilde{g}_j(\alpha)]$	Error function for $0 \leq \alpha \leq UB_j$

As an example, we can define the following error function from Eq. (3.2):

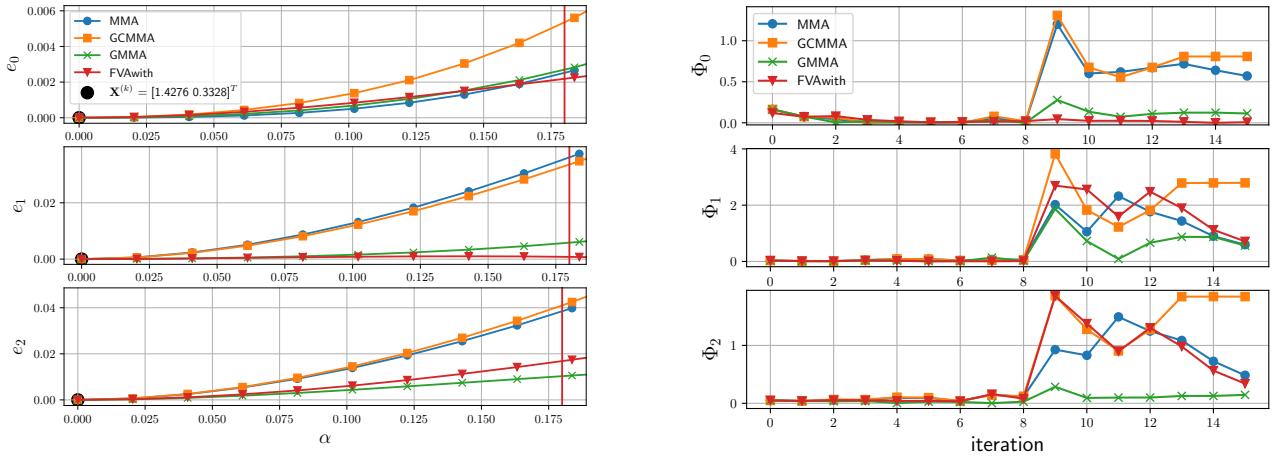
$$e_j(\alpha) = |g_j(\mathbf{X}(\alpha)) - \widetilde{g}_j(\mathbf{X}(\alpha))| \ , \qquad \forall j \tag{3.3}$$

This is the function used Fig. 3.4a, where $m+1$ plots are generated at an arbitrary point $\mathbf{X}^{(k)}$. The last step is to integrate Eq. (3.2) in order to get a scalar valued index for each approximation function at each point, i.e. Φ_j:

$$\Phi_j = \int_0^{UB_j} e_j(\alpha)\, d\alpha \ , \qquad \forall j$$

$$UB_j \triangleq \arg\max_{\mathbf{X}} \alpha(\mathbf{X}) = \begin{cases} \min\limits_{approx} \left\{ \min\limits_i \{x_i + \delta_i,\ U_i,\ \overline{x}_i\} \right\}, & if\ \Delta x_i^{(k)} > 0 \\ \max\limits_{approx} \left\{ \max\limits_i \{x_i - \delta_i,\ L_i,\ \underline{x}_i\} \right\}, & if\ \Delta x_i^{(k)} < 0 \end{cases} , \qquad \forall j \tag{3.4}$$

Clearly, the best approximation for a function $g_j(\mathbf{X})$ at a point $\mathbf{X}^{(k)}$, is the one that has the lowest value of Φ_j at iteration (k).



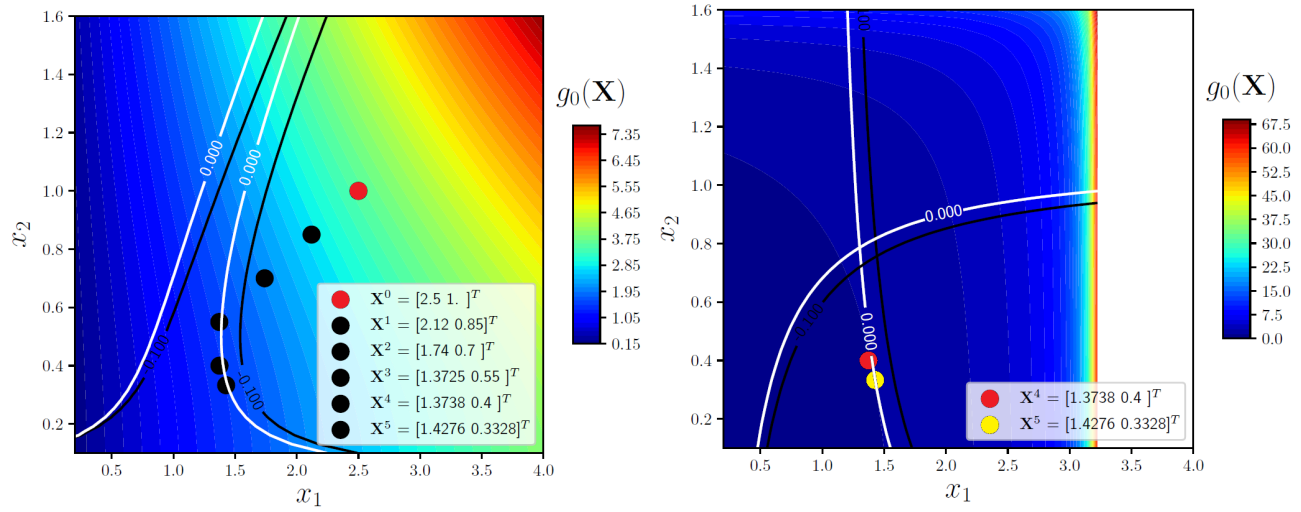
(a) Error function e_j for the 5^{th} iteration given by Eq. (3.3)

(b) Index Φ_j given by Eq. (3.4)

Figure 3.4: Evaluation of approximation quality index Φ_j, $\forall \mathbf{X}^{(k)}$ found by MMA in Eq. (3.1)

Contour plots (only for 2D)

For the special case of 2D problems, we can also generate contour plots for the functions involved. The benefit of using such plots is that by looking at the solution path of an approximation scheme, see Fig. 3.5a, one can get a clear overview of the optimization. For example, using such plots, macroscopic phenomena like clustering in the vicinity of a local optima – even before the step-size $\Delta\mathbf{X}^{(k)}$ reduces – can be detected. This information can be exploited by adding criteria for the density of visited points within a hyper-sphere of a certain radius around the current point $\mathbf{X}^{(k)}$. That way, proximity to optimum can be identified as early as possible and – if used properly (e.g. choosing more conservative approximations, reducing step-size, etc.) – this can lead to a further reduction of the number of iterations required to converge. In order to analyze the behaviour of the optimizer shown in Fig. 3.5a, an additional contour plot of the current approximate sub-problem $\widetilde{\mathcal{P}}_{\text{NLP}}$ can be generated every iteration, wherein the initial point $\mathbf{X}^{(k)}$ and the local optimum $\mathbf{X}^{*(k)}$ are displayed, see Fig. 3.5b.



(a) $\mathcal{P}_{\text{NLP}}$: First 5 points of MMA's solution path

(b) $\widetilde{\mathcal{P}}_{\text{NLP}}$ for the 4^{th} iteration of MMA

Figure 3.5: Exact problem $\mathcal{P}_{\text{NLP}}$ (a) and approximate sub-problem $\widetilde{\mathcal{P}}_{\text{NLP}}$ (b)

3.2.2 Local approximation quality plots

After recognizing an occurrence of undesirable behaviour using the global convergence plots (e.g. small step-size, oscillations, bad approximation quality, etc.), a more in-depth investigation is performed using the local plots. To put it simply, with the global plots an issue is detected and by using the local plots one tries to understand what causes it. This is an important step needed to handle the arising approximation issues efficiently when designing an adaptive mixed scheme.

Pair analysis plots

Although Fig. 3.5b and Fig. 3.4a can be regarded as local plots as well, by far the most insightful category of post-optimization performance measures found in this report is the pair analysis plots. Our numerical experiments indicate that one can infer from them much more information about the quality of an approximation than any other local plot generated herein. Inspired by the approach of [12], every detrimental situation of the optimization problems found in Chapter 5 is analyzed on a fundamental level using pair plots. The biggest advantage of these plots is that they offer a simple and clear image of how each response function $g_j(\mathbf{X})$ depends on each design variable x_i. Thus all unwanted behaviours can be explained by superposing $\{x_i, \widetilde{g}_j\}$ on $\{x_i, g_j\}$ plots, as in Fig. 3.6. However, as a problem's characteristic size (i.e. $n \cdot (m+1)$) – and therefore the number of pair plots – increases, this performance measure becomes impractical and to the author's knowledge there is no aggregated alternative.

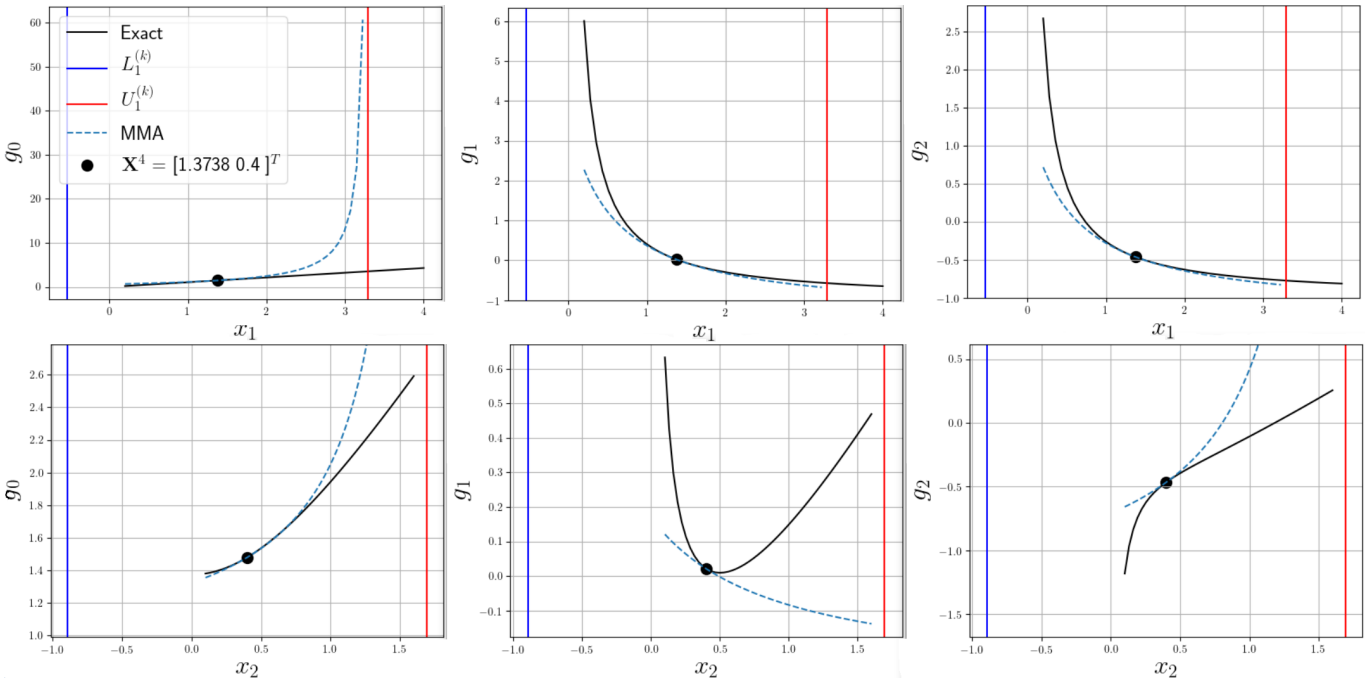


Figure 3.6: Pair analysis plots for the 4^{th} iteration of Eq. (3.1) with MMA. Good $(\{x_1, g_0\}, \{x_1, g_1\}, \{x_1, g_2\}, \{x_2, g_0\})$ and bad $(\{x_2, g_1\}, \{x_2, g_2\})$ approximation quality for the respective design variables

3.3 Real-time behaviour indicators

After detecting an approximation issue and understanding its cause (e.g. oscillatory behaviour caused by approximating a non-monotonous $g_j(\mathbf{X})$ with a monotonous $\widetilde{g}_j(\mathbf{X})$), the next step one needs to take towards generating a robust and efficient mixed scheme is to prevent the issue's occurrence. This is achieved by evaluating certain indicators from the available information, as the optimization is running (i.e. real-time). Taking into account the large-scale nature of the TO problems we ultimately aim to address, our approach requires storing information of the last 3 visited points. Our numerical experiments showed that – for the studied problems of Chapter 5 – 3 points were enough to estimate the behaviour of functions $g_j(\mathbf{X})$ and predict the aforementioned issues from occurring.

By analyzing several problems, a list of important checks to prevent bad approximations was generated. These are

common causes for undesired optimizer behaviour. This list is by no means exhaustive and by including more problems, more causes for low approximation quality are likely to be found.

- **Linearity/Concavity**
  Since the implemented solver can only handle convex functions $\widetilde{g}_j(\mathbf{X})$, the best approximation for a non-convex (or linear) region of $g_j(\mathbf{X})$ is the linear one. Those instances are detected by the following checks:

$$\left| \left.\frac{\partial g_j}{\partial x_i}\right|_{\mathbf{X}^{(k)}} - \left.\frac{\partial g_j}{\partial x_i}\right|_{\mathbf{X}^{(k-1)}} \right| < \varepsilon_1 \ , \qquad \forall i,j \tag{3.5a}$$

$$\left(\left.\frac{\partial g_j}{\partial x_i}\right|_{\mathbf{X}^{(k)}} - \left.\frac{\partial g_j}{\partial x_i}\right|_{\mathbf{X}^{(k-1)}} \right) \cdot \left(x_i^{(k)} - x_i^{(k-1)} \right) < 0 \ , \qquad \forall i,j \tag{3.5b}$$

  for which, our numerical experiments show that a reasonable value could be $\varepsilon_1 \approx 10^{-5}$.

- **Proximity to optimum**
  As the optimizer approaches the true local minimum of $\mathcal{P}_{\text{NLP}}$ (i.e. $\mathbf{X}^*$), oscillatory behaviour often occurs. Taking too large of a step in any direction can lead to redundant iterations and consequently unnecessary function evaluations. Therefore, in the vicinity of $\mathbf{X}^*$, one must opt for the most conservative approximation functions $\widetilde{g}_j(\mathbf{X})$ to ensure that a relatively large step-size does not turn the approximate local minimum of $\widetilde{\mathcal{P}}_{\text{NLP}}$ (i.e. $\mathbf{X}^{*(k)}$) away from $\mathbf{X}^*$. Proximity to optimum $\mathbf{X}^*$ is detected by:

$$\left| \frac{x_i^{(k)} - x_i^{(k-1)}}{\overline{x}_i - \underline{x}_i} \right| < \varepsilon_2 \ , \qquad \forall i \tag{3.6}$$

  for which, our numerical experiments show that a reasonable value could be $\varepsilon_2 \approx 10^{-2}$.

- **Monotonicity**
  In order to achieve the 'best-fit' of $\widetilde{g}_j(\mathbf{X})$ on $g_j(\mathbf{X})$, the structure of those functions must be alike. Therefore, it is desirable to approximate a monotonous (non-monotonous) $g_j(\mathbf{X})$ with with a monotonous (non-monotonous) $\widetilde{g}_j(\mathbf{X})$. Otherwise, either convergence is slowed down by small steps, or oscillations occur due to large steps. Monotonicity is checked as follows:

$$\left(x_i^{(k)} - x_i^{(k-1)} \right) \cdot \left(x_i^{(k-1)} - x_i^{(k-2)} \right) > 0 \ , \qquad \forall i \tag{3.7a}$$

$$\left(g_j(\mathbf{X}) - g_j(\mathbf{X}^{(k-1)}) \right) \cdot \left(g_j(\mathbf{X}^{(k-1)}) - g_j(\mathbf{X}^{(k-2)}) \right) > 0 \ , \qquad \forall j \tag{3.7b}$$

$$\left.\frac{\partial g_j}{\partial x_i}\right|_{\mathbf{X}^{(k)}} \cdot \left.\frac{\partial g_j}{\partial x_i}\right|_{\mathbf{X}^{(k-1)}} > 0 \ , \qquad \forall i,j \tag{3.7c}$$

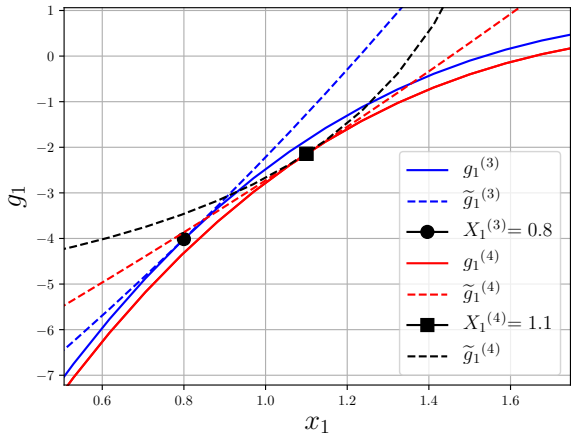
So far we have mentioned indicators that have already been applied to other mixes schemes of the literature, e.g. [12]. In the coming section, a set innovative indicators will be presented that aims to improve the outcome of the approximation function selection.

3.4 Unreported inadequacies of approximation enhancement

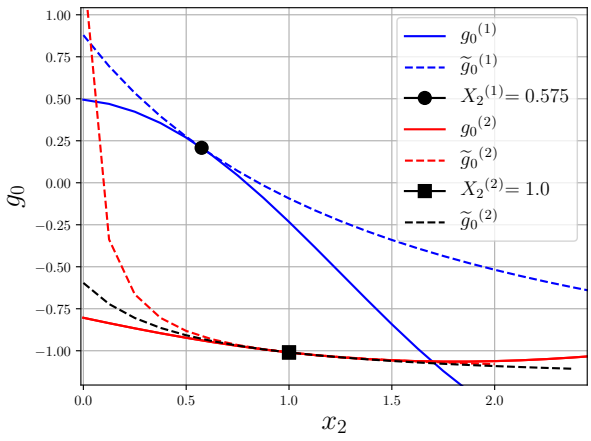
Although using history information to improve the 'fit' of $\widetilde{g}_j(\mathbf{X})$ on $g_j(\mathbf{X})$ at a given point $\mathbf{X}^{(k)}$ sounds intuitively beneficial, it is often not the case. What is more, as the number of variables (n) increases, enriching $\widetilde{g}_j(\mathbf{X})$ with previous point information is more likely to be detrimental. To the author's knowledge, at the time of writing this report there is no research available in the corresponding literature on when enriching a function $\widetilde{g}_j(\mathbf{X})$ with history information is beneficial or not.

In this section we report the undesired effect history information may have on the function approximation quality. More specifically, we attempt to answer the following question: when should one use the available information stored in previous iterations to enrich an approximation $\widetilde{g}_j(\mathbf{X})$? Moreover, since using previous point information is common practice in SAO algorithm enhancement (see Section 2.1), our conclusions are expected to be extrapolatable on other approximation methods (e.g. [42]), besides the MMA family of approximations that was implemented.

Using the pair analysis plots described in Section 3.2.2 on low-dimensional analytical problems, the behaviour of the exact functions with respect to all variables (i.e. exact pair of $\{x_i, g_j\}$), as well as their respective approximations (i.e. approximate pair of $\{x_i, \widetilde{g}_j\}$), can be captured in $n \cdot (m+1)$ plots, see Fig. 3.6. Then, by superposing the pair plots of two successive iterations (i.e. at $\mathbf{X}^{(k-1)}$ and $\mathbf{X}^{(k)}$) as in Fig. 3.7, one can clearly see when the use of the previous point $x_i^{(k-1)}$ may improve or worsen the 'fit' of $\{x_i, \widetilde{g}_j\}$ on $\{x_i, g_j\}$. Looking at Fig. 3.7, if the exact function of $\{x_i, g_j\}$ shown in the solid curves has changed significantly between the two iterations due to a change in the remaining design variables of the problem, forcing the current approximate pair (i.e. red dashed curve) to satisfy the previous point (i.e. $\widetilde{g}_j(x_i^{(k-1)}) = g_j(x_i^{(k-1)})$) and/or its partial derivative (i.e. $\partial\widetilde{g}_j/\partial x_i(x_i^{(k-1)}) = \partial g_j/\partial x_i(x_i^{(k-1)})$) is not advisable, see Fig. 3.7b. On the contrary, when the exact pair of $\{x_i, g_j\}$ is not influenced as much by a change in the remaining design variables, enriching the current approximate pair with previous point information is beneficial, see Fig. 3.7a. From an implementation point of view, fitting the current approximation through the previous point requires adjusting its asymptotes by an iterative Newton-Raphson scheme, see Eq. (2.10). Thus, it only makes sense to do so when the accuracy of the approximation is likely to be improved, as in Fig. 3.7a. In any other case, see Fig. 3.7b, attempting to fit the current approximation (red dashed curve) through the previous point (black circle) will result in unnecessary Newton-Raphson iterations and – possibly – worsen the approximation quality. This can be handled to some extent by imposing move limits on the asymptotes, improving the stability of methods that enrich their approximations with previous point information (e.g. GMMA). Nevertheless, it is completely unnecessary to perform these computationally intensive iterations when it can be predicted that the previous point is likely not usable and therefore the risk of worsening the approximation quality is high.



