Application of Evolutionary Structural Optimization to Reinforced Concrete Structures

Andrea De Marco

Thesis submitted in partial fulfilment of the requirements for the Degree of:

Master of Science

in

Civil Engineering

Track: Structural Engineering
Specialization: Structural Mechanics

Faculty of Civil Engineering and Geosciences,
Delft University of Technology, The Netherlands.

June 2018

Author:

Andrea De Marco
Student n. 4321421
Delft University of Technology

Committee:

Dr. ir. M.A.N. Hendriks
Delft University of Technology
Faculty of Civil Engineering and Geosciences
Department of Structural Mechanics

Prof. dr. ir. J.G. Rots
Delft University of Technology
Faculty of Civil Engineering and Geosciences
Department of Structural Mechanics

Dr. ir. drs. C.R. Braam
Delft University of Technology
Faculty of Civil Engineering and Geosciences
Department of Concrete Structures

Dr. ir. M. Langelaar
Delft University of Technology
Faculty of Mechanical, Maritime and Materials Engineering
Department of Precision and Microsystems Engineering

Keywords:

Evolutionary Structural Optimization, Bi-directional Evolutionary Structural Optimization, Computational Optimization, Topology Optimization, Structural Optimization, Heuristic Optimization, ESO, BESO, Optimization Algorithm, Optimization Procedure, Reinforced Concrete Structures, Reinforced Concrete Design, Reinforced Concrete

Abstract

The present thesis has the objective to create a procedure for the automatic preliminary design of reinforced concrete structures, based on the Evolutionary Structural Optimization method (ESO). The developed algorithm performs heuristic topology optimization based on multiple criteria, in subsequent optimization cycles executed in series. In each ESO cycle, it is possible to perform material addition, removal or transition between a couple of materials, i.e. steel, concrete and void. Each optimization cycle is governed by an optimality criterion chosen from: stiffness, Von Mises, Drucker-Prager, tension stress criteria or linear interpolation of previous ones. The finite element analysis used in the algorithm regards all materials as linear elastic. The reaching of maximum strength and failure of materials is not taken into account.

The automated design process of reinforced concrete structures has been separated in two problems: the form-finding of reinforced concrete structure as a whole and the definition of internal distribution between steel and concrete. The first problem is solved, in the first optimization cycle, with a "standard" ESO procedure, with the only difference in choosing a Von Mises optimization criterion over a "classical" stiffness criterion. The second problem is solved, in the second and third optimization cycles, combining in series two additional "modified" ESO procedures and introducing gradual transition in optimization criteria through linear interpolation of sensitivity numbers. The combined criteria for the second cycle are a Von Mises and Drucker-Prager optimization criteria, while for the third cycle, a Drucker-Prager and tensile stress optimization criteria. Moreover, additional geometrical constraints are applied, to ensure a set minimal distance between steel and outer boundaries, i.e. concrete cover, and to ensure optimal angle preservation for steel members found during optimization, introducing two new "density" matrices called *cover* and *mask*. Summarizing, the preliminary design of reinforced concrete structures is dealt with three cycles of ESO optimization procedures executed in series, with optimization criteria that gradually change with continuity between the different cycles over the whole process.

The developed procedure has been tested on several case studies, both from ESO and reinforced concrete literature. The defined ESO process has been found to generate solutions with steel correctly placed in tensile stress zones both to resist bending and shear. Generally, the presence of remaining tensile stresses in concrete zones in the solutions is absent or very limited, and in latter cases their magnitude is very small compared to other stresses. Obtained solutions cannot be directly used as such for reinforcement layouts definition but, in combination with principal stresses plots and engineering judgement, they are able to suggest useful resulting reinforcement layouts.

Application of automatic concrete cover around holes, which are generated during the optimization process and whose position and shape is not previously known, has been proven possible. Moreover, application of concrete cover has been observed to bring improvements to the steel distribution in the solutions. Also the presence of internal holes in the geometry, and thus of the related stress concentrations, has been found to be favorable in the same regard.

Careful choice of target steel volumes, in the input of the algorithm, has to be performed to get solutions of interest. With the present procedure, the target steel volume parameter is generally needed to be set considerably higher compared to values used in real-life applications. Therefore, in interpreting the resulting steel zones in the solutions, it is more appropriate to refer to them as zones of structure where reinforcements are desired. This implies the need of an additional step of engineering interpretation to bridge the gap of applicability for the solutions.

Special attention has to be payed in presence of resulting thin members in the geometry, either due to closeness of imposed holes to the edges of the structure or to the size of mesh compared to the parameters for filtering of sensitivities and concrete cover. In such cases, incorrect steel elements removal may occur, resulting in the presence of concrete members loaded in tension in the solution.

In conclusion, it has been observed that Evolutionary Structural Optimization is well suited approach for the design process of reinforced concrete structures and offers new insights and perspectives. For such application, ESO method could be used both to optimize the overall structure geometry and to find

internal distribution between concrete and steel.

As result of the specificities introduced for addressing the design of reinforced concrete structures, the ESO method has been extended for the case of topological optimization of composite materials with macro structure, with different optimization criteria applied to component materials, with one component material that has different resistance in tension and compression and with geometrical constraints for one component material.

The developed algorithm could be extended to address analogous cases from other disciplines by including other material parameters, adapting optimality criteria and updating or formulating new geometrical constraints. Since the code of the present thesis has been developed modularly, it could be upgraded with computationally more efficient parts of code, with non-linear or sequentially-linear finite elements solvers and/or with additional algorithms for improving steel distributions obtained by current procedure. The complete MATLAB code is published in the annex.

Preface by the Author

"In a matter of speaking, optimization is the way to harmonize existence and endeavor within the space governed by the laws of Nature. Maybe, optimization is embedded in ourselves, and in all Nature's creations, as one of our basic principles of existence and, along with the growth of our mathematical and computational capabilities, we move forward in the pursuit of our optimum form, organization and endeavor, within all our dimensions of living."

The deep roots of optimization have been, and continue to be, discovered in several physical phenomena which well correlate to optimized results of minimization/maximization problems. For example, the Dirichlet Principle:
"Local strict minima of potential energy are stable equilibrium configurations of a conservative mechanical system",
and the Fermat Principle:
"Rays of light follow the path of least travel time between two points".

On the other hand, optimization is an important and recurrent topic in human culture as well. Over the years, more and more scientists, engineers, entrepreneurs, managers and policy makers look for optimization in processes, products, projects and, at the same time, researchers look for better ways and algorithms to perform such optimizations.

The multiplicity and multidisciplinarity of optimality criteria required for addressing real-world practical applications, makes the use of rigorous mathematical optimization models quite laborious and arduous in terms of programming and of computational expenses. Therefore, throughout the years, heuristic optimization relevance grew due to its simplicity in implementation, its elasticity of incorporating and combining multiple criteria and its possibilities to tweak and adapt algorithms. As a matter of fact, in heuristic optimization it is required to think "outside the box" to find new ways in combining multiple criteria and parameters to get the needed solutions in the most effective ways.

For such reasons, a heuristic optimization method, namely the Evolutionary Structural Optimization, has been chosen in this thesis to address the automatic design of reinforced concrete structures through optimization procedure.

Contents

1 Introduction **7**
- 1.1 History of optimization . . . 7
- 1.2 Literature review . . . 8
- 1.3 Applications . . . 8
- 1.4 Applications in Structural Engineering . . . 9
- 1.5 Scope of the thesis . . . 14
- 1.6 Objective of the thesis . . . 14
- 1.7 Outline of the thesis . . . 14

2 Methods **15**
- 2.1 Standard ESO formulation . . . 15
- 2.2 Modified ESO formulation . . . 19
- 2.3 Strategy and key choices . . . 23
- 2.4 MATLAB code explained . . . 28
  - 2.4.1 ESOscript.m code explained . . . 29
  - 2.4.2 Comments on FEM.m code . . . 41
  - 2.4.3 Comments on POST.m code . . . 42
  - 2.4.4 Comments on SENS.m code . . . 42
  - 2.4.5 Comments on COVER.m code . . . 43

3 Results **47**
- 3.1 Case study n.1 . . . 50
- 3.2 Case study n.2 . . . 51
- 3.3 Case study n.3 . . . 51
- 3.4 Case study n.4 . . . 54
- 3.5 Case study n.5 . . . 54
- 3.6 Case study n.6 . . . 57
- 3.7 Case study n.7 . . . 60
- 3.8 Case study n.8 . . . 60
- 3.9 Case study n.9 . . . 62
- 3.10 Case study n.10 . . . 62
- 3.11 Case study n.11 . . . 65
- 3.12 Case study n.12 . . . 68

4 Discussion **73**
- 4.1 Discussion about chosen procedure . . . 73
  - 4.1.1 Choice of the structure of optimization algorithm . . . 73
  - 4.1.2 Smooth change of sensitivity numbers . . . 73
  - 4.1.3 Use of $mask$ and y density matrices . . . 74
  - 4.1.4 Order of materials in optimization loops . . . 76
  - 4.1.5 Different materials in finite element analysis . . . 77
  - 4.1.6 Choice of sensitivity numbers formulations . . . 77
- 4.2 Discussion about solutions for case studies . . . 78
  - 4.2.1 Target volume of reinfocred concrete . . . 78
  - 4.2.2 Target volume of steel . . . 78

4.2.3 Target volume of steel in intermediate ESO loop . . . 80
4.2.4 Effect of stress concentrations . . . 81
4.2.5 Effect of concrete cover . . . 82
4.2.6 Effect of mask . . . 82
4.3 Evaluation and interpretation of solutions for case studies . . . 84
4.3.1 Case study n.1 . . . 85
4.3.2 Case study n.2 . . . 89
4.3.3 Case study n.3 . . . 93
4.3.4 Case study n.4 . . . 97
4.3.5 Case study n.5 . . . 101
4.3.6 Case study n.6 . . . 105
4.3.7 Case study n.7 . . . 109
4.3.8 Case study n.8 . . . 113
4.3.9 Case study n.9 . . . 117
4.3.10 Case study n.10 . . . 121
4.3.11 Case study n.11 . . . 125
4.3.12 Case study n.12 . . . 129

5 Conclusion **133**
5.1 Conclusions . . . 133
5.2 Directions for further development . . . 135
5.2.1 Problems with self-weight . . . 135
5.2.2 Improvement of steel layout optimization . . . 137

Chapter 1

Introduction

1.1 History of optimization

Optimization has been a studied problem since the ancient Greeks, that used geometry to approach minimization and maximization problems.
Along with the advent of calculus, in the seventeen century, new mathematical tools to approach optimization problems appeared, namely differential and integral calculus. Not long after then, in the eighteenth century, the latter tools were refined and structured in a solid framework for approaching problems in generic contexts. So, the field of Calculus of Variations was born, which provided instruments to develop analytical solutions for optimization problems.
In the nineteenth century, the first optimization algorithms were established, i.e. the "least squares problem" and "gradient method", which offered new possibilities to study minimization and maximization problems and so develop numerical solutions.
Active research continued in the past two centuries in both the field of analytic solutions, through Calculus of Variations, and the field of numerical solutions, through many types of algorithms established ever since. The latter methods became more and more viable, thanks to the strong development and increasing accessibility to electronic computations and mathematical programming. [27]

In the field of algorithm development for structural optimization, we see further distinction in two branches, the first which comprehend the rigorous mathematical programming methods:

- SIMP "Solid Isotropic Material with Penalization",
- MMA "Method of Moving Asymptotes"
- Homogenization Method

and the second which comprehend the heuristic methods:

- ESO "Evolutionary Structural Optimization",
- GA "Genetic Algorithms".

The ones listed above are generally understood as the most famous and researched methods to date, but in literature it is possible to find several dozen other methods, as well as variants and combinations of the above ones. Although heuristic methods do not guarantee to reach a global optimum they are still of big interest in engineering applications. It is to note that due to constructability/manufacturing issues, and due to engineering local phenomena caused by environmental/materials issues, the absolute optimum in mathematical terms is not guaranteed to be the engineering optimum. In engineering practice, the optimization techniques currently could solely take the role of an assistant and additional source of information rather than a substitute for designing process. This is especially true in the field of reinforced concrete structures due to its complex behavior and its related phenomena.

The valuable resource of optimization algorithms is their innate freedom from traditional construction schemes, layouts and the human design experience. In fact, the only data that is "fed" to the algorithm

is constituted by the physical laws, the boundary conditions (related to displacements and forces) and the criteria to be used to create/optimize the design. Therefore, solutions can correlate and validate customary design schemes or not resemble them at all and so provide new directions to explore or to bring attention to issues overlooked before. With the future evolution of mathematical models, optimization algorithms and computing power, the influence of programming and structural optimization is expected to grow massively their influence on the way structures are designed, built and repaired, eventually becoming one of the fundamental tools in the toolkit of the future engineer.

1.2 Literature review

Over the past 30 years a lot of research has been carried on for the multiple types of optimization methods available, mainly the Homogenization Method, MMA, SIMP, ESO and GA, which have found several applications ever since. The SIMP algorithm has been the first of them to become efficient, robust and widely used. As result, SIMP optimizers have recently been introduced in some of the main finite element packages all over the world. For a comprehensive review of SIMP and Homogenization methods see the book by Sigmund and Bendsoe [1]. Moreover, it is to note that a MATLAB code implementation of SIMP has been open to the public since the year 2001 [7]. A revised and updated version of the code has been published by Sigmund et al. in 2011 [8]. At comparable levels of development and popularity as of SIMP, there is ESO.

The ESO method originally proposed by Xie and Steven [6] is built on the basic heuristic principle: *"Slowly removing inefficient materials, the structure evolves towards an optimum"*.
Initially ESO was implemented solely as a material removal method, which meant that removed parts could not be restored afterwards. However, this led to convergence issues and mesh-dependence. The latter problems were overcome in the extension of the method called BESO "Bidirectional Evolutionary Structural Optimization" that allowed both material addition and removal, as first presented in the paper [9]. Two complete books summarizing and containing the major results from research articles are [2] for ESO and [3] for BESO. In the thesis, both methods will be referred to using the acronym ESO only.

In subsequent implementations, the ESO method borrowed some concepts from SIMP, remarkably the strategy to use a bi-phase material as adopted in the paper of Xie [10] written in 2008. With the addition of the so-called material interpolation scheme, material is never removed but shifts from and to a "softened" version of itself, which has a steep reduction of its mass, density, elastic modulus and its other properties. This method solved ultimately numerical instabilities and gave the required robustness to the ESO method to become of wide interest and use. Simultaneously, the very same article [10] opened the doors to multi-material optimization in the ESO method.

In spite of using similar material interpolation schemes, ESO development goes on quite distinctively from the one of SIMP and offers a more flexible environment for integrating new design requisites and optimization criteria, thus offering an extended potential range of applications.

1.3 Applications

As of today, ESO method has been applied for the optimization of:

- stiffness,
- displacement,
- von mises stress,
- natural frequency
- stiffness, considering design-dependent loads (self-weight),
- stiffness, considering multiple materials,
- stiffness, considering fluid and earth interaction,

- stiffness, considering periodic structures
- stiffness, considering material and geometrical nonlinearities.

For a comprehensive review of the above ESO developments and applications till the year 2016, see the article of Xie [11]. In such paper, applications of ESO to both macro and micro scale are presented, along with sample MATLAB codes for the basic cases of stiffness optimization in both scales. It is remarkable that the macro and micro scale ESO optimization algorithms presented in such paper can also be performed simultaneously.

Beyond the applications reviewed in the article of Xie [11], it is important to point out other implementations found in literature, however developed with the first version of ESO algorithm only, now superseded:

- tensile-only and compressive-only materials [12],
- simultaneous multi-criteria optimization for single material optimization
    - o stiffness-inertia [13],
    - o stiffness-natural frequency [14],
    - o stiffness-stress [15].

It is relevant to note that the biggest part of ESO applications found in literature refer to isotropic materials with same resistance in tension and compression (e.g. steel) or composite materials with micro structure (e.g. composites in the aerospace industry). For such cases, there is a wide range of real-world applications especially in the fields of mechanical and aerospace engineering. Applications in structural engineering, however, are still relatively limited except a few notable cases.

1.4 Applications in Structural Engineering

The most relevant applications of ESO in structural engineering and architecture to date are based on few main figures from Japan, namely: Mutsuro Sasaki, Toshihiko Iijima and Hiroshi Ohmori.

- Mutsuro Sasaki has applied ESO for roof supports design and for shells optimization in several projects in collaboration with the top-notch architects: Arata Isozaki, Toyo Ito and Kazuyo Sejima + Ryue Nishizawa (SANAA);
- Toshihiko Iijima, in collaboration with F-tai architects, has built the first ESO-designed building in reinforced concrete in 2004, applying ESO for load bearing facade design;
- Hiroshi Ohmori, has developed the variation of ESO method, called “Extended ESO”, used by Mutsuro Sasaki and Toshihiko Iijima in their projects.

In the text below, the main projects curated by Mutsuro Sasaki’s and Iijima’s structural design offices are shown. As a remark, in the reinforced concrete structure designed by Iijima structural design office, the ESO method was employed only to determine the outer shape of the structure and not to determine the distribution between concrete and steel inside it. Details regarding the specific features of the “Extended ESO” method used by Sasaki and Iijima, can be found by interested readers in the works by Sasaki [5],[26] and Ohmori [26], [24].

It is to be noted that several other practical applications of ESO can be found in the fields of mechanical and aerospace engineering. However, since those are beyond the scope of this thesis, they are not reported or discussed further in the text.

Figure 1.1: Mutsuro Sasaki Structural Design Office and Arata Isozaki Atelier, “Illa de Blanes” in Blanes, Spain, 2002 [22]

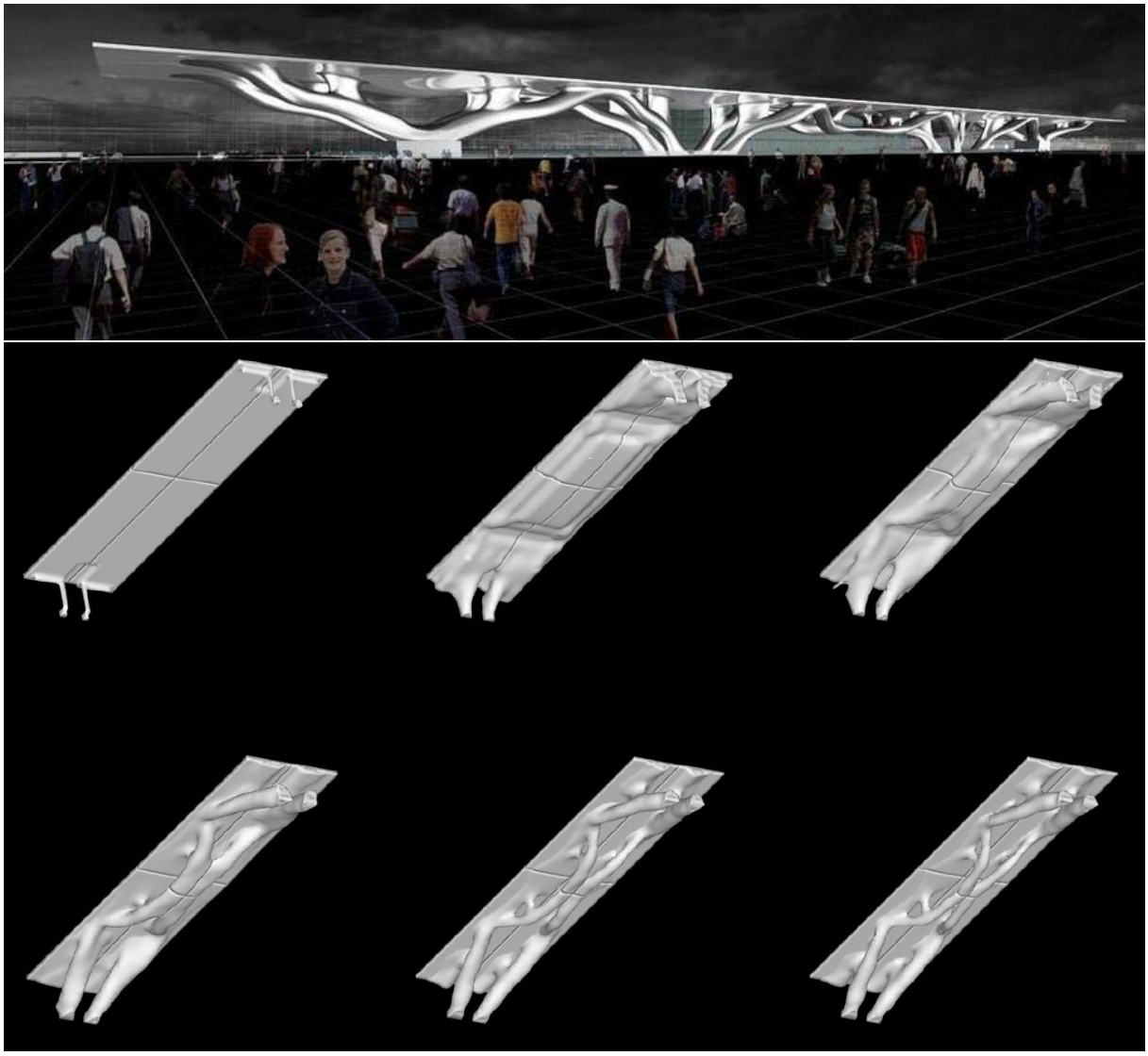
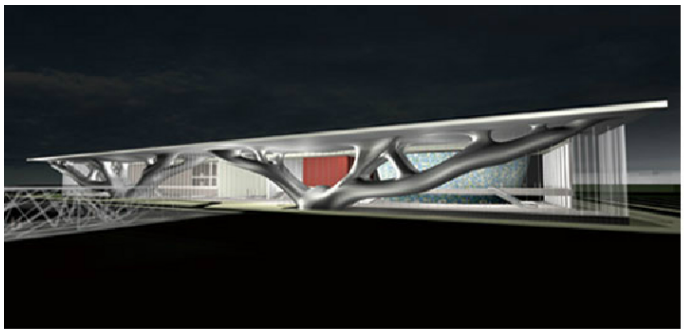
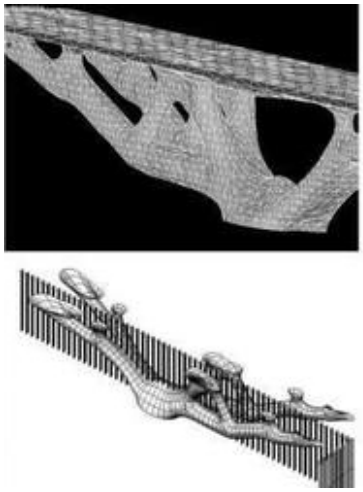


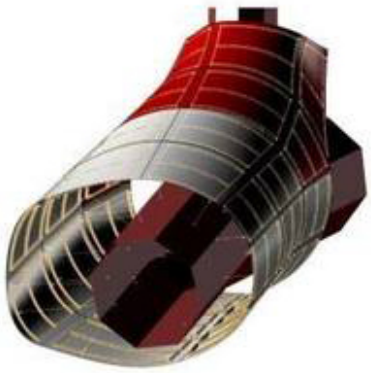
Figure 1.2: Mutsuro Sasaki Structural Design Office and Arata Isozaki Atelier, 3D model of the train station Santa Maria Novella in Florence, 2003 [22], [28]

(a) Photo of finished building



(b) Render (left), Photo during construction (right)





(c) Optimized mesh (left), Construction detail (center), Photo (right)

Figure 1.3: Mutsuro Sasaki Structural Design Office and Arata Isozaki Atelier, Qatar National Convention Centre in Doha, 2008 [29], [22], [30]

Figure 1.4: Mutsuro Sasaki Structural Design Office and Toyo Ito & Associates, Crematorium in Kakamigahara Gifu, Japan, 2006 [22]

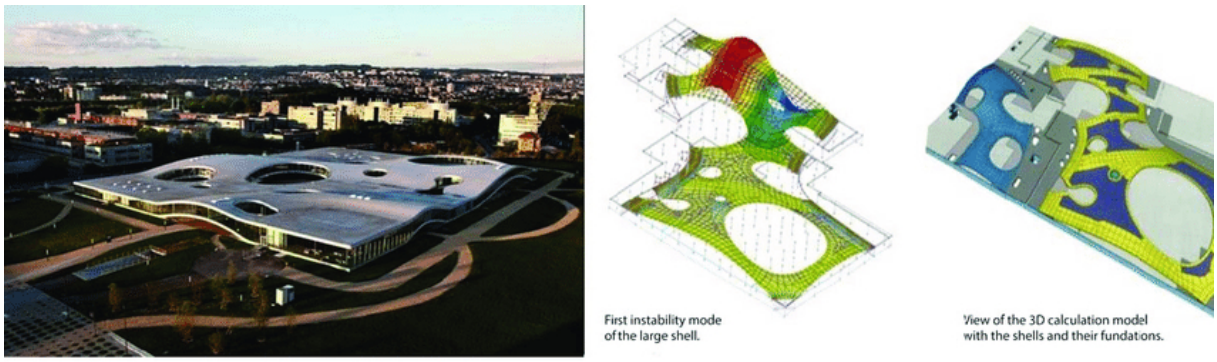
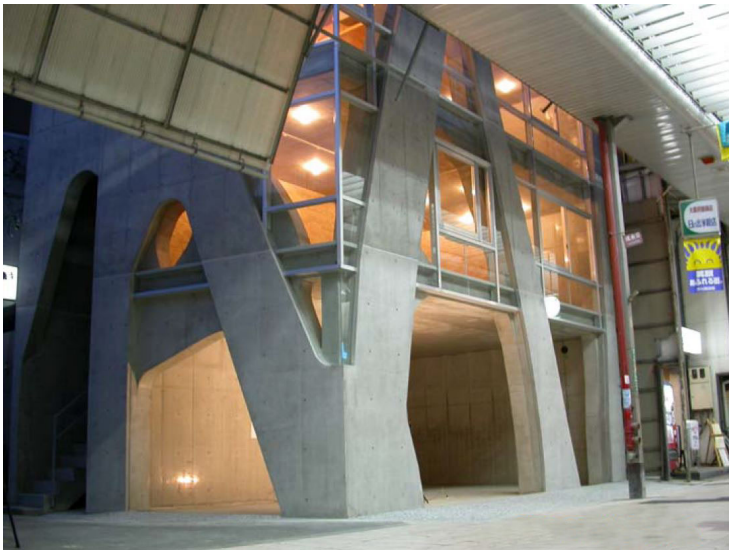


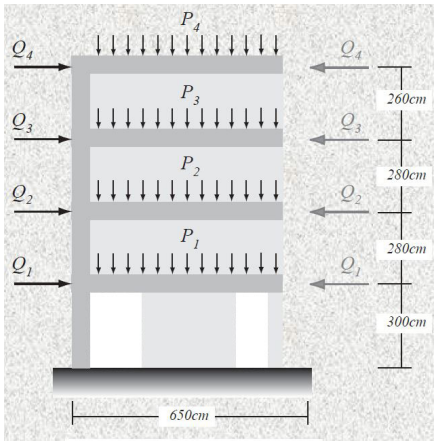
Figure 1.5: Mutsuro Sasaki Structural Design Office and Kazuyo Sejima + Ryue Nishizawa (SANAA), Rolex Learning Centre in Lausanne, Switzerland, 2008 [22]

(a) Photo - West facade

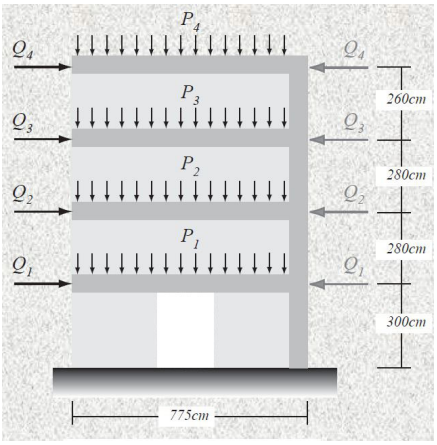


(b) Photo - South facade

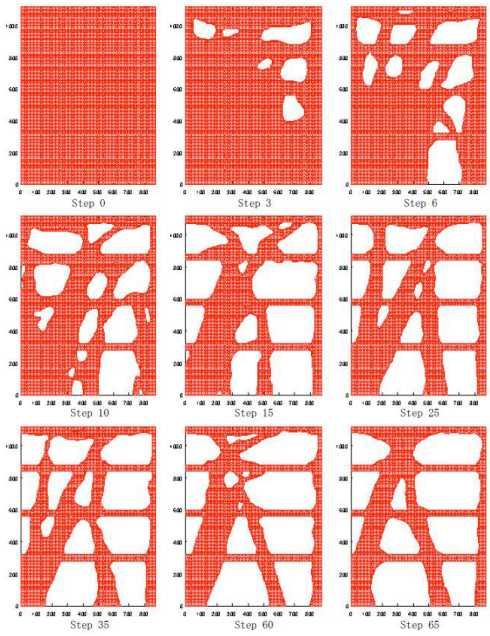
Figure 1.6: Iijima Structural Design Office and F-tai Architects, Akutagawa River Side in Takatsuki, Japan, 2004 [28]



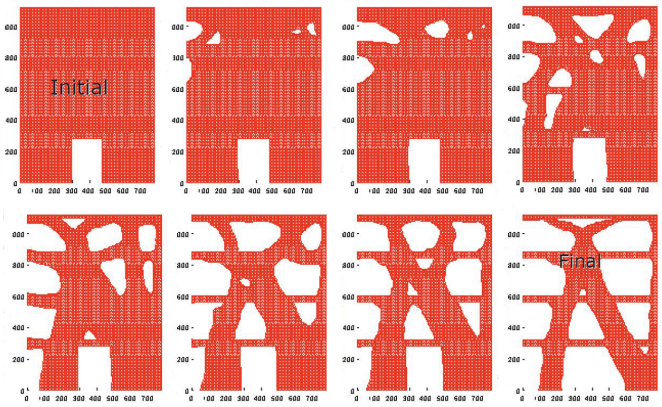
(a) Problem statement - West facade



(b) Problem statement - South facade



(c) Optimization procedure - West facade



(d) Optimization procedure - South facade

(e) Project render - West facade

(f) Project render - South facade

Figure 1.7: Iijima Structural Design Office and F-tai Architects, Akutagawa River Side in Takatsuki, Japan, 2004 [25], [23], [28]

1.5 Scope of the thesis

As discussed in the section "Literature review", several types of optimization within the ESO method are available for isotropic materials with same resistance in tension and compression (e.g. steel) and for composite materials with microstructure. The case of reinforce concrete, however, does not fall in either one of them. In fact, reinforced concrete is a composite material with macro structure. And, moreover, one of the components (i.e. concrete) has different resistance in tension and compression.

Regarding composite materials with macrostructure, the research found in ESO literature is very limited. To the knowledge of the author, in fact, there has been only one relevant application [10] of ESO algorithm to composite materials with macro scale, where the same optimization criteria has been applied on all materials of the composite. In the case of reinforced concrete, however, it is needed for each material of the composite - concrete and steel - to undergo optimization based on different criteria.

The scope of this thesis is then to extend the ESO method for the particular case of:

"Topological optimization of composite materials with macro structure, with different optimization criteria applied to the component materials,"

and, more specifically, to reinforced of concrete, that adds to the previous:

"with one component material that has different resistance in tension and compression (i.e. concrete), and with geometrical constraints for one component material (i.e. concrete cover applied on steel)."

1.6 Objective of the thesis

On a more practical level, the objective of the thesis is to:

"Define a procedure, through a MATLAB code, to perform automatic preliminary design of reinforced concrete structures, using an Evolutionary Structural Optimization process."

In other words:

"Given a starting geometry, a set of boundary conditions (constraints and applied forces & displacements), and set of target volumes for concrete and steel,"

the procedure developed in the thesis, in the form of a MATLAB code, should be able to:

"define the optimal shape of the structure, and the optimal topology & distribution between concrete and steel, also taking concrete cover requirements into consideration."

1.7 Outline of the thesis

The thesis is organized as follows. In chapter 2, "Methods", the optimization problem & process is formulated, starting from its simplest case and, step by step, evolved to its required complexity, to include all the specificities for addressing reinforced concrete. Then, the developed MATLAB code is presented and explained. In chapter 3, "Results", the code is tested on several case studies from literature to assess its ability to reach the set objective and to evaluate the quality of solutions obtained with it. In chapter 4, "Discussion", the reasons and effects of choices and methods for the developed procedure are discussed along with the solutions of case studies. Principal stress plots and example reinforcement layout interpretations are shown for all case studies. In chapter 5, the conclusions are drawn and few directions for further research and development are given. Finally, in the annex, the MATLAB code developed in the thesis is reported in its entirety.

Chapter 2

Methods

For better understanding the optimization procedure used in the thesis, the chapter starts with presenting the simplest case of single material stiffness ESO optimization, that is step by step modified and extended in the chapter for the purposes of the thesis. Consequently, the strategy and specific choices for the optimization process of the thesis are presented, and the developed procedure is outlined in its steps and explained along its code.

2.1 Standard ESO formulation

Material interpolation scheme

In standard ESO 2-dimensional formulation, the geometry is represented by a rectangular finite element mesh composed by bi-linear 4-noded rectangular elements in condition of plane stress. To each element, it is assigned a "density" parameter x_i, which can assume the values $x_i = x_{min}$ or 1, where x_{min} is a very small number, usually set to 0.001. The standard ESO method considers each element to be composed of a bi-phase material, (e.g. steel and void, or concrete and void, or steel and concrete), and the parameter x_i is used to determine the current material phase in an element. When $x_i = 1$, the i^{th} element is composed by the 1^{st} phase of the material, while when $x_i = x_{min}$, it is composed by the 2^{nd} phase. Therefore, the standard ESO problem is reduced to the determination of the values of the matrix x.

In the biggest part of cases in literature, the bi-phase material is comprised by a chosen material and void. Then, the optimization problem can be referred to as material removal and addition. It is used the following material "density" definition:

$$x_i = \begin{cases} 1 & \text{for material} \\ \\ x_{min} & \text{for void} \end{cases} \tag{2.1}$$

and it is used a material interpolation scheme as follows:

$$E(x_i) = Ex_i^p \qquad \forall i = 1, ..., N \tag{2.2}$$

where p is the penalty exponent, usually chosen as $p = 3$. At this point, it is useful to remark that the choice of setting $x_{min} = 0.001 = 10^{-3}$ is done to avoid singularity of the stiffness matrix during computations, for the cases when a bi-phase material comprised by material and void is considered. In such cases, when the material is turned to void, its stiffness is notably reduced (by 10^{-9}) but it remains different from zero.

In another part of cases, the bi-phase material is comprised by two non-void material. Then, the problem can be referred to as material phase change. It is used the following material "density" definition:

$$x_i = \begin{cases} 1 & \text{for } 1^{st} \text{ phase material} \\ \\ x_{min} & \text{for } 2^{nd} \text{ phase material} \end{cases} \tag{2.3}$$

and it is used a material interpolation scheme as follows:

$$E(x_i) = x_i^p E_1 + (1 - x_i^p) E_2 \qquad \forall i = 1, ..., N \tag{2.4}$$

where p is the penalty exponent, usually chosen as $p = 3$, and the materials are ordered from greatest to least in terms of stiffness $E_1 > E_2$.

There are other cases when there are more than two materials in total, and thus one bi-phase material would not suffice. In such cases, it is needed to define multiple bi-phase materials. The paper of Xie [10], addresses multi-material optimization of n materials, with $n - 1$ bi-phase materials being optimized "in parallel" in a single ESO procedure for stiffness maximization. The materials are ordered from greatest to least in terms of stiffness $E_1 > E_2 > E_3 > ... > E_n$ and it is used the following material "density" definition:

$$x_{ij} = \begin{cases} 1 & \text{for materials with: } E_i \geq E_j \\ \\ x_{min} & \text{for materials with: } E_i \leq E_{j+1} \end{cases} \tag{2.5}$$

where x_{ij} represents the density of the i^{th} element for the j^{th} bi-phase material, and it is used the following material interpolation scheme:

$$E(x_{ij}) = x_{ij}^p E_j + (1 - x_{ij}^p) E_{j+1} \qquad \forall i = 1, ..., N \quad \forall j = 1, ..., n - 1 \tag{2.6}$$

where p is the penalty exponent, usually chosen as $p = 3$.

Problem formulation

In the case of single bi-phase material optimization (then in both cases of material/void and material_1/material_2 optimization), the minimization problem appears as follows:

$$\text{minimize:} \qquad Obj = F(x_i, u) \tag{2.7}$$

$$\text{subject to:} \qquad V^* - \sum_{i=1}^{N} V_i x_i = 0 \tag{2.8}$$

$$\text{with:} \qquad x_i = x_{min} \text{ or } 1 \qquad \forall i = 1, ..., N \tag{2.9}$$

where Obj is the objective function to be minimized, x_i is the material density 2.1 used in the material interpolation scheme 2.2 and 2.4, u is the displacement vector, V^* is the prescribed volume to reach during the optimization process, N is the number of finite elements in the mesh, V_i is the volume of the i^{th} element in the mesh.

In the case of multiple bi-phase material optimization (as in the paper of Xie [10]), the minimization problem appears as follows:

$$\text{minimize:} \qquad Obj_j = F(x_{ij}, u) \tag{2.10}$$

$$\text{subject to:} \qquad V_j^* - \sum_{i=1}^{N} V_i x_{ij} - \sum_{i=1}^{j-1} V_i^* = 0 \qquad \forall j = 1, ..., n - 1 \tag{2.11}$$

$$\text{with:} \qquad x_{ij} = x_{min} \text{ or } 1 \qquad \forall i = 1, ..., N \quad \forall j = 1, ..., n - 1 \tag{2.12}$$

where u, N and V_i are the same parameters as described for previous case, Obj_j is the objective function for the bi-phase material j, x_{ij} is the material density 2.5 used in the material interpolation scheme 2.6, V_j^* is the prescribed volume for the bi-phase material j, and n is the number of materials.

Sensitivity number formulation

The elements are removed and added, or their phase is shifted, according to their ranking based on a so called *sensitivity number*. In standard ESO algorithms, for single bi-phase material optimization (material/void or material_1/material_2 cases) it is set:

$$Obj = \sum_{i=1}^{N} \alpha_i \tag{2.13}$$

and for multiple bi-phase material optimization (more than two materials in total) it is set:

$$Obj_j = \sum_{i=1}^{N} \alpha_{ij} \tag{2.14}$$

In the standard case of stiffness maximization, the sensitivity number is derived substituting the interpolation scheme 2.2, or 2.4, or 2.6, in the compliance $C = \frac{1}{2} u^T K u$, computing the gradient $\frac{\partial C}{\partial x_i}$ or $\frac{\partial C}{\partial x_{ij}}$, and, for convenience, setting $\alpha_i = -\frac{1}{p}\frac{\partial C}{\partial x_i}$ or $\alpha_{ij} = -\frac{1}{p}\frac{\partial C}{\partial x_{ij}}$.
In case of a bi-phase material comprised by a chosen material and void, the sensitivity number for stiffness optimization is expressed as:

$$\boldsymbol{\alpha_i = \frac{x_i^{p-1}}{2} u_i^T K_i u_i \qquad \forall i = 1, ..., N} \tag{2.15}$$

where K_i is the elemental stiffness matrix for the i^{th} element.
In case of a bi-phase material comprised by two non-void materials, the sensitivity number for stiffness optimization is expressed as:

$$\boldsymbol{\alpha_i = \frac{x_i^{p-1}}{2} (u_i^T K_{i,1} u_i - u_i^T K_{i,2} u_i) \qquad \forall i = 1, ..., N} \tag{2.16}$$

where $K_{i,1}$ and $K_{i,2}$ are the elemental stiffness matrices for the i^{th} element calculated respectively with the 1^{st} and 2^{nd} material.
In the case of multiple bi-phase materials, the sensitivity number for stiffness optimization is expressed as:

$$\boldsymbol{\alpha_{ij} = \frac{x_{ij}^{p-1}}{2} (u_i^T K_{i,j} u_i - u_i^T K_{i,j+1} u_i) \qquad \forall i = 1, ..., N \quad \forall j = 1, ..., n-1} \tag{2.17}$$

where $K_{i,j}$ and $K_{i,j+1}$ are the elemental stiffness matrices for the i^{th} element calculated respectively with the j^{th} and $(j+1)^{th}$ material.