(a) Beneficial use of $\mathbf{X}^{(k-1)}$ by GMMA (red) compared to not using it, as in MMA (black)



(b) Detrimental use of $\mathbf{X}^{(k-1)}$ by GMMA (red) compared to not using it, as in MMA (black)

Figure 3.7: Pair plots of $\{x_i, g_j\}$ (solid lines) and $\{x_i, \widetilde{g}_j\}$ (dashed lines) for the 1^{st} (blue) and 2^{nd} (red for GMMA and black for MMA) iterations of Eq. (3.1)

As mentioned before, this analysis is only possible for low-dimensional problems for which we can evaluate all the functions involved at any point we desire. Consequently, after understanding the logic of when using history information is beneficial or not, one must include this information in an approximation selection criterion, e.g. Eq. (3.10), in order to generate a robust and efficient approximation scheme. Thus, stemming from the observations of Fig. 3.7, and in addition to the real-time indicators presented in Section 3.3, we propose the following criterion to assess the usability of $\mathbf{X}^{(k-1)}$:

- **Usability of $\mathbf{X}^{(k-1)}$**
  We first introduce the following backward finite difference scheme:

$$A_{ij}^{(k)} = \frac{g_j(\mathbf{X}^{(k)}) - g_j(\mathbf{X}^{(k-1)})}{x_i^{(k)} - x_i^{(k-1)}} \qquad \forall i \ , \qquad \forall j \tag{3.8}$$

  Subsequently, one can formulate a ratio between the exact partial derivative and its approximation by the afore-

mentioned finite difference scheme:

$$B_{ij}^{(k)} = \frac{\frac{\partial g_j}{\partial x_i}\big|_{\mathbf{X}^{(k)}}}{A_{ij}^{(k)}} = \frac{\frac{\partial g_j}{\partial x_i}\big|_{\mathbf{X}^{(k)}} \cdot \left(x_i^{(k)} - x_i^{(k-1)}\right)}{g_j(\mathbf{X}^{(k)}) - g_j(\mathbf{X}^{(k-1)})} = \frac{\Delta g_j\big|_{predicted}}{\Delta g_j\big|_{actual}} \quad , \quad \forall i \ , \quad \forall j \tag{3.9}$$

Using Eq. (3.8) and Eq. (3.9) one can formulate the following conditions:

$$0.8 \leq \ B_{ij}^{(l)} \ \leq 1.2 \quad , \quad \forall i \ , \quad \forall j \ , \quad l = k-1, \ k \tag{3.10a}$$

$$\left.\frac{\partial g_j}{\partial x_i}\right|_{\mathbf{X}^{(k)}} \leq A_{ij}^{(k)} \ \leq \left.\frac{\partial g_j}{\partial x_i}\right|_{\mathbf{X}^{(k-1)}} \quad , \quad \forall i \ , \quad \forall j \tag{3.10b}$$

$$\left.\frac{\partial g_j}{\partial x_i}\right|_{\mathbf{X}^{(k-1)}} \leq A_{ij}^{(k)} \ \leq \left.\frac{\partial g_j}{\partial x_i}\right|_{\mathbf{X}^{(k)}} \quad , \quad \forall i \ , \quad \forall j \tag{3.10c}$$

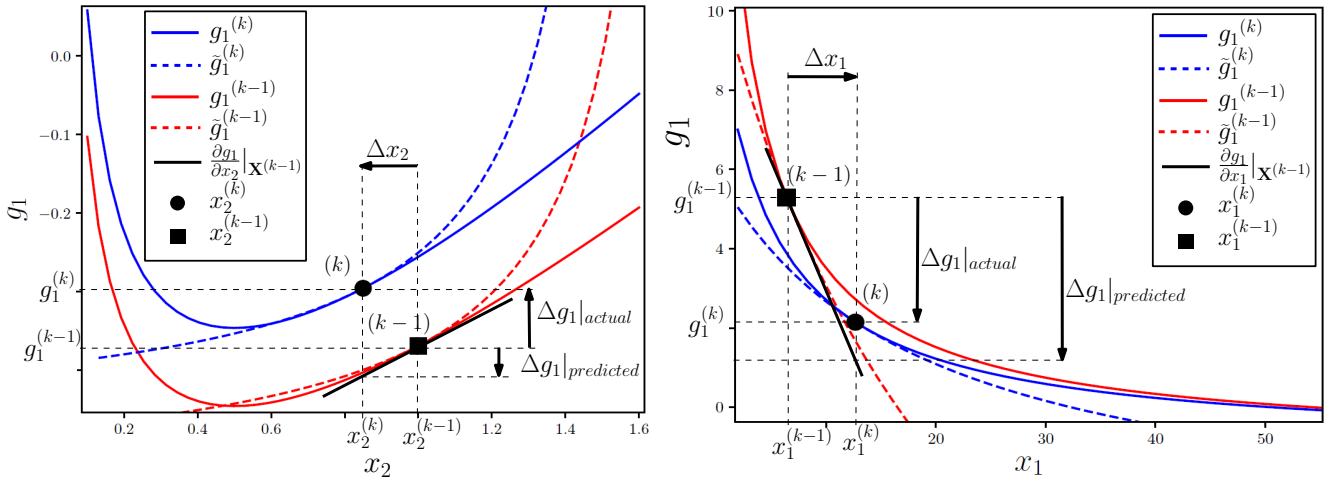
which are then used to assess the usability of $\mathbf{X}^{(k-1)}$ according to the Algorithm 1 below:

Algorithm 1: Algorithm to assess the usability of $\mathbf{X}^{(k-1)}$

for *i in range(1, n)*:
  for *j in range(0, m)*:
    if *Eq.* (3.10a), $\forall l$ == *True*:
      $\mathbf{X}^{(k-1)}$ can be used for the pair $\{x_i, g_j\}$
    elif *(Eq.* (3.10b) == *True)* ***or*** *(Eq.* (3.10c) == *True)*:
      $\mathbf{X}^{(k-1)}$ can be used for the pair $\{x_i, g_j\}$
    else:
      $\mathbf{X}^{(k-1)}$ cannot be used for the pair $\{x_i, g_j\}$
    j = j + 1
  i = i + 1

As seen in Eq. (3.9), $B_{ij}^{(k)}$ is the ratio between the change in function value that occurred (denominator), compared to the change that could be predicted based on only the change of the considered variable (nominator). Assuming small changes in all design variables, which is guaranteed by applying the move limits of Eq. (1.6), the dominant source of discrepancies between the nominator and the denominator of Eq. (3.9) is the change in the remaining design variables. Consequently, Eq. (3.10a) can be interpreted as follows: if $B_{ij}^{(k)}$ is close to 1, enriching the approximation with previous point information will likely be beneficial. On the other hand, if $B_{ij}^{(k)}$ diverges from unity, either the function is highly nonlinear with respect to x_i, or the remaining design variables have caused a significant change in $g_j(\mathbf{X})$ in the vicinity of $\mathbf{X}^{(k)}$. In both cases, using information from $x_i^{(k-1)}$ is risky, see Fig. 3.8a.

A criterion like Eq. (3.10a) can not predict all the detrimental scenarios an optimizer might encounter. For example, as the step-size increases, the finite difference scheme of Eq. (3.8) becomes less accurate and Eq. (3.10a) might incorrectly classify the previous point unusable. This possibility is accounted for by applying Eq. (3.10b) and Eq. (3.10c). If either of those conditions hold and the response function $g_j(\mathbf{X})$ is monotonous with respect to x_i between $[x_i^{(k-1)}, x_i^{(k)}]$, the previous point is likely usable, see Fig. 3.8b. Our numerical experiments, see Chapter 5, indicate that Algorithm 1 – in combination with the behaviour indicators presented in Section 3.3 – can reduce the number of iterations required by an SAO method to converge. More information on the generated mixed scheme can be found on the next chapter.



(a) $\mathbf{X}^{(k-1)}$ is not usable due to Eq. (3.10a)

(b) $\mathbf{X}^{(k-1)}$ is usable due to Eq. (3.10b) and Eq. (3.10c)

Figure 3.8: Applying Algorithm 1 to assess the usability of $\mathbf{X}^{(k-1)}$

3.5 Additional indicators

This section contains several additional indicators that have proven to be useful in assessing the performance of an approximation scheme.

- **Direction index**
  This is a scalar valued metric that indicates the consistency in the direction of the solution path an optimization algorithm follows. A boundary filter is applied to remove the undesired influence of variables that have reached their bounds (within a tolerance ε_i).

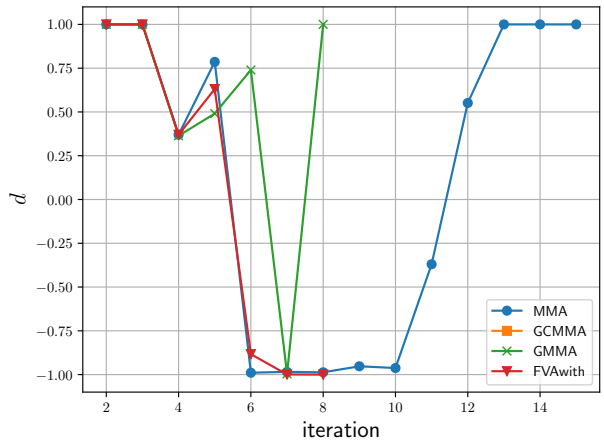
$$d = \frac{\left(\mathbf{F}\Delta\mathbf{X}^{(k)}\right)^T \cdot \left(\mathbf{F}\Delta\mathbf{X}^{(k-1)}\right)}{\left|\left|\mathbf{F}\Delta\mathbf{X}^{(k)}\right|\right|_2 \cdot \left|\left|\mathbf{F}\Delta\mathbf{X}^{(k-1)}\right|\right|_2}\ , \qquad -1 \leq d \leq 1 \tag{3.11a}$$

$$F_{ii} = \tanh\left(\frac{3(x_i - \underline{x}_i)}{\varepsilon_i}\right)\tanh\left(\frac{3(-x_i + \overline{x}_i)}{\varepsilon_i}\right)\ , \qquad \forall i \tag{3.11b}$$

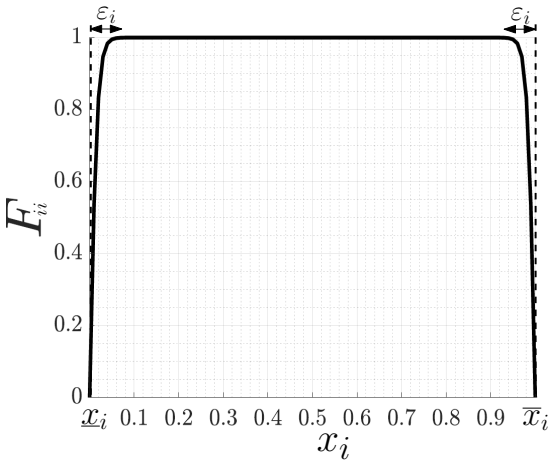
  for which a reasonable value could be $\varepsilon_i \approx (\overline{x}_i - \underline{x}_i) \cdot 10^{-2}$. In Fig. 3.9 one can see how Eq. (3.11) can be applied to the template problem of Eq. (3.1). Furthermore, 3 distinctive cases for the values Eq. (3.11a) can be found below:

$$\begin{aligned} &\text{if } \ d = 1 && \implies && \Delta\mathbf{X}^{(k)} \uparrow\uparrow \Delta\mathbf{X}^{(k-1)} \\ &\text{if } \ d = -1 && \implies && \Delta\mathbf{X}^{(k)} \uparrow\downarrow \Delta\mathbf{X}^{(k-1)} \\ &\text{if } \ d = 0 && \implies && \Delta\mathbf{X}^{(k)} \text{ and } \Delta\mathbf{X}^{(k-1)} \text{ are unrelated} \end{aligned}$$

  One can observe from Fig. 3.9a that after the 6^{th} iteration, $d \approx -1$ for most approximations methods. This means that these optimizers oscillate around a point, as the steps taken have opposite directions for several iterations.



(a) Evolution of direction index for different SAO methods used to solve Eq. (3.1)



(b) Filter of Eq. (3.11b) to remove the influence of variables that are closer than ε to their bounds

Figure 3.9: Direction index and boundary filter used

- **Expectation index**
  Eq. (3.9) can be aggregated to a scalar valued index for each response function g_j. Although some information is lost in this aggregation, the macroscopic view of a function's behaviour that is provided by the following index can be useful. This is an index that essentially shows the average degree of nonlinearity of a function g_j. It is defined as follows:

$$e_j = \frac{\left(\Delta \mathbf{X}^{(k)}\right)^T \cdot \left(\frac{dg_j}{d\mathbf{X}}\Big|_{\mathbf{X}^{(k)}}\right)}{g_j^{(k)} - g_j^{(k-1)}} \quad , \qquad \forall j \tag{3.13}$$

  The most distinctive values of Eq. (3.13) can be found below:

$$\begin{aligned} &\text{if} \;\; e_j > 1 \quad &\Longrightarrow \quad &g_j \text{ resembles concave functions for } \mathbf{X} \in [\mathbf{X}^{(k-1)}, \mathbf{X}^{(k)}] \\ &\text{if} \;\; e_j = 1 \quad &\Longrightarrow \quad &g_j \text{ resembles linear functions for } \mathbf{X} \in [\mathbf{X}^{(k-1)}, \mathbf{X}^{(k)}] \\ &\text{if} \;\; e_j < 1 \quad &\Longrightarrow \quad &g_j \text{ resembles convex functions for } \mathbf{X} \in [\mathbf{X}^{(k-1)}, \mathbf{X}^{(k)}] \end{aligned}$$

  Obviously, when Eq. (3.13) is used, one must make sure that $\Delta g_j^{(k)} = g_j^{(k)} - g_j^{(k-1)} \neq 0$. Therefore, towards the end of the optimization (when $\Delta g_j^{(k)} \to 0$ and/or $\Delta \mathbf{X}^{(k)} \to 0$) the use of this index is not suggested. For example, in Fig. 3.10a one can see that at the 7^{th} iteration of MMA numerical complications arise because $\Delta g_1^{(k)} \to 0$. What is more, although e_2 suggests that g_2 is behaving linearly around the points MMA visits from the 9^{th} iteration onward, this is not the case. This happens because the step-size taken by MMA is reduced substantially, yielding $\Delta \mathbf{X}^{(k)} \to 0$. Reasonably, any continuous function resembles a linear one within a region defined by $\mathbf{X} \in [\mathbf{X}^{(k-1)}, \mathbf{X}^{(k)}]$ if that region is infinitesimal.

- **Participation index**
  This index shows the participation of a partial derivative $\frac{\partial g_j}{\partial x_i}\Big|_{\mathbf{X}^{(k)}}$ in the change of each variable $\Delta x_i^{(k)}$. It holds that:

$$p_j = \frac{\left(\mathbf{F}\Delta \mathbf{X}^{(k)}\right)^T \cdot \left(\mathbf{F}\frac{dg_j}{d\mathbf{X}}\Big|_{\mathbf{X}^{(k)}}\right)}{\left|\left|\mathbf{F}\Delta \mathbf{X}^{(k)}\right|\right|_2 \cdot \left|\left|\mathbf{F}\frac{dg_j}{d\mathbf{X}}\Big|_{\mathbf{X}^{(k)}}\right|\right|_2} \quad , \qquad \forall j \tag{3.15}$$

  where,

$$-1 \leq p_j \leq 1 \quad , \qquad \forall j$$

  Similarly to the direction index, the filter of Fig. 3.9b is applied to remove the influence of variables that have

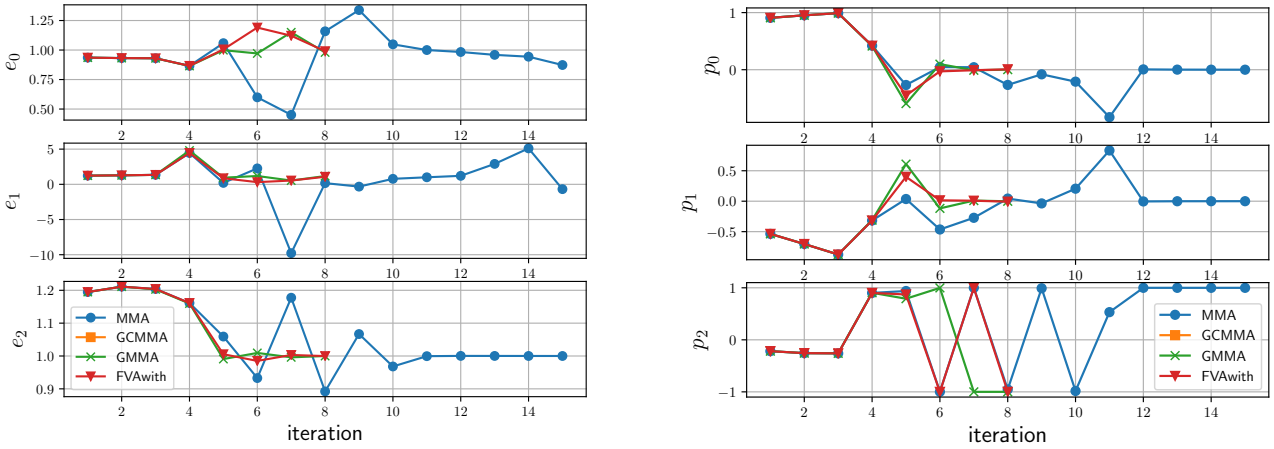
essentially reached their bounds. The most distinctive cases for Eq. (3.15) can be found below:

$$\text{if } \; p_j = 1 \quad \Longrightarrow \quad \left.\frac{dg_j}{d\mathbf{X}}\right|_{\mathbf{X}^{(k)}} \uparrow\uparrow \Delta\mathbf{X}^{(k)}$$

$$\text{if } \; p_j = -1 \quad \Longrightarrow \quad \left.\frac{dg_j}{d\mathbf{X}}\right|_{\mathbf{X}^{(k)}} \uparrow\downarrow \Delta\mathbf{X}^{(k)}$$

$$\text{if } \; p_j = 0 \quad \Longrightarrow \quad \left.\frac{dg_j}{d\mathbf{X}}\right|_{\mathbf{X}^{(k)}} \text{and } \Delta\mathbf{X}^{(k)} \text{ are unrelated}$$

The use of Eq. (3.15) can be seen in Fig. 3.10b, wherein one can observe the oscillatory behaviour of most SAO schemes depicted. Furthermore, it can be concluded that these fluctuations are attributed to constraint g_2, since p_2 oscillates between -1 and 1 from the 6^{th} iteration onward. On the contrary, p_0 and p_1 converge to 0 for most of the depicted schemes, indicating a lack of participation in the steps taken from most schemes after the 6^{th} iteration.



(a) Expectation index **(b)** Participation index

Figure 3.10: Evolution of direction and participation indices $\forall g_j$ of Eq. (3.1) for different approximation methods

- **Switching frequency array**
  This is a useful metric that can be used to oversee the utilization of a mixed scheme. It is a 2D array of size $(m+1)\times n$ that displays the number of times the approximation function of $\{g_j, x_i\}$ is changed. By using it, one gains further insight on how a mixed scheme behaves. For example, it is possible to detect 'over-switching' occurrences that could lead to oscillatory behaviour. One can clearly see in Fig. 3.11 that switching the approximation function for $\{g_{11}, x_9\}$ 20 times in an optimization that converges after 29 iterations is – at best – unnecessary.

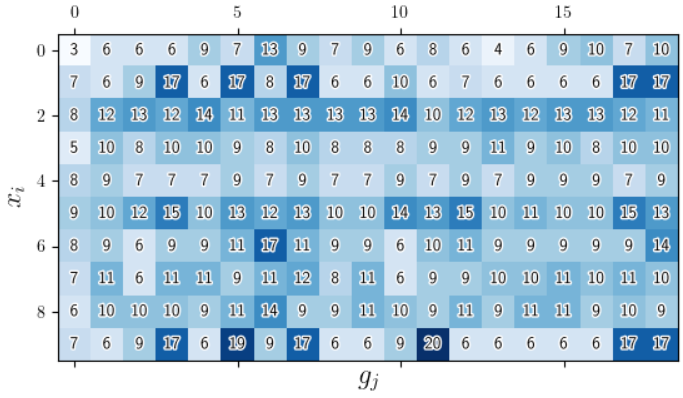


Figure 3.11: The transpose of an array of size $(m+1)\times n$ that displays the number of times a mixed scheme changes the approximation function for the pair of $\{g_j, x_i\}$. This array refers to the 10-bar truss problem of Section 5.1 that converges after 29 iterations.

3.6 Performance profiles

Comparing different approximation schemes objectively is far from trivial. There are many factors that need to be considered, all of which are hard to be captured in a single comparison technique. For example, when different optimizers are compared with respect to their performance on a specific problem set, non-convergence of an optimizer in a particular problem is an issue. One can must either remove the problem from the considered set, or penalize its performance. Both approaches are problematic: the former causes information loss and the latter depends strongly on the penalization factor used. Furthermore, averaging/summing performance measures can be misleading, as outliers can influence the results in an undesirable way.

A *performance profile* is generated in order to obtain an indication of the relative performance of different methods (e.g. approximation schemes), instead of their relative capability to solve certain problems [15]. Obviously, this method becomes more reliable as the number and variety of the included problems increase. Introduced by [51], performance profiles enable the comparison of n optimization methods (i.e. $s_1, s_2, ..., s_n$) with respect to a single performance measure Φ (e.g. the number of iterations needed for convergence, the objective value obtained, etc.) when a problem set of m different problems (i.e. $p_1, p_2, ..., p_m$) is considered. Assuming the minimization of Φ is desired, by solving the problem p_1 with all methods s_j, one can obtain the method with the best performance s_{best}, for which $\Phi_{best} = min\{\Phi_1, ..., \Phi_n\}$. Moreover, one can also obtain a *performance ratio* $\forall s_j$ defined as: Φ_j/Φ_{best}. Obviously, $\Phi_j/\Phi_{best} \geq 1$ and the equality holds only for the method(s) with the top performance for p_1.

A *performance profile* of a method s_j indicates the number of problems solved by this method within a certain ratio of Φ_j/Φ_{best} from the method with the top performance (for that particular problem). For example, in Fig. 3.12 we can see that Method A has the best performance for 70% of the problems (since for $\Phi_A/\Phi_{best} = 1$ the value of the profile is 70%). For Method B, 60% of the problems can be solved successfully with a 10% relaxed performance measure (i.e. $\Phi_B/\Phi_{best} = 1.1 \implies \Phi_B = 1.1 \cdot \Phi_{best}$). Another important characteristic of those plots is that non-convergence can also be addressed. If we assume that the maximum allowed performance measure is 20% more than the optimal one (i.e. $\Phi_{max}/\Phi_{best} = 1.2$), one can see that Method A solves 100% of the considered problems, whereas Methods B and C solve around 85% and 79% respectively. This means that overall convergence and robustness can also be evaluated through these profiles.

What is more, one can see that Method C plateaus for $1 \leq \Phi_C/\Phi_{best} \leq 1.05$ and increases with a steep slope for $1.05 \leq \Phi_C/\Phi_{best} \leq 1.08$. On the one hand, a plateau region means that the considered method will not solve successfully more of the implemented problems for the coming increase in performance ratio. On the other hand, a steep slope signifies a large increase in the number of problems that are successfully solved with a small increase in performance ratio. In short, profile curves that are higher than others perform better – in terms of the considered performance measure Φ – for the considered problem set.

Taking the above into consideration, this method is applied on the approximation schemes under comparison in Section 5.3. This concise – yet informative – representation of the results offers a clear image of the relative performance of the aforementioned schemes and facilitates the derivation of conclusions.