For better understanding, for the specific case when all materials have the same Poisson's ratio, substituting the density parameter definitions 2.1 or 2.3 or 2.5, the sensitivity numbers can be explicitly expressed as follows,
- for a bi-phase material comprised by a chosen material and void:

$$\alpha_i = \left\{ \begin{array}{ll} \frac{1}{2} u_i^T K_i u_i & \text{when } x_i = 1 \text{ (for material)} \\ \\ \frac{x_{min}^{p-1}}{2} u_i^T K_i u_i \overset{p \to +\infty}{\longrightarrow} 0 & \text{when } x_i = x_{min} \text{ (for void)} \end{array} \right\} \forall i = 1, ..., N$$

- for a bi-phase material comprised by two non-void materials:

$$\alpha_i = \left\{ \begin{array}{ll} \frac{1}{2}[1 - \frac{E_2}{E_1}] u_i^T K_{i,1} u_i & \text{when } x_i = 1 \text{ (for material 1)} \\ \\ \frac{1}{2} \frac{x_{min}^{p-1}(E_1 - E_2)}{x_{min}^{p-1} E_1 + (1 - x_{min}^{p-1}) E_2} u_i^T K_{i,2} u_i \overset{p \to +\infty}{\longrightarrow} 0 & \text{when } x_i = x_{min} \text{ (for material 2)} \end{array} \right\} \forall i = 1, ..., N$$

- for multiple bi-phase materials:

$$\alpha_{ij} = \left\{ \begin{array}{ll} \frac{1}{2}[1 - \frac{E_{j+1}}{E_j}] u_i^T K_{i,j} u_i & \text{when } x_{ij} = 1 \text{ (for materials 1,...,j)} \\ \\ \frac{1}{2} \frac{x_{min}^{p-1}(E_j - E_{j+1})}{x_{min}^{p-1} E_j + (1 - x_{min}^{p-1}) E_{j+1}} u_i^T K_{i,j+1} u_i \overset{p \to +\infty}{\longrightarrow} 0 & \text{when } x_{ij} = x_{min} \text{ (for materials j+1,...,n)} \end{array} \right\} \begin{array}{l} \forall i = 1, ..., N, \\ \forall j = 1, ..., n-1 \end{array}$$

In the ESO algorithm, the presence of the density x_i is not limited to within the definition of the sensitivity numbers, but can also appear (as it does in the present thesis) inside the finite element analysis in the following form:

$$\begin{array}{rl} \text{Elemental stiffness matrix:} & K_i = x_i^p K_{i,1} + (1 - x_i^p) K_{i,2} \\ \text{Global stiffness matrix:} & K = Assembly(K_i) \end{array}$$

wherein the elemental stiffness matrices $K_{i,1}$ and $K_{i,2}$ are calculated with the following formulas:

$$K_{i,1} = \sum_{gp=1}^{4} w_{gp}\, B^T\, D_1\, B\, det(J)\, h \qquad K_{i,2} = \sum_{gp=1}^{4} w_{gp}\, B^T\, D_2\, B\, det(J)\, h$$

where w_{gp} is the weight relative to the current gauss point, B is the strain-displacement matrix, D_1 and D_2 are the stress-strain matrices respectively for material 1 and 2, $det(J)$ is the determinant of the Jabobian, and h is the plate thickness.

Sensitivity numbers are used to rank elements during the ESO process for the purpose of element removal/addition or shift in its phase, according to a *design update scheme* as presented further in the text.

Filtering schemes for sensitivity numbers

To avoid numerical instabilities, such as the so called *checkerboard pattern* [11], and to ensure mesh independency, it is adopted a *filter scheme* to average sensitivity numbers of neighboring elements and smooth out the sensitivity numbers field. The technique used the most in the past ten years of ESO research is comprised by the following steps:

- average sensitivity numbers α_i of connected elements to define nodal sensitivity numbers α_h^n,
- determine a radius of influence r_{min} to use in the averaging process of α_h^n for the determination of the *filtered* elemental sensitivity numbers $\tilde{\alpha}_i$, *(all nodes inside the circular domain Ω_i with radius r_{min} concur to the calculation of the $\tilde{\alpha}_i$ with different weight in the sum process)*,
- calculate the weight factor $w(r_{ih})$, dependent to the distance r_{ih} of the nodes to the center of the element with the following formula:

$$w(r_{ih}) = \begin{cases} r_{min} - r_{ih} & \text{for } r_{ih} < r_{min} \quad \text{(nodes inside } \Omega_i\text{)} \\ 0 & \text{for } r_{ih} \geq r_{min} \quad \text{(nodes outside } \Omega_i\text{)} \end{cases} \tag{2.18}$$

- calculate the *filtered* sensitivity numbers $\tilde{\alpha}_i$ for each element with the following formula:

$$\hat{\alpha}_i = \frac{\sum_{h=1}^{M} w(r_{ih})\alpha_h^n}{\sum_{h=1}^{M} w(r_{ih})} \qquad \forall i = 1, ..., N \tag{2.19}$$

where M is the number of nodes, indicated with the index h, and N is the number of elements, indicated with the index i.

In addition to spatial averaging of sensitivity numbers, it is performed a "time" averaging between each of two consecutive iterations during the evolutionary process:

$$\tilde{\alpha}_i = \frac{1}{2}(\hat{\alpha}_{i,k} - \hat{\alpha}_{i,k-1}) \qquad \forall i = 1, ..., N \tag{2.20}$$

where, k is the index of current iteration and $k-1$ the index of previous one. In such a way, *historical information* is taken into account in each iteration, resulting in improved convergence of the evolutionary process.

Convergence of optimization process

The optimization is carried on in an iterative process which is stopped when the values of the objective function Obj are converging to a stable value $\overline{Obj} \pm \tau$, where τ is a set tolerance, usually set as $\tau = 0.001 = 0.1\%$. In other words, it is set a convergence criterion expressed as:

$$error = \frac{|\sum_{l=1}^{L}(Obj_{k-l+1} - Obj_{k-L-l+1})|}{\sum_{l=1}^{L} Obj_{k-l+1}} < \tau \tag{2.21}$$

where k is the current iteration number and L, usually set as $L = 5$, corresponds to half of the "time" interval wherein convergence is checked. In other words, the index l, where $l = 1, ..., L$, is used to check convergence in an interval of $2L$ successive iterations. In the standard case of $L = 5$, eq. 2.21 is true if the convergence has been stable on $2L = 10$ iterations.

Steps in the optimization process

Step 1 Discrete the design domain with a rectangular mesh of bilinear 4-noded elements;

Step 2 Set boundary conditions regarding applied forces, prescribed displacements and supports;

Step 3 Set material parameters (E, ν);

Step 4 Set ESO parameters:

- target volume V^*,
- evolutionary ratio ER, to determine volume decrease in each iteration,
- radius of influence r_{min} for filtering sensitivities,
- allowable convergence error τ, to determine when to stop the ESO loop;

Step 5 Start iterations of ESO process, **while loop**;

Step 6 Determine the target volume for the current iteration with the formula: $max(V_i(1 - ER), V^*)$ where V^* is the end target volume;

Step 7 Execute finite element analysis;

Step 8 Calculate sensitivity numbers;

Step 9 Update sensitivity numbers by filtering and averaging with historical information;

Step 10 Update design (x matrix) switching material density from 1 to x_{min} and vice versa, according to ranking of sensitivity numbers;

Step 11 Repeat steps 6-10 until the end volume V^* is reached and the convergence criterion $error \leq \tau$ is satisfied;

Step 12 End ESO process.

2.2 Modified ESO formulation

The standard ESO formulation described in previous section has to be extended and modified for addressing the case of reinforced concrete. In fact, the required procedure has to respond to the following criteria:

Requirements for RC		Requirements for ESO procedure
multiple materials to optimize	$\Rightarrow$	more than one bi-phase material required (thus more than one x_i and α_i)
every material optimized with different criteria	$\Rightarrow$	different sensitivity numbers definitions for α_i required
different minimal sizes of members for each material	$\Rightarrow$	different radius of sensitivity filter r_{min} required

This implies that multiple ESO loops are needed. They could be arranged either *in parallel* or *in series*.

An arrangement *in parallel* can be found in the article [10], where multi-material stiffness optimizations are performed simultaneously in a single ESO process. The procedure goes on as follows. The materials are ranked by their stiffness $E_1 > E_2 > E_3$, (e.g. mat.1=steel, mat.2=concrete, mat.3=void). The optimization starts with mat.1=100% of volume in the structure, while mat.2 and mat.3 start at

0% of volume. The mat.1 is turned to mat.2 until the target volume for mat.1 is reached. After that, the volume of mat.1 is kept constant and mat.2 is turned to mat.3 until the target volume for mat.2 is reached, and so on in case of more materials. It is to note that, once a material reaches its target volume, only its volume is kept constant, while its distribution in the structure is allowed to change. In case a similar ESO process executed *in parallel* had to be applied to reinforced concrete, the direction of transformation would be: steel → concrete → void.

The arrangement *in series*, however, brings the following advantages. There can be as many loops as desired. Materials can be processed/transformed in any order desired, as far as the two material phases for any ESO loop are ordered according to their stiffness $E_1 > E_2$. It is possible to separate the two optimization problems of outer geometry topology definition and inner material distribution determination. In the specific case of reinforced concrete, this translates into the possibility of separating the problem of reinforced concrete structure shape optimization and the problem of steel-concrete internal distribution finding.

Therefore, the solution chosen in the thesis is to execute several ESO loops *in series*.

To obtain the formulation of the modified ESO procedure, new sensitivity numbers definitions have to be introduced. Additionally, some remarks have to be made on the material interpolation schemes to be used. These two points will be addressed below in this section and, after that, the steps of the modified ESO process will be shown. The filtering scheme and convergence criteria formulations used in the modified ESO procedure will be the same as the ones for the standard procedure.
It is important to specify that, since the ESO cycles performed in series will be in fact executed separately, in each ESO loop a problem formulation as in the standard case of single bi-phase material will be considered.

Material interpolation scheme

In each ESO cycle it is possible to choose the two phases for the bi-phase material between three materials: steel, concrete and void, as long as the first phase is chosen stiffer than the second phase. According to the presence or absence of void in the bi-phase material, it is chosen the appropriate material interpolation scheme from:

$$\begin{array}{lll} \text{mat. / void:} & E(x_i) = x_i^p E & \forall i = 1, ..., N \\ \text{mat.1 / mat.2:} & E(x_i) = x_i^p E_1 + (1 - x_i^p) E_2 & \forall i = 1, ..., N \quad \text{with: } E_1 > E_2 \end{array}$$

Sensitivity number formulation

In the present thesis, three optimization criteria are used:

- Von Mises criterion
- Drucker-Prager criterion
- Tensile stress criterion

The above criteria are not found in ESO literature applied to the current version of ESO procedure (post 2008). But, it is only possible to find applications of Von Mises and Tensile stress criterion to the first ESO method which did not make use of sensitivity numbers, and executed only element removal, with an overall different procedure.
Therefore, in the present thesis, a heuristic choice for the sensitivity numbers definitions of the above criteria is being made by the author.

Sensitivity number formulation - Von Mises criterion

Starting from the definition of the Von Mises criterion:

$$F(\sigma) = \sqrt{J_2(\sigma)} - H \leq 0 \tag{2.22}$$

where J_2 is the second invariant of the deviatoric stress tensor:

$$J_2(\sigma) = \frac{1}{3}\sigma^T V \sigma \tag{2.23}$$

where, in plane stress condition:

$$\sigma = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix}$$

$$V = \begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

and H is a parameter to represent material strength. Usually H is defined as:

$$H = \frac{\sigma_y}{\sqrt{3}}$$

and σ_y is called yield strength of the material.

In ESO literature [2], Von Mises criterion has been applied to the first ESO version, where elements were removed according to a rejection ratio RR_k, updated every iteration k, with a criterion $\frac{\sigma_{i,vm}}{\sigma_{max,vm}} < RR_k$, where $i = 1, ..., N$ is the index of the element.

In the following it is applied the Von Mises criterion on the latest version of ESO method and it is chosen a definition of sensitivity number as follows:

$$\boldsymbol{\alpha_i = \sqrt{J_2}\, x_i^{p-1}} \tag{2.24}$$

Sensitivity number formulation - Drucker-Prager criterion

Starting from the definition of the Drucker-Prager criterion:

$$F(\sigma) = \aleph I_1(\sigma) + \sqrt{J_2(\sigma)} - H \leq 0 \tag{2.25}$$

where J_2 is the second invariant of the deviatoric stress tensor (as in eq. 2.23), and I_1 is the first invariant of the stress tensor:

$$I_1(\sigma) = W^T \sigma \tag{2.26}$$

where, in plane stress condition:

$$\sigma = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix}$$

$$W = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

and H and $\aleph$ (aleph) are the parameters that represent material strength, usually defined as follows:

$$H = \frac{2\sigma_c\sigma_t}{\sqrt{3}(\sigma_c + \sigma_t)} \qquad \aleph = \frac{\sigma_c - \sigma_t}{\sqrt{3}(\sigma_c + \sigma_t)}$$

and σ_c and σ_t are respectively the compressive and tensile strength of the material. In the limit case when $\sigma_c = \sigma_t = \sigma_y$, the material parameters become $H = \frac{\sigma_y}{\sqrt{3}}$ and $\aleph = 0$, thus the Drucker-Prager criterion becomes equivalent to the Von Mises criterion.

In literature, Drucker-Prager criterion has been used in the paper of Kang [16] in a mixed technique optimization method comprising a gradient-based optimization algorithm in conjunction with the adjoint-variable sensitivity analysis and MMA algorithm to update the design variables. In the above paper it is used a "rigorous" derivative of the function 2.25 and an aggregation process to use the criterion at global scale.

In the thesis it is chosen a sensitivity number defined as follows:

$$\boldsymbol{\alpha_i = (\aleph I_1 + \sqrt{J_2})\, x_i^{p-1}} \tag{2.27}$$

Sensitivity number formulation - Tensile stress criterion

In literature, in the article by Xie of 1999 [12], the optimal design for a tension (or compression) dominated structure has been addressed with the first version of ESO, where elements were removed according to a rejection ratio RR_k, updated every iteration k, with a criterion $|\sigma_{i,11}| < RR_k\,|\sigma_{max,11}|$ *when* $\sigma_{i,22} \leq 0$, where $i = 1, ..., N$ is the index of the element. With $\sigma_{i,11}$ and $\sigma_{i,22}$ it is indicated, respectively, the first and second principal stress of the i^{th} element, expressed as:

$$\sigma_1 = \frac{\sigma_{xx} + \sigma_{yy}}{2} + \tau_{max} \qquad \sigma_2 = \frac{\sigma_{xx} + \sigma_{yy}}{2} + \tau_{min}$$

with:

$$\tau_{max} = \sqrt{\frac{\sigma_{xx} - \sigma_{yy}}{2}^2 + \sigma_{xy}^2} \qquad \tau_{min} = -\sqrt{\frac{\sigma_{xx} - \sigma_{yy}}{2}^2 + \sigma_{xy}^2}$$

where, for brevity, it is omitted the index i indicator of the element. In another example in literature, in the article of Xie of 2005 [17], the tension criterion had been updated and rewritten as $\sigma_{i,11} + \sigma_{i,22} < \sigma_{th}$, where σ_{th} is a stress threshold, that takes into account the maximum allowable stress, the rejection ratio and the volume removal rate of current iteration. For more details about the formulation, please refer to the original article [17]. For the latest formulation of ESO, with sensitivity numbers and material removal/addition process, a correspondent formulation has not been given yet.

In the thesis, it is used a sensitivity number formulation defined as follows:

$$\boldsymbol{\alpha_i} = \Big(\boldsymbol{\sigma_{1,i}} + \boldsymbol{\sigma_{2,i}} + \big|\,\mathbf{min}(\boldsymbol{\sigma_1})\big| + \big|\,\mathbf{min}(\boldsymbol{\sigma_2})\big|\Big)\,\boldsymbol{x_i^{p-1}} \tag{2.28}$$

Steps in the optimization process

Step 1 Discrete the design domain with a rectangular mesh of bilinear 4-noded elements, with possibility of defining holes of various forms;

Step 2 Set boundary conditions on external forces, prescribed displacements and constraints;

Step 3 Set the number of materials and the material parameters (E, ν, ratio between σ_c and σ_t);

Step 4 Set concrete cover type and size, if desired;

Step 5 Set the number of ESO cycles to be performed in series and set, for each cycle, two materials which will compose the bi-phase material during optimization, starting material density field x_i, material density mask which will influence x_i, and ESO parameters such as:

- total volume of current bi-phase material,
- target end volume V^* for first component of bi-phase material,
- evolutionary ratio ER, to determine volume decrease in each iteration,
- optimization criteria, to determine sensitivity numbers α_i,
- radius of influence r_{min} for filtering sensitivities,
- allowable convergence error τ, to determine when to stop each ESO cycle;

Step 6 Set mask definition/update rule, if desired;

Step 7 Start ESO process, **for loop**;

Step 8 Start iterations of current ESO cycle, **while loop**, selecting the chosen two materials and loading the material density field x_i and the mask field;

Step 9 Determine the target volume for current iteration with $max(V_i(1 - ER), V^*)$, where V^* is the end volume of current ESO cycle;

Step 10 Execute finite element analysis;

Step 11 Post-process to determine stresses;

Step 12 Calculate sensitivity numbers (mask is applied on sensitivities during their calculation);

Step 13 Update sensitivity numbers by filtering and averaging with historical information;

Step 14 Enforce voids and solids (parts that cannot be perforated), overwriting the values of sensitivity matrix in the prescribed positions;

Step 15 Enforce concrete cover, overwriting sensitivity numbers of related elements (and adjust total volume for iteration, in the moment when concrete cover is applied);

Step 16 Update design, switching material density from 1 to x_{min} and vice versa, according to ranking of sensitivity numbers;

Step 17 Repeat steps 9-16, until the end volume V^* of current ESO cycle is reached and the convergence criterion $error \leq \tau$ is satisfied;

Step 18 End current ESO cycle;

Step 19 Repeat steps 8-18, until all ESO cycles are performed;

Step 20 End optimization process, the resulting design is the solution to the ESO problem;

Step 21 Post-process to determine final stresses, if stress plots are desired.

2.3 Strategy and key choices

Overall procedure strategy

The chosen strategy in the thesis is to separate the problem of outer structure boundaries optimization and the problem of internal steel/concrete distribution definition. By trial and error, it is chosen the following approach:

- the first ESO loop is used to determine the optimized outer boundaries of the reinforced concrete structure as a whole, starting with the whole geometry filled by steel, using a bi-phase material constituted by steel and void, and using the Von Mises optimization criterion to transform steel into void;
- the second and third ESO loops are used to determine the internal steel/concrete distribution, starting with the optimized structure geometry from the first ESO loop, constituted by steel, using a bi-phase material constituted by steel and concrete, and using a transition from Von Mises to Drucker-Prager optimization criterion in the second loop and a transition from Drucker-Prager to Tensile stress optimization criteria in the third ESO loop, to transform steel into concrete.