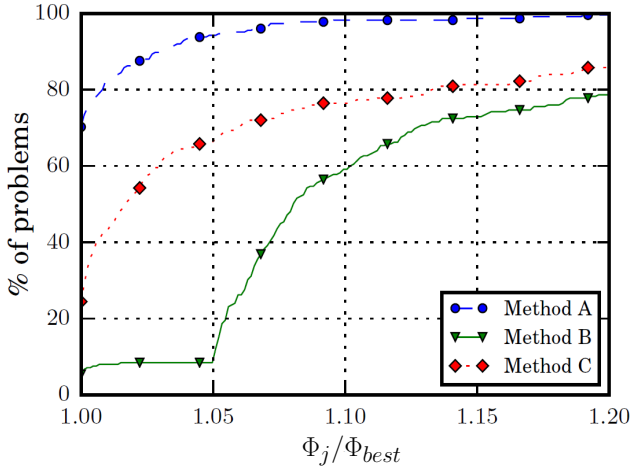


Figure 3.12: Example of performance profiles of 3 optimizers when tested on 225 minimum compliance problems [15]

4 | Novel mixed schemes

4.1 Classification of mixed schemes

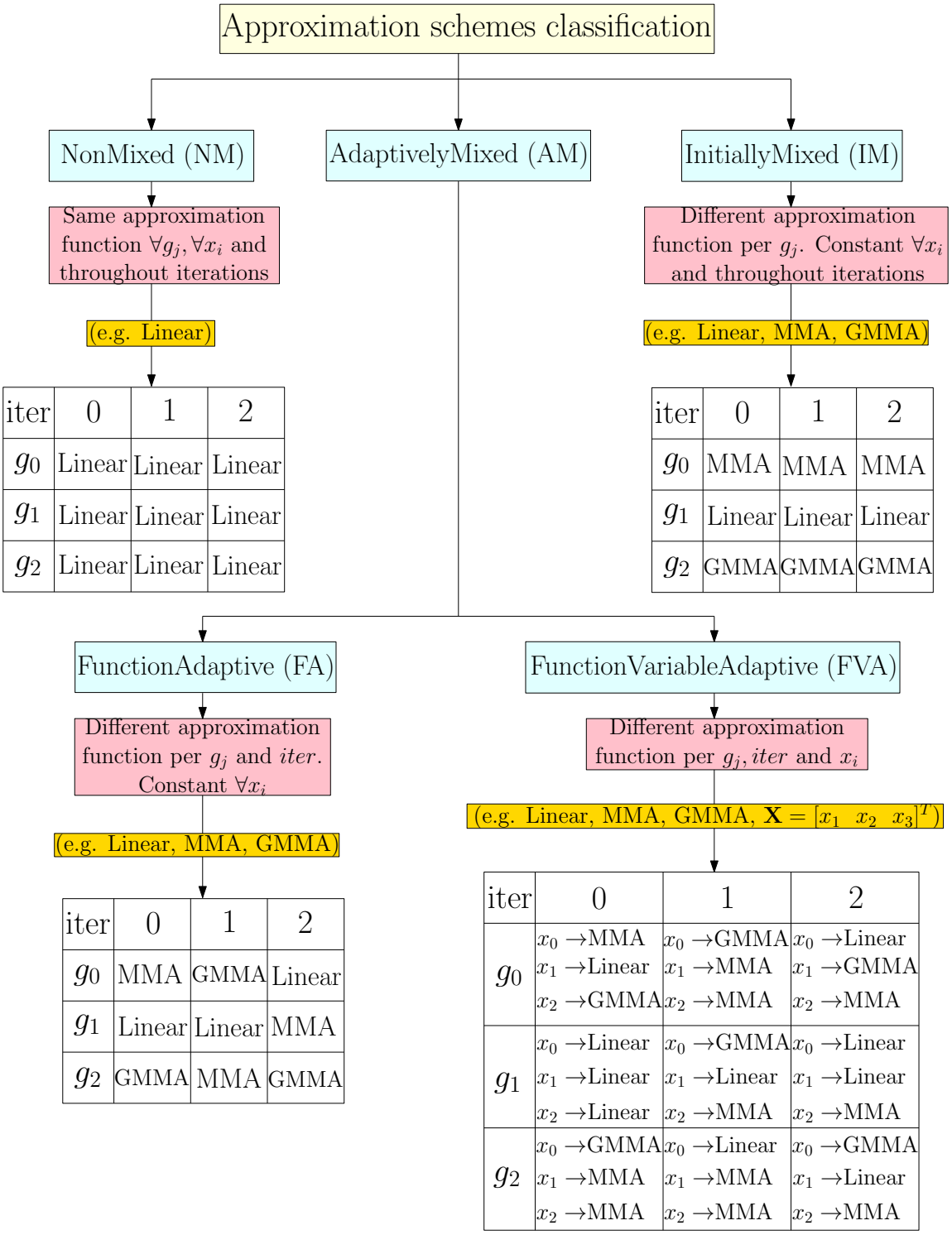
In this chapter we present a set of novel mixed schemes for function approximations based on the MMA family of approximations. Throughout this report we have presented argumentation in favor of mixed schemes and the benefits of their use in SAO algorithms. Thus, considering the impact an efficient mixed scheme could have on such algorithms, it is of utmost importance to explore all the possible features such schemes may offer. To this end, a novel classification method for the approximation schemes is introduced in Fig. 4.1 and used in the Optimization Lab of Appendix A. All possible approximation schemes are categorized according to their degree of adaptiveness.

Definitions of scheme types

One can find 3 generic types of adaptive approximation schemes in Fig. 4.1:

- **Non-Mixed (NM)**
  This type uses a single approximation method $\forall g_j, \forall x_i$ throughout all iterations of the optimization. One can see in Fig. 4.1 that – if the Linear approximation method is chosen – all responses $g_j(\mathbf{X})$ at all iterations are approximated by it. Therefore this type is not a mixed scheme and its adaptiveness depends on the inherent adaptiveness of the approximation method used. For example, a Linear approximation method does not adapt to the local characteristics of the response $g_j(\mathbf{X})$ it is applied on, whereas MMA can adjust its convexity by moving its asymptotes accordingly between iterations.
- **Initially-Mixed (IM)**
  This type of mixed schemes can assign a different approximation method to each response $g_j(\mathbf{X})$. This assignment remains constant throughout all iterations of the optimization (with the exception perhaps of a few initial 'test' iterations, wherein one can collect some data in order to assign the most appropriate method to each response). All the variables of that response, i.e. $\forall x_i \in g_j(\mathbf{X})$, are approximated by the same approximation method (e.g. MMA is used for g_0, Linear is used for g_1 and GMMA is used for g_2, see Fig. 4.1).
- **Adaptively-Mixed (AM)**
  This type of mixed schemes is far more adaptive than the previous ones. Furthermore, it is divided into 2 sub-categories: Function-Adaptive (FA) schemes use the same approximation method $\forall x_i$ of a response $g_j(\mathbf{X})$, whereas for Function-Variable-Adaptive (FVA) schemes that is not the case. When FVA schemes are used, one can assign a different approximation method to the contribution of each variable x_i to a response function $g_j(\mathbf{X})$. Since all methods used herein are separable, the contribution of each variable to each function can be extracted. This feature may be particularly useful for problems with dissimilar behaviour of design variables, see Section 1.4.

It is important to note that by making a scheme more adaptive (i.e. Non-Mixed $\rightarrow$ Initially-Mixed $\rightarrow$ Adaptively-Mixed) the computational cost of the approximation process performed every iteration increases significantly. Therefore, taking into account the specific characteristics of the addressed problem (e.g. dissimilar behaviour of design variables, see Section 1.4), one must choose between a 'flexible and expensive' AM scheme, or a 'stiff and inexpensive' one like IM schemes. In other words, there is a trade-off between the adaptiveness of the generated approximation and its computational cost.



iter	0	1	2
g_0	Linear	Linear	Linear
g_1	Linear	Linear	Linear
g_2	Linear	Linear	Linear

iter	0	1	2
g_0	MMA	MMA	MMA
g_1	Linear	Linear	Linear
g_2	GMMA	GMMA	GMMA

iter	0	1	2
g_0	MMA	GMMA	Linear
g_1	Linear	Linear	MMA
g_2	GMMA	MMA	GMMA

iter	0	1	2
g_0	$x_0 \rightarrow$MMA $x_1 \rightarrow$Linear $x_2 \rightarrow$GMMA	$x_0 \rightarrow$GMMA $x_1 \rightarrow$MMA $x_2 \rightarrow$MMA	$x_0 \rightarrow$Linear $x_1 \rightarrow$GMMA $x_2 \rightarrow$MMA
g_1	$x_0 \rightarrow$Linear $x_1 \rightarrow$Linear $x_2 \rightarrow$Linear	$x_0 \rightarrow$GMMA $x_1 \rightarrow$Linear $x_2 \rightarrow$MMA	$x_0 \rightarrow$Linear $x_1 \rightarrow$Linear $x_2 \rightarrow$MMA
g_2	$x_0 \rightarrow$GMMA $x_1 \rightarrow$MMA $x_2 \rightarrow$MMA	$x_0 \rightarrow$Linear $x_1 \rightarrow$MMA $x_2 \rightarrow$MMA	$x_0 \rightarrow$GMMA $x_1 \rightarrow$Linear $x_2 \rightarrow$MMA

Figure 4.1: Classification of approximation schemes with respect to their degree of adaptiveness. The main categories are non-mixed (NM), initially-mixed (IM) and adaptively-mixed schemes (AM). The latter ones are divided in function-adaptive (FA) and function-variable-adaptive (FVA) schemes depending on their adaptive capabilities.

Given the importance of this chapter, an overview of the implementation of the different mixed schemes that are included in the Optimization Lab of Appendix A can be seen in the coming sections. At the time of writing this report, the framework within which different approximation functions are combined to create a mixed scheme resembles the GMMA approximation. To be specific, all approximations to be used are transformed in a GMMA-compatible format

(i.e. by having a set of $p_{ij}^{(k)}, q_{ij}^{(k)}, L_{ij}^{(k)}, U_{ij}^{(k)}, \alpha_i^{(k)}, \beta_i^{(k)}, r_j^{(k)}$) that can be used to assemble the constituent approximations to a $\widetilde{g}_j(\mathbf{X})$, as in Eq. (4.1):

$$\begin{aligned} \widetilde{g}_j(\mathbf{X}) &= r_j^{(k)} + \sum_{i=1}^{n} \left\{ p_{ij}^{(k)} \frac{1}{U_{ij}^{(k)} - x_i} + q_{ij}^{(k)} \frac{1}{x_i - L_{ij}^{(k)}} \right\} = r_j^{(k)} + \sum_{i=1}^{n} \widetilde{g}_{ij}(x_i) \\ r_j^{(k)} &= g_j(\mathbf{X}^{(k)}) - \sum_{i=1}^{n} \left\{ p_{ij}^{(k)} \frac{1}{U_{ij}^{(k)} - x_i^{(k)}} + q_{ij}^{(k)} \frac{1}{x_i^{(k)} - L_{ij}^{(k)}} \right\} \end{aligned} \tag{4.1}$$

However, one could generate a similar mixed scheme for any other family of approximation functions with minimal changes to the Optimization Lab. Different families of approximation functions can also be combined, as long as the assembly of Eq. (4.1) includes only separable and convex functions. For example, a mixed scheme that includes the exponential approximation of Eq. (2.16) could be useful in problems that include functions of similar – exponential – structure. In that case, $\widetilde{g}_{ij}(x_i)$ terms must be used to assemble the approximation $\widetilde{g}_j(\mathbf{X})$, instead of $p_{ij}^{(k)}, q_{ij}^{(k)}$. That way, one need not be limited to using approximations that are compatible with MMA. This generalized approach, along with several other approximation methods of Section 2.2, will be realized in an updated version of the Optimization Lab.

4.2 A Non-Mixed scheme

Non-Mixed schemes have been studied extensively in literature, see Chapter 2. However, in order to better understand the functionality of the generated mixed schemes that will follow, a concise flowchart of the implementation of Non-Mixed schemes in the Optimization Lab is given in Fig. A.4a.

4.3 An Initially-Mixed scheme

The significant advantage of Initially-Mixed schemes is that they are easily applicable to large problems, since their additional computational cost is marginal compared to the Non-Mixed schemes that are currently being used. However, it is questionable how to determine a-priori – or within a few 'test' iterations – which approximation function fits better the 'structure' of a response $g_j(\mathbf{X})$ without prior knowledge of that response's properties. What is more, even if an approximation function fits well a response for several iterations, one cannot guarantee that this will not change as the optimization progresses. Consequently, if an IM scheme is selected and no prior knowledge of the response functions is available, it is assumed that the initial iterations are representative of the whole optimization. Namely, there are 3 enhancements these schemes can offer to the currently used NM schemes:

- Different approximation method per response function
- When available, including information on a response function's structure (e.g. linear volume constraints, reciprocal stress/displacement constraints for sizing variables, etc.)
- Sampling data at initial iterations and estimating the structure of response functions

Although these schemes are computationally efficient, their use is more advantageous in cases where prior knowledge of the functions' structure is available. Currently, no criterion has been found that can predict robustly the structure of unknown responses by sampling data from initial iterations. Therefore, a simple criterion based on the approach of Fig. 2.7 is used, see Fig. A.6. However, the benefits of these schemes can still be shown by choosing a-priori the approximation method for (some of) the functions involved. For example, in Eq. (3.1) we can estimate that the objective function (weight of truss) is almost linear with respect to its design variables and that the stress constraints will have a reciprocal-like behaviour, with respect to x_1. The effect of including this information on the approximation can be seen in the figures of Section 5.3. There, two IM schemes are used: one that includes information on responses whenever available (i.e. 'IMwith'), and one that does not (i.e. 'IMwithout'). In Fig. A.5, one can see a flowchart of the implementation of such schemes in the context of the Optimization Lab.

4.4 An Adaptively-Mixed scheme

Although the FA and FVA schemes that constitute Adaptively-Mixed schemes may look inherently different, to the author's knowledge they share the same computational cost. This is attributed to the fact that by using a different method (and therefore its corresponding update formula) every iteration, the adaptiveness of an MMA-based approximation is compromised and oscillations occur. More specifically, the ability of using any approximation method (e.g. MMA, GMMA, Linear) on any response $g_j(\mathbf{X})$ at any iteration, requires one of the 2 following options: one must either have available all approximation methods $\forall g_j, \forall x_i$ at every iteration, or, use information from the approximation method used in the previous iteration. Either way, information must be passed from one iteration to the next in order to preserve the adaptiveness of MMA-based methods. An example of this can be seen in the FA scheme of Fig. 4.1: for the 2^{nd} iteration of g_0, one must either update MMA's asymptotes (calculated by Eq. (2.7)), or, have available GMMA's asymptotes for the 1^{st} iteration (calculated by Eq. (2.13)) even though it was not used. The first option adds a significant computational cost to the approximation process, whereas the second one introduces oscillatory behaviour due to the different update formulas of the constituent approximations' asymptotes. Therefore, the only viable option – so far – for a mixed scheme of this type is to compute all its constituent approximations, $\forall g_j, \forall x_i$ at every iteration. However, this could be addressed to some extent with parallel computing, since all the constituent approximations can be computed simultaneously (i.e. in parallel).

Being the most general and adaptive of the approximation schemes described herein, FVA schemes can be reduced to any other type. Furthermore, adding prior knowledge of response function types – as in IM schemes – is also possible. However, the accompanying high computational cost of making redundant function approximations every iteration needs to be addressed. All the scheme types mentioned in the previous section have advantages and disadvantages depending on the peculiarities of the problem at hand. The approximation methods used in the generated FVA scheme – all of which are convex and separable – can be seen below:

- Linear , MMA [35] , GCMMA [35] , GMMA [39] , GBMMA family [4]

As mentioned before, it is important to note that approximation functions of any other family can be added to such a scheme with minimal changes to the Optimization Lab. The only constraints for the approximation functions that are included in Eq. (4.2) are convexity and separability.

In order to capture the behaviour of $\mathcal{P}_{\text{NLP}}$ more accurately, we adjust the convexity – and therefore the asymptotes – of the contribution of each variable x_i to each response $g_j(\mathbf{X})$ separately. The idea is that by doing so, one can achieve a better local fit of $\widetilde{\mathcal{P}}_{\text{NLP}}$ on $\mathcal{P}_{\text{NLP}}$, as the former becomes more adaptive/flexible. As seen in Eq. (4.1), and since all approximations used herein are separable and compatible with MMA, one can write the resulting approximate function $\widetilde{g}_j(\mathbf{X})$, $\forall j$ of a mixed scheme as follows:

$$\begin{aligned}\widetilde{g}_j(\mathbf{X}) &= g_j(\mathbf{X}^{(k)}) + \sum_{i=1}^{n}\left\{p_{ij}^{(k)}\left(\frac{1}{U_{ij}^{(k)}-x_i}-\frac{1}{U_{ij}^{(k)}-x_i^{(k)}}\right)+q_{ij}^{(k)}\left(\frac{1}{x_i-L_{ij}^{(k)}}-\frac{1}{x_i^{(k)}-L_{ij}^{(k)}}\right)\right\}\\ &= r_j^{(k)} + \sum_{i=1}^{n}\widetilde{g}_{ij}(x_i)\end{aligned} \tag{4.2}$$

where, $\widetilde{g}_{ij}(x_i)$ is the contribution of variable x_i to response $\widetilde{g}_j(\mathbf{X})$. At each $\mathbf{X}^{(k)}$ we construct all the aforementioned approximations for all the functions involved independently of one another. This means that we have a set of $\widetilde{g}_{ij}(x_i)$ $\forall i, j$, for all the above approximations. Now, we must select the most 'appropriate' approximation for each term of $\widetilde{g}_{ij}(x_i)$ in order to construct a $\widetilde{g}_j(\mathbf{X})$ that resembles $g_j(\mathbf{X})$ as much as possible.

The selection criterion is based on information stored from previously visited points, see Section 3.3. In Fig. A.4b, one can see a flowchart of the implementation of AM schemes in the context of the Optimization Lab, and in Fig. A.8 one can see the flowchart of the approximation selection criterion for the generated FVA scheme. It goes without saying that this scheme is not the 'best' scheme for all the problems of the SO field. However, after building a library of diverse problems, see Chapter 5, our goal is to show that such a mixed scheme can outperform its constituent members for a large set of problems. Obviously, this means that the approximation selection criterion must be designed carefully.

4.5 A Hybrid scheme

Ideally, one would like exploit the benefits and suppress the weaknesses of the aforementioned scheme types. The goal is to use the adaptiveness of AM schemes, while maintaining the low computational cost of IM schemes. To this end, the aforementioned schemes can be combined to form a hybrid scheme (herein referred to as HY). Such a scheme can be designed as follows:

- Several AM (FA or FVA) schemes are designed and tested in several problems
- The most promising selection criteria are located
- These are then added to an efficient IM scheme
- The process is iterated several times

For example, one can see the outcome of this methodology in Fig. A.9. There, the flowchart of an HY scheme that is composed of the IM scheme of Fig. A.5 with the addition of the gray blocks of Fig. A.8 is presented. The use of these blocks was found to be beneficial for the problem set of Section 5.1. With this additional 'feature', one can estimate when a function $g_j(\mathbf{X})$ is nonmonotonous with respect to a variable x_i, and use a nonmonotonous function $\widetilde{g}_{ij}(x_i)$ to approximate its contribution. Our numerical experiments suggest that the additional computational cost is a small price to pay for the accompanying increase in approximation quality, see Section 5.3.

In general, every additional feature that is to be included in an HY scheme should be reviewed carefully. The benefits of including it must be weighted against the additional computational cost. Reasonably, only if the enhanced performance outweighs the increased cost including it is beneficial. A criterion that can be used to provide estimated values for this trade-off can be found in Eq. (5.4).

5 | Numerical examples

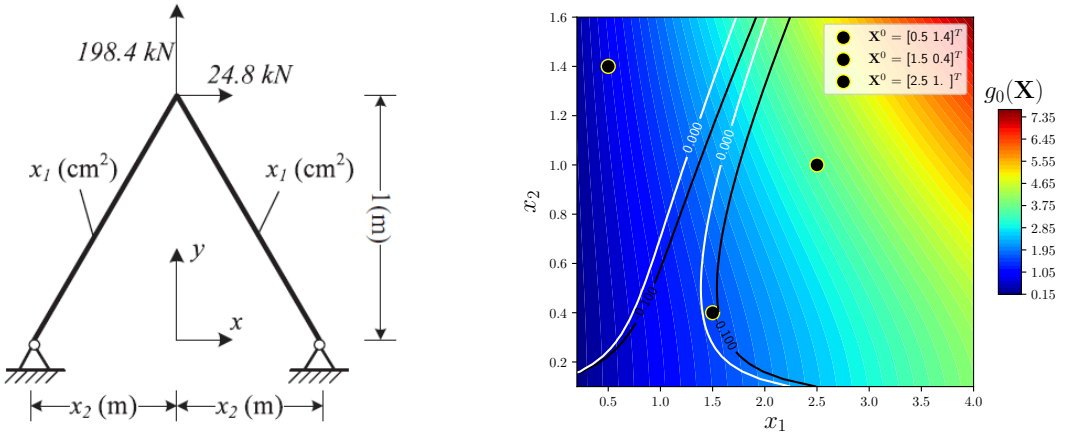
5.1 Problems used

The performance of any optimizer is problem-specific. Consequently, when an SO algorithm is evaluated in comparison to others, one cannot extrapolate the observed results on other – perhaps dissimilar – problems. There are several papers in the literature on optimization methods showing the difficulty of objective algorithm comparison, see [52]. Creating a mixed approximation scheme for an SAO setting and evaluating its performance is no exception. Consequently, a diverse library of problems that can simulate the scenarios such a scheme might be exposed to in practice must be used in the Optimization Lab of Appendix A. What is more, due to the complexity of large-scale TO problems we eventually intend to apply our schemes to, the problems used in the designing and evaluating phase of our mixed schemes must be chosen carefully. To this end, a set of 10 low-dimensional, hard to optimize, structural and analytical problems was chosen as a base for generating and evaluating our schemes. Reasonably, further research on the behaviour of mixed schemes in an SAO setting requires expanding the above set of problems. In a subsequent stage, any generated scheme must be tested in TO problems that are known to cause difficulties to the state-of-the-art approximation methods of Section 2.1.

In this section we present the aforementioned benchmark problems that were used as a base for the generation of our mixed schemes. The initial points that were used for all problems are displayed in Table B.1. It is important to note that although these problems exhibit useful characteristics we exploited while designing our schemes (e.g. structural response functions of trusses, high non-linearity, dissimilar response functions, etc.), they are by no means exhaustive. Thus, one can extend this library of problems by including problems that are relevant to SO and whose behaviour cannot be captured by the considered set.

2-bar truss (analytical)

This is the problem of Eq. (3.1) used throughout Chapter 3. It is comprised of analytical functions that are easy to implement, has been used by other researchers – which is important for validation purposes – and is low-dimensional, which is advantageous for the analysis associated with generating local plots, see Section 3.2.2.



(a) Configuration of 2-bar truss

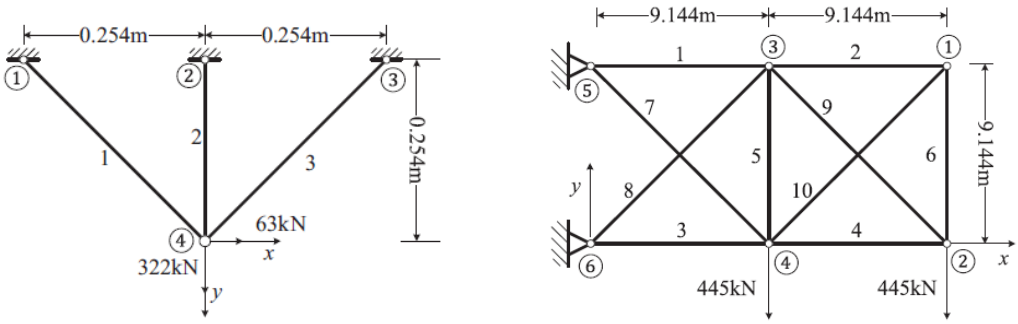
(b) Contour plot for $\mathcal{P}_{\text{NLP}}$ of Eq. (3.1)

Figure 5.1: 2-bar truss problem [12] with 3 different starting points

3-bar and 10-bar trusses (FEA)

Along the same lines, the next problems considered are truss configurations of 3-bars (Fig. 5.2a) and 10-bars (Fig. 5.2b) respectively. As before, the design variables to be optimized are the cross-sectional areas of the truss elements and the motivation for selecting them is similar to the 2-bar truss problem. Note however that for these problems

there is no analytical description of the exact functions of $\mathcal{P}_{\text{NLP}}$ and a linear FEA was implemented to evaluate $g_j(\mathbf{X}^{(k)})$. Moreover, the sensitivities $\frac{\partial g_j}{\partial x_i}\big|_{\mathbf{X}^{(k)}}$ were calculated by the finite difference method. By doing so, we aim to imitate the simulation-based SO optimization problems we ultimately address our method to. What is more, for truss configurations, the problem's dimension rapidly increases with the number of elements. To be specific, the characteristic size of the 3-bar truss reads as $n \cdot (m+1) = 18$, whereas for the 10-bar configuration it holds that $n \cdot (m+1) = 190$. Further details on these problems can be found in [12], wherein several other truss configurations are considered.

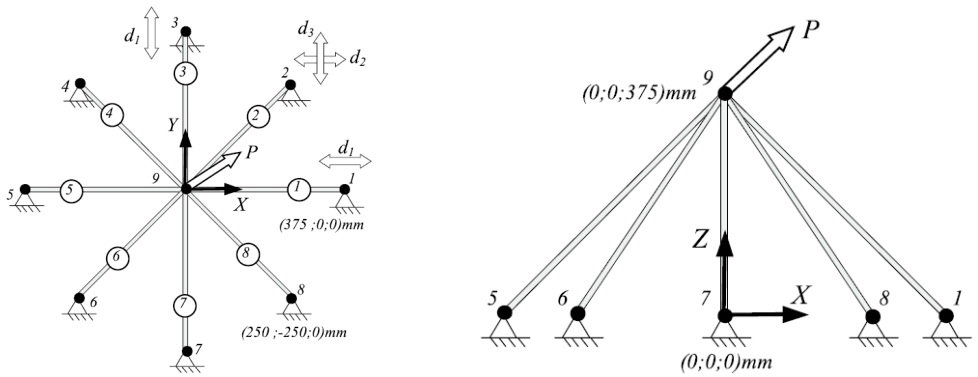


(a) 3-bar truss **(b)** 10-bar truss

Figure 5.2: Truss configurations solved by a linear FEA [12]

8-bar truss (FEA)

This truss configuration problem is inherently dissimilar to the ones considered above. The design variables under optimization are the nodal coordinates (i.e. $\mathbf{X} = [d_1\ d_2\ d_3]^T$ due to symmetry planes XZ, YZ) instead of the elements' cross-sectional areas. This means that the response functions involved (i.e. $g_j(\mathbf{X})$) will be significantly different to the ones of the aforementioned truss problems.



(a) Top view **(b)** Front view

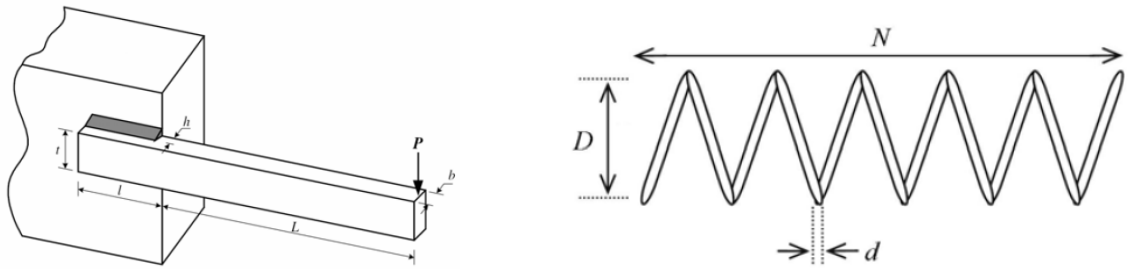
Figure 5.3: 8-bar truss configuration solved by a linear FEA [4]

Welded beam (analytical)

This is a highly non-linear low-dimensional analytical problem, wherein the geometry of a welded beam with a rectangular cross-sectional area is optimized, see Fig. 5.4a. The design variables to be optimized are the beam's length (L), height (t), width (b) and the thickness of weldment (h) under stress. The feasible domain is bounded by stress, displacement, load and geometric constraints, and the objective is to minimize its fabrication cost. A detailed description of this problem along with a list of all the parameters used can be found in [16].

Axial spring (analytical)

Similarly to the welded beam, the axial spring design problem is also inherently non-linear and low-dimensional. Here, the design variables to be optimized are the number of active coils (N), the mean coil diameter (D) and the wire diameter (d). The objective is to minimize its weight under deflection, stress and surge frequency constraints, see Fig. 5.4b. Further details on this problem, as well as several other non-linear optimization problems, can be found in [16].



(a) Welded beam design problem **(b)** Axial spring design problem

Figure 5.4: Analytical nonlinear problems of a welded beam (left) and an axial spring (right) [16]

Analytical benchmark functions

These are some of the most famous analytical benchmark functions for non-linear optimization. Important to note is that all the functions listed in Appendix B are shifted with respect to their design variables (i.e. $\mathbf{X} = [x_1\ x_2]$) to alleviate the optimizer from the numerical complications that arise when x_i changes sign. We justify this choice by the fact that the SO problems we address our method to include exclusively positive design variables (i.e. $x_i > 0\ ,\ \forall i$). The interested reader is referred to Appendix B for the exact formulations of these problems.

5.2 Influence of parameters

Achieving a fair comparison between different schemes is far from trivial. After selecting a diverse set of structural and benchmark problems to apply our schemes to, we must ensure that the observed results have a general character and are – to some extent – parameter insensitive. To this end, the influence of the involved parameters must be investigated and assessed. Thus, in the coming sections a parameter sweep for the initial point selection, the convergence criterion used and the tolerance that terminates the optimization is performed. By doing so, the value of the conclusions drawn from the results of Section 5.3 is increased.

Initial point

The selection of initial point influences greatly the performance of optimization algorithms. In general, its influence can be observed more clearly when gradient-based methods are used to solve problems with multiple local minima. There, one can expect that a different starting point might result in a different optimal point of the domain. Although all optimization methods used herein are gradient-based, the problems described in Section 5.1 do not necessarily have many local optima. Consequently, the different starting points must be chosen such that the influence of a local approximation function becomes apparent. Considering the above, 3 different types of initial points are chosen for each problem:

- **A point used by other researchers**: By comparing the performance of the implemented approximation methods with the results found in literature, one can confirm the correct implementation thereof. Obviously, this is only applicable to the problems of Section 5.1 that were used by other researchers. Nonetheless, validating the correct implementation of the approximation methods used is an important step.
- **A point near the optimum**: By choosing such a starting point, the oscillatory behaviour of an optimizer in the proximity of the optimum can be studied in detail, removing any influence from the path taken before reaching that region. Our numerical experiments have shown that most NM schemes fluctuate for several iterations before settling to a stationary point.

- **A point far from the optimum**: By starting the optimization far from the optimum the effect of different optimizers will become more clear. This is attributed to the fact that by increasing the number of iterations an algorithm takes to converge, the number of generated approximations do so as well, and the scenarios the optimizers face are more diverse. As a result, the possibility of creating approximations of low quality is increased and the robustness of an approximation scheme can be pushed to its limits.

In Table B.1 one can find all the initial points used in the present thesis, as well as the optima obtained for all the considered problems.

Convergence criteria

The use of different convergence criteria can also influence the comparison of optimizers. Thus, 4 different criteria are used for the implemented problems and the results are combined in the average performance profiles of the coming section:

- **Norm of KKT conditions**: See Eq. (1.4) and Fig. 3.3 for more information. This criterion reads as:

$$||KKT||_2 < \varepsilon \tag{5.1}$$

- **Norm of design change**: A feasible point (i.e. all constraints are satisfied) for which the design does not change anymore, see Eq. (3.6) and Fig. 3.2b. This criterion reads as:

$$\left|\left|\frac{\Delta \mathbf{X}^{(k)}}{\overline{\mathbf{X}} - \underline{\mathbf{X}}}\right|\right|_2 = \left|\left|\frac{\mathbf{X}^{(k)} - \mathbf{X}^{(k-1)}}{\overline{\mathbf{X}} - \underline{\mathbf{X}}}\right|\right|_2 < \varepsilon \tag{5.2}$$

- **Absolute value of objective change**: A feasible point for which the objective does not change anymore. This criterion reads as:

$$\left|\frac{\Delta g_0^{(k)}}{g_0^{(k)}}\right| = \left|\frac{g_0^{(k)} - g_0^{(k-1)}}{g_0^{(k)}}\right| < \varepsilon \tag{5.3}$$

- **All of the above**: Here, equations Eq. (5.1), Eq. (5.2) and Eq. (5.3) must be satisfied simultaneously to attain convergence. By using this criterion one can push each approximation scheme to its limits, since it is the hardest to satisfy. Therefore, if an approximation scheme is designed for robustness, this is the most appropriate criterion.

Moreover, a parameter sweep for the allowed convergence tolerance (i.e. ε) is also possible. In the present research, numerical experiments were conducted for $\varepsilon = \{10^{-3}, 10^{-4}, 10^{-5}\}$.

5.3 Results and discussion

The performance of different mixed schemes compared to the most prominent approximation methods of the SO literature can be seen in the figures below. By no means do we suggest that the generated methods are superior to all other approximations for any optimization problem one can formulate. Despite fine-tuning our schemes by using the problems of Section 5.1, the present report argues in favor of the feasibility of generating a mixed scheme that is superior to its constituent members. Although further research is necessary to improve the robustness of such schemes against the numerous detrimental scenarios any SAO algorithm encounters in practice, the versatility, efficiency and adaptiveness of mixed schemes are hard to ignore.

The current section is divided as follows: first, a performance profile is generated according to Section 3.6 for the initial points found in literature (whenever available). It is followed by a profile corresponding to starting points near the optimum, and another one for which the initial points are selected to be far from it. Lastly, an overall analysis is presented. For all starting points, a maximum threshold of 100 iterations is used as well as all the convergence criteria and termination tolerances seen in Section 5.2, increasing the validity of the present research. The performance measure Φ, see Section 3.6, used to generate these profiles is the number of iterations necessary to attain convergence for all the considered cases. For the types of problems that were used (i.e. low-dimensional problems with a small number of local minima), this is the most relevant performance measure. However, other options are possible if they are of interest (e.g. objective function value obtained). Furthermore, assuming a normal distribution of data points around their mean, error bars that correspond to a confidence level of 95% are given in all plots.

Initial points from literature

Fig. 5.5 shows the performance profile of the generated IM, HY and FVA schemes ('with' and 'without' a-priori knowledge of function types) in comparison to the most prominent approximation methods found in literature. It is clear that, although MMA and GMMA perform similarly for the set of initial points considered here, the novel mixed schemes perform better. This plot is essentially an average profile of data points generated by applying the 4 convergence criteria and the 3 corresponding termination tolerances found in Section 5.2. As expected, Fig. 5.5 suggests that – for the current selection of initial points – MMA and GMMA are the least robust methods. For GMMA, there are no significant improvements from the use of history information, as the 2 profiles cannot be distinguished with statistical confidence. The monotonous structure that characterises both MMA and GMMA can explain in part why for more than 30% of the considered problems they cannot attain convergence – even for large performance ratios. Although GMMA converges faster than MMA in (most of) the problems that they both converge, in terms of robustness, it appears to be marginally worse than MMA. Several comments found in literature about the overall – albeit slow – convergence of GCMMA can also be confirmed by Fig. 5.5. Although it is clearly the slowest algorithm, it achieves convergence for 80% of the implemented problems, surpassing beyond doubt all the other variants of the MMA family except for the mixed schemes. This is also to be expected, since it has been reported that the nonmonotonous GCMMA becomes increasingly conservative, resulting in unnecessarily small steps and thus more iterations [13].

Moreover, one can clearly see that the worst mixed scheme is the IM without information on the functions' structure, i.e. 'IMwithout'. This is to be expected, as it is the simplest mixed scheme found herein. It uses a simple combination of MMA and GCMMA (see Fig. A.6) and therefore cannot provide significant improvements. However, one can observe an improved performance compared to its constituent members. By comparing the dashed lines to their solid counterparts of the same colour in Fig. 5.5, one can also conclude that adding information on the functions' structure improves the performance of all mixed schemes. What is more, the FVA and HY schemes notably outperform all their competitors, with the 'HYwithout' even surpassing 'FVAwithout'. Although the difference is marginal and their respective error bars overlap greatly, this can be explained by the fact that changing the approximation function too often makes an approximation scheme prone to oscillatory behaviour, see Fig. 3.11. In other words, the methodology of enhancing IM schemes with criteria found by AM schemes (like FVA) appears to be successful for this set of initial points.

Initial points near the optimum

For initial points near the optimum the results are somewhat similar, as seen in Fig. 5.6. Most mixed schemes clearly outperform the state-of-the-art methods, whose performance is similar as before. More specifically, the trend in Fig. 5.6 is the following: MMA and GMMA have the worst performance by converging slower than their competitors for less than 70% of the considered problems and failing to converge in the remaining 30% of them. Next, GCMMA converges slightly faster than MMA and GMMA in most problems and attains convergence eventually in almost 90% of them. As in Fig. 5.5, the FVA scheme clearly outperforms its competitors (including the HY scheme this time) by having the top performance in 52% of the problems and eventually attaining convergence in more than 92% of them when no information is available (i.e. 'FVAwithout'). Obviously, adding information on the functions' structure further improves its performance by increasing these numbers to 82% and 98% respectively. However, as in Fig. 5.5, one must note that this additional information improves the performance of all mixed schemes.

Generally, in Fig. 5.6 the picture is somewhat similar to Fig. 5.5. The IM scheme is the worst mixed scheme, followed by an improved HY scheme and a clearly better FVA scheme. Compared to the previously used set of initial points, one can observe that the difference between the state-of-the-art methods and the mixed schemes is slightly smaller. Especially when no information on the functions' structure is available, i.e. solid curves. This is attributed to the fact that the effect of multiple iterations (and therefore approximations) adds up and amplifies the performance differences. Therefore, fewer iterations reduce the effect of a mixed scheme. However, amongst the mixed schemes seen in Fig. 5.6, the adaptiveness of FVA appears to be important in the proximity of the optimum, since the difference between the performance of FVA schemes and their HY and IM counterparts is intensified.

One can also look at this plot as an opportunity for improvements. Since the effect of the mixed schemes compared to the state-of-the-art methods is smaller for the case of unknown types of functions, one can estimate that this is the area where the schemes have the most room for improvements. Obviously, by no means does this mean that in all other areas they have reached their full potential. Nonetheless, the behaviour of a mixed scheme near the optimum is a good place to start.

Another interesting observation is the similarity Fig. 5.5 and Fig. 5.6. This effectively means that – to some extent –

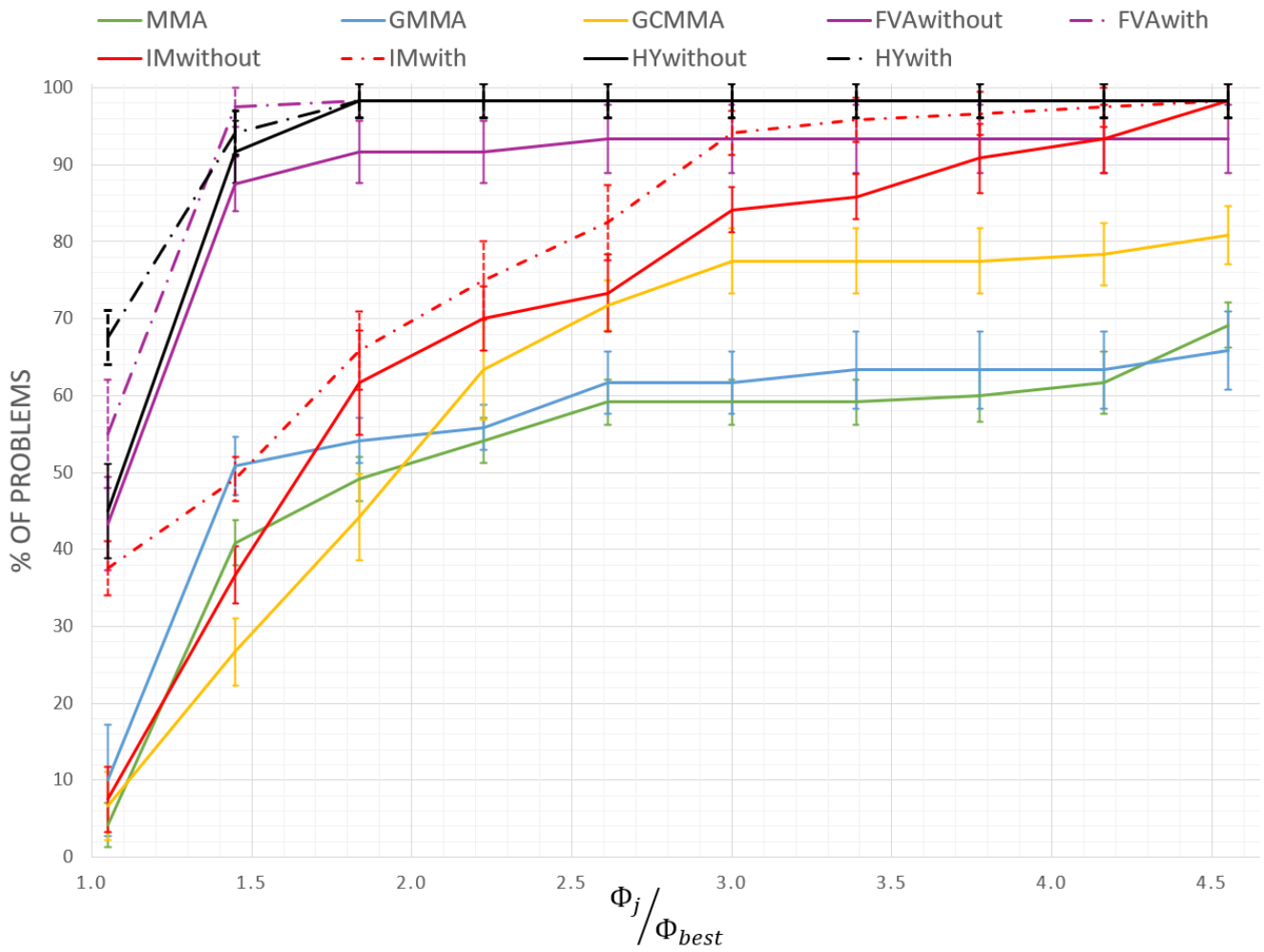


Figure 5.5: Average performance profiles of various SAO schemes for different convergence criteria and termination tolerances. **Starting points implemented in other research papers are used**. The error bars indicate a 95% confidence level, assuming a normal distribution of data points. Mixed schemes 'with' and 'without' information on the response functions' structure are compared.

the non-convergence of approximation schemes for several problems is insensitive to the selection of starting point. Since these methods do not converge even in the vicinity of the optimum, one can estimate that it is the 'structure' of the response functions involved in the aforementioned problems that determines convergence (or lack thereof). This is also confirmed by the addition of information on the functions' structure to the considered mixed schemes, see dashed curves. By doing so, the overall convergence is improved, indicating that a sub-optimal 'fit' of $\widetilde{g}_j(\mathbf{X})$ on $g_j(\mathbf{X})$ is responsible – to some extent – for the lack of convergence. Thus, the inability the most prominent NM schemes of the SAO field to capture the behaviour of different combinations of response functions near the optimum is confirmed.

Initial points far from the optimum

The performance profiles obtained for starting points far from the optimum are given in Fig. 5.7. In comparison to Fig. 5.6, it can be seen that the differences of the approximation methods under investigation are somewhat intensified (especially when no information on the functions' structure is available). As mentioned before, this is because of the increased number of iterations for all methods used. Once again the trend is similar, which is a strong indication of validity in the results. The IM scheme performs similarly to the state-of-the-art methods, the FVA scheme outperforms all others, and HY lies in between. As before, adding information on the type of the involved functions improves overall performance regardless of the scheme used.

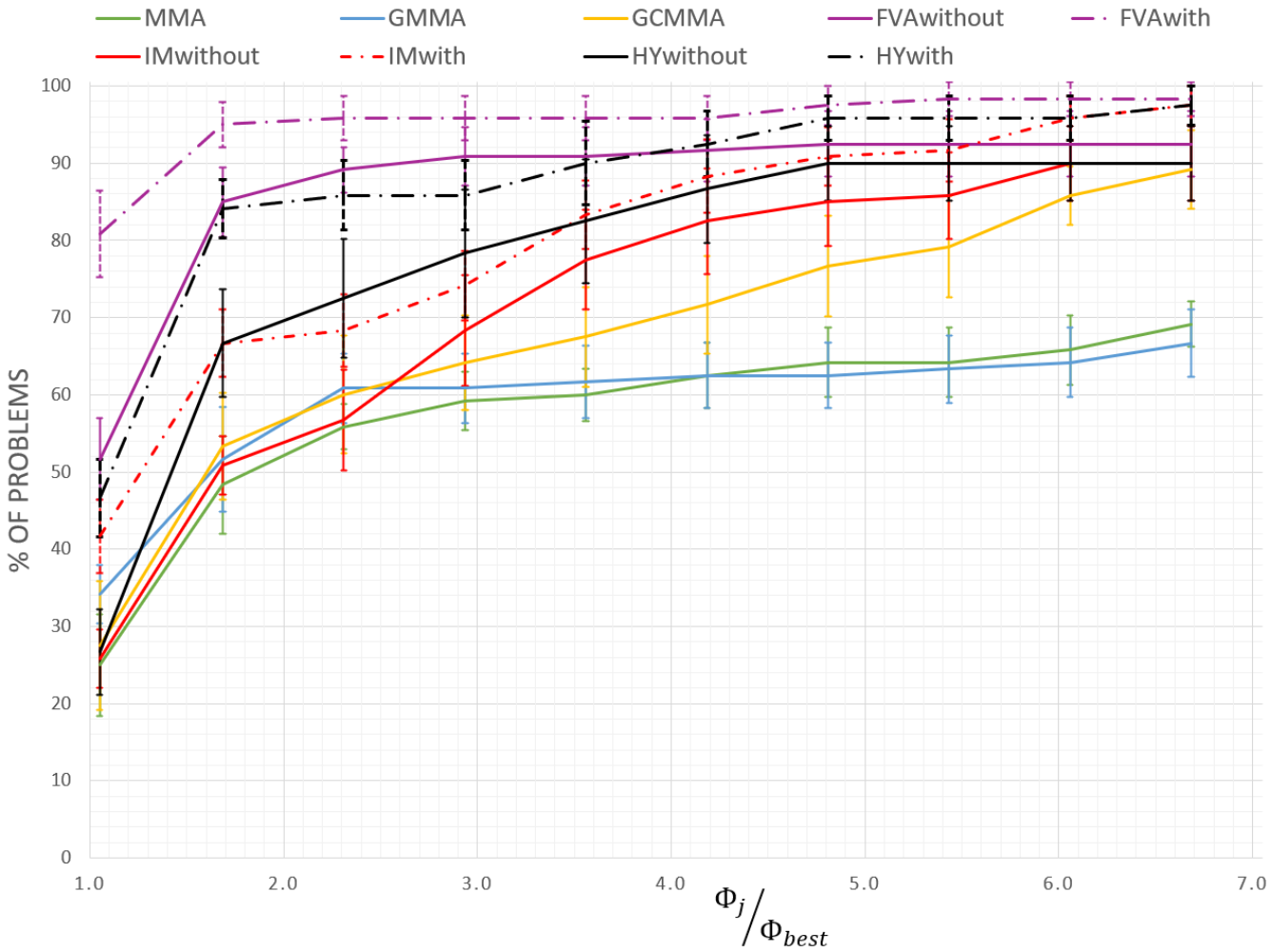


Figure 5.6: Average performance profiles of various SAO schemes for different convergence criteria and termination tolerances. **All starting points are selected near the optimum**. The error bars indicate a 95% confidence level, assuming a normal distribution of data points. Mixed schemes 'with' and 'without' information on the response functions' structure are compared.