Seeing the process from another point of view:

- in the first loop, the voids are inserted in the geometry to reduce its weight,
  void elements are inserted in lowly stressed zones, irrespective of the sign of stresses, tensile (+) or compressive (-) ones;
- in the second loop, the optimal positions and directions of steel members are determined,
  concrete is inserted in lowly stressed zones at the beginning (starting from Von Mises criterion) irrespective of the sign of stresses (+) or (-) and slowly favoring the addition of concrete in compressive (-) zones over the tensile (+) ones (when it turns to Drucker-Prager criterion);
- in the third loop, the steel volume is reduced as much as possible and only steel members in tensile zones are kept,
  at first, the remaining steel in compressive (-) zones is substituted with concrete and secondly, concrete is added in lowly stressed tensile zones (+) around the steel members in tensile zones.

Summarizing:

ESO loop n.:	1^{st}	2^{nd}	3^{rd}
1^{st} phase material:	Steel	Steel	Steel
2^{nd} phase material:	Void	Concrete	Concrete
Criterion:	Von Mises	transition from Von Mises to Drucker-Prager	transition from Drucker-Prager to Tensile stress
Action:	Insert voids in lowly stresses zones (both + and -)	Find positions and directions of steel members, insert concrete in lowly stressed zones (more - than +)	Remove steel in compressive zones (-) and reduce size of steel members in tensile zones (+)

It is to note that, the choice of Von Mises criterion for the first ESO loop, highlights the "hidden" choice of optimizing the outer boundaries of the reinforced concrete structure, in such a way to have equally strong members in compression and tension.

Linearly interpolated sensitivity numbers

After executing tests of optimization with different criteria in each optimization loop, it has been noted that the abrupt change in sensitivity numbers between two consecutive loops introduced sudden changes in geometry and difficulties in convergence. By trial and error, it has been chosen to use linear interpolated sensitivity numbers from the second ESO loop onwards, to have a smooth change in sensitivities all over the ESO process. The sensitivity numbers for second and third ESO loop are defined as follows:

$$\alpha_{i,lin} = t_{len}\,\alpha_{i,a} + (1 - t_{len})\,\alpha_{i,b}$$

where $\alpha_{i,a}$ is the sensitivity number of current loop, $\alpha_{i,b}$ is the sensitivity number of previous loop, t_{len} is a parameter that defines the number of iterations over which the transition takes place, in the thesis set as $t_{len} = 20$.

Additional density matrix y

Since the final configuration of the geometry comprises three materials - i.e. steel, concrete and void - only one material density matrix x is not enough (because x matrix is comprised by "binary" parameters 1 and 0.001). Then, a new material density called y is defined and the following meaning is given to the x and y matrices, for their final configuration:

- y determines the geometry of the reinforced concrete structure as a whole, identifying with $y_i = 1$ the presence of structure and with $y_i = 0.001$ the presence of void;
- x determines the distribution of steel and concrete, in the zones where there is the reinforced concrete structure, identifying with $x_i = 1$ the presence of steel and with $x_i = 0.001$ the presence of concrete.

The final structure is determined by interpretation of the two matrices y and x together:

- $y_i = 1$ and $x_i = 1 \Rightarrow$ steel
- $y_i = 1$ and $x_i = 0.001 \Rightarrow$ concrete
- $y_i = 0.001$ and $x_i = 1 \Rightarrow$ void
- $y_i = 0.001$ and $x_i = 0.001 \Rightarrow$ void

In the thesis, the following color scheme is used to plot the final solutions:

- steel $\rightarrow$ black
- concrete $\rightarrow$ grey
- void $\rightarrow$ white

With the use of the matrix y, the implementation for the cases when the outer reinforced concrete structure is not to be optimized, results straightforwardly setting $y_i = 1 \ \ \forall i = 1, ..., N$, where N is the total number of elements.

The matrix y appears in the finite element calculation in the following form:

$$K_i^* = K_i \, y_i^p = (x_i^p \, K_{i,1} + (1 - x_i^p) \, K_{i,2}) \, y_i^p$$

where K_i is the elemental stiffness matrix of the i^{th} element, the additional indexes 1 and 2 on K_i indicate the number of material used for its computation, and p is the same penalty exponent used on x_i.

Additional "density" matrix *mask*

After trial and error, it has been chosen to add another density matrix called *mask* which stores the final configuration of previous ESO loop and uses it to influence current loop. The reasons and effects of such decision are written further in the chapter "Discussion". The additional density matrix *mask* is used in the calculation of sensitivity numbers with the following formula:

$$\alpha_i^* = \alpha_i \, mask_i^{pm}$$

where pm is an additional penalty exponent specially defined for the mask, usually set as $pm = 2$. By setting $pm = 0$, it is possible to eliminate completely the effect of the mask in the procedure.

Additional "density" matrix *passive*

As in the paper of Sigmund [7], it is used a matrix called *passive* of the same size of x, but filled with three possible values 0, 1 and 2. The value 0 does not make any effect. The value 1 correspond to manually enforcing a void element in such position, i.e. for defining openings (e.g. for the passage of pipes, windows or doors). The value 2 correspond to manually enforcing a non-void element in such position, i.e. for defining pavements, ceilings, containment walls, etc.

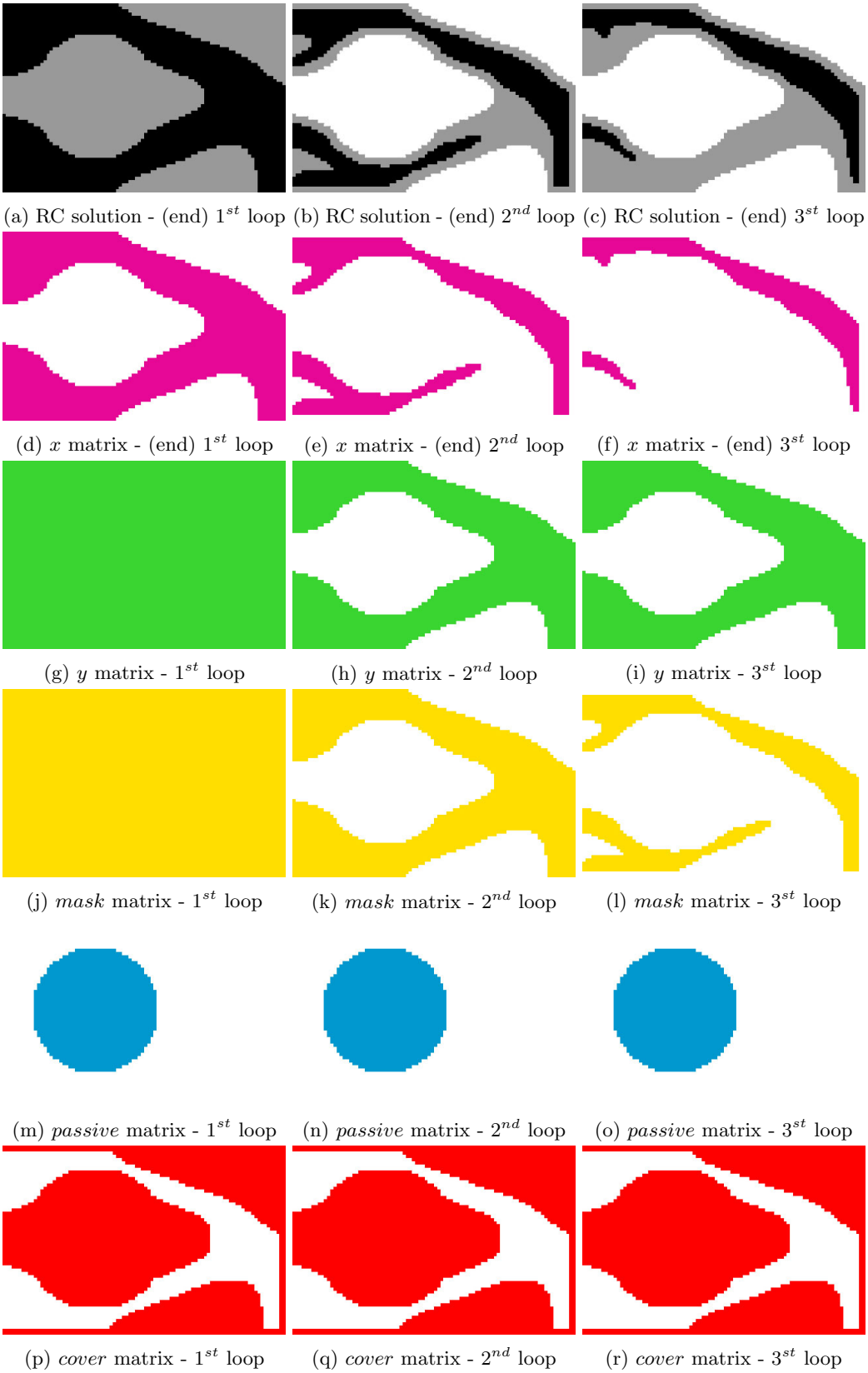
Since the design domain in the ESO procedure is always rectangular, it is necessary to use the matrix *passive* to determine voids in the standard square rectangular domain to define a non-rectangular starting geometry. As a result, any kind of starting geometry can be defined in the algorithm of the thesis by subtraction from a bigger rectangular domain.

Additional "density" matrix *cover*

In a similar way as for the matrix *passive*, an additional "density" matrix is introduced called *cover*. In this matrix, two possible values can be assumed by the elements: 0 and 1. The value 0 does not make any effect. The value 1 correspond to manually enforcing a non-steel element in corresponding position, defining the so-called concrete cover. The concrete cover is the set minimal thickness, between the surface of the embedded reinforcement and the outer surface of the reinforced concrete structure. The principal reasons for its application are:

- *chemical:* to prevent corrosion of steel,
- *structural:* to provide enough bonding strength (to ensure composite action and prevent slip),
- *thermal:* to provide thermal insulation of steel in case of fire.

The concrete cover in the thesis can be defined in two ways: manually and automatically. Setting the cover "manually" means defining an area (e.g. rectangular, circular, rhomboidal, oval) by setting manually its center and size and this must be done by the user himself/herself. Setting the cover "automatically" means that the cover is applied automatically by the algorithm in the procedure, without any input by the user, other than setting its thickness. In the latter case, the cover is applied directly around the voids generated in the structure. It is also possible to set various combinations of both manual and automatic covers. This is in most cases necessary, since the current automatic cover algorithm is not able to apply concrete cover around outer edges of the rectangular design domain, which have to be set and added manually.



(a) RC solution - (end) 1^{st} loop (b) RC solution - (end) 2^{nd} loop (c) RC solution - (end) 3^{st} loop

(d) x matrix - (end) 1^{st} loop (e) x matrix - (end) 2^{nd} loop (f) x matrix - (end) 3^{st} loop

(g) y matrix - 1^{st} loop (h) y matrix - 2^{nd} loop (i) y matrix - 3^{st} loop

(j) *mask* matrix - 1^{st} loop (k) *mask* matrix - 2^{nd} loop (l) *mask* matrix - 3^{st} loop

(m) *passive* matrix - 1^{st} loop (n) *passive* matrix - 2^{nd} loop (o) *passive* matrix - 3^{st} loop

(p) *cover* matrix - 1^{st} loop (q) *cover* matrix - 2^{nd} loop (r) *cover* matrix - 3^{st} loop

Figure 2.1: Case study n.5 - solutions and matrices x, y, *mask*, *passive*, *cover* for each ESO loop

Example of use matrices x, y, *mask*, *passive*, *cover*

For case study n.5 (analyzed and discussed in next chapters), the plots for the matrices x, y, *mask*, *passive*, *cover*, along with the reinforced concrete solutions (which are the results of the combination of x and y) are shown in figure 2.1 for each ESO loop. In the plots, the colored elements represent the value 1 and the white elements represent the values 0 or 0.001 (depending on which matrix is shown).

Distinction of materials in Finite Element Analysis

Another choice made, was to take into account the distinction of the different materials' elastic moduli and Poisson's ratios in the finite element analysis during optimization.

In case of a stiffness-based optimization as in [10], this is not a necessary choice for the validity of the multi-material optimization, since the differences in stiffness are already considered in the sensitivity number definition. And in the article of Xie [10] it is not stated if it has been taken into account also in finite element analysis or not.

In case of a stress-based optimization, it is possible as well to choose not to take into account the difference in stiffness in the finite element analysis, without affecting the validity of the multi-material optimization. In such case, it would be just needed to consider such difference in the post-processing function, when calculating the stresses used in the sensitivity numbers calculation.

In most cases, the ESO process can already take into account of material differences through the sensitivity numbers definitions. Therefore, the choice to regard or disregard the difference of materials stiffness (and Poisson's ratio) in the finite element analysis is not so obvious and can be considered to be a chosen assumption in the context of the ESO procedure.

Distinction of materials in Post-processing

To distinguish between the different materials in the post processing function, it is used a similar formula as for the material interpolation scheme:

$$\sigma_i = x_i^p \, D_1 \, B \, u_i + (1 - x_i^p) \, D_2 \, B \, u_i$$

Summarizing, the choices which can be seen as distinctive aspects in the thesis and/or novel approaches are grouped and listed here below.

Distinctive aspects in the thesis

Based on the literature research performed, it is possible to distinguish in this thesis some, what appear to be, novel approaches to ESO optimization:

- not only one optimization loop is performed, but a multiple number of loops is executed in series to perform a multi-material optimization;
- a concept of "mask" in the sensitivity analysis is introduced, which takes into account the field of sensitivity numbers of a chosen previous loop (or potentially a user defined field representing a preferred geometry) that influences the design update in combination with sensitivity numbers;
- transition between materials is carried on with different optimization criteria in the different optimization loops in series;
- the smooth changing of optimization criteria during an optimization loop is introduced, linear interpolating two different definitions of sensitivity numbers along a predefined number of iterations;
- alternative formulations of sensitivity numbers for Von Mises, Drucker-Prager and tensile stress criteria from the ones found in literature are used;
- Von Mises, Drucker-Prager and tensile stress criteria are used without taking into account the strengths of the materials, σ_y, σ_c and σ_t, but only the ratio between σ_c and σ_t;
- a method to enforce concrete cover automatically is introduced, which takes into account the evolution of boundaries of the structure as well.

The consequences and reasons of these choices are explained further in the chapter "Discussion".

2.4 MATLAB code explained

The basic ESO code, for single material stiffness optimization, has been taken from the articles of Sigmund [7] [8] and Xie [11] [18], and has been modified to accomplish the goals of the thesis according to the strategy and key choices outlined in previous sections.

The readers not interested in the details on the programming side, can skip the reading of current section from this point and continue from chapter 3 "Results".

The code is explained step by step here below, to enable the reader to gain enough familiarity for using, adapting, editing, and basically, working interactively with it.

Code structure

The MATLAB code is comprised by 5 files for the main code:

- Evolutionary Structural Optimization script - file *"ESOscript.m"*
- Finite Element Analysis function - file *"FEM.m"*
- Sensitivity Analysis function - file *"SENS.m"*
- Post-processing function - file *"POST.m"*
- Automatic concrete cover function - file *"COVER.m"*

and one file to plot results:

- Plot field (defined on element-level) script - file *"plotELEMfield.m"*

The file *"ESOscript.m"* is the main script to be executed in the command line of the MATLAB software. In this file, there is the code for the ESO process and the various inputs (materials, geometry, ESO parameters, etc.), to be edited for analyzing the various case studies of the thesis, or for adding new ones, or for computing alternative solutions. To execute the script in MATLAB, it is just needed to type in the command line:

```
ESOscript
```

The files *"FEM.m"*, *"SENS.m"*, *"POST.m"* and *"COVER.m"* are separate functions called automatically during the ESO process from inside the *"ESOscript.m"* file.

The script *"plotELEMfield.m"* can be executed in the command line of the MATLAB software, after the script *"ESOscript.m"* has been terminated, to plot a certain field, e.g. a selected stress or strain, over the optimized geometry, in its deformed or undeformed state. Such script needs to be called with some user input data, as in the following example:

```
plotELEMfield(ELEM,NODE,0,ELEM.sigma2,6)
```

where `ELEM` and `NODE` contain information about structure's geometry and displacements, `0` represents the set scale for displaying the deformation in the plot (in the example, no deformation is shown in the plot), `ELEM.sigma2` corresponds to the field to be plotted (in the example, the second principal stress defined on element-level) and `6` represents the chosen color scale for the plot (in the example, the red-blue colorbar with 0.5% of values removed from top and bottom of the scale). To understand the different colorbars available, please read the code of the *"plotELEMfield.m"* script.

Parameters organization in the code

Since the quantities are stored in MATLAB in several multidimensional arrays (column vectors, row vectors, matrices, tensors) it is important to understand to which point/element/d.o.f./node the values in a line or column refer to. Therefore, the variables in the MATLAB code has been grouped using the MATLAB class `struct` to avoid confusion with quantities defined on different variables. For example:

- the parameters `ELEM.ny` , `ELEM.n`, `ELEM.epsxx`, `ELEM.sigma1`, etc. are defined on the elements,
- the parameters `NODE.xy` , `NODE.n_mat`, `NODE.connect`, etc. are defined on the nodes,
- the parameters `DOF.K` , `DOF.f`, `DOF.u`, etc. are defined on the degrees of freedom (in the following order: node-1-x-dir, node-1-y-dir, node-2-x-dir, etc.),
- the parameters `GP.strains` , `GP.stresses` `GP.sigma2`, etc. are defined on the Gauss points,
- the parameters `ESO.er` , `ESO.criteria`, `ESO.rmin`, `ESO.startvol`, etc. are parameters that determine the ESO procedure,
- the parameters `SET.bctype` , `SET.solidvoid`, `SET.cover`, `SET.coverthick`, etc. are parameters used for settings,
- the parameters `GEOM.bc` , `GEOM.lenght`, `GEOM.height`, `GEOM.thickness`, are parameters that define the geometry of the structure,
- the parameters `VOID.E` , `VOID.nu`, `VOID.lambda`, `VOID.mu`, are defined for the material *void*,
- the parameters `STEEL.E` , `STEEL.nu`, `STEEL.lambda`, `STEEL.mu`, are defined for the material *steel*,
- the parameters `CONCR.E` , `CONCR.nu`, `CONCR.lambda`, `CONCR.mu`, are defined for the material *concrete*.

The values that are changed and set for each optimization cycle are stored in the MATLAB class `cell`. Those are:

- `x{i}`, the result material density matrix of the i^{th} ESO cycle;
- `startx{i}`, the initial material density matrix set for (and to be modified during) the i^{th} ESO cycle;
- `y{i}`, the material density matrix used to define reinforced concrete structure boundaries during the i^{th} ESO cycle;
- `mask{i}`, the matrix used to influence the sensitivity numbers during the i^{th} ESO cycle;
- `MAT1{i}` and `MAT2{i}`, respectively the two materials chosen for the bi-phase material for the i^{th} ESO cycle.

2.4.1 ESOscript.m code explained

The file `ESOscript.m` is explained, step by step, in the text below. It is comprised of four main parts: data input, initialization, optimization and post-processing.

ESOscript.m - part n.1: Data input

```
%-1---INPUT-------------------------------------------------------------
```

The first part of the code is dedicated to the input of data.

```
%-1.1---INPUT-GEOMETRY-------------------------
ELEM.nx = 100; ELEM.ny = 60; %number of elements used to discretize design area
GEOM.lenght = ELEM.nx; GEOM.height = ELEM.ny;  %length and height of design area
GEOM.thickness = 1;
```

The design domain is discretized with a rectangular mesh of size `ELEM.nx` along x direction and `ELEM.ny` along y direction. The real length is defined by the parameters `GEOM.lenght` and `GEOM.height`. In the attached code `GEOM.lenght` and `GEOM.height` are automatically set to `ELEM.nx` and `ELEM.ny`. It is possible to set a scale by specifying a multiplicator in the definition of `GEOM.lenght` and `GEOM.height`, for example defining: `GEOM.lenght = 2*ELEM.nx`, or `GEOM.lenght = 5*ELEM.nx`, etc.

```
%-1.2---INPUT-SETTINGS------------------------
SET.inttype=1; %1=2x2integration, 2=analytical
SET.bctype=2; %1=manual, 2=cantilever with central force, 3=half MBB, 4=half wheel,
              %5=cantilever with bottom force ... see section 1.5 for others
SET.solidvoid=1; %1=no solids and voids, 2=round hole, 3=square hole, 4=rhombus hole,
                 %5=rectangular hole ... see section 1.6 for others
SET.boundaries=2; %1=keep original boundaries, 2=optimize boundaries
SET.covertype=8; %1=no concrete cover, 2=top&bottom, 3=left&right, 4=automatic cover,
                 %5=automatic cover + top&bottom, 6=automatic cover + left&right
                 %7=automatic cover + top&bottom + left&right
                 %8=automatic cover + top&bottom + right
SET.coverthick=2; %concrete cover thickness (expressed in number of elements)
SET.translenght=20; %n. of iterations for transition between two optimization criteria
```

The parameter `SET.inttype` enables to choose finite elements integration type: 1 corresponds to numerical Gauss integration with 2x2 integration scheme and 2 corresponds to analytical integration as in the paper of Sigmund [8]. In the thesis, numerical integration is always used.