Overall results

After sweeping a diverse set of starting points, using the most prominent convergence criteria one can encounter and applying several termination tolerances, the aggregated results are summarized in Fig. 5.8. There, all the different parameters of Section 5.2 are considered, as this plot summarizes 360 optimization runs that include:

- **10 approximation methods**: MMA, GMMA, GCMMA, IMwith, IMwithout, HYwith, HYwithout, FVAwith, FVAwithout
- **4 convergence criteria**: Eq. (5.2), Eq. (5.1), Eq. (5.3) and all of them simultaneously
- **3 sets of initial points**: A set used by other researchers, another set near the optimum and one far from it
- **3 termination tolerances**: $\varepsilon = \{10^{-3}, 10^{-4}, 10^{-5}\}$

This plot is effectively an average of all possible optimization settings and is the most general comparison plot of the present thesis. The corresponding error bars of $\pm 2\sigma$ result in a confidence level of 95%, assuming a normal distribution of data points, indicate that the mixed schemes confidently outperform the state-of-the-art methods for the considered problem set. The trend for all profiles is the similar to the figures above: FVA outperforms its competitors, followed by HY. Then, IM performs slightly better than GCMMA. Although GCMMA outperforms MMA and GMMA overall, in the problems that all of 3 of them converge, GCMMA is the slowest. This can be seen in low performance ratios, i.e. for $\Phi_j/\Phi_{best} \leq 2$. The performance of MMA and GMMA is very similar and cannot be distinguished with statistical

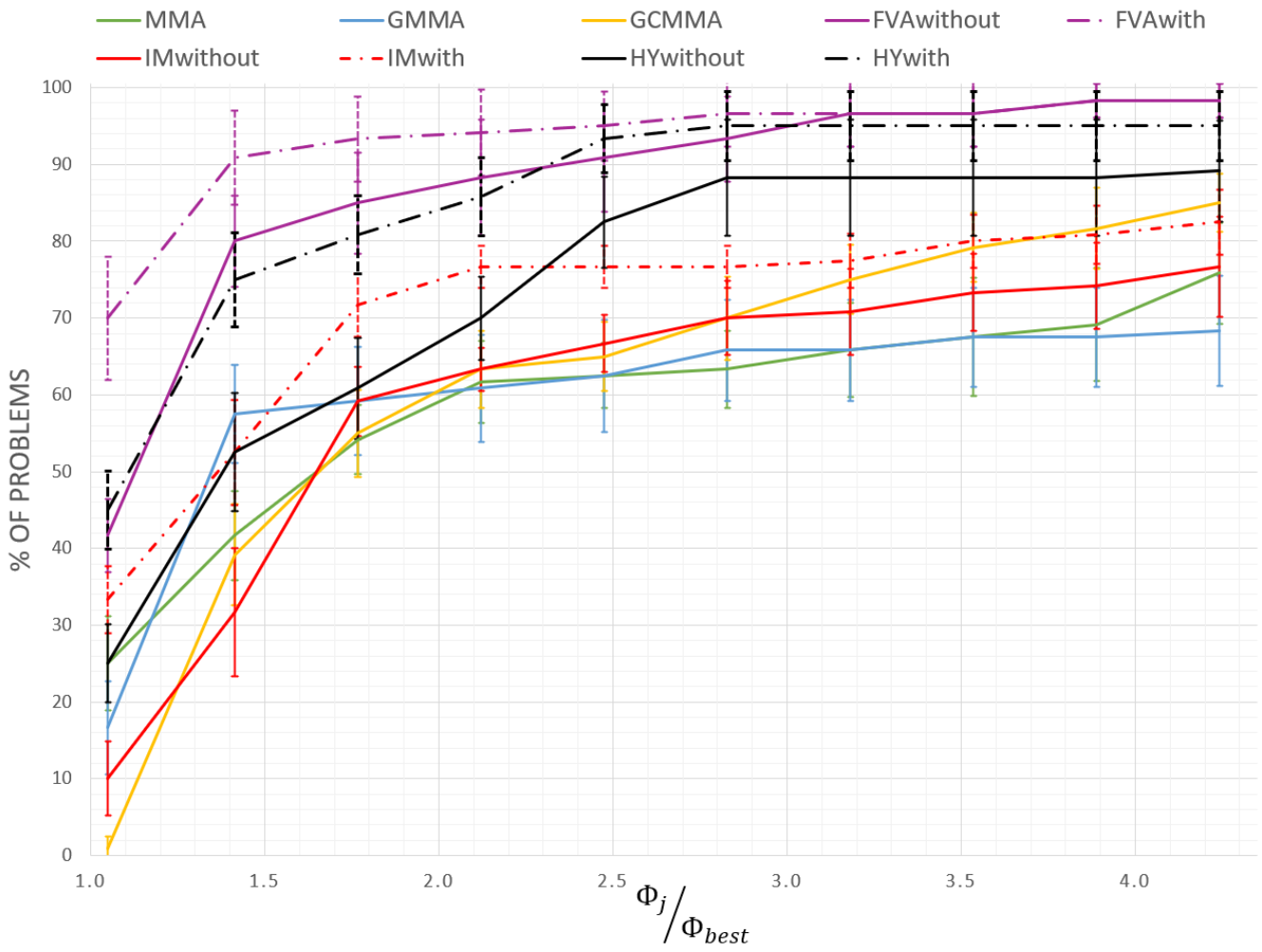


Figure 5.7: Average performance profiles of various SAO schemes for different convergence criteria and termination tolerances. **All starting points are selected far from the optimum**. The error bars indicate a 95% confidence level, assuming a normal distribution of data points. Mixed schemes 'with' and 'without' information on the response functions' structure are compared.

confidence. However, MMA appears to be slightly more robust and GMMA seems somewhat faster in the problems that they both converge.

Overall, one can see in Fig. 5.8 that (when no additional information is available on the functions involved) FVAwithout has the best performance for 46% of the considered problems and eventually converges in 95% of them for $\Phi_{\text{FVAwithout}}/\Phi_{best} \approx 3.7$. HYwithout comes next with the top performance in 32% of the problems and overall convergence in 92% of them for $\Phi_{\text{HYwithout}}/\Phi_{best} \approx 3.7$. IMwithout and GCMMA perform similarly by having the top performance in approximately 15% of the problems and eventually attaining convergence in 88% and 85% of them respectively for $\Phi_j/\Phi_{best} \approx 5.2$. Finally, although MMA and GMMA appear to be the optimal choice in approximately 20% of the problems, they can only converge in less than 70% of them. Thus, one can conclude that they have the worst overall performance – as far as robustness in concerned. Lastly, one can see in Fig. 5.8 that the performance of all mixed schemes is improved when the structure of (some of) the response functions is known, as all dashed profiles are shifted vertically by a notable amount compared to their solid counterparts.

Combining of all convergence criteria results in a single dataset, as in Fig. 5.8, gives a representation of the 'average performance' assuming a representative set of users would use a similar mix of criteria. In addition to such a profile, it is interesting to make a performance profile only for the strictest convergence criterion of Section 5.2. That is when convergence is achieved only if Eq. (5.1), Eq. (5.2) and Eq. (5.3) are satisfied simultaneously. In conclusion, by comparing Fig. 5.8 to Fig. 5.9 one can argue that the superiority of mixed schemes compared to their constituent

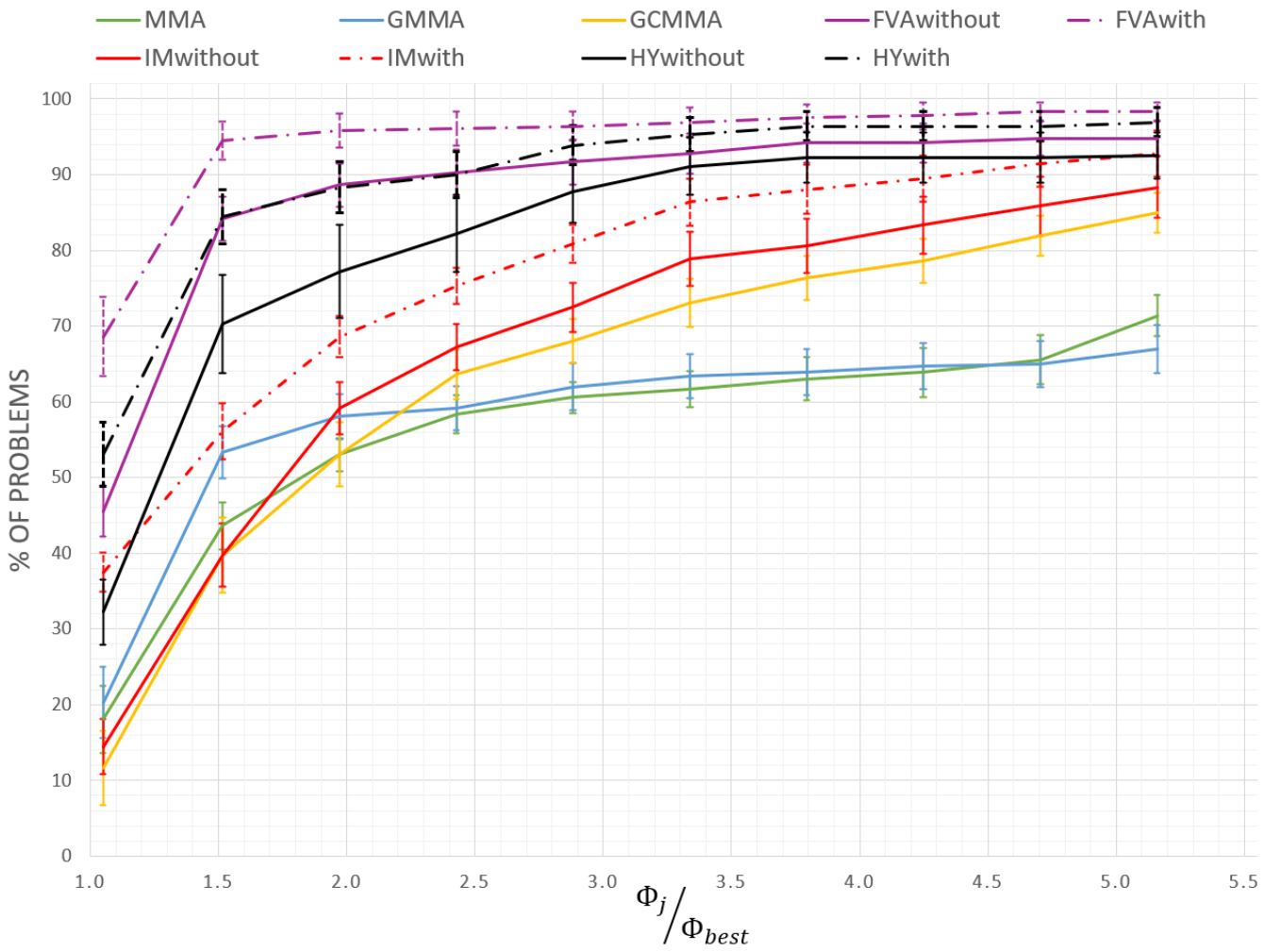


Figure 5.8: Average performance profiles of Fig. 5.5, Fig. 5.6 and Fig. 5.7. This figure considers **all possible combinations of initial points, convergence criteria and termination tolerances**. The error bars indicate a 95% confidence level, assuming a normal distribution of data points. Mixed schemes 'with' and 'without' information on the response functions' structure are compared.

members – as far as the considered problem set is concerned – is parameter insensitive.

Computational cost considerations

The implemented Optimization Lab of Appendix A is used to perform some measurements on the efficiency of the generated mixed schemes compared to the state-of-the-art methods of SAO. Since MMA is the reference method in the SAO literature, the measurements are normalized accordingly. In Fig. 5.10 one can clearly see that the improved performance mentioned above does not come without cost. As the characteristic size of a problem $n \cdot (m+1)$ increases, so does the time required by any approximation scheme to generate its approximation. However, the increase is significant and needs to be addressed. Although it is widely accepted within the SO community that, for the large-scale problems we ultimately aim to address, the computational cost of the approximation part is insignificant compared to the cost of simulation-based function evaluations (i.e. FEA), any generated scheme should be closely monitored with respect to the currently used methods in terms of approximation time spent per iteration. However, it is important to note that a large part of the computations conducted by AM schemes can be done in parallel since, at each iteration, all approximations can be generated simultaneously. This can reduce the elapsed real time (i.e. wall time) required for the approximation process of such schemes significantly.

What is more, one must understand to what extent enriching an approximation function with history information is cost effective. Taking into account the fact that researchers have reported that the wall time of the simulation-based

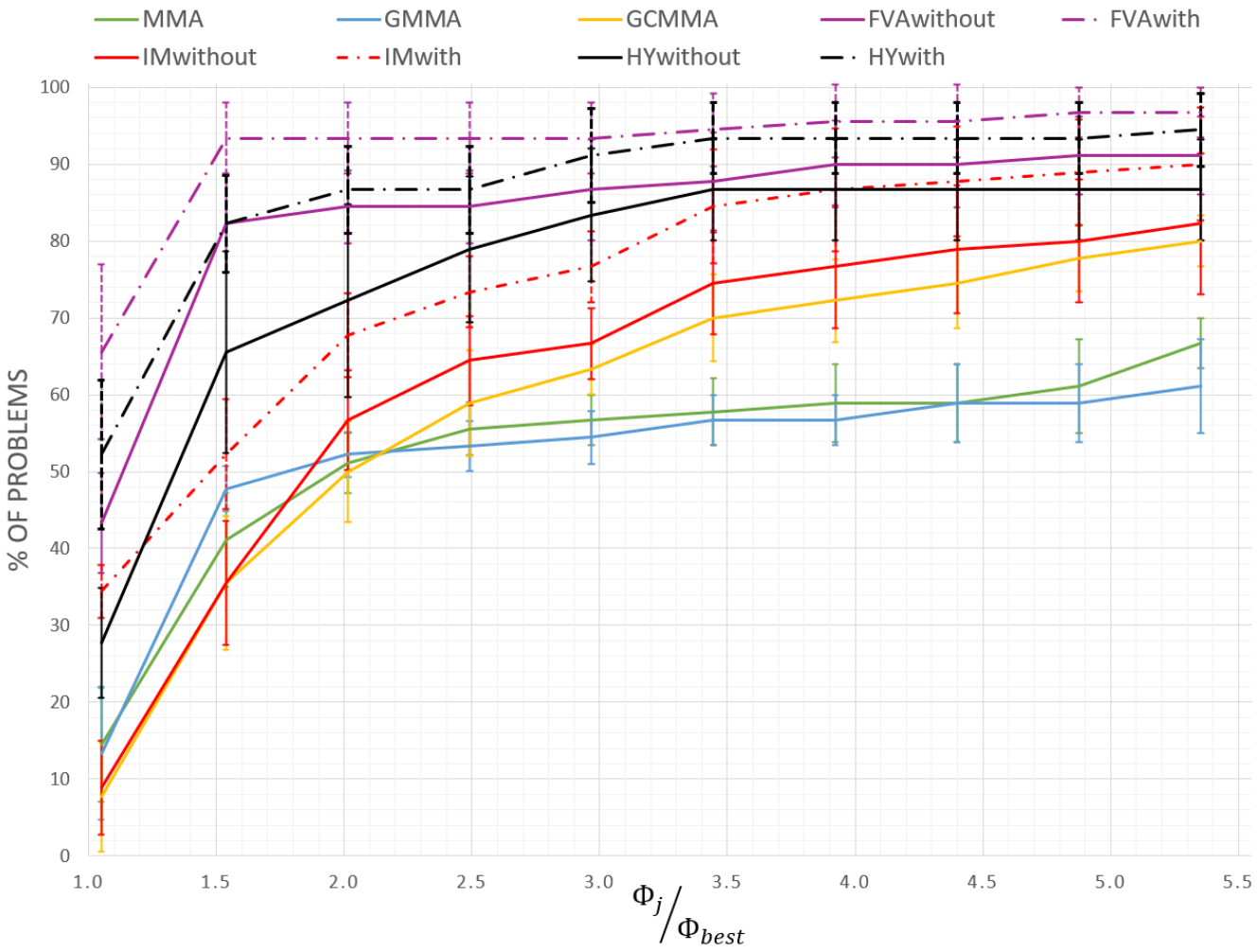


Figure 5.9: Average performance profiles of **all initial points and termination tolerances when the most strict convergence criterion is used**, see Section 5.2. This figure considers 9 different combinations of initial points and termination tolerances. The error bars indicate a 95% confidence level, assuming a normal distribution of data points. Mixed schemes 'with' and 'without' information on the response functions' structure are compared.

function evaluations, i.e. FEA, of large-scale TO problems takes approximately 97% of the total wall time of an SAO iteration [17], one can obtain a rough estimation of the viable region for the approximation enrichment. Under the assumptions seen below, the ratio of a mixed scheme's approximation wall time with respect to MMA's is given as a function of the respective ratio of the number of required iterations until convergence is attained.

$$\left.\begin{aligned} t_{tot} &= k \cdot t_{loop} \\ t_{loop} &\triangleq t_{fea} + t_{approx} \\ t_{fea} &\triangleq 0.97 \cdot t_{loop} \end{aligned}\right\} \Longrightarrow \frac{t_{mixed}}{t_{MMA}} \leq 33.33 \cdot \frac{1}{\frac{k_{mixed}}{k_{MMA}}} - 32.33 \tag{5.4}$$

As seen in Fig. 5.11, a larger reduction in the number of iterations by a mixed scheme will permit a larger increase in the wall time of its approximation process. As long as a scheme operates within the green area of Fig. 5.11, the total wall time is reduced and its use is beneficial. In any other case, using a mixed scheme is not cost effective, even for large-scale TO problems. As an example, one can compare Fig. 5.10 and Fig. 5.11 for a problem of $n \cdot (m+1) = 190$. For this case, FVA schemes must provide a minimum reduction in the number of iterations of around 40%, HY schemes must reduce it by approximately 25%, and IM schemes by 15%.

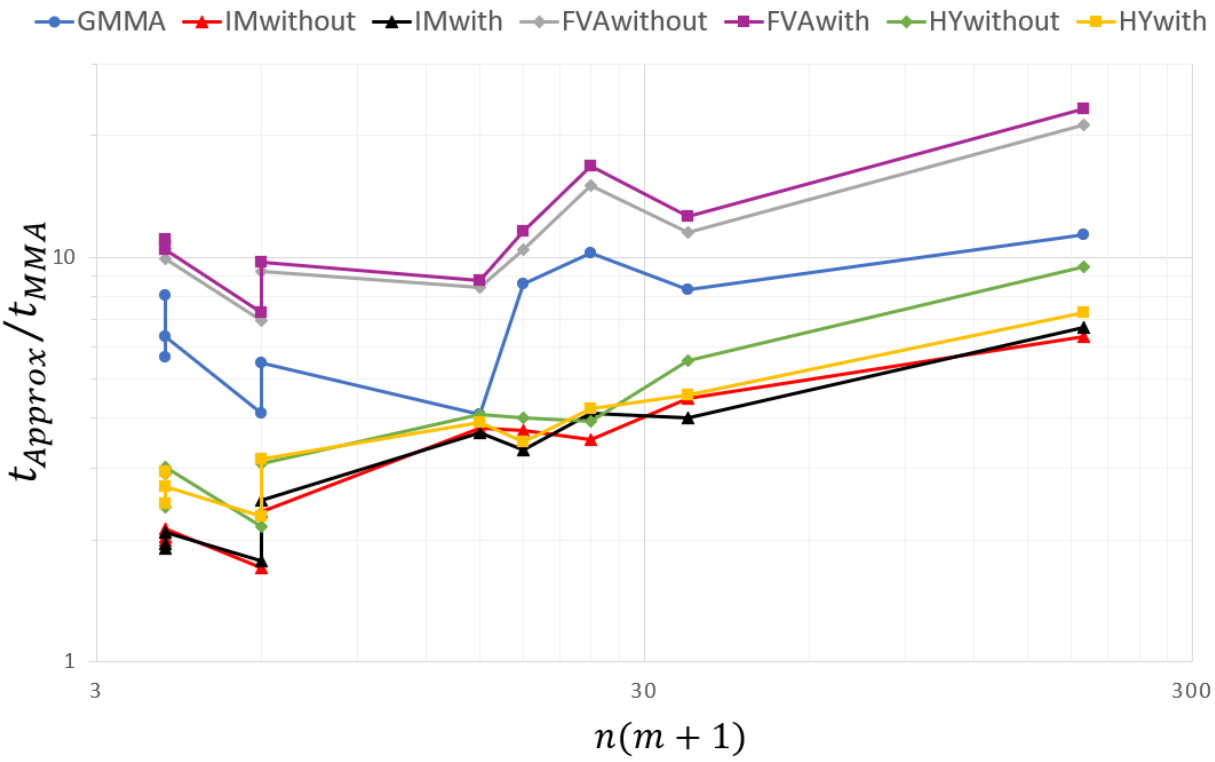


Figure 5.10: Elapsed real time (i.e. 'wall' time) of a single iteration for different approximation schemes. Measurements for problems of increasing characteristic size are normalized with respect to MMA.

Further discussion

The use of previous point information to enrich approximation functions needs to be addressed carefully. As an example, although GMMA can improve the approximation quality in some cases, the additional computational cost of the required Newton-Raphson scheme that accompanies the enrichment of the generated approximation, in combination with the instabilities caused by its potentially detrimental use, does not seem a fair trade-off. Likewise, our numerical experiments have shown that other MMA variants that use a similar approach (e.g. GBMMA3 and GBMMA4, see [4]) do not seem to provide adequate improvements compared to their additional computational cost. To make matters worse, this is expected to intensify as the number of design variables increases and more intricate problems are included. The computational cost of the Newton-Raphson scheme will increase greatly and the risk of misusing history information will be even higher, as the local 'structure' of a function will likely change more radically between consecutive iterations because of the combined influence of a large number of variables. However, when handled appropriately, exploiting information that is readily available from previously visited points can lead to faster convergence by improving the generated approximation quality.

It should be clear by now that the aim of the present thesis is not to find a mixed approximation scheme that will outperform any other for every problem. This is impossible. The goal is to prove that a mixed scheme – being mixed – can combine advantages of different methods and, if handled appropriately, suppress their weaknesses. We are confident that more research on mixed schemes will shed more light on the topic and will result in improved schemes that are able to outperform their constituent members for larger sets of problems, as there is plenty of room for further improvements.

Revisiting the inadequacies addressed in Section 1.4, the above results show that the complications that arise when response functions with dissimilar design variables are used can be handled successfully. Similarly, high quality function approximations of dissimilar response functions is also possible, due to the inherent adaptiveness that characterizes IM and AM mixed schemes (as well as their hybrid counterparts). These observations provide adequate motivation to continue the research on such adaptive SAO schemes by applying them to large-scale problems.