The boundary conditions and voids inside the rectangular domain are predefined for some basic geometries of the case studies from literature and can be selected choosing the parameters `SET.bctype` and `SET.solidvoid`. New geometries can be set manually. New preset geometries can be added in the section 1.5 and section 1.6 of the code `ESOscript.m`, simply adding new "case" inside the MATLAB command "switch". With `SET.solidvoid=1` there are no voids imposed on the structure. With `SET.bctype=1` the boundary conditions can be set completely manually.

With `SET.boundaries` it is specified if the outer geometry of reinforced concrete has to be optimized or has to be kept unmodified "rectangularly-shaped" as specified originally in `SET.bctype` and `SET.solidvoid`.

A concrete cover can be set in a similar way as for voids inside the geometry (what we will refer to as manual mode) and/or in an automatic way using the function written in the file `COVER.m`. Two parameters `SET.cover` and `SET.coverthick` are defined respectively to choose the type and the size of the concrete cover (expressed in number of elements). With `SET.cover=1` there is no concrete cover imposed. With `SET.cover=2,...,3` there is manual concrete cover applied, as defined in section 1.7 of the code `ESOscript.m`. With `SET.cover=4,...,8` there is an automatic concrete cover imposed, with or without an additional manual concrete cover, as defined in the function in file `COVER.m`.

The parameter `SET.translenght` is used for the optimization criteria n. 5, 6, 7 (defined in file `SENS.m`) where, not only one criterion for optimization is used, but two criteria linearly interpolated along a set number of iterations, defined by `SET.translenght`. Taking the example of optimization criteria n.5, the sensitivity numbers start from Von Mises optimization criterion with 100% influence and slowly transition to Drucker-Prager criterion, until the Von Mises criterion has 0% influence on the sensitivity numbers and Drucker-Prager criterion has the full 100% influence. This transition is completed once `SET.translenght` iterations have been performed inside the ESO loop.

```
%-1.3---INPUT-MATERIAL-DATA-------------------
VOID.E  = 0;
VOID.nu = 0;
VOID.lambda = 0;
VOID.mu     = 0;
STEEL.E  = 10.0;
STEEL.nu = 0.3;
STEEL.lambda = STEEL.nu*STEEL.E/( (1+STEEL.nu)*(1-2*STEEL.nu) );
STEEL.mu     = STEEL.E/( 2*(1+STEEL.nu) );
CONCR.E  = 1.0;
CONCR.nu = 0.2;
CONCR.lambda = CONCR.nu*CONCR.E/( (1+CONCR.nu)*(1-2*CONCR.nu) );
CONCR.mu     = CONCR.E/( 2*(1+CONCR.nu) );
% factors for drucker prager criterion
sigmac=1;
sigmat=0.2;
```

```
DPalpha=(sigmac-sigmat)/(sqrt(3)*(sigmac+sigmat));
clear sigmac sigmat
% for selfweight problems only
gravity =  [ 0.0    %gravity acceleration x-dir
            -1.0];  %gravity acceleration y-dir
VOID.density= 0.0; %material density (rho)
VOID.bodyforce = VOID.density*gravity;  %specific weight(gamma=rho*g)
STEEL.density= 0.0;
STEEL.bodyforce = STEEL.density*gravity;
CONCR.density= 0.0;
CONCR.bodyforce = CONCR.density*gravity;
clear gravity
```

Material parameters are defined for the materials: void, steel and concrete.

The factors for Drucker-Prager criterion has to be inserted in the parameters `sigmac` and `sigmat`. One of them can be expressed as unitary value and the other as the ratio between the two, since they are only used to calculate `DPalpha` the parameter $\aleph$ used in the Drucker-Prager principle, which is only dependent to their ratio. In the example code the ratio between the `sigmac` and `sigmat` is set to 20%.

There is a section of code for properties to be defined for selfweight problems. Even though in this thesis they are not treated, the code already supports such implementation, with the remark that the sensitivity numbers must be changed and defined accordingly. For principles and examples on how to derive sensitivity numbers definitions for design-dependent optimization problems, see the article of Xie [11].

```
%-1.4---INPUT-ESO-PARAMETERS--------------------------
%-1.4.1--INITALIZATION-ESO-PARAMETERS----
x0=ones(ELEM.ny,ELEM.nx); %start geometry
ESO.cycles=3; %determine the number of cycles
x=cell(1,ESO.cycles); %initialize eso density matrix
for cycle=1:ESO.cycles; x{cycle}(1:ELEM.ny,1:ELEM.nx)=1.; end
ESO.penal = 3.; %penalty exponent applied to x
ESO.penalmask = 2.; %penalty exponent applied to mask
%set ESO.penalmask=2 to have same effect as of ESO.penal on x
%set ESO.penalmask=0 to nullify effect of mask (works only for SET.covertype=1,2,3)
%-1.4.2--INPUT-VARIABLE-ESO-PARAMETERS----------------------
%ESO settings (1 column = 1 cycle) to be changed for each case study
ESO.volfrac=          [0.60,        0.30,          0.15           ];
%-1.4.3--INPUT-FIXED-ESO-PARAMETERS---------------------------------
%ESO settings (1 column = 1 cycle) usually not needed to be changed
ESO.conv_err=         [0.001        0.001          0.001          ]; %allowable conv. error
ESO.er=               [0.04,        0.03,          0.02           ];
ESO.rmin=             [8,           3,             2              ];
ESO.criteria=         [2,           5,             7              ];
MAT1=                 {STEEL,       STEEL,         STEEL          };
MAT2=                 {VOID,        CONCR,         CONCR          };
ESO.startvol=         [1,           ESO.volfrac(1), ESO.volfrac(2) ];
update_startx=@(x0,x) {x0,          x{1},          x{2}           }; %values of ESO.start
update_mask=@(x0,x)   {x0,          x{1},          x{2}           }; %values of mask
switch SET.boundaries   % set values of y (concrete boundaries)
case 1   % preserve original boundaries
update_y=@(x0,x)      {x0,          x0,            x0             };
case 2   % optimize boundaries
update_y=@(x0,x)      {x0,          x{1},          x{1}           };
end
% startx, y and mask are set in the beginning and updated inside the eso loops
% at end of each main-iteration because x{1}, x{2} etc are calculated inside the
% eso loops, to do this MATLAB function handles are used
```

```
startx=update_startx(x0,x);
y=update_y(x0,x);
mask=update_mask(x0,x);
% In y it is stored the information about concrete boundaries
```

A neutral mask and initial full density matrix is defined with `x0` setting all values to 1.

The number of ESO cycles is defined with `ESO.cycles`. The number of cycles can be increased or decreased changing the size of the arrays `ESO.conv_err`, `ESO.volfrac`, `ESO.er`, `ESO.criteria`, `ESO.startvol`, of the cells `MAT1`, `MAT2` and of the handle functions `update_startx`, `update_mask` and `update_y`.

A penalty exponent (`ESO.penal`) of 3 is set on the density matrix, as in the article of Xie [11]. On the mask it is set a penalty exponent (`ESO.penalmask`) of 2. However, the exponent `ESO.penalmask=2` has the same effect on mask as the exponent `ESO.penal=3` has on the density matrix, because in the sensitivity numbers definitions the `x` is to the power of `penal-1` while the mask is to the power of `penalmask`.

The parameters `ESO.volfrac` and `ESO.er` respectively defines the target volume for each ESO loop and the speed with which the volume is decreased in each iteration for each ESO loop. In the example, `ESO.volfrac=0.6` it means that the target volume is 60% of the total volume and `ESO.er=0.04` means that the volume is decreased in each iteration for the 4% of the volume in previous iteration. This means that volume in the example changes with the following values: 100%, 96%, 92.2%, 88.5%, 84.9%, ..., 60.0%. The parameter `ESO.volfrac` is the only ESO parameter that needs to be changed for each case study to suit the particularities of each example problem. Depending on each case study it is wished to reach a certain reinforced concrete total volume, or to preserve a certain simplicity of the outer structure boundaries (not to push the optimization too much). Also the target steel volume can be lowered or must be increased depending on complexity of steel layout. A more complex steel layout usually requires a higher target volume, not to turn to a trivial solution. Further discussion about this aspect is carried in the chapter "Discussion".

The parameter `ESO.conv_err` determine the allowable convergence error, it is advised to keep its value to 0.001 unless the number of iterations needed to complete loops starts to be higher than 40-50. In such case, to lower the computational expenses, it is viable to change its value to 0.002 or 0.003 to save considerable computational time, in exchange of a solution less close to the convergent one, which in case of ESO optimization would not show remarkable difference in most of the cases.

With `ESO.rmin` is possible to set the size of the filter scheme, to avoid checkerboard patterns. Setting `ESO.rmin=1` is equivalent to disable the filter on sensitivities. In simple terms, the effect given by this parameter is that the bigger the `ESO.rmin` number, the bigger the minimal size of members inside the resulting design. All the members with "thickness" smaller than the size defined by `ESO.rmin` (expressed in number of finite elements) will be cut/removed from the design. This is an important parameter to correctly drive the solution for the reinforced concrete designs for the following two reasons: to avoid creating too much slender concrete members and to prevent removal very thin but relevant steel parts inside of the concrete.

With `ESO.criteria` is possible to choose the optimization criteria (defined in the file `SENS.m`) to be used during the optimization loops with the following numbers:

1. Stiffness criterion
2. Von Mises criterion
3. Drucker-Prager criterion
4. Tensile stress criterion
5. Transition from Von Mises to Drucker-Prager criterion
6. Transition from Von Mises to tensile stress criterion
7. Transition from Drucker-Prager to tensile stress criterion

Other criteria, outside this list, could be added inserting a new "case" in the MATLAB command "switch" in the file `SENS.m`, with the relative expression for calculating sensitivity numbers.

With the MATLAB cells `MAT1`, `MAT2` it is defined the first and second material that take part in the bi-phase material for each ESO optimization loop.

The parameters `ESO.startvol`, `update_startx`, `update_mask` and `update_y` define for each ESO optimization loop, respectively, the start volume, the associated material density matrix, the mask used to steer the optimized solution and the matrix defining the (reinforced concrete) structure boundaries. In the example, it is set that in the first ESO loop the start volume is 100%, and both the density matrices `startx`, `y` and `mask` are the unitary matrices. In the second ESO loop the start volume and density matrix are set from the end results of the first ESO loop and, similarly, in the third loop, using end results of second loop. The density matrix `y`, however, is set to be kept unitary, in case the structure boundaries are to be kept unmodified or is set to be equal to the result of first ESO loop, in case the structure boundaries have to be optimized.

Handle functions are used so that the parameters `startx`, `mask` and `y` can be easily updated after each ESO cycle inside the ESO optimization loop, in an automatic way with the calls of the handle functions.

```
%-1.5---INPUT-BOUNDARY-CONDITIONS--------------
% GEOM.bc=[# # # #]'=[node n., bctype, dof, bcvalue]'
% (bctype=0 for force, bctype=1 for displacement)
switch (SET.bctype)
...
see annex for rest of code
```

Boundary conditions are set through a matrix where, in each line, it is specified a condition on a node, with four parameters: number of node, type of condition (0=force, 1=displacement), degree of freedom (1=x-direction, 2=y-direction) and value of force / prescribed displacement (value=0 for constraint). Such matrix is transposed and assigned to a matrix called `GEOM.bc`. Conditions are applied in file `FEM.m` during finite element analysis, applying prescribed forces on the `f` vector before solving the system and not resolving equations with prescribed displacements and applying them to solutions in vector `u`. Boundary conditions can be defined manually or chosen from the following presets:

1. manually defined;
2. cantilever with downward force on center of right edge;
3. half "MBB" beam with downward force in middle of full beam;
4. "half wheel design" beam with downward force in middle of lower edge;
5. cantilever with downward force in lower right corner;
6. plate fixed on four edges with upward force in the middle of upper half of plate;
7. plate fixed on four edges with downward force in the center of plate;
8. wall for case study n.9, defined for mesh of 98 x 60 elements only (not scalable), with two downward forces applied on 1/3 and 2/3 of upper edge;
9. wall for case study n.10, defined for mesh of 150 x 94 elements only (not scalable), with one eccentric downward forces applied upper edge
10. corbel jointed to a column of case study n.11, defined for mesh of 88 x 216 elements only (not scalable), to be used in combination with `SET.solidvoid=8`;
11. corbel with a ledge support of case study n.12, defined for mesh of 88 x 88 elements only (not scalable), to be used in combination with `SET.solidvoid=9`;

Presets defined from n.2 to n.7 can be used for any proportion and size of meshes (for any scale as well), without need to make any change on the code. Therefore, such presets are readily usable for mesh refinement studies on the code developed in the thesis.

```
%-1.6---VOIDS-AND-SOLIDS-IN-GEOMETRY-----------
% set ESO.passive = 1 (holes) or 2 (solids)
switch (SET.solidvoid)
...
see annex for rest of code
```

Holes on the geometry (for example for passage of pipes, for windows, doors, etc.) are applied through a matrix `ESO.passive` which is enforced on the matrix of sensitivities in the ESO loop, just after filtering and before the design update step. With the same procedure, there can be imposed on the structure some parts that must remain solid and cannot be perforated during optimization. The latter are referred to in this thesis as "solids". Holes and solids can be defined semi-automatically from the following presets:

1. no holes, no solids;
2. circular hole/solid(*);
3. square hole/solid(*);
4. rhombus hole/solid(*);
5. rectangular hole/solid(*);
6. two rectangular holes for case study n.9, for mesh of 98 x 60 elements only (not scalable), to be used in combination with boundary conditions `SET.bctype=8`;
7. one eccentric rectangular hole for case study n.10, for mesh of 150 x 94 elements only (not scalable), to be used in combination with boundary conditions `SET.bctype=9`;
8. two rectangular holes (to create a colum + corbel structure from a rectangular domain), for case study n.11, for mesh of 88 x 216 elements only (not scalable), to be used in combination with boundary conditions `SET.bctype=10`;
9. one rectangular holes for case study n.12 (to create a corbel structure from a rectangular domain), for mesh of 88 x 88 elements only (not scalable), to be used in combination with boundary conditions `SET.bctype=11`.

In the cases marked with * it is needed to define manually diameter and center of hole. It possible to choose to enforce holes or solids, setting `ESO.passive=1` for holes and setting `ESO.passive=2` for solids.

```
%-1.7---CONCRETE-COVER------------------------
switch (SET.covertype)
...
see annex for rest of code
```

Concrete cover can be defined manually in the same way as holes are defined in part of code 1.6, but these are "holes" that apply only to the steel density matrices. They are applied through a matrix `ESO.cover` in the ESO loop just after the holes/solids are enforced to the sensitivity matrix and before the design update step. The presets in the code are the following:

1. no concrete cover;
2. concrete cover in top and bottom of rectangular design domain;
3. concrete cover in left and right of rectangular design domain;
4. automatic concrete cover around holes;
5. automatic concrete cover around holes + top and bottom cover on rectangular design domain;
6. automatic concrete cover around holes + left and right cover on rectangular design domain;
7. automatic concrete cover around holes + top, bottom, left and right cover on rectangular design domain;
8. automatic concrete cover around holes + top, bottom and right cover on rectangular design domain.

The size of the concrete cover is determined in n. of elements by the parameter `SET.coverthick`.

ESOscript.m - part n.2: Initialization

```
%-2---INITIALIZATION-------------------------------------------------------
```

The second part of the code is dedicated to the initialization of geometry and the creation of the mesh, the definition of the integration scheme and the pre-calculations for the filtering scheme.

```
%-2.1---INITIALIZATION-GEOMETRY------------------
% creation of node and dof numbers
% - matrix of nodes numbers
NODE.n_mat = reshape(1:(1+ELEM.nx)*(1+ELEM.ny),1+ELEM.ny,1+ELEM.nx);
% - node number on bottom right corner
DOF.n_mat_col_1 = reshape(2*NODE.n_mat(1:end-1,1:end-1)+1,ELEM.nx*ELEM.ny,1);
% - matrix of dof numbers in the element
DOF.n_mat = repmat(DOF.n_mat_col_1,1,8)+repmat([0 1 2*ELEM.ny+[2 3 0 1] -2 -1],ELEM.nx*ELEM.ny,1);
% creation of matrix of nodes coordinates (order top to bottom, right to left)
[NODE.x_mat,NODE.y_mat] = ...
                  meshgrid(linspace(0,GEOM.lenght,ELEM.nx+1),linspace(0,-GEOM.height,ELEM.ny+1));
NODE.xy = [NODE.x_mat(:) NODE.y_mat(:)];
NODE.n = size(NODE.xy, 1);
% creation of connections of nodes for members (order top to bottom, right to left)
NODE.connect = [(DOF.n_mat(:,2)/2) (DOF.n_mat(:,4)/2) (DOF.n_mat(:,6)/2) (DOF.n_mat(:,8)/2)];
ELEM.n = size(NODE.connect, 1);
```

Given the number of elements along the x and y direction, the mesh is built assembling the matrix `NODE.n_mat`, with the node numbers placed in the right geometrical order as in the mesh. Each row and column in the matrix correspond respectively to a row and column of nodes in the finite element mesh.

From the latter, it is a created another matrix called `DOF.n_mat`, to link the degrees of freedom to the elements (8 d.o.f. per element, per row). The order of enumeration is from bottom left node and in a counterclockwise direction. The 2 d.o.f. for each node are gathered in couples: 1&2, 3&4, 5&6, etc. Odd numbers correspond to d.o.f. in x direction, while even numbers to d.o.f. in y direction.

After that, the coordinates for each node are calculated and put into two matrices in corresponding positions to the nodes, the first for x coordinates `NODE.x_mat` and the second for y coordinates `NODE.y_mat`. In the matrix `NODE.xy` the nodes coordinates are reordered vertically in lines (2 coordinates per node, per row).

The matrix `NODE.connect` is assembled as well. Within each line of this matrix, there are stored the node numbers that form a corresponding finite element (4 nodes per element, per row). The order of enumeration is from bottom left node and in a counterclockwise direction. The enumeration convention for elements, nodes and d.o.f. is shown in the image 2.2 with:

- element n°, highlighted in yellow;
- node n°, in red font;
- d.o.f. n°, in black font.

The convention for numeration is the same as in the paper of Sigmund [7].
The coordinate system is set with origin in top left corner (origin on node n.1), with x axis pointing rightwards and y axis pointing upwards.

```
%-2.2---INITIALIZATION-INTEGRATION-SCHEME---------
% 2x2 point Gauss integr
GP.def = zeros(3,4);
GP.def(1,:) = 1.0; %weight of gp,
GP.def(2,1) = -1.0/sqrt(3.0); GP.def(3,1) = -1.0/sqrt(3.0); %eta and xi coord of gp
GP.def(2,2) =  1.0/sqrt(3.0); GP.def(3,2) = -1.0/sqrt(3.0);
GP.def(2,3) =  1.0/sqrt(3.0); GP.def(3,3) =  1.0/sqrt(3.0);
GP.def(2,4) = -1.0/sqrt(3.0); GP.def(3,4) =  1.0/sqrt(3.0);
```

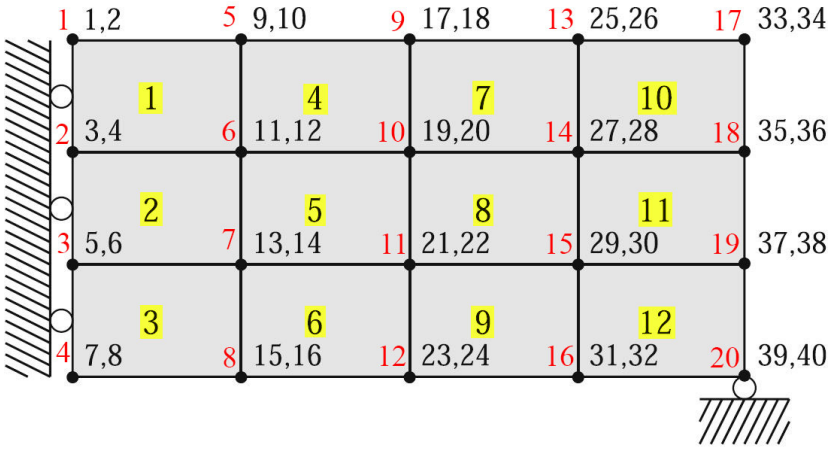


Figure 2.2: Numeration convention

Table 2.1: `NODE.n_mat`

1	5	9	13	17	row1
2	6	10	14	18	row2
3	7	11	15	19	row3
4	8	12	16	20	row4
col.1	col.2	col.3	col.4	col.5	

Table 2.2: `DOF.n_mat`

3	4	11	12	9	10	1	2	elem.1
5	6	13	14	11	12	3	4	elem.2
7	8	15	16	13	14	5	6	elem.3
11	12	19	20	17	18	9	10	elem.4
13	14	21	22	19	20	11	12	elem.5
15	16	23	24	21	22	13	14	elem.6
19	20	27	28	25	26	17	18	elem.7
21	22	29	30	27	28	19	20	elem.8
23	24	31	32	29	30	21	22	elem.9
27	28	35	36	33	34	25	26	elem.10
29	30	37	38	35	36	27	28	elem.11
31	32	39	40	37	38	29	30	elem.12

Table 2.3: `NODE.connect`

2	6	5	1	elem.1
3	7	6	2	elem.2
4	8	7	3	elem.3
6	10	9	5	elem.4
7	11	10	6	elem.5
8	12	11	7	elem.6
10	14	13	9	elem.7
11	15	14	10	elem.8
12	16	15	11	elem.9
14	18	17	13	elem.10
15	19	18	14	elem.11
16	20	19	15	elem.12

A standard 2x2 Gauss integration scheme is chosen with weight equal to 1 and integration points coordinates:

$$\left(\pm \frac{1}{\sqrt{3}}, \pm \frac{1}{\sqrt{3}} \right)$$

```
%-2.3---INITIALIZATION-MESH-INDEPENDENCY-FILTER-----
for cycle=1:ESO.cycles
iH = ones(ELEM.nx*ELEM.ny*(2*(ceil(ESO.rmin(ESO.cycles))-1)+1)^2,1);
jH = ones(size(iH)); sH = zeros(size(iH)); k = 0;
for i1 = 1:ELEM.nx
  for j1 = 1:ELEM.ny
    e1 = (i1-1)*ELEM.ny+j1;
    for i2 = max(i1-(ceil(ESO.rmin(cycle))-1),1):min(i1+(ceil(ESO.rmin(cycle))-1),ELEM.nx)
      for j2 = max(j1-(ceil(ESO.rmin(cycle))-1),1):min(j1+(ceil(ESO.rmin(cycle))-1),ELEM.ny)
        e2 = (i2-1)*ELEM.ny+j2; k = k+1;
        iH(k) = e1; jH(k) = e2;
        sH(k) = max(0,ESO.rmin(cycle)-sqrt((i1-i2)^2+(j1-j2)^2));
      end
    end
  end
end
ESO.H{cycle} = sparse(iH,jH,sH);
ESO.Hs{cycle} = sum(ESO.H{cycle},2); %the arrays needed for filtering are ESO.H and ESO.Hs
clear e1 e2 i1 i2 j1 j2 iH jH sH k cycle
end
```

The same filtering scheme used as in Sigmund [8] and Xie [11]. The filter is "prepared" before the ESO loops and applied during each iteration with simple matrix products. This is done in such a procedure to reduce computational efforts, with respect to original filter application as in previous papers of Sigmund [7] and Xie [18].

ESOscript.m - part n.3: Optimization

```
%-3---OPTIMIZATION-LOOP-------------------------------------------------------
for cycle=1:ESO.cycles
```

The third part of the code comprise the core of the optimization process. A `for` cycle is initialized to execute a series of consecutive ESO optimization loops. All what is written below is therefore executed for each ESO optimization loop. In the thesis three ESO loops are used.

```
  %-3.1--INITIALIZATION i-(th) ESO PROCEDURE-------------
  iter = 0; change = 1.;
  x{cycle}=startx{cycle};
  vol=ESO.startvol(cycle);
```

The initial sensitivity matrix and volume for the i^{th} ESO procedure is loaded. After that a `while` loop is started for the i^{th} ESO procedure.

```
  %-3.2--START ITERATIONS FOR i-(th) ESO PROCEDURE-------
  while change > ESO.conv_err(cycle)
```

Each ESO procedure performs a number of iterations inside the `while` loop until the optimization procedure is convergent.

```
%-3.2.1--volume and historical data
    iter = iter + 1;
    if iter >1
```

```
   ESO.oldapha = ESO.alpha;
end
if cycle==2 && iter==1 && SET.covertype>1
   %in first iter when it is applied cover don't reduce volume by ER
else
   vol = max(vol*(1-ESO.er(cycle)),ESO.volfrac(cycle));
end
```

In `ESO.oldapha` it is stored previous sensitivity matrix for historical averaging of sensitivities. With the formula `vol = max(vol*(1-ESO.er(cycle)),ESO.volfrac(cycle))`, the volume is decreased within each iteration and, once reached the target volume, kept stable. Only in the moment when the concrete cover is applied (first iteration of second ESO loop), the volume is chosen not to be reduced by the evolutionary ratio `ESO.er`, but only by the concrete cover application.

```
%-3.2.2--fem analysis
[DOF.K, DOF.f, DOF.u, ELEM.KE1, ELEM.KE2]=...
    FEM(ELEM,NODE,DOF,GEOM,GP,SET,MAT1{cycle},MAT2{cycle},x{cycle},y{cycle},ESO.penal);
%-3.2.3--postprocess for stress based eso criteria
if ESO.criteria(cycle) ~= 1
[GP.strains, GP.stresses, GP.xy]=...
 POST(ELEM,NODE,DOF,GP, MAT1{cycle}, MAT2{cycle},x{cycle},ESO.penal);
end
```

Finite element analysis and post processing are executed as MATLAB functions from the separate files `FEM.m` and `POST.m`.

```
%-3.2.4--sensitivity analysis
[ESO.alpha, SET] = ...
SENS(ELEM,DOF,GP,SET,ESO.criteria(cycle),x{cycle},mask{cycle},ESO.penal,ESO.penalmask,...
DPalpha,iter);
%calculate total objective function
ESO.obj(iter) = sum(sum(ESO.alpha));
%filtering sensitivities
ESO.alpha(:) = ESO.H{cycle}*ESO.alpha(:)./ESO.Hs{cycle};
%stabilizing sensitivities
if iter > 1
    ESO.alpha = (ESO.alpha+ESO.oldapha)/2.;
end
```

Sensitivities are calculated calling in the function from the separate file `SENS.m`. The object function is simply defined as the sum of all sensitivity numbers. The sensitivities are filtered through matrix multiplication with the arrays `H` and `Hs` calculated outside the `while` and `for` loops, in the section 2.3 of the code.

```
%-3.2.5--enforce voids and solids
ESO.alpha(ESO.passive==1) = min(min(ESO.alpha));
ESO.alpha(ESO.passive==2) = max(max(ESO.alpha));
%for cases when concrete boundaries are not optimized (SET.boundaries==1)
%enforce voids and solids directly in x0
if iter==1 && SET.boundaries==1
x0(ESO.passive==1) = 0.001; %enforced voids by codes in section 1.6
x0(ESO.passive==2) = 1;     %enforced solids by codes in section 1.6
end
```

The elements that are forced to be voids have their sensitivity number overridden as the minimum of all sensitivity values. Vice versa, it is used the maximum for the elements forced to be solids. In the cases when the outer geometry is kept not optimized, the voids and solids have to enforced on the initial density matrix `x0` (which is used to create the matrix `y`, i.e. the reinforced concrete boundaries, as seen in section 1.4.3 of the code).

```
%-3.2.6--enforce concrete cover
if cycle>1
   if SET.covertype>=4
       %compute automatic concrete cover (for cases SET.covertype>=4)
       %and apply steel-free zone outside concrete boundaries (done inside function COVER)
       [ESO.cover]=COVER(ELEM,SET,y,cycle);
   else
       %concrete cover already set before (for cases SET.covertype<4)
       %then only apply steel-free zone outside concrete boundaries
       ESO.alpha(y{cycle}~=1)=min(min(ESO.alpha));
   end
  if cycle==2 && iter==1 %only the first iteration on 2 cycle it adjusts volume
      removedsteel=0;
      for indx=1:ELEM.n
      %if there is steel inside concrete boundary and it needs to be
      %deleted due to application of concrete cover in first iteration
          if y{cycle}(indx)==ESO.cover(indx)
             removedsteel=removedsteel+1;
          end
      end
      %modify volume
      vol=vol-removedsteel/ELEM.n;
   end
   %concrete cover is enforced overriding ESO.alpha
   ESO.alpha(ESO.cover==1) = min(min(ESO.alpha));
end
```

The concrete cover is set to have effect only after the first ESO loop has been terminated, since it has been chosen that only voids are defined in the first loop and while the distribution between steel and concrete is determined in next loops.

- For automatic cover cases (`SET.covertype>=4`), it is executed the function from the file `COVER.m`, which fills the matrix `ESO.cover` with 1 in holes and around them with a thickness equal to `SET.coverthick`.
- For semi-manual cover cases (`SET.covertype<4`) it is not needed to fill the matrix `ESO.cover` because it already done in section 1.7 of the code, but it is only needed to enforce the voids also on the steel. This is done overriding the sensitivities with the following formula: `ESO.alpha(y{cycle}~=1)=min(min(ESO.alpha))`.

Moreover, in all cases when concrete cover is applied, it is needed to adjust the total volume, in the first iteration of the second loop (`if cycle==2 && iter==1`). This is done through a specially defined parameter `removedsteel` that counts the steel elements removed due to the concrete cover. The cover is applied overriding sensitivities with the following formula: `ESO.alpha(ESO.cover==1) = min(min(ESO.alpha))`.

```
%-3.2.7--beso design update
l1 = min(ESO.alpha(:)); l2 = max(ESO.alpha(:));
while abs((l2-l1)/(l1+l2)) > 1.0e-9
    th = (l1+l2)/2.0;
    x{cycle} = max(0.001,sign(ESO.alpha-th)) ;
    if mean(x{cycle}(:)) - vol > 0
        l1 = th;
    else
        l2 = th;
    end
end
clear l1 l2 th
```

Before the design update step is run for the first time in the code, it is known that all the elements are set in x to assume the first phase of the material, as assigned in section 1.4.3 of the code.

The ESO design update used is the same as in the paper of Xie [11] with two threshold parameters `l1` and `l2`, and works as explained here below. At first, it is set `l1` as the minimum of the sensitivity numbers from `ESO.alpha` and `l2` as the maximum. It is calculated a mean value `th`. With `sign(ESO.alpha-th)` it is found a matrix of -1 and +1 indicating corresponding elements with sensitivities, respectively, lower and higher than `th`. The elements with sensitivity lower than `th` are set to 0.001 in the matrix `x`. In other words, those elements are turned to the second phase of the material. The elements with sensitivity higher than `th` are set to 1 in the matrix `x`. In other words, those elements are kept to the first phase of the material. While the target volume is still to be reached (`mean(x{cycle}(:)) - vol > 0`), iteratively, the threshold `l1` is set equal to `th`, a new `th` is calculated and additional elements are removed. In case the target volume is crossed (`mean(x{cycle}(:)) - vol < 0`), the threshold `l2` is set equal to `th`, a new `th` is calculated and, from the elements already removed, the ones with sensitivity higher than the new value `th` are restored/re-added. This is done simply overriding their previous value of 0.001 in the matrix `x` with their new value 1. This process continues iteratively until the volume `mean(x{cycle}(:))` converges (oscillating from top and bottom) to the target volume `vol`. At that point, the values of `l1`, `l2` and `th`, as a matter of fact, assume the same value. Therefore, it is used the convergence criterion: `abs((l2-l1)/(l1+l2)) > 1.0e-9`.

```
    %-3.2.8--output results
    %print results
    if iter>10 && vol == ESO.volfrac(cycle)
    %if minimum 10 iterations are performed and target volume has been reached
    change=abs(sum(ESO.obj(iter-9:iter-5))-sum(ESO.obj(iter-4:iter)))/sum(ESO.obj(iter-4:iter));
    end
    disp([' It.: ' sprintf('%4i',iter) ' Obj.: ' sprintf('%10.4f',ESO.obj(iter)) ...
           ' Vol.: ' sprintf('%6.3f',sum(sum(x{cycle}))/(ELEM.nx*ELEM.ny)) ...
            ' ch.: ' sprintf('%6.3f',change )])
    %plot design
    if cycle ==1
    colormap(gray); imagesc(-x{cycle}); axis equal; axis tight; axis off;pause(1e-6);
    else
    colormap(gray); imagesc(-y{cycle}); hold on; fingeom=imagesc(-x{cycle}); hold off;
    alpha(fingeom,.6); axis equal; axis tight; axis off;pause(1e-6);
    end
  end   %end of while loop of i-(th) ESO procedure
  clear vol change iter
```

In this point of the code, the while loop of i^{th} ESO procedure ends. The results of the i^{th} ESO loop, i.e. the `x` and `y` matrices, are plotted contemporaneously with two `plot` functions, such that the color indicates the resulting material:

black = steel

gray = concrete

white = void

```
  %-3.3---UPDATE-x-y-mask-WITH-i-(th)-ITERATION-CALCULATIONS-
  %it is needed because x{1} x{2} etc. are calculated during the loops
  startx=update_startx(x0,x);
  y=update_y(x0,x);
  mask=update_mask(x0,x);
end  %end of for loop of the series of ESO optimizations
```

In the end of the i^{th} ESO procedure, having obtained the resulting density matrix `x{i}` of the i^{th} ESO loop, the matrices `startx`, `y` and `mask` are updated through the handle functions `update_startx`, `update_y` and `update_mask`.

After this point in the code, the `for` loop of the series of ESO optimization ends. Sections of code 3.1, 3.2, 3.3 are executed for each ESO loop, until the established number of ESO loops is reached. In the thesis, the total number of ESO loops is 3.

ESOscript.m - part n.4: Post-processing

```
%-4--POST-PROCESSING-------------------------------------------------------
[GP.strains, GP.stresses, GP.xy]=...
                        POST(ELEM,NODE,DOF,GP, MAT1{cycle}, MAT2{cycle},x{cycle},ESO.penal);
% assign nan value to void elements for not printing stresses and strains of void parts
gp_id=0;
for i_element=1:ELEM.n
    for gauss_point = GP.def
        gp_id = gp_id +1;
        if y{cycle}(i_element)~= 1 %get the boundaries of the concrete
            GP.stresses(gp_id,:) = nan;
            GP.strains(gp_id,:) = nan;
        end
    end
end
... rest of the code in the annex
```

The fourth part of the code is dedicated to the post processing, needed only in the case it is wished to perform analyses of stresses, strains, on the resulting design of the optimization. The post processing function `POST` is executed once again, but outside the ESO loop and on the final design.

The stresses and strains of void elements are set to "nan" (not-a-number), in order to conveniently get white areas on corresponding void elements, on the plots of stresses and strains drawn with the MTALAB `plot` function.

It is to note that the post processing function calculates stresses and strains on Gauss points. In parts of code not shown here (but written in the annex) the following operations are performed:

- values of stresses and strains from Gauss points are reordered separate arrays;
- Von Mises and principal stresses are calculated on Gauss points;
- stresses and strains are extrapolated at element level (averaging values on 4 Gauss points of each element) for reducing computational times for plots.

2.4.2 Comments on FEM.m code

The file `FEM.m` is a classic finite element analysis code with just the following exemptions:

1. two element stiffness matrices are defined, one for each phase of the material, and assembly is carried on with the following formula:

   ```
   KEx(:,:,iel)=(K_e1(:,:)*x(iel)^penal+K_e2(:,:)*(1-x(iel)^penal))*y(iel)^penal;
   ```

   executing linear interpolation between the two stiffness matrices according to the density matrix `x(iel)`. Moreover, the obtained values are multiplied by the y density `y(iel)` to penalize the void elements in the FEM analysis;

2. two 3D tensors are defined, one for each phase of the material, to store all elemental stiffness matrices, along the third dimension, with the formula:

   ```
   KE1(:,:,iel)=K_e1(:,:);
   KE2(:,:,iel)=K_e2(:,:);
   ```

These arrays are later on stored in a MATLAB `struct` object, respectively, `ELEM.KE1` and `ELEM.KE2` and used in the sensitivity analysis function `SENS`.

The formula used on to assemble `KEx` (and therefore `K`) is similarly used to assemble the `f`. However, as a matter of fact, assembly on vector `f` is needed only for selfweight problems, not treated in the current thesis.

2.4.3 Comments on POST.m code

The file `POST.m` is a classic post-processing function, with the only addition of the following snippet of code for distinguishing between the two phases of the material, according to the value of the density `x`:

```
stressesGP(gp_id,:)=D1*B*DOF.u(DOF.n_mat(i_element,:))*x(i_element)^penal+...
           D2*B*DOF.u(DOF.n_mat(i_element,:))*(1-x(i_element)^penal);
```

2.4.4 Comments on SENS.m code

The formulas for sensitivity numbers from the file `SENS.m` are shown in the text below. The equations n.2.16, n.2.24, n.2.27 and n.2.28 are multiplied by $mask^{penalmask}$ and deployed in the code.