On the other hand, the performance of mixed schemes in problems that include design-dependent and aggregated

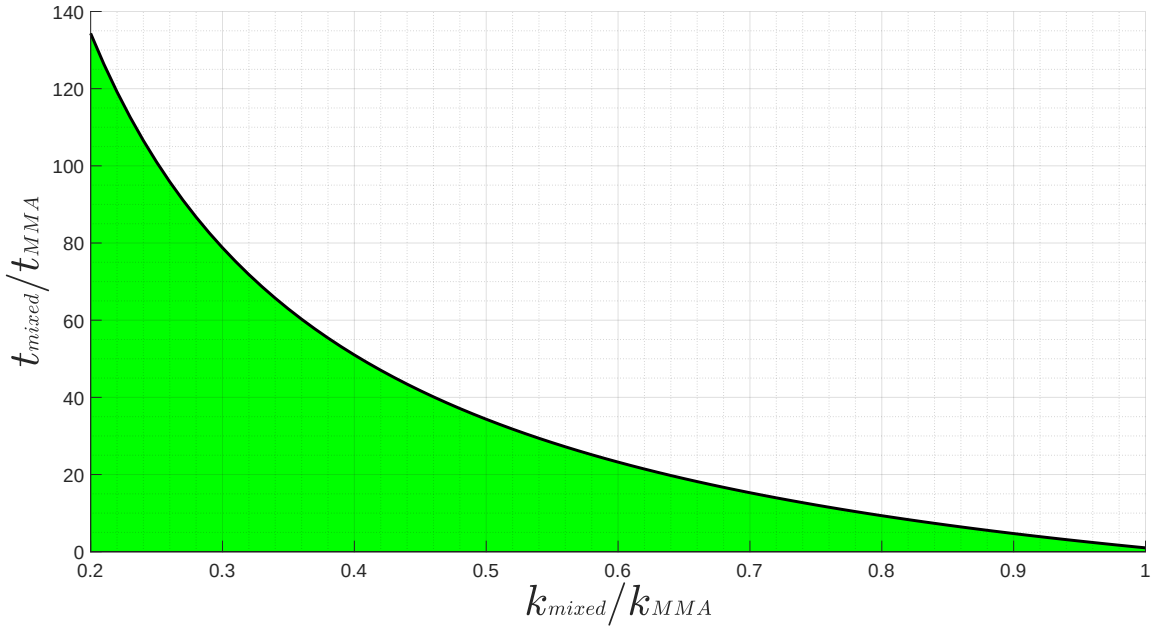


Figure 5.11: Area wherein approximation enrichment is beneficial (green area): The ratio of wall time for a mixed scheme relative to MMA is given as a function of the respective ratio for the number of iterations, assuming large-scale TO problems wherein $t_{fea} \approx 0.97 \cdot t_{loop}$ [17]

response functions could not be addressed in detail. Testing the generated schemes in such problems requires the enrichment of the implemented problem set with several large-scale TO problems. However, it is the author's belief that the adaptive character of mixed approximation schemes would be capable of 'fitting' much better/faster than the current approaches on the abruptly changing and highly nonlinear functions that the aforementioned problems typically involve. However, caution is needed not to abuse the adaptiveness of mixed schemes, as this could lead to oscillatory behaviour deriving from switching between approximation methods too often.

6 | Conclusions and future work

6.1 Conclusions

In an SAO setting, the approximation quality of the response functions involved affects significantly the total computational cost of the optimization process. As the cost of function evaluations increases (e.g. by addressing more complex and multi-disciplinary optimization problems), the benefits/consequences of a good/bad approximation are amplified.

The main objective of this research is to improve SAO algorithms by minimizing the number of 'expensive' function evaluations needed. This is possible by improving the generated function approximation quality (e.g. by enhancing an approximation with history information), which causes a reduction in the number of iterations that are necessary to attain convergence. In short, the results indicate that this is possible for a small set of low-dimensional – albeit challenging – problems.

More specifically, all the goals of the present research are reassessed below:

- **Review of current methods**: An extensive literature review was performed on the state-of-the-art SAO algorithms and an in-depth understanding of them was obtained on a fundamental level. It appears that MMA is the most widely used method and that its variants have not been tested yet in large-scale problems. Several other enriched Taylor-like expansions are available, none of which have been used extensively in practice. As far as mixed schemes are concerned, although the benefits of adaptive mixed schemes are self-evident, there has not been any significant breakthrough during the last decades. Thus, there is plenty of room for further research in that direction. Furthermore, although the solver was not under the scope of the present thesis, due to its inherent coupling with the approximation process, it requires some research as well. Although the literature on SAO methods seems to agree that purely dual solvers are the most efficient option, other alternatives (e.g. primal-dual interior point algorithms) have been used extensively without issues. This is important for the future of SAO methods, as purely dual solvers have strict prerequisites that limit the complexity – and therefore the information – one can include in an approximation function. Obviously, there is a trade-off between the efficiency of the solver and the amount of information it is capable of handling.
- **Locating and understanding their inadequacies**: In combination with the aforementioned literature survey, an implementation of several different problems made this possible. Namely, different types of design variables, diverse 'structures' of response functions, as well as mishandling history information were the most important causes for inadequate performance of the currently used methods.
- **Toolbox for mixed schemes**: A modular toolbox for classifying, generating, analyzing and evaluating fairly different adaptive structural optimization schemes was implemented. An efficient object-oriented architecture was realized, minimizing the effort needed to add further modules in the future. Different problems, approximations, solvers and convergence criteria can enrich this Optimization Lab without any significant changes to the current structure.
- **Generation of novel mixed schemes**: Apart from the implementation of the currently used methods, a framework for generating different types of adaptive approximation schemes was introduced, along with the necessary tools to perform an in-depth analysis on them. Several indices that offer great insight on a scheme's behaviour on a fundamental level are presented, thus facilitating the generation and oversight of robust and predictable mixed schemes. More specifically, 2 different types of mixed schemes were generated: IM schemes that can provide a different approximation per response function, and AM schemes, that can additionally change the approximation method of each response function as the optimization progresses. IM schemes are almost computationally equivalent to the currently used methods (e.g. MMA) but require prior knowledge of the functions' structure to be most efficient, whereas AM schemes are far more adaptive, their efficiency is – to some extent – decoupled from the use of a-priori knowledge of the functions' structure, but have an increased computational cost. Depending on the peculiarities of the addressed problem, the use of some scheme types may be advantageous over others. The use of hybrid schemes (referred to as HY schemes) is also an option that is investigated and compared with its generic counterparts.
- **Implementation of challenging problems to test the generated schemes**: A problem set of 10 low-dimensional – albeit challenging – problems were implemented to test all the methods used herein. Although

this goal was achieved, implementing some of the large-scale problems we ultimately address our method to was not realized. However, this will become possible in the near future, as the modular architecture of the aforementioned toolbox allows the expansion of the problem set that is being used to evaluate the methods under investigation.

- **Algorithm performance evaluation**: The most prominent performance evaluation method one can find in the literature on optimization algorithms was applied on the methods under comparison. For the aforementioned problem set, numerical experiments were performed, the influence of parameters involved was addressed and – to some extent – minimized. Different sets of starting points, termination criteria and allowed tolerances were used. The results were averaged in performance profile plots. To be specific, the numerical experiments conducted herein show that – on average – an FVA scheme outperforms the most prominent methods of the MMA family in 46% of the problems in terms iterations used, followed by the HY scheme that is the optimal method for 32% of the problems. The IM scheme, MMA, GMMA and GCMMA are the best choices for approximately 15% of the considered problems. As far as robustness is concerned, the FVA mixed scheme outperforms its competitors once again. Eventually, 95% of problems converge to the optimum when FVA is used. The HY and IM schemes come next by successfully solving 92% and 88% of the problems respectively, and GCMMA follows by doing so for 85% of them. MMA and GMMA are the least robust methods by converging to 72% and 68% of the problems respectively. Lastly, it was concluded that the performance of all mixed schemes is improved notably when information on the response functions' structure is available.
- **Viability of approximation enhancement**: The computational cost of using adaptive mixed schemes in an SAO framework was investigated. The cost reduction that accompanies the (hopefully) reduced number of iterations was compared to the increased approximation cost for large-scale systems. The result is an estimated area within which all generated schemes must lie in order to achieve a reduced overall optimization wall time. This is an important test, capable of estimating whether the use of an 'expensive' and adaptive mixed scheme is beneficial or not compared to the currently used methods.

Overall, the use of previous point information to enrich approximation functions needs to be addressed carefully. If it is used wisely, it can improve approximation quality and accelerate convergence. If not, it adds an unnecessary (and high) computational cost and creates instabilities. To make matters worse, this is expected to intensify as the number of design variables increases and more intricate problems are addressed.

In summary, throughout the present report a strong case in favor of mixed schemes compared to non-mixed ones emerges. Although the mixed schemes generated thus far are far from perfect, there are encouraging indications that an adaptive mixed scheme can outperform its constituent members for a large subset of any considered problem set. Finally, it should be clear at this point that designing a robust approximation selection criterion for any mixed scheme is crucial for its overall performance.

6.2 Future work

Further investigation on adaptive approximation schemes is necessary, as there is still plenty of room for improvements. Namely, the most important suggestions for future work can be seen below.

After confirming that, for a set of challenging low-dimensional problems, function approximations of high quality – achieved by adaptive mixed schemes – lead to a reduced number of iterations, and by extension to the number of function evaluations required, one must apply such schemes to high-dimensional problems. Given the ultimate aim of this research, the most important next step is to apply the generated schemes to TO problems. Although many features of the generated schemes offer great capabilities when applied to low-dimensional problems, their scalability is still questionable (e.g. FVA schemes). However, a large part of the computations conducted by AM schemes can be done in parallel, reducing the wall time of the approximation process to (hopefully) acceptable values. In short, a set of intricate TO problems must test the limitations of each type of mixed scheme. At this stage, it is of utmost importance to include such problems in order ensure compatibility with large-scale systems.

Since these schemes are designed for large-scale TO problems, one must make sure they are implemented efficiently. Although the current architecture is already efficient, larger dimensions will require further code optimization and may even necessitate a switch to compiled code. As we have already shown, relative efficiency graphs are very useful to visualize the scalability of new schemes (or lack thereof). After applying such schemes to large-scale problems, the cost of the approximation process itself must be profiled with respect to the total cost of the SAO algorithm. Only

after obtaining a detailed insight on each scheme's computational cost can one conclude on the problems it can address efficiently.

Moreover, the Optimization Lab can be enriched. More approximations, solvers and performance indices can be included. The performance profile generation can also be automated. By doing so, testing schemes in large problem sets will produce interpretable results much faster.

The numerical experiments showed that near the optimum all methods used herein – including the generated mixed schemes – can be improved. A possible direction could be to create a local response surface by fitting certain parameters. Although traditional fitting tools are prohibitively expensive (e.g. a quadratic function in n-dimensional design space requires $(n+2)\cdot(n+1)/2$ data points to fit the corresponding coefficients) it would be interesting to see if simplified curve fitting tools – in combination with dimensionality reduction techniques – could be used. For example, variables at their bounds would be irrelevant to such a fit, as they would remain constant for all the data points considered. This fitting could be assisted by using the stored information from the previously visited points. Nonetheless, the computational cost of such a procedure must be evaluated with respect to the benefits of using it.

Last but not least, new approximation selection criteria must be generated, tested and evaluated for all types of approximation schemes. It is certain that including more problems will lead to inadequacies of the current schemes and upgrades will be necessary. Therefore, this step must be reiterated several times before one can claim to have generated a robust mixed scheme for large-scale SO problems. It is the author's belief that when the characteristic size of the addressed problems increases by several orders of magnitude, the approach must include some machine learning. A very large amount of data will be generated, the necessary supervision model will be quite simple, and human oversight by aggregated indices will be insufficient. Training an algorithm to monitor several indices that can represent the behaviour of an approximation function on a fundamental level will be much more effective when scaled to higher dimensions. After all, most cases that share similar characteristics (i.e. an approximation selection criterion can be seen as a classification problem) are already being solved much more effectively by machines than humans.

A | Optimization Lab architecture

This appendix describes the structure of the software library we herein refer to as 'Optimization Lab'. All different modules and their respective inputs/outputs are given, as well as a detailed list of the library's features. In Fig. A.1 one can find a flowchart of the basic modules used in an optimization run.

General layout

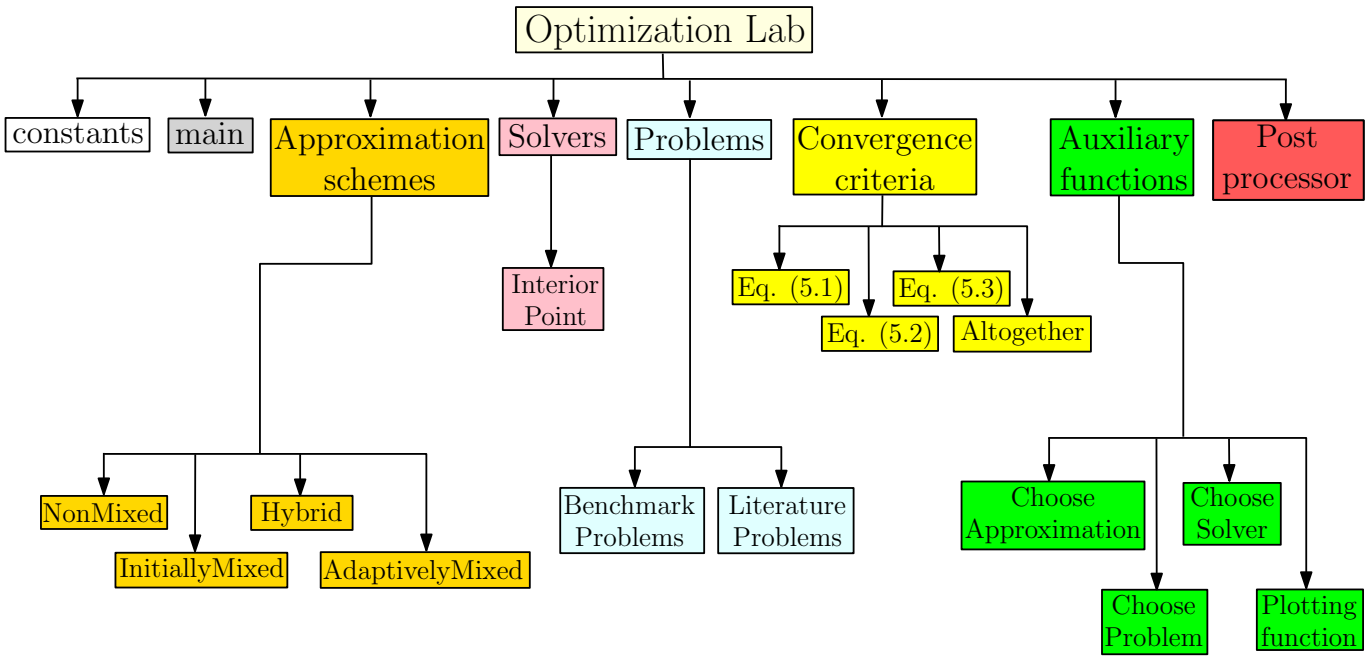


Figure A.1: A color-coded general layout of the Optimization Lab

More specifically, the different modules of the above figure are described below:

- **constants**
  This file contains all the constants used throughout the Optimization Lab. Lists of names for problems, approximations and solvers, along with several flags that determine the optimization options can be found herein.
- **main(problem list, approximation list, solver, criterion)**
  This is the main function from which all other modules/functions are called. It takes as arguments a list of names for the problems to be solved, the solver to be used, the termination criterion to stop the optimization and the list of approximations under comparison. A detailed flowchart that explains its functionality can be seen in Fig. A.2.
- **Approximation schemes**
  This directory contains all the implemented approximation schemes. Each scheme is implemented as a separate class, and more advanced functions/schemes (e.g. GMMA) inherit attributes and methods from their generic counterparts (e.g. MMA). More details on the structure of approximation schemes can be seen in Fig. A.3c.
- **Solvers**
  This directory contains all the available solvers. At the time of writing this report, only a primal-dual interior-point algorithm has been implemented. In the future, we hope the influence of the solver can be investigated as well by performing a comparison of the most prominent ones, as done with approximation schemes. More details on the structure of solvers can be seen in Fig. A.3b.
- **Problems**
  Thus far, only test problems and the problems of Section 5.1 are included in the library. In addition, a linear

truss solver is also included herein, in order to define and solve any 2D or 3D truss problem one can conceive. Any problem compatible with Fig. A.3a can be added to and solved by the Optimization Lab. It is of utmost importance to extend this module with additional problems in order to increase further the validity of the present research.

- **Convergence criteria**
  A class of convergence criteria is defined according to Section 5.2. This class includes methods that return the value of the different termination criteria. This means that one can select any termination criterion without any changes to the main function.
- **Auxiliary functions**
  This directory contains the following:
  - *Choose approximation*: A function that selects and initializes a user-specified approximation scheme.
  - *Choose problem*: A function that selects and initializes a user-specified problem.
  - *Choose solver*: A function that selects and initializes a user-specified solver.
  - *Plotting function*: A class of functions that contains all the possible plots one can generate with the Optimization Lab. A list of (some of) the available options can be found in Section 3.1 and some examples in Section 3.2.
- **Post processor**
  The class of the PostProcessor stores all the necessary information one may need to perform an in-depth analysis of the results after the optimization process ends. Obviously, depending on the characteristic size of the addressed problem, different options apply. For example, a low-dimensional problem can be subject to the local plots of Section 3.2.2, whereas a large-scale problem cannot. This is taken into consideration automatically by using certain flags defined in the *constants* file.

Module flowcharts

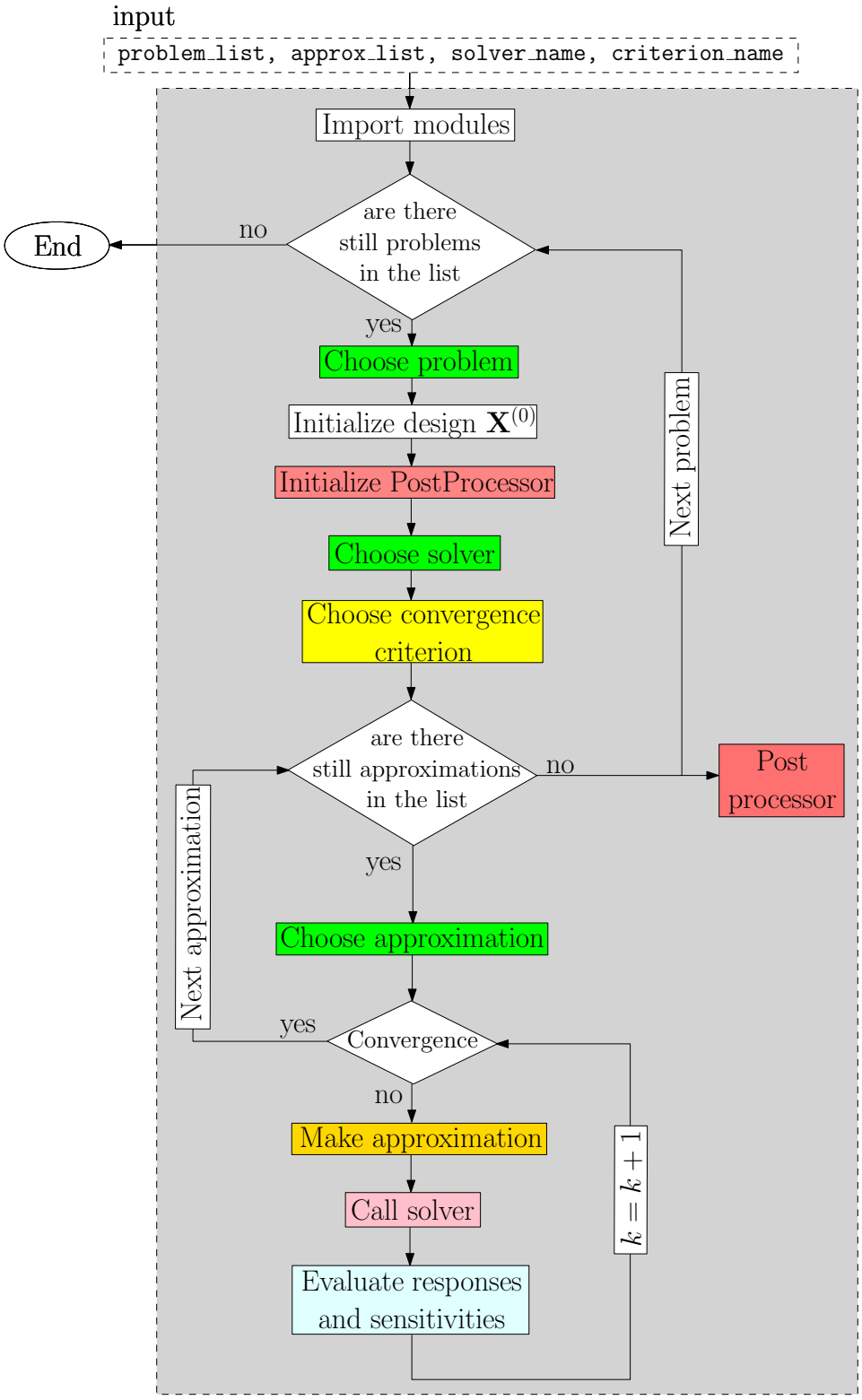
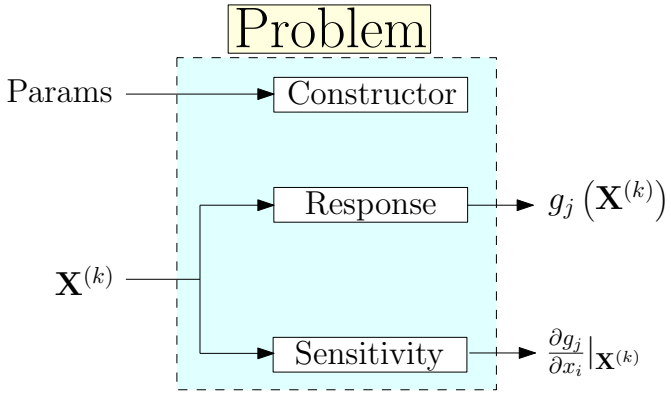
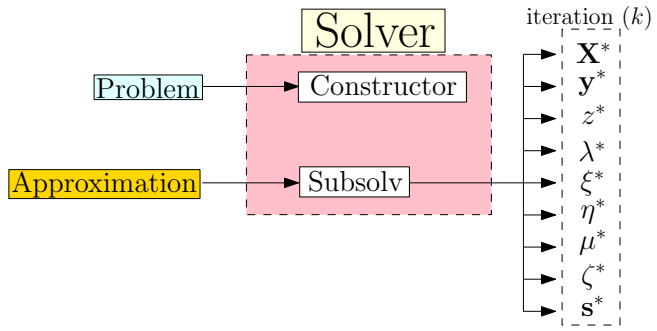


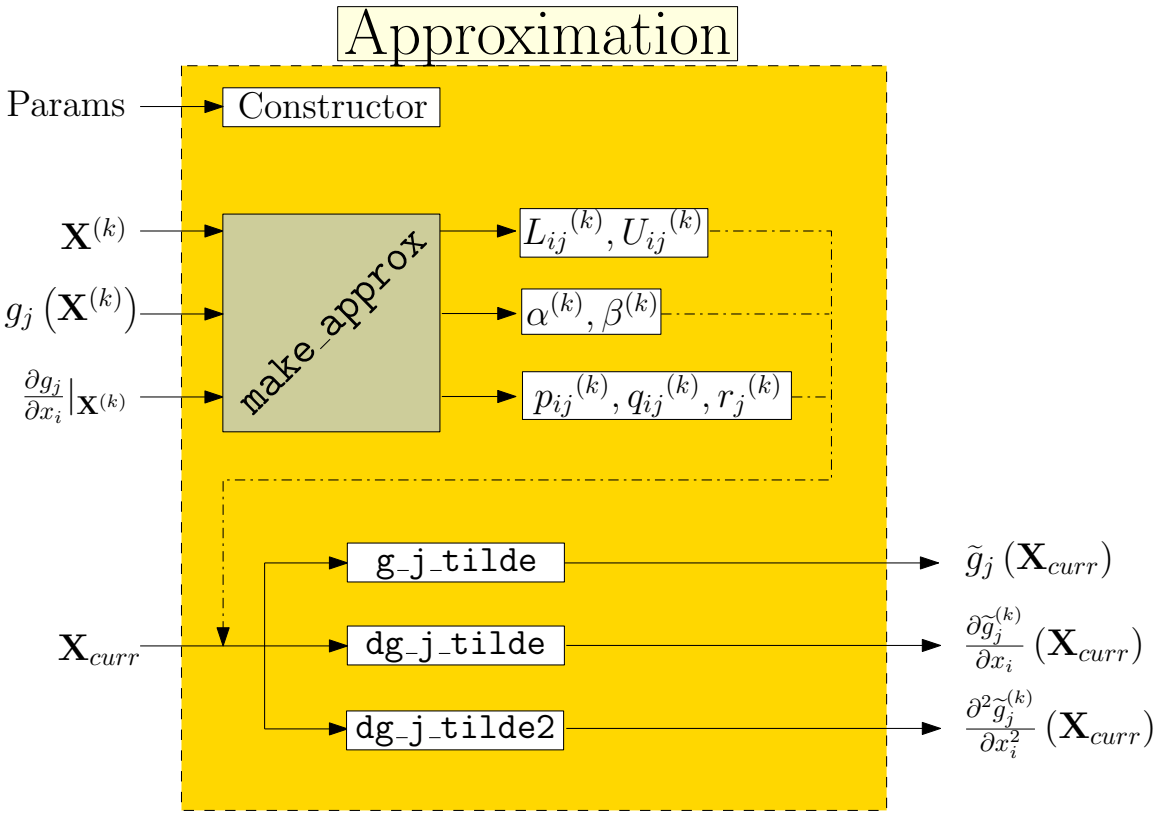
Figure A.2: A flowchart of the main function of the Optimization Lab: this is the function from which all other functions/modules are called. Color-coded in accordance with Fig. A.1.