1. *Stiffness criterion;*
   from formula 2.16, reported here below:

$$\alpha_i = \frac{x_i^{p-1}}{2}(u_i^T K_{i,1} u_i - u_i^T K_{i,2} u_i) \qquad \forall i = 1, ..., N$$

   it is derived the following equation for alpha:

```
for iiel=1:ELEM.n
    alphavec(iiel)=0.5 *  x(iiel)^(penal-1) *  mask(iiel)^(penalmask)* ...
    ( dot(DOF.u(DOF.n_mat(iiel,:))'*ELEM.KE1(:,:,iiel) , DOF.u(DOF.n_mat(iiel,:)))-...
      dot(DOF.u(DOF.n_mat(iiel,:))'*ELEM.KE2(:,:,iiel) , DOF.u(DOF.n_mat(iiel,:))) );
end
alphamat=reshape(alphavec,ELEM.ny,ELEM.nx);
```

2. *Von Mises criterion;*
   from formula 2.24, reported here below:

$$\alpha_i = \sqrt{J_2}\, x_i^{p-1}$$

   it is derived the following equation for alpha:

```
alphamat=(sqrt(ELEM.J2mat)).*x.^(penal-1).*mask.^(penalmask);
```

3. *Drucker-Prager criterion;*
   from formula 2.27, reported here below:

$$\alpha_i = (\aleph I_1 + \sqrt{J_2})\, x_i^{p-1}$$

   it is derived the following equation for alpha:

```
alphamat=(DPalpha*ELEM.I1mat+sqrt(ELEM.J2mat)).*x.^(penal-1).*mask.^(penalmask);
```

4. *Tensile stress criterion;*
   from formula 2.28, reported here below:

$$\alpha_i = \left(\sigma_{1,i} + \sigma_{2,i} + \left|\min(\sigma_1)\right| + \left|\min(\sigma_2)\right|\right) x_i^{p-1}$$

   it is derived the following equation for alpha:

```
alphamat=( ELEM.sigma1mat+ELEM.sigma2mat...
    +abs(min(min(ELEM.sigma1mat)))+abs(min(min(ELEM.sigma2mat))) )...
    .*x.^(penal).*mask.^(penalmask);
```

5. *Transition from Von Mises to Drucker-Prager criterion;*
   linearly interpolating formula 2.24 and 2.27 it is derived the following equation for alpha:

```
alphamat=(SET.trans/SET.translenght*DPalpha*ELEM.I1mat+sqrt(ELEM.J2mat))...
    .*x.^(penal-1).*mask.^(penalmask);
```

6. *Transition from Von Mises to tensile stress criterion;*
   linearly interpolating formula 2.24 and 2.28 it is derived the following equation for alpha:

```
alphamat=((1-SET.trans/SET.translenght)*(sqrt(ELEM.J2mat))...
    +(SET.trans/SET.translenght)*( ELEM.sigma1mat+ELEM.sigma2mat...
    +abs(min(min(ELEM.sigma1mat)))+abs(min(min(ELEM.sigma2mat))) ) )...
    .*x.^(penal-1).*mask.^(penalmask);
```

7. *Transition from Drucker-Prager to tensile stress criterion;*
   linearly interpolating formula 2.27 and 2.28 it is derived the following equation for alpha:

```
alphamat=((1-SET.trans/SET.translenght)*(DPalpha*ELEM.I1mat+sqrt(ELEM.J2mat))...
    +(SET.trans/SET.translenght)*( ELEM.sigma1mat+ELEM.sigma2mat...
    +abs(min(min(ELEM.sigma1mat)))+abs(min(min(ELEM.sigma2mat))) ) )...
    .*x.^(penal-1).*mask.^(penalmask);
```

In the cases when there is a transition between two criteria (case 5, 6 7), it is set an interpolation parameter `SET.trans` which increases within each iteration and is defined by the following formula:

```
if iter<=SET.translenght; SET.trans=iter; end
```

so that, when `SET.trans=SET.translenght`, the parameter `SET.trans` remains constant, and the transition from one criterion to the other is complete on 100%.

2.4.5 Comments on COVER.m code

The file `COVER.m` is used in all cases when an automatic concrete cover is desired. Automatic means that the cover takes place around any geometry, no matter its shape and/or the number of holes, without specifying the geometry of the cover manually.

The main feature of `COVER` function is to generate a matrix named `cover` (later on stored into `ESO.cover`), of the same size of the mesh and filled with 0 and 1. The value 1 is stored in the positions corresponding to elements affected by concrete cover, while the value 0 is kept for other elements, unaffected by concrete cover. The discriminating parameter is the distance of the element from the edges, in relation to the specified concrete cover thickness.

The technique developed in the thesis is to check, for each element, a surrounding circular area, of radius equal to the cover thickness plus one (the "plus one" has to be applied because only the elements entirely within the circle in fact are considered). In this area it is checked the presence of void elements (which have correspondent values of 0.001 in the matrix `y`). If a void element is present in the circular area, the central element is "taken out" by the concrete cover, setting `cover=1`. In other words, such elements are set to be turned to second phase material (concrete), no matter its sensitivity number. This simply results in an offset of the steel area around all holes in the geometry. The automatic concrete cover is created with the following snippet of code:

```
for ii = 1:ELEM.nx
  for jj = 1:ELEM.ny
    %check a close range of surrounding elements with radius = cover distance
    covercheck1=0;
    covercheck2=0;
    for hh = 1:ELEM.nx
        for kk = 1:ELEM.ny
           %sqrt((kk - center_y)^2+(hh - center_x)^2) < half length
            if sqrt((kk-jj)^2+(hh-ii)^2) < SET.coverthick + 1
                covercheck1=covercheck1+1; %total max area elements in close range
                covercheck2=covercheck2+y{cycle}(kk,hh); %total real area in close range
            end
        end
    end
    %check if there are void elements in close range
    if covercheck1==covercheck2
        cover(jj,ii) = 0;
    else
        cover(jj,ii) = 1;
    end
  end
end
```

However, this kind of code is not able to apply cover around the edges of the rectangular domain. Therefore, for such scope, it is necessary to add a code for "taking out" the elements neighboring the design domain edges. The code to apply top and bottom cover is:

```
for ii = 1:ELEM.nx
  for jj = 1:ELEM.ny
    %max([abs(jj - center_y)/y_scale,abs(ii - center_x)/x_scale] < half length
    if max([abs(jj-1),abs(ii-ELEM.nx/2)/ELEM.nx]) < SET.coverthick
    cover(jj,ii) = 1;
    end
    if max([abs(jj-ELEM.ny),abs(ii-ELEM.nx/2)/ELEM.nx]) < SET.coverthick
    cover(jj,ii) = 1;
    end
  end
end
```

while for left and right cover it is:

```
for ii = 1:ELEM.nx
  for jj = 1:ELEM.ny
    %max([abs(jj - center_y)/y_scale,abs(ii - center_x)/x_scale] < half length
    if max([abs(jj-ELEM.ny/2)/ELEM.ny,abs(ii-1)]) < SET.coverthick
    cover(jj,ii) = 1;
    end
    if max([abs(jj-ELEM.ny/2)/ELEM.ny,abs(ii-ELEM.nx)]) < SET.coverthick
    cover(jj,ii) = 1;
    end
  end
end
```

As it is possible to see it is created a rectangular area centered in the middle of the edge, having lenght as the edge itself and having thickness as double the concrete thickness. It results that the half of the rectangle inside the design domain has exactly width as the concrete thickness. To define the rectangular area it is used the following formula:

```
max([abs(jj - center_y)/y_scale,abs(ii - center_x)/x_scale] < half length
```

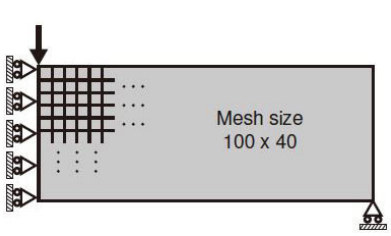
Clearly, it is possible to apply concrete cover only to top, bottom, left, right, or any combination of them, inserting only the relative rectangular area definitions.

In case only "manual" concrete cover (around the edges of design domain) is wished, it is preferable to set it outside the ESO loop to save computation expenses, as it is exactly done for the cases `SET.covertype=2` and `SET.covertype=3` defined in section 1.7 of the code. In cases when it is needed to deploy "automatic" cover, with or without "manual" cover, it is necessary to set it inside the ESO loop, via the `COVER` function, because the holes in the geometry are continuously changed within the optimization procedure.

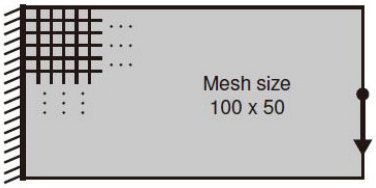
Chapter 3

Results

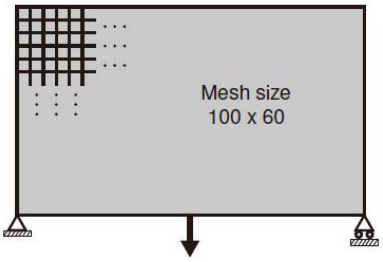
The procedure developed in the thesis has been tested on 12 case studies from literature. The first three case studies are the classic benchmark tests for ESO method, as found in the article of Xie [11].



(a) Case study n.1

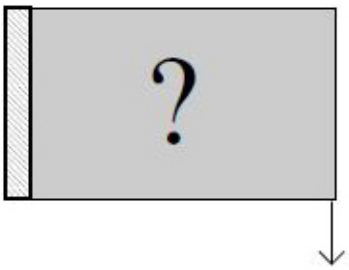


(b) Case study n.2

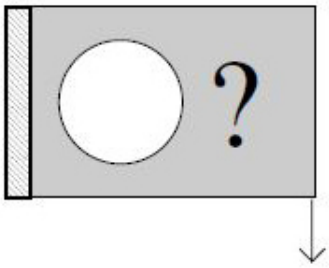


(c) Case study n.3

The fourth and fifth case study are two additional benchmark tests from SIMP literature, as found in the article of Sigmund [7].



(a) Case study n.4



(b) Case study n.5

The sixth case study is a more slender version of the second case study (having ratio lenght/height=4), as found in the article of Xie [18]. In ESO literature, the cantilever benchmark test is usually addressed for different slenderness ratios. The same is done in this thesis.

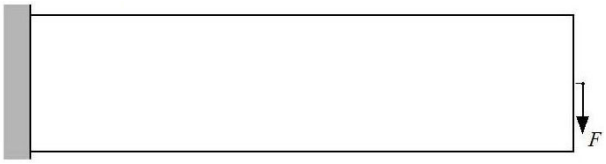
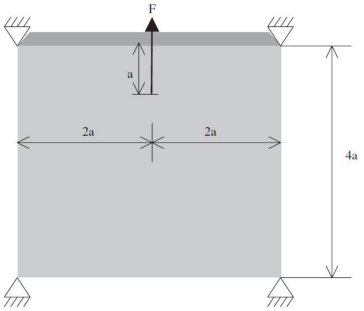


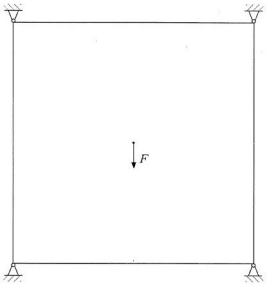
Figure 3.3: Case study n.6

The seventh and eight case study are two square deep beams, as found in the articles of Xie [17] and [19], which are two of the few applications in literature that address materials with different resistance

in tension and compression. In literature, the first case study was solved with a tension optimization criterion (applied on the first and obsolete formulation of ESO method) and the second with a nonlinear finite element analysis optimization with Drucker-Prager material definition.

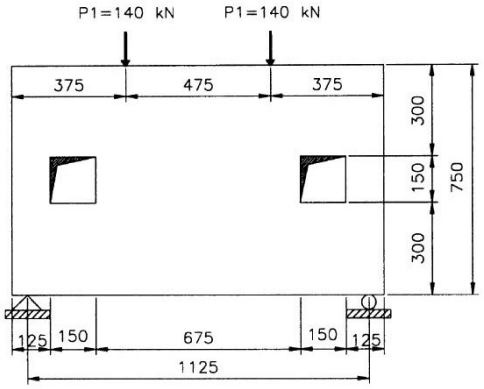


(a) Case study n.7

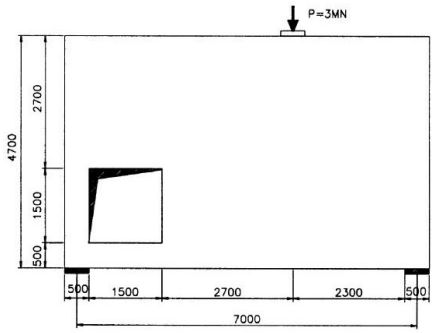


(b) Case study n.8

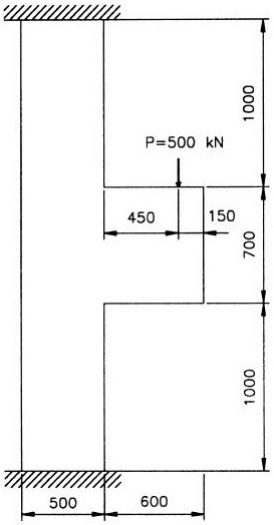
The last four case studies are taken from the research carried on by Liang around the years 2000-2005 [20][4]. Liang has created a variation of standard ESO method, called "Performance-Based Optimization", which uses a specially defined performance index to verify convergence and stop the procedure. He tested his method on several standard case studies, also from concrete literature. His method was applied also for the automatic generation of strut and tie models for reinforced concrete, employing a displacement-type optimization criteria. More information on the method, along with other case studies and applications to reinforced concrete strut and tie models, can be found in his book [4].



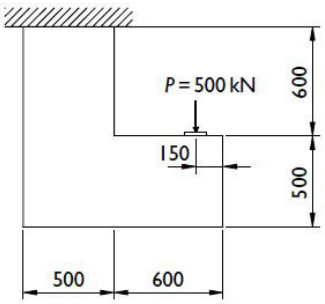
(a) Case study n.9



(b) Case study n.10



(c) Case study n.11



(d) Case study n.12

Following in the chapter, there are presented solutions for the above case studies derived by the ESO method of this thesis, along with the solutions taken from literature. All parameters used in the procedure are reported for each case study.

Analysis and evaluation of the results is presented separately in chapter 4 "Discussion".

Parameters selection

As an introductory remark, it is important to state that the material parameters and loads chosen in the solutions are not up to scale with real variables. Therefore, the solutions shown are to be considered as academic examples, especially when evaluating the magnitude of stresses, as shown later on in the text.

The chosen material parameters are:

Material parameters in common for all case studies

```
STEEL.E  = 10.0;
STEEL.nu = 0.3;
CONCR.E  = 1.0;
CONCR.nu = 0.2;
sigmac=1;
sigmat=0.2;
DPalpha=(sigmac-sigmat)/(sqrt(3)*(sigmac+sigmat));
```

Since the loads are not up to scale, the only relevant factor in the choice of material stiffnesses is their ratio. It has been chosen a ratio of stiffness between steel and concrete of 10.
It is to note also that, the strength of concrete in tension (`sigmat`) and compression (`sigmac`) appear in the procedure only through their ratio via the Drucker-Prager parameter $\aleph$ (indicated in the code as `DPalpha`). Then, in the present procedure, their magnitude results irrelevant. The only relevant factor remains their ratio. It has been chosen a ratio of concrete stress strengths between tension and compression of 0.2.

The other shared parameters are:

ESO parameters in common for all case studies

```
ESO.penal = 3.; %penalty exponent applied to x
ESO.penalmask = 2.; %penalty exponent applied to mask
ESO.conv_err=          [0.001        0.001           0.001           ]; %allowable conv. error
ESO.er=                [0.04,        0.03,           0.02            ];
ESO.rmin=              [8,           3,              2               ];
ESO.criteria=          [2,           5,              7               ];
MAT1=                  {STEEL,       STEEL,          STEEL           };
MAT2=                  {VOID,        CONCR,          CONCR           };
ESO.startvol=          [1,           ESO.volfrac(1), ESO.volfrac(2)  ];
update_startx=@(x0,x) {x0,           x{1},           x{2}            }; %values of ESO.start
update_mask=@(x0,x)   {x0,           x{1},           x{2}            }; %values of mask
switch SET.boundaries   % set values of y (concrete boundaries)
case 1   % preserve original boundaries
update_y=@(x0,x)      {x0,           x0,             x0              };
case 2   % optimize boundaries
update_y=@(x0,x)      {x0,           x{1},           x{1}            };
end
```

The optimization criteria sequence (as defined in array `ESO.criteria`) has been chosen for all solutions of case studies in the following manner:

- n.2 - Von Mises criterion - in the first optimization loop, when the void is determined

- n.5 - Transition from Von Mises to Drucker-Prager criterion - in the second optimization loop, when steel is replaced with concrete in compressive and low stressed tensile zones
- n.7 - Transition from Drucker-Prager to tensile stress criterion - in the third optimization loop, when steel boundary is refined, removing remaining steel still present in compressive zones

Other settings in common for all case studies

Transition lenght for linear interpolation between criteria during optimization loop (expressed in n. iterations):
`SET.translenght=20;`
In case original external geometry is to be kept (non-optimized), it is set:
`SET.boundaries=1;`
while in the case the external geometry is optimized it is set the parameter:
`SET.boundaries=2;`

The remaining parameters were set individually for each case study. The chosen values are indicated further in this chapter in separate sections for each case study.

Individual settings, different for all case studies

`ELEM.nx`
`ELEM.ny`
`SET.bctype`
`SET.solidvoid`
`SET.covertype`
`SET.coverthick`
`ESO.volfrac`

3.1 Case study n.1

The case study n.1 is the classic half-MBB beam present in most of the articles about structural optimization. The problem statement is presented in figure 3.6.

The solution in figure 3.7 is taken from the article of Xie [11] and is obtained with a stiffness optimization of a 100 by 40 elements design domain, with a target volume of 60% , an evolution ratio `er` of 0.02 and `rmin` equal to 0.6.

The solution in figure 3.8 is taken from the article of Xie [10] and is obtained with a multi-material stiffness optimization whereas E1 = 1GPa, E2 = 0.1GPa. The Poisson ratio is 0.3 for all material. The objective volume of materials 1 and 2 are set to be 15% and 25% of the whole design domain. It was chosen an evolution ratio `er` of 0.02 and `rmin` equal to 0.15.

The solutions derived using the MATLAB code presented in this thesis used the following settings:

Solution in figure 3.9

```
ELEM.nx = 100; ELEM.ny = 40;
SET.bctype=3;
SET.solidvoid=1;
SET.covertype=1;
ESO.volfrac=[0.70, 0.35, 0.20];
```

Solution in figure 3.10

```
ELEM.nx = 100; ELEM.ny = 40;
SET.bctype=3;
SET.solidvoid=1;
SET.covertype=8;
```

```
SET.coverthick=2; *(see below)
ESO.volfrac=[0.70, 0.35, 0.20];
```

In `SET.bctype=3` the boundary conditions have been programmed for this case study and hold even for different size of design domain. With `SET.solidvoid=1` there are no voids of solids imposed on the solution.
* In the solution 3.10(b) it has been used `SET.coverthick=1` instead of `SET.coverthick=2` because of the small resolution of 100x40 elements and the small thickness of the tensile member in the lower right corner. With this resolution, a bigger concrete cover thickness enforcement could cause the loss of steel members in thin tensile zones.

3.2 Case study n.2

The case study n.2 is the classic cantilever beam present in most of the articles about structural optimization, together with the previous case of half MBB beam. The problem statement is presented in figure 3.11.

The solution in figure 3.12 is taken from the article of Xie [11] and is obtained with a stiffness optimization of a 100 by 50 elements design domain, with a target volume of 60% , an evolution ratio `er` of 0.02 and `rmin` equal to 0.6.

The solutions derived using the MATLAB code presented in this thesis used the following settings:

Solution in figure 3.13

```
ELEM.nx = 100; ELEM.ny = 50;
SET.bctype=2;
SET.solidvoid=1;
SET.covertype=1;
ESO.volfrac=[0.60, 0.30, 0.15];
```

Solution in figure 3.14

```
ELEM.nx = 100; ELEM.ny = 50;
SET.bctype=2;
SET.solidvoid=1;
SET.covertype=8;
SET.coverthick=2;
ESO.volfrac=[0.60, 0.30, 0.15];
```

In `SET.bctype=2` the boundary conditions have been programmed for this case study and hold even for different size of design domain. With `SET.solidvoid=1` there are no voids of solids imposed on the solution.

3.3 Case study n.3

The case study n.3 is the classic half-wheel beam, a famous example from Mitchell optimal solution for trusses. The problem statement is presented in figure 3.15.

The solution in figure 3.16 is taken from the article of Xie [11] and is obtained with a stiffness optimization of a 100 by 60 elements design domain, with a target volume of 60% , an evolution ratio `er` of 0.02 and `rmin` equal to 0.6.