(a) Problem module

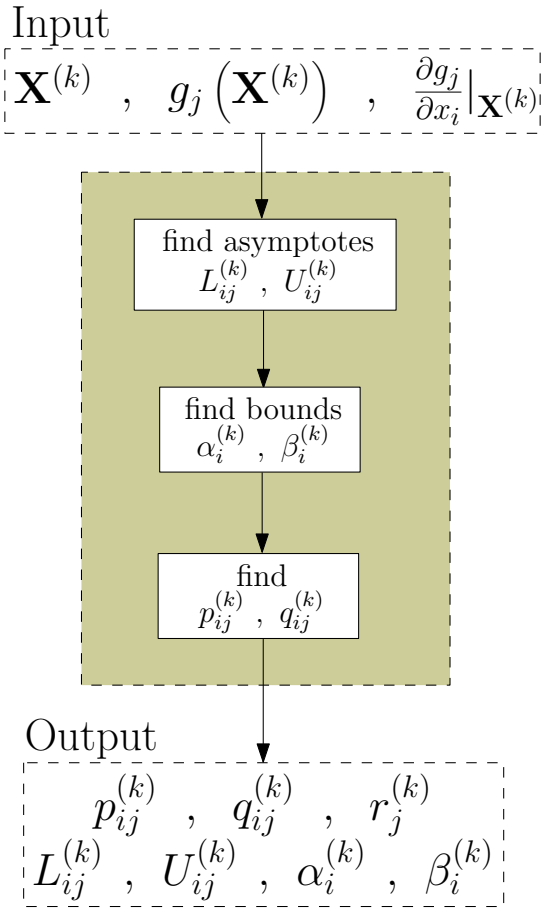


(b) Solver module

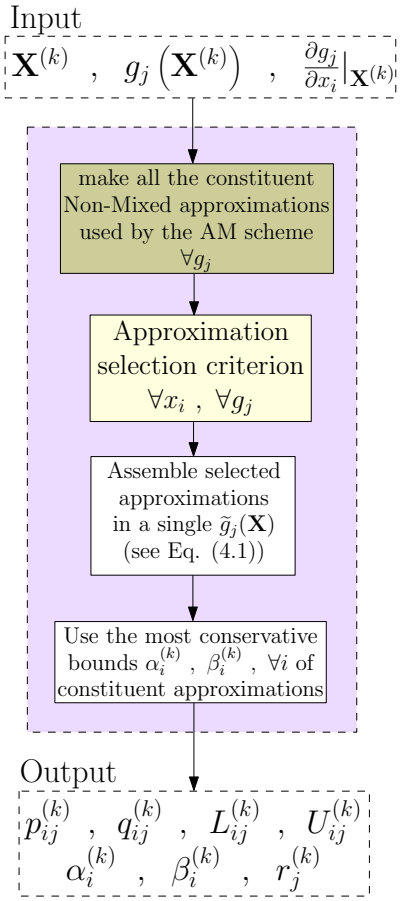


(c) Approximation module

Figure A.3: The architecture of the most important modules used in the Optimization Lab. Color-coded in accordance with Fig. A.1.



(a) A Non-Mixed scheme: The brown area represents the method `make_approx` of Fig. A.3c when Non-Mixed schemes are selected



(b) An Adaptively-Mixed scheme: The purple area represents the method `make_approx` of Fig. A.3c that overrides Fig. A.4a when Adaptively-Mixed schemes are selected. The brown and yellow areas in this figure is equivalent to the ones found in Fig. A.4a and Fig. A.8 respectively.

Figure A.4: Implementation of a Non-Mixed scheme of the MMA family (left) and an Adaptively-Mixed one (right)

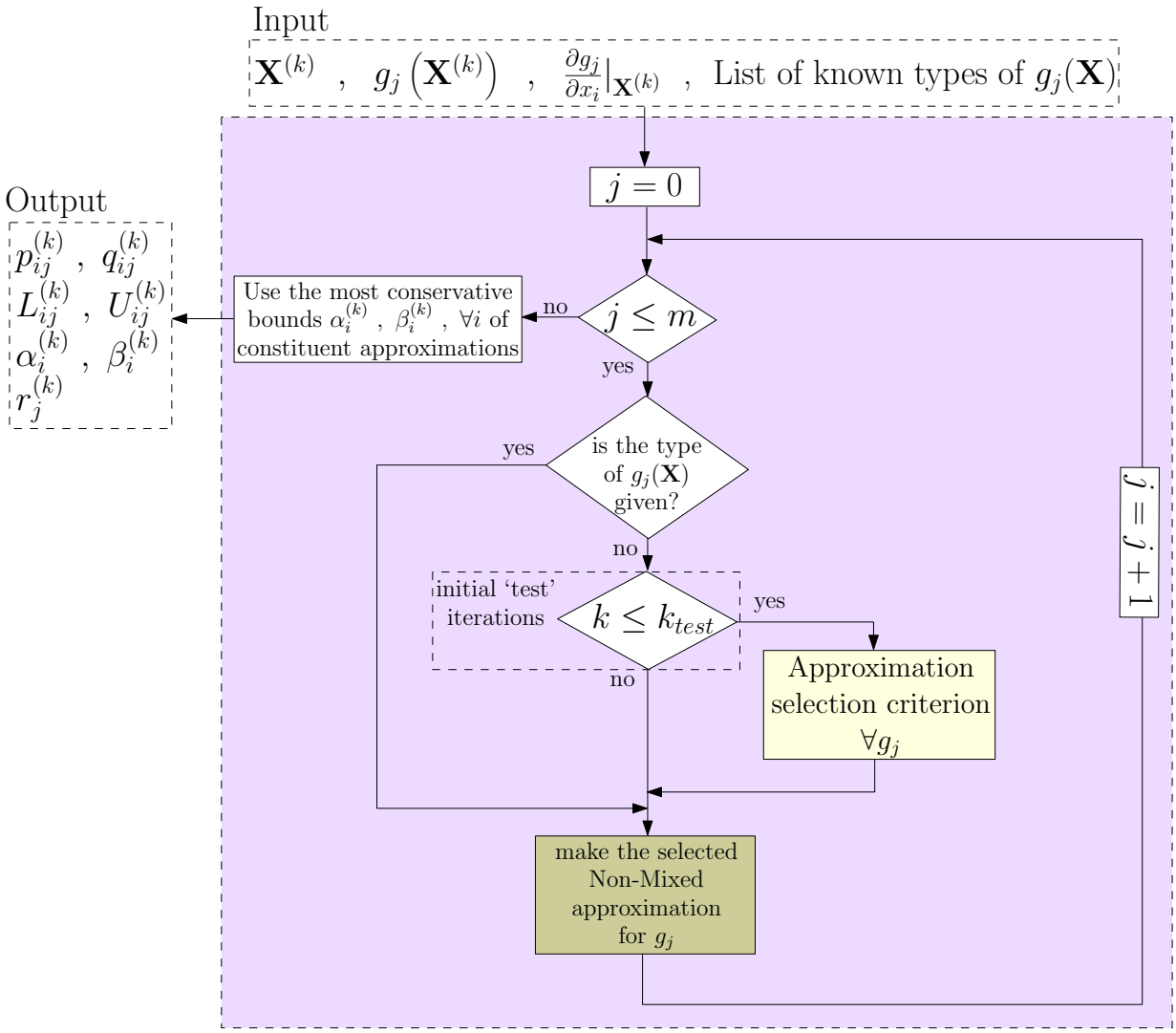


Figure A.5: Implementation of an Initially-Mixed scheme in the Optimization Lab. The purple area represents the method `make_approx` of Fig. A.3c that overrides the one seen in Fig. A.4a when Initially-Mixed schemes are selected. The brown area in this figure is equivalent to the one in Fig. A.4a, and the yellow one corresponds to Fig. A.6.

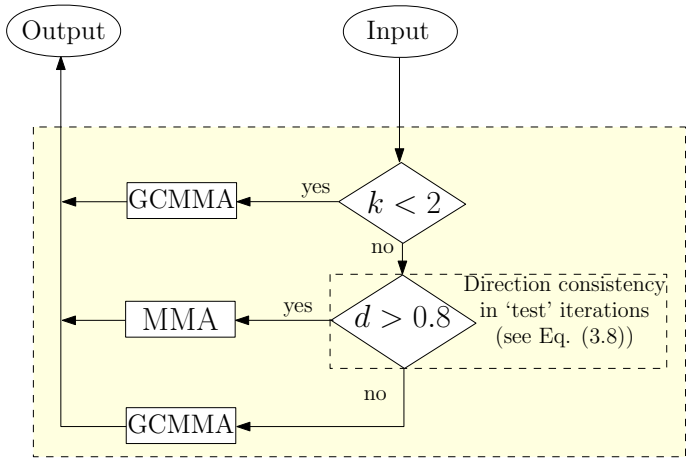
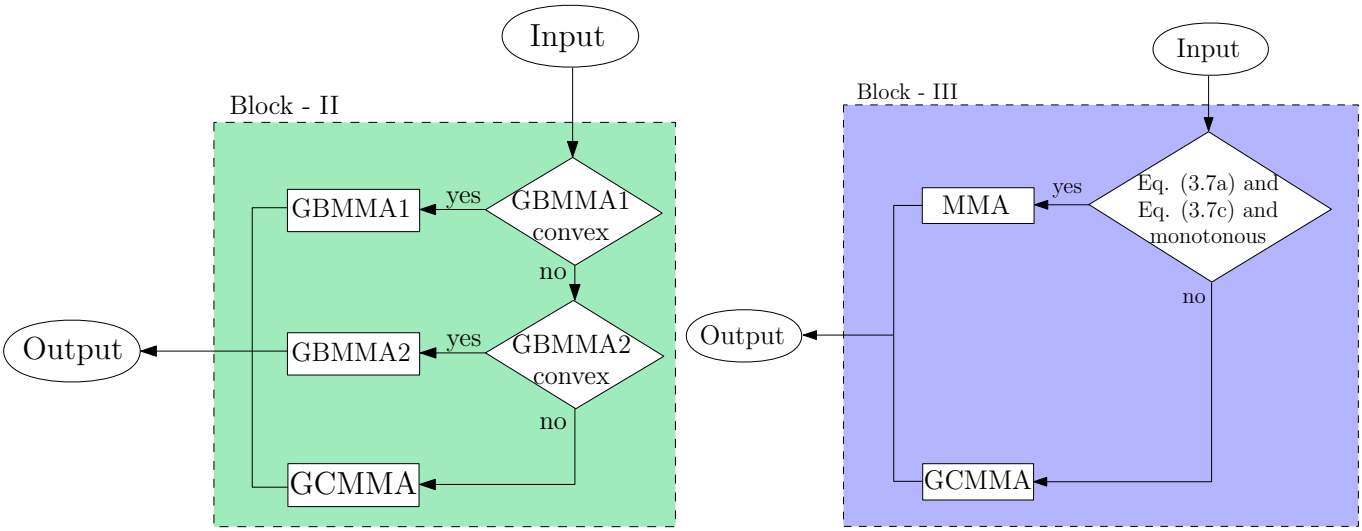


Figure A.6: Flowchart of an approximation selection criterion for a novel IM scheme. The light yellow area in this figure is equivalent to the one in Fig. A.5



(a) Block II: Applied to non-monotonous pairs of $\{x_i, g_j\}$ [13]

(b) Block III: Applied when previous point is not usable

Figure A.7: Detailed view of blocks II (left) and III (right) found in Fig. A.8

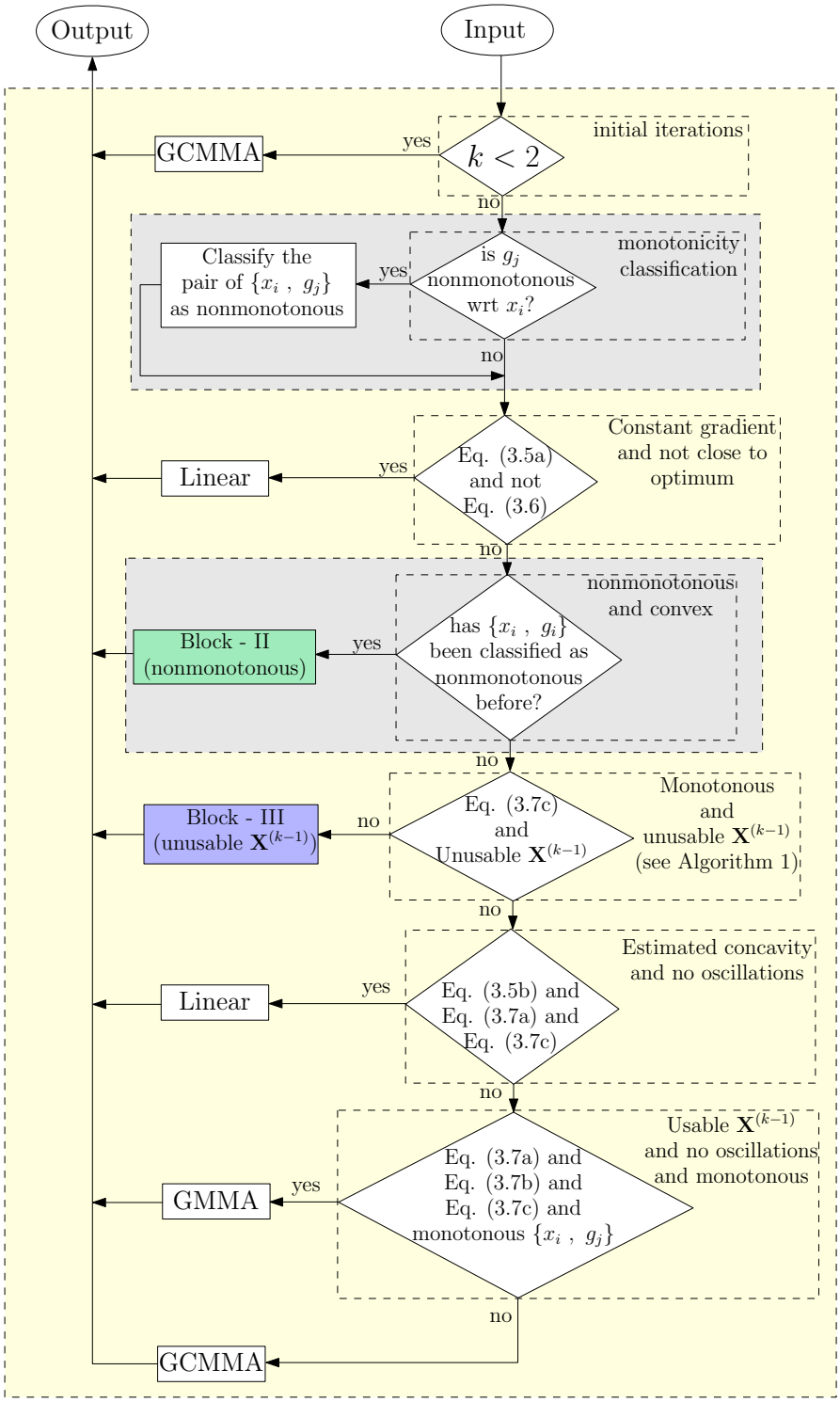


Figure A.8: Flowchart of an approximation selection criterion for a novel FVA scheme. The yellow area in this figure is equivalent to the one in Fig. A.4b, and the green and blue blocks can be seen in more detail in Fig. A.7

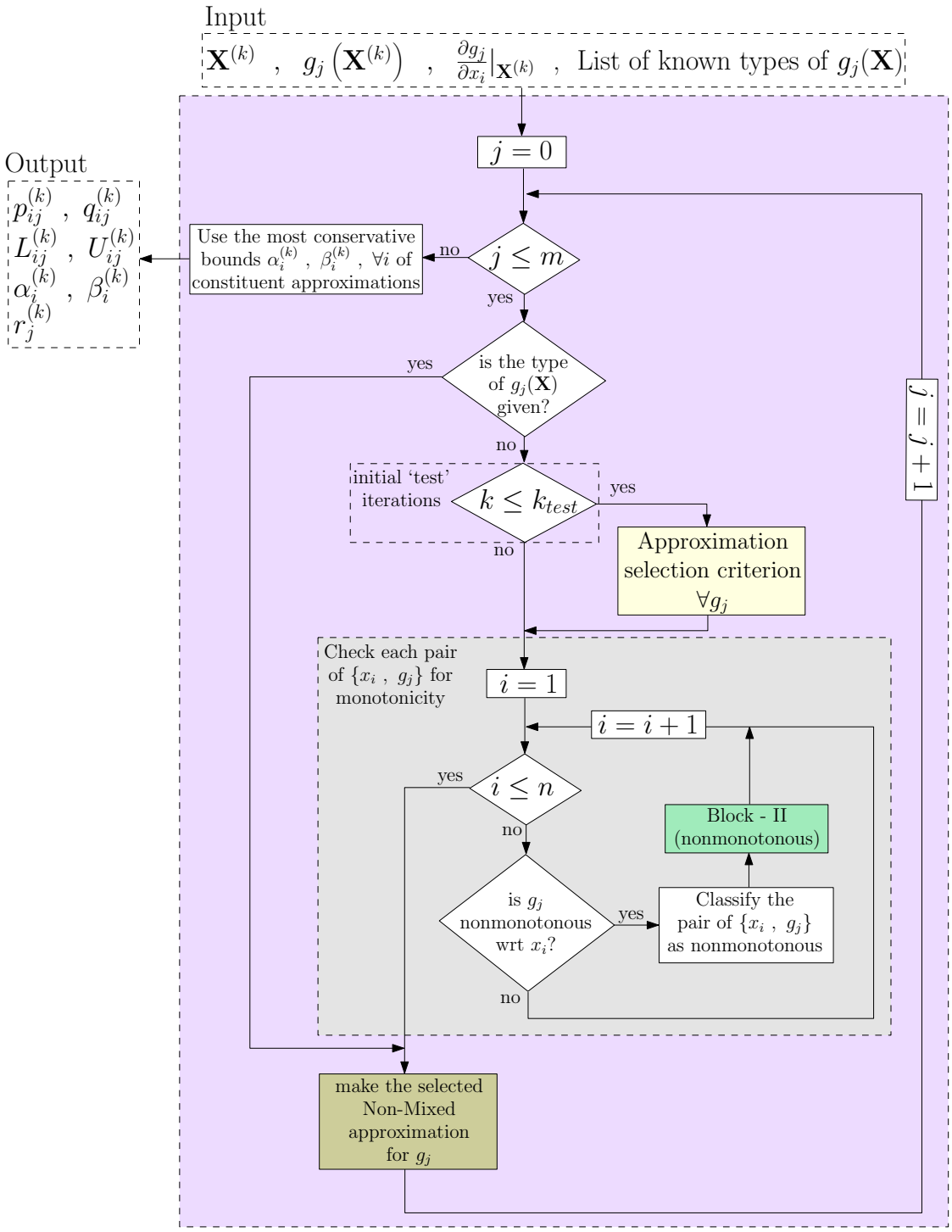


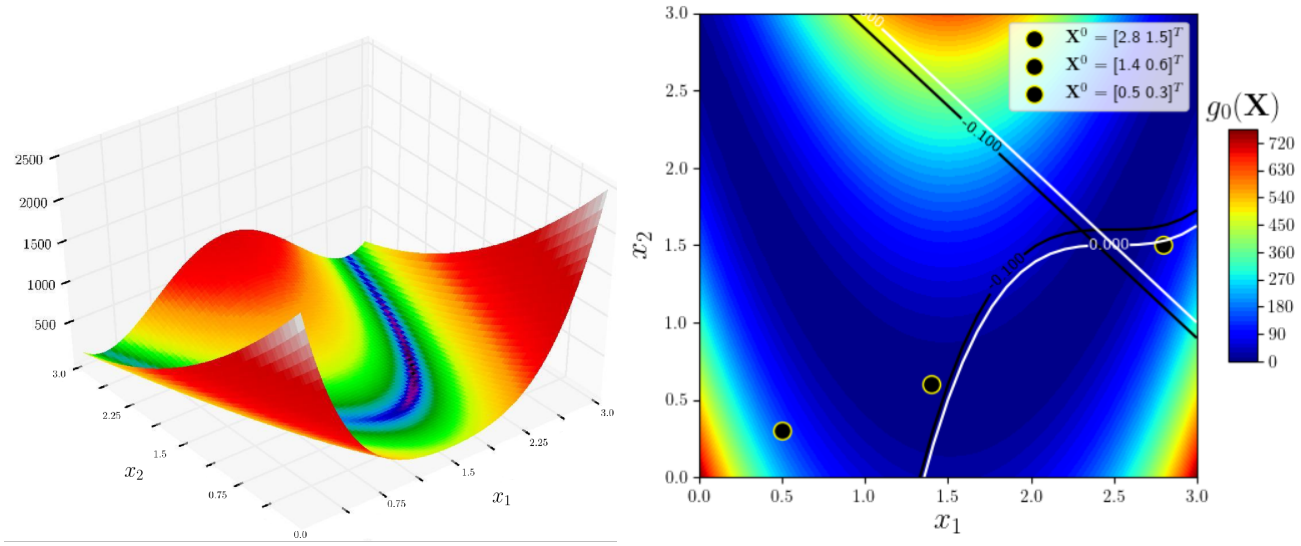
Figure A.9: Implementation of a hybrid scheme. The use of the gray blocks of Fig. A.8 (introduced for the first time by [13]) were found to be beneficial and therefore were added to the scheme of Fig. A.5. The resultant hybrid scheme combines – to some extent – the low cost of IM schemes and the enhanced performance of AM schemes.

B | Numerical examples' details

This appendix contains details on the numerical examples of Section 5.1. More specifically, the exact equations of all benchmark functions used, a graphical representation of the respective solution spaces, as well as a table with the initial points selected and the optima obtained for all optimization runs can be found herein.

Rosenbrock's function

$$\mathcal{P}_{\text{NLP}} = \begin{cases} \underset{\mathbf{X}}{\text{minimize}} & g_0(\mathbf{X}) = (2.5 - x_1)^2 + 100 \cdot ((x_2 - 0.5) - (x_1 - 1.5)^2)^2 \\ \text{s.t.} & g_1(\mathbf{X}) = (x_1 - 2.5)^3 - x_2 + 1.5 \leq 0 \\ & g_2(\mathbf{X}) = x_1 + x_2 - 4 \leq 0 \end{cases}$$



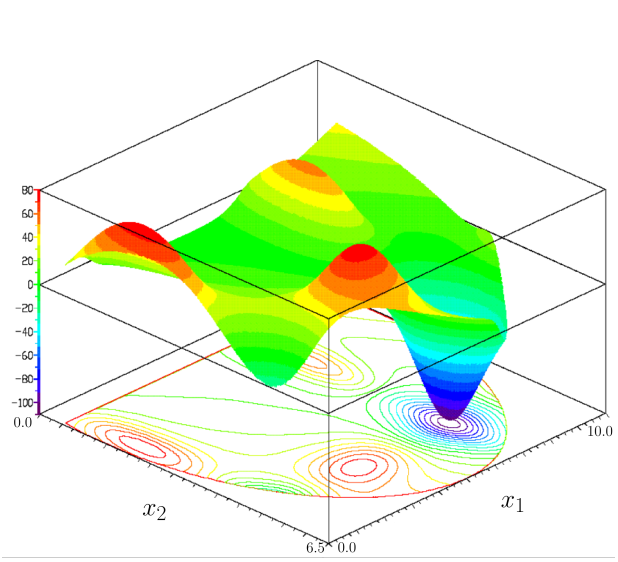
(a) Isometric view of objective function [53]

(b) Top view of objective and constraint functions

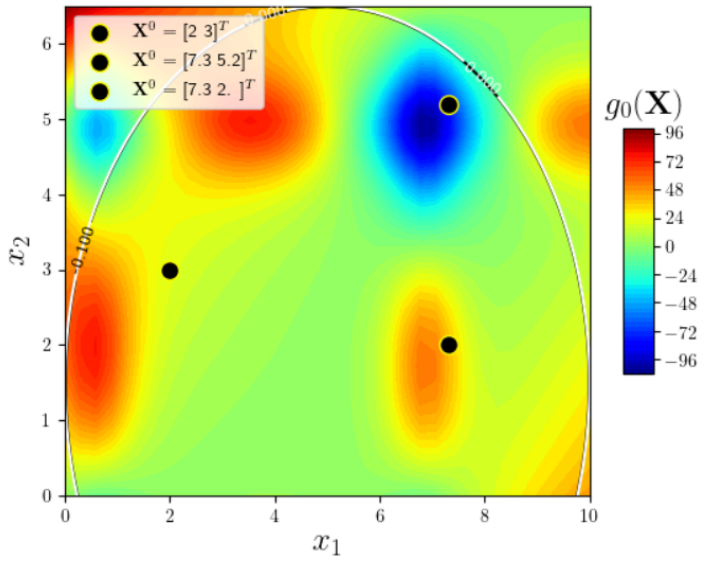
Figure B.1: Rosenbrock's function with cubic and linear constraints [18]

Mishra's bird function

$$\mathcal{P}_{\text{NLP}} = \begin{cases} \underset{\mathbf{X}}{\text{minimize}} & g_0(\mathbf{X}) = \sin(x_2 - 6.5) \cdot e^{[1-\cos(x_1-10)]^2} + \cos(x_1 - 10) \cdot e^{[1-\sin(x_2-6.5)]^2} + (x_1 - x_2 - 3.5)^2 \\ \text{s.t.} & g_1(\mathbf{X}) = (x_1 - 5.)^2 + (x_2 - 1.5)^2 - 25 \leq 0 \end{cases}$$



(a) Isometric view of objective function [19]



(b) Top view of objective and constraint functions

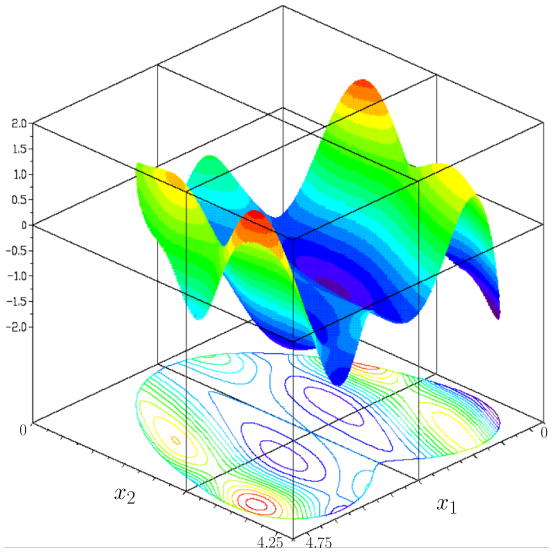
Figure B.2: Mishra's bird function with circular constraint [19]

Townsend's function

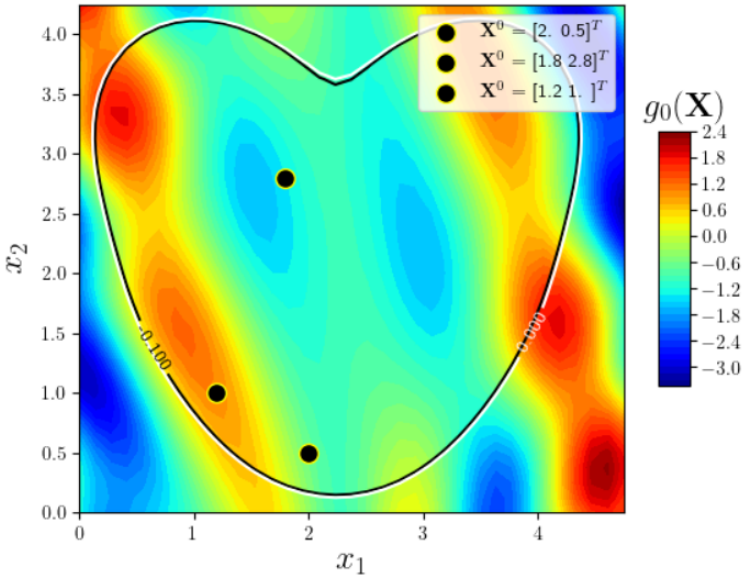
$$\mathcal{P}_{\text{NLP}} = \begin{cases} \underset{\mathbf{X}}{\text{minimize}} & g_0(\mathbf{X}) = -(\cos((x_1 - 2.35) \cdot (x_2 - 2.5)))^2 - (x_1 - 2.25) \cdot \sin(3 \cdot (x_1 - 2.25) + x_2 - 2.5) \\ \text{s.t.} & g_1(\mathbf{X}) = (x_1 - 2.25)^2 + (x_2 - 2.5)^2 - [2\cos(t) - 0.5\cos(2t) - 0.25\cos(3t) - \frac{1}{8}\cos(4t)]^2 \\ & \qquad - [2\sin(t)]^2 \leq 0 \end{cases}$$

where,

$$t = \arctan2(x_1 - 2.25,\ x_2 - 2.5)$$



(a) Isometric view of objective function [20]



(b) Top view of objective and constraint functions

Figure B.3: Townsend's function [20]

Simionescu's function

$$\mathcal{P}_{\text{NLP}} = \begin{cases} \underset{\mathbf{X}}{\text{minimize}} & g_0(\mathbf{X}) = 0.1 \cdot (x_1 - 1.25) \cdot (x_2 - 1.25) \\ \text{s.t.} & g_1(\mathbf{X}) = (x_1 - 1.25)^2 + (x_2 - 1.25)^2 - \left[1 + 0.2 \cos\left(8 \arctan \dfrac{(x_1 - 1.25)}{(x_2 - 1.25)}\right)\right]^2 \end{cases}$$

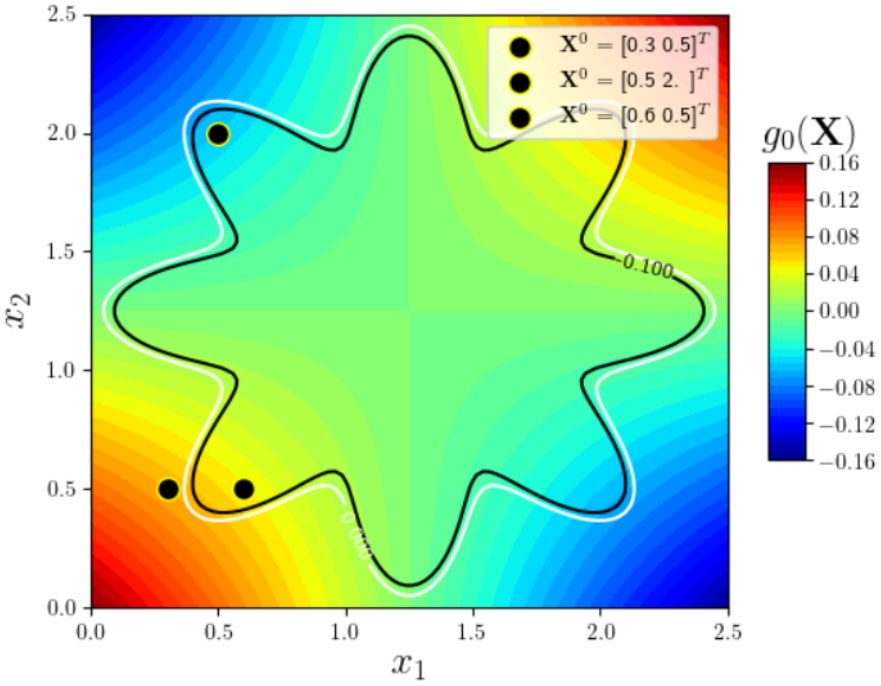


Figure B.4: Simionescu's function [21]

Problem	Literature	Near	Far	Optimum
2-bar	$[2.5\ 1]^T$	$[1.5\ 0.4]^T$	$[0.5\ 1.4]^T$	$[1.411\ 0.377]^T$
3-bar	$[6.4\ 6.4\ 6.4]^T$	$[48\ 63\ 24]^T$	$[10\ 10\ 10]^T$	$[49.17\ 64.51\ 23.33]^T$
10-bar	$[20\ 20\ 20\ 20\ 20\ 20\ 20\ 20\ 20\ 20]^T$	$[30\ 0.2\ 24\ 15\ 0.2\ 0.2\ 9\ 21\ 21\ 0.1]^T$	$[0.2\ 20\ 0.2\ 0.2\ 20\ 20\ 0.2\ 0.2\ 0.2\ 0.2]^T$	$[30.7\ 0.1\ 23.9\ 14.7\ 0.1\ 0.1\ 8.5\ 20.9\ 20.8\ 0.1]^T$
8-bar	$[375\ 250\ 250]^T$	$[110\ 75\ 40]^T$	$[375\ 375\ 375]^T$	$[116.5\ 73\ 37.9]^T$
Welded beam	$[3\ 6.5\ 8\ 2.35]^T$	$[0.3\ 6.3\ 9\ 0.5]^T$	$[9.5\ 0.2\ 0.2\ 9.5]^T$	$[0.24\ 6.22\ 8.29\ 0.24]^T$
Axial spring	$[7\ 0.6\ 0.2]^T$	$[11.2\ 0.4\ 0.06]^T$	$[5\ 1\ 0.9]^T$	$[10.93\ 0.36\ 0.05]^T$
RosenCubic	$[0.5\ 0.3]^T$	$[0.5\ 0.3]^T$	$[0.5\ 0.3]^T$	$[1.5011\ 0.5033]^T$
MishraBird	$[7.3\ 2]^T$	$[7.3\ 5.2]^T$	$[2\ 3]^T$	$[6.87\ 4.92]^T$
Townsend	$[1.2\ 1]^T$	$[1.8\ 2.8]^T$	$[2\ 0.5]^T$	$[1.52\ 2.71]^T$
Simionescu	$[0.6\ 0.5]^T$	$[0.5\ 0.2]^T$	$[0.3\ 0.5]^T$	$[2.1\ 0.4]^T$

Table B.1: Table with initial point selection and the respective optima found for all considered problems

Bibliography

[1] Airbus APWorks. Optimized bike frame. https://apworks.de, 2019. Accessed: 2019-11-25.

[2] David G. Luenberger and Yinyu Ye. *Linear and Nonlinear Programming*. Springer Publishing Company, Incorporated, 2015.

[3] Claude Fleury and Lucien A. Schmit. Dual Methods and Approximation Concepts in Structural Synthesis. *NASA Contractor Reports*, 1980.

[4] Michael Bruyneel, Pierre Duysinx, and Claude Fleury. A family of MMA approximations for structural optimization. *Structural and Multidisciplinary Optimization*, 24(4):263–276, 2002.

[5] W H Zhang, M Domaszewski, and Claude Fleury. A new mixed convex approximation method with applications for truss configuration optimization. *Structural Optimization*, 15:237–241, 1998.

[6] Michael Bruyneel and Claude Fleury. Composite structures optimization using sequential convex programming. In *Advances in Engineering Software*, volume 33, pages 697–711. Elsevier, jul 2002.

[7] Eduardo Fernández, Kai ke Yang, Stijn Koppen, Pablo Alarcón, Simon Bauduin, and Pierre Duysinx. Imposing minimum and maximum member size, minimum cavity size, and minimum separation distance between solid members in topology optimization. *Computer Methods in Applied Mechanics and Engineering*, 368:113157, 2020.

[8] M. Bruyneel and P. Duysinx. Note on topology optimization of continuum structures including self-weight. *Structural and Multidisciplinary Optimization*, 29(4):245–256, 2005.

[9] Em Dede. Multiphysics topology optimization of heat transfer and fluid flow systems. *proceedings of the COMSOL Users Conference*, 2009.

[10] G. I.N. Rozvany. On design-dependent constraints and singular topologies. *Structural and Multidisciplinary Optimization*, 21(2):164–172, 2001.

[11] J F M Barthelemy and Raphael T Haftka. Structural Optimization structural design. *Structural Optimization*, 144(6):129–144, 1993.

[12] Lei Li and Kapil Khandelwal. An adaptive quadratic approximation for structural and topology optimization. *Computers and Structures*, 151:130–147, 2015.

[13] Pierre Duysinx, Michaël Bruyneel, and Claude Fleury. Solution of large scale optimization problems with sequential convex programming. Technical report, Institute of Mechanics and Civil Engineering, University of Liege, 2009.

[14] Albert A. Groenwold and L. F.P. Etman. A quadratic approximation for structural topology optimization. *International Journal for Numerical Methods in Engineering*, 2010.

[15] J. van Schoubroeck. Investigation of the effect of initial design choices in density-based topology optimization. http://resolver.tudelft.nl/uuid:5cba2fb3-ed0b-4d1e-b3c9-99b585ebb2be, 2018. Accessed: 2020-09-20.

[16] Koziel Slawomir and Yang (Eds.) Xin-She. *Computational Optimization,Methods and Algorithms*, volume 356. Springer, 2011.

[17] Thomas Borrvall and Joakim Petersson. Large-scale topology optimization in 3D using parallel computing. *Computer Methods in Applied Mechanics and Engineering*, 190(46-47):6201–6229, 2001.

[18] P.-A. Simionescu and D.G. Beale. New concepts in graphic visualization of objective functions. In *International Design Engineering Technical Conference and Computers and Information in Engineering Conference*, volume 2, pages 891–897, 2002. cited By 7.

[19] Sudhanshu K Mishra. Some new test functions for global optimization and performance of repulsive particle swarm method. *University Library of Munich, Germany, MPRA Paper*, 08 2006.

[20] Alex Townsend. Constrained optimization in chebfun, January 2014.

[21] P.A. Simionescu. *Computer-aided graphing and simulation tools for AutoCAD users*. Taylor & Francis Group, 2014. cited By 11.

[22] Peter W. Christensen and Anders Klarbring. *An introduction to structural optimization*. Springer, 2008.

[23] K. Maute and M. Allen. Conceptual design of aeroelastic structures by topology optimization. *Structural and Multidisciplinary Optimization*, 27(1):27–42, May 2004.

[24] Baotong Li, Liu Honglei, and Shuai Zheng. Multidisciplinary topology optimization for reduction of sloshing in aircraft fuel tanks based on sph simulation. *Structural and Multidisciplinary Optimization*, 05 2018.

[25] J. S. Arora and Q. Wang. Review of formulations for structural and mechanical system optimization. *Structural and Multidisciplinary Optimization*, 30(4):251–272, Oct 2005.

[26] Albert A. Groenwold, L. F P Etman, Schalk Kok, Derren W. Wood, and Simon Tosserams. An augmented Lagrangian approach to non-convex SAO using diagonal quadratic approximations. *Structural and Multidisciplinary Optimization*, 38(4):415–421, 2009.

[27] Krister Svanberg. The method of moving asymptotes — a new method for structural optimization. *International Journal for Numerical Methods in Engineering*, 24(2):359–373, feb 1987.

[28] Albert A Groenwold and L. F. P. Etman. SAOi: an algorithm for very large scale optimal design. *9th World Congress on Structural and Multidisciplinary Optimization*, 2010.

[29] James E. Falk. Lagrange multipliers and nonlinear programming. *Journal of Mathematical Analysis and Applications*, 19(1):141–159, 1967.

[30] Derren Wesley Wood. *Dual Sequential Approximation Methods in Structural Optimisation*. PhD thesis, Stellenbosch University, 2012.

[31] Suqiang Xu and Ramana V. Grandhi. Effective two-point function approximation for design optimization. *AIAA Journal*, 36(12):2269–2275, 1998.

[32] Albert A. Groenwold, L. F.P. Etman, and Derren W. Wood. Approximated approximations for SAO. *Structural and Multidisciplinary Optimization*, 41(1):39–56, 2010.

[33] Claude Fleury. *Mathematical Programming Methods For Constrained Optimization: Dual Methods*, chapter 7, pages 123–150. AIAA, 1993.

[34] Albert A. Groenwold and L. F.P. Etman. Sequential approximate optimization using dual subproblems based on incomplete series expansions. *Structural and Multidisciplinary Optimization*, 36(6):547–570, 2008.

[35] Krister Svanberg. The method of moving asymptotes – modelling aspects and solution schemes. *Lecture notes for the DCAMM course Advanced Topics in Structural Optimization*, 1998.

[36] Bing Chung Chen and Noboru Kikuchi. Topology optimization with design-dependent loads. *Finite elements in analysis and design*, 37(1):57–70, 2001.

[37] Lei Li and Kapil Khandelwal. Two-point gradient-based MMA (TGMMA) algorithm for topology optimization. *Computers and Structures*, 131:34–45, 2014.

[38] Albert A. Groenwold and L. F. P. Etman. On the equivalence of optimality criterion and sequential approximate optimization methods in the classical topology layout problem. *International Journal for Numerical Methods in Engineering*, 73(3):297–316, 2008.

[39] Wei-Hong Zhang and Claude Fleury. A Modification of Convex Approximation Methods for Structural Optimization. *Computers and Structures*, 64(4), 1997.

[40] T. Jiang and P Y Papalambros. A first order method of moving asymptotes for structural optimization. In *Fourth International Conference on Computer Aided Optimum Design of Structures – OPTI95*, volume 14. WIT Press, Jan 1995.

[41] Lucien A. Schmit and Miura Hirokazu. Approximation Concepts for efficient structural synthesis. Technical report, University of California, Los Angeles, 1976.

[42] Albert A. Groenwold, L. F.P. Etman, J. A. Snyman, and J. E. Rooda. Incomplete series expansion for function approximation. *Structural and Multidisciplinary Optimization*, 34(1):21–40, 2007.

[43] Claude Fleury and Vincent Braibant. Structural optimization: A new dual method using mixed variables. *International Journal for Numerical Methods in Engineering*, 23(3):409–428, 1986.

[44] Pierre Duysinx, Wei-Hong Zhang, and Claude Fleury. A new separable approximation scheme for topological problems and optimization problems characterized by a large number of design variables. Technical report, Institute of Mechanics and Civil Engineering, University of Liege, 1995.

[45] Krister Svanberg. A class of globally convergent optimization methods based on conservative convex separable approximations. *SIAM Journal on Optimization*, 2002.

[46] G. M. Fadel, M. F. Riley, and J. M. Barthelemy. Two point exponential approximation method for structural optimization. *Structural Optimization*, 2(2):117–124, 1990.

[47] Min Soo Kim, Jong Rip Kim, Jae Young Jeon, and Dong Hoon Choi. Efficient mechanical system optimization using two-point diagonal quadratic approximation in the nonlinear intervening variable space. *KSME International Journal*, 15(9):1257–1267, 2001.

[48] Jong Rip Kim and Dong Hoon Choi. Enhanced two-point diagonal quadratic approximation methods for design optimization. *Computer Methods in Applied Mechanics and Engineering*, 197(6-8):846–856, 2008.

[49] J A Snyman and N Stander. New successive approximation method for optimum structural design. *AIAA Journal*, 32(6):1310–1315, 1994.

[50] J. A. Snyman and A. M. Hay. The spherical quadratic steepest descent (SQSD) method for unconstrained minimization with no explicit line searches. *Computers and Mathematics with Applications*, 2001.

[51] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, Jan 2002.

[52] Susana Rojas-Labanda and Mathias Stolpe. Benchmarking optimization solvers for structural topology optimization. *Structural and Multidisciplinary Optimization*, 52(3):527–547, sep 2015.

[53] H. H. Rosenbrock. An Automatic Method for Finding the Greatest or Least Value of a Function. *The Computer Journal*, 3(3):175–184, 01 1960.