The solutions derived using the MATLAB code presented in this thesis used the following settings:

Solution in figure 3.17

```
ELEM.nx = 100; ELEM.ny = 60;
SET.bctype=4;
SET.solidvoid=1;
SET.covertype=1;
ESO.volfrac=[0.60, 0.30, 0.15];
```

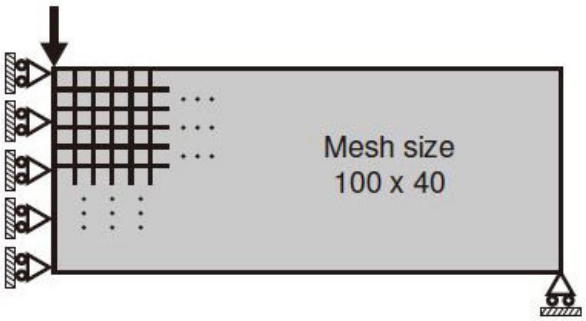


Figure 3.6: Case study n.1 - problem statement

Figure 3.7: Case study n.1 - solution by Xie [11] with ESO stiffness optimization

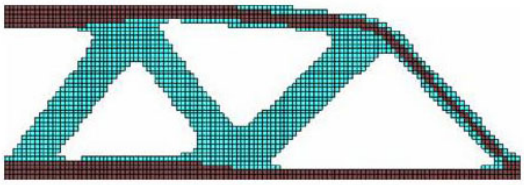


Figure 3.8: Case study n.1 - solution by Xie [10] with multi-material ESO stiffness optimization with materials equally resistant in compression and tension but with different stiffness

(a) maintaining external geometry (b) optimizing external geometry

Figure 3.9: Case study n.1 - solution derived by the code developed in this thesis for reinforced concrete - no concrete cover

(a) maintaining external geometry (b) optimizing external geometry

Figure 3.10: Case study n.1 - solution derived by the code developed in this thesis for reinforced concrete - with concrete cover

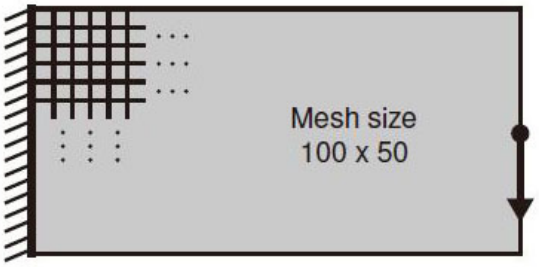
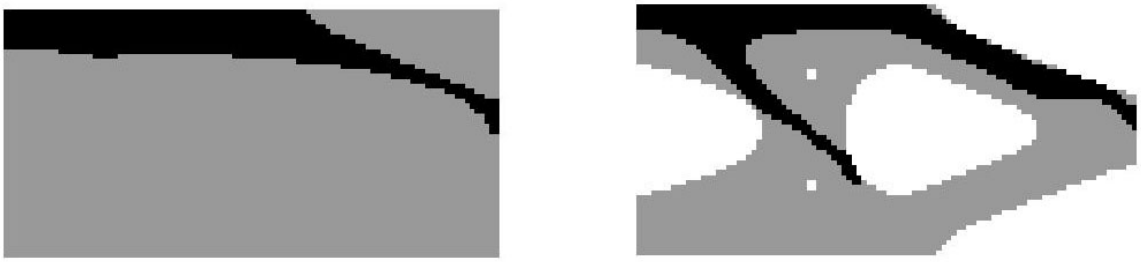


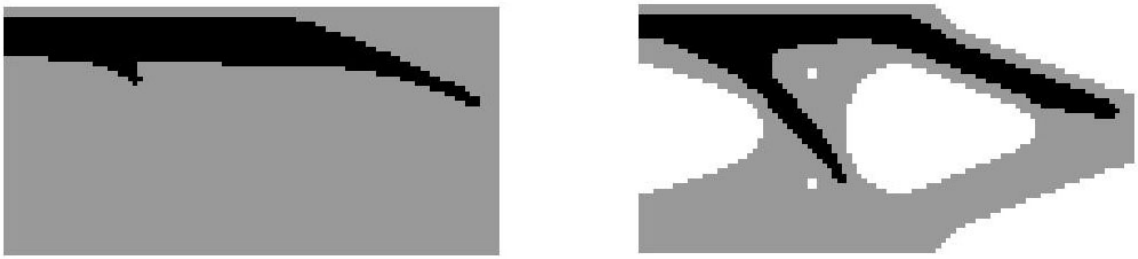
Figure 3.11: Case study n.2 - problem statement

Figure 3.12: Case study n.2 - solution by Xie [11] with ESO stiffness optimization



(a) maintaining external geometry (b) optimizing external geometry

Figure 3.13: Case study n.2 - solution derived by the code developed in this thesis for reinforced concrete - no concrete cover



(a) maintaining external geometry (b) optimizing external geometry

Figure 3.14: Case study n.2 - solution derived by the code developed in this thesis for reinforced concrete - with concrete cover

Solution in figure 3.18

```
ELEM.nx = 100; ELEM.ny = 60;
SET.bctype=4;
SET.solidvoid=1;
SET.covertype=7;
SET.coverthick=2;
ESO.volfrac=[0.60, 0.30, 0.15];
```

In `SET.bctype=4` the boundary conditions have been programmed for this case study and hold even for different size of design domain. With `SET.solidvoid=1` there are no voids of solids imposed on the solution.

3.4 Case study n.4

The case study n.4 is taken from the paper of Sigmund [7] and is a cantilever beam loaded in the lower left corner. The problem statement is presented in figure 3.19.

The solution in figure 3.20 is taken from the article of Sigmund [7] and is obtained with a stiffness optimization of a 30 by 20 elements design domain, with a target volume of 40% and `rmin` equal to 1.2.

The solutions derived using the MATLAB code presented in this thesis used the following settings:

Solution in figure 3.21

```
ELEM.nx = 90; ELEM.ny = 60;
SET.bctype=5;
SET.solidvoid=1;
SET.covertype=1;
ESO.volfrac=[0.50, 0.25, 0.15];
```

Solution in figure 3.22

```
ELEM.nx = 90; ELEM.ny = 60;
SET.bctype=5;
SET.solidvoid=1;
SET.covertype=8;
SET.coverthick=2;
ESO.volfrac=[0.50, 0.25, 0.15];
```

In `SET.bctype=5` the boundary conditions have been programmed for this case study and hold even for different size of design domain. With `SET.solidvoid=1` there are no voids of solids imposed on the solution.

3.5 Case study n.5

The case study n.5 is taken from the paper of Sigmund [7] and is a cantilever beam loaded in the lower left corner with a fixed hole (e.g. a hole for a pipe). The problem statement is presented in figure 3.23. The hole is defined as a circle with radius $\frac{\text{ELEM.ny}}{3}$ and center $\left(\frac{\text{ELEM.nx}}{3}, \frac{\text{ELEM.ny}}{2}\right)$.

The solution in figure 3.24 is taken from the article of Sigmund [7] and is obtained with a stiffness optimization of a 45 by 30 elements design domain, with a target volume of 50% and `rmin` equal to 1.5.

The solutions derived using the MATLAB code presented in this thesis used the following settings:

Solution in figure 3.25

```
ELEM.nx = 90; ELEM.ny = 60;
SET.bctype=5;
SET.solidvoid=2;
SET.covertype=1;
ESO.volfrac=[0.50, 0.25, 0.15];
```

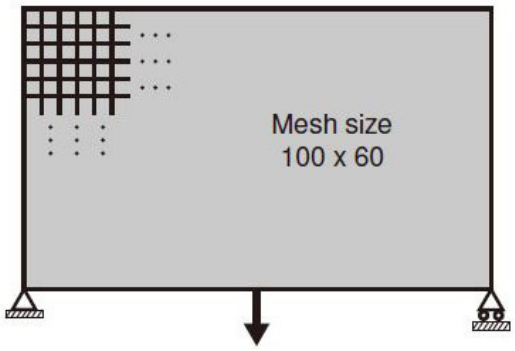


Figure 3.15: Case study n.3 - problem statement

Figure 3.16: Case study n.3 - solution by Xie [11] with ESO stiffness optimization

(a) maintaining external geometry (b) optimizing external geometry

Figure 3.17: Case study n.3 - solution derived by the code developed in this thesis for reinforced concrete - no concrete cover

(a) maintaining external geometry (b) optimizing external geometry

Figure 3.18: Case study n.3 - solution derived by the code developed in this thesis for reinforced concrete - with concrete cover

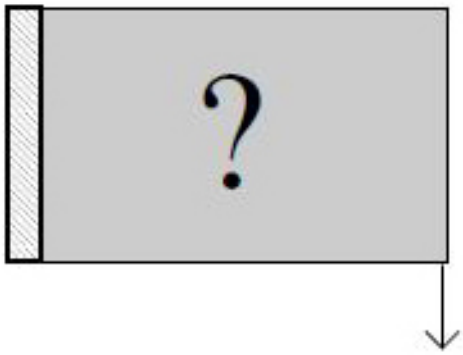


Figure 3.19: Case study n.4 - problem statement

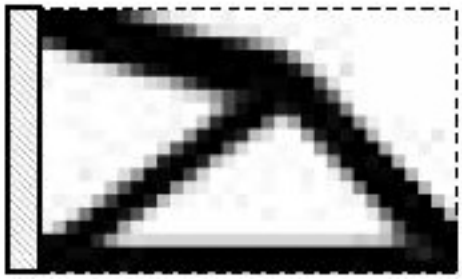


Figure 3.20: Case study n.4 - solution by Sigmund [7] with SIMP stiffness optimization

(a) maintaining external geometry (b) optimizing external geometry

Figure 3.21: Case study n.4 - solution derived by the code developed in this thesis for reinforced concrete - no concrete cover

(a) maintaining external geometry (b) optimizing external geometry

Figure 3.22: Case study n.4 - solution derived by the code developed in this thesis for reinforced concrete - with concrete cover

Solution in figure 3.26

```
ELEM.nx = 90; ELEM.ny = 60;
SET.bctype=5;
SET.solidvoid=2;
SET.covertype=8;
SET.coverthick=2;
ESO.volfrac=[0.50, 0.25, 0.15];
```

In `SET.bctype=5` the boundary conditions have been programmed for this case study and hold even for different size of design domain.

With `SET.solidvoid=2` it is imposed a circular hole with chosen parameters for center and radius as in the original case study with radius $\frac{\text{ELEM.ny}}{3}$ and center $\left(\frac{\text{ELEM.nx}}{3}, \frac{\text{ELEM.ny}}{2}\right)$.

The hole is imposed with the following code:

```
for ii = 1:ELEM.nx
  for jj = 1:ELEM.ny
      %sqrt((jj - center_y)^2+(ii- center_x)^2) < radius
    if sqrt((jj-ELEM.ny/2)^2+(ii-ELEM.nx/3)^2) < ELEM.ny/3
    ESO.passive(jj,ii) = 1;
    end
  end
end
```

3.6 Case study n.6

The case study n.6 is the classic cantilever beam similar to the one in case study n.2 but more slender. The problem statement is presented in figure 3.27.

The solution in figure 3.28 is taken from the article of Xie [18] and is obtained with a stiffness optimization of a 160 by 40 elements design domain, with a target volume of 50% , an evolution ratio `er` of 0.02 and `rmin` equal to 3.

The solutions derived using the MATLAB code presented in this thesis used the following settings:

Solution in figure 3.29

```
ELEM.nx = 160; ELEM.ny = 40;
SET.bctype=2;
SET.solidvoid=1;
SET.covertype=1;
ESO.volfrac=[0.70, 0.40, 0.25];
```

Solution in figure 3.30

```
ELEM.nx = 160; ELEM.ny = 40;
SET.bctype=2;
SET.solidvoid=1;
SET.covertype=8;
SET.coverthick=2;
ESO.volfrac=[0.70, 0.40, 0.25];
```

In `SET.bctype=2` the boundary conditions have been programmed for this case study and hold even for different size of design domain. With `SET.solidvoid=1` there are no voids of solids imposed on the solution.

In the solution 3.30(b) it has been used `SET.coverthick=1` instead of `SET.coverthick=2` because due to the small resolution of 160x40 elements and the small thickness of the tensile member in the upper right corner. With this resolution, a bigger concrete cover thickness enforcement could cause the loss of the steel member in the tensile zone.

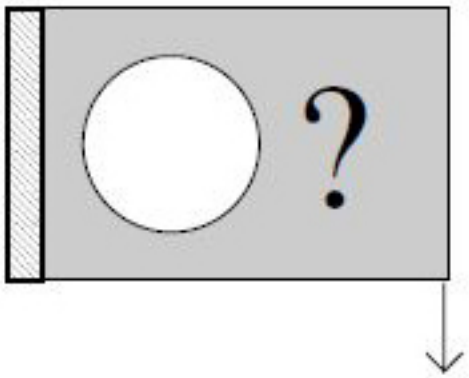


Figure 3.23: Case study n.5 - problem statement

Figure 3.24: Case study n.5 - solution by Sigmund [7] with SIMP stiffness optimization

(a) maintaining external geometry (b) optimizing external geometry

Figure 3.25: Case study n.5 - solution derived by the code developed in this thesis for reinforced concrete - no concrete cover

(a) maintaining external geometry (b) optimizing external geometry

Figure 3.26: Case study n.5 - solution derived by the code developed in this thesis for reinforced concrete - with concrete cover

Figure 3.27: Case study n.6 - problem statement

Figure 3.28: Case study n.6 - solution by Xie [18] with ESO stiffness optimization

(a) maintaining external geometry (b) optimizing external geometry

Figure 3.29: Case study n.6 - solution derived by the code developed in this thesis for reinforced concrete - no concrete cover

(a) maintaining external geometry (b) optimizing external geometry

Figure 3.30: Case study n.6 - solution derived by the code developed in this thesis for reinforced concrete - with concrete cover

3.7 Case study n.7

The case study n.7 is a deep beam with an upper force applied in middle-top part. The problem statement is presented in figure 3.31.

The solutions in figure 3.32 are taken from the article of Xie [17] and are obtained with one of first ESO optimization algorithm (which did not yet use sensitivity numbers but performed element removal based on stress ranking and a set rejection ratio) respectively with a tension criterion defined as $\sigma^e = \sigma_1 + \sigma_2$ and a compression criterion $\sigma^e = -\sigma_1 - \sigma_2$. ESO parameters were not specified in the paper.

The solutions derived using the MATLAB code presented in this thesis used the following settings:

Solution in figure 3.33

```
ELEM.nx = 80; ELEM.ny = 80;
SET.bctype=6;
SET.solidvoid=1;
SET.covertype=1;
ESO.volfrac=[0.50, 0.25, 0.10];
```

Solution in figure 3.34

```
ELEM.nx = 80; ELEM.ny = 80;
SET.bctype=6;
SET.solidvoid=1;
SET.covertype=6;
SET.coverthick=2;
ESO.volfrac=[0.50, 0.25, 0.10];
```

In `SET.bctype=6` the boundary conditions have been programmed for this case study and hold even for different size of design domain. With `SET.solidvoid=1` there are no voids of solids imposed on the solution.

3.8 Case study n.8

The case study n.8 is a deep beam with a downward force applied in middle. The problem statement is presented in figure 3.35.

The solutions in figure 3.36 are taken from the article of Xie [19] and are obtained respectively with ESO stiffness optimization performed with linear elastic FEA, and ESO for elastic-plastic strain energy optimization performed with material nonlinear (Drucker-Prager model) FEA. The number of finite elements used for the discretization of the design domain had not been specified in the paper. ESO parameters were chosen as follows: target volume of 20% , evolution ratio `er` of 0.02 and `rmin` equal to 1.

The solutions derived using the MATLAB code presented in this thesis used the following settings:

Solution in figure 3.37

```
ELEM.nx = 80; ELEM.ny = 80;
SET.bctype=7;
SET.solidvoid=1;
SET.covertype=1;
ESO.volfrac=[0.50, 0.25, 0.10];
```

Solution in figure 3.38

```
ELEM.nx = 80; ELEM.ny = 80;
SET.bctype=7;
SET.solidvoid=1;
SET.covertype=6;
SET.coverthick=2;
ESO.volfrac=[0.50, 0.25, 0.10];
```

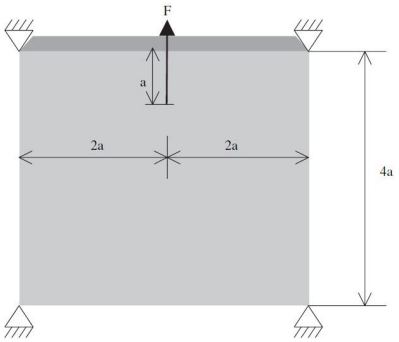


Figure 3.31: Case study n.7 - problem statement

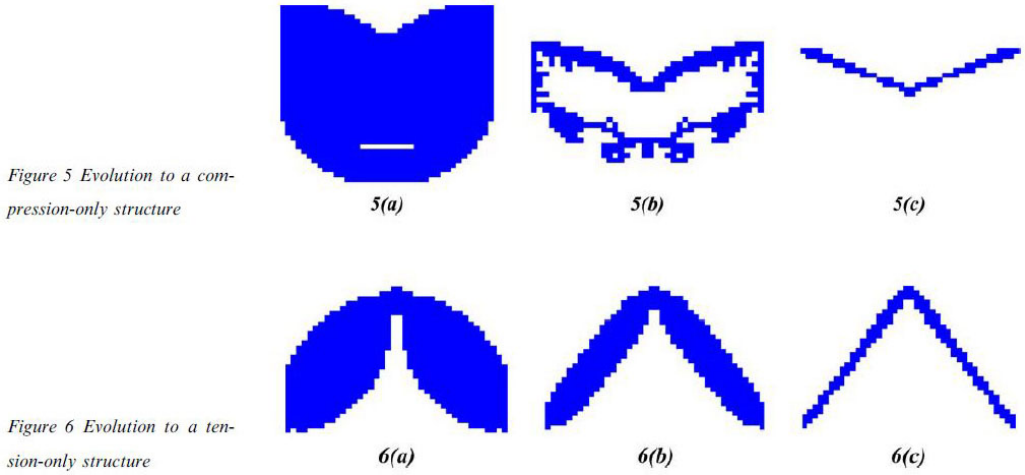


Figure 3.32: Case study n.7 - solution by Xie [17] with first generation ESO optimization with tensile and compressive criteria

(a) maintaining ext. geometry (b) optimizing ext. geometry

Figure 3.33: Case study n.7 - solution derived by the code developed in this thesis for reinforced concrete - no concrete cover

(a) maintaining ext. geometry (b) optimizing ext. geometry

Figure 3.34: Case study n.7 - solution derived by the code developed in this thesis for reinforced concrete - with concrete cover

In `SET.bctype=7` the boundary conditions have been programmed for this case study and hold even for different size of design domain. With `SET.solidvoid=1` there are no voids of solids imposed on the solution.

3.9 Case study n.9

The case study n.9 is a deep beam with two web square openings and symmetric loading. The problem statement is presented in figure 3.39.

The solution in figure 3.40 is taken from the article of Xie and Liang [20] and is derived with a performance based ESO optimization using a displacement constraint. Sensitivity numbers are defined to evaluate the influence of element removal to the change in the constrained displacement. It is neither a stress or stiffness-based optimization, but a method specially defined by Liang to generate strut and tie models in reinforced concrete. The number of finite elements used for the discretization of the design domain was 49 by 30. ESO parameter of evolution ratio used `er` was 0.01. The optimized result shown in the picture was the one with the highest performance index with respect to final volume and set displacement constraint. The same case study is as also present in the book of Liang [4].

The solutions derived using the MATLAB code presented in this thesis used the following settings:

Solution in figure 3.41

```
ELEM.nx = 98; ELEM.ny = 60;
SET.bctype=8;
SET.solidvoid=6;
SET.covertype=1;
ESO.volfrac=[0.70, 0.35, 0.20];
```

Solution in figure 3.42

```
ELEM.nx = 98; ELEM.ny = 60;
SET.bctype=8;
SET.solidvoid=6;
SET.covertype=7;
SET.coverthick=2;
ESO.volfrac=[0.70, 0.35, 0.15];
```

In `SET.bctype=8` the boundary conditions have been programmed for this case study and hold only for size of design domain of 98 by 60. With `SET.solidvoid=6` there are two square holes dimensioned and positioned for a design domain of 98 by 60.

3.10 Case study n.10

The case study n.10 is a deep beam with one web square opening and asymmetric loading. The problem statement is presented in figure 3.43. The beam is simply supported. In the computation the boundary conditions have been modeled with a roller on the right and a hinge on the left.

The solution in figure 3.44 is taken from the article of Xie and Liang [20] and is derived with a performance based ESO optimization using a displacement constraint. Sensitivity numbers are defined to evaluate the influence of element removal to the change in the constrained displacement. It is neither a stress or stiffness-based optimization, but a method specially defined by Liang to generate strut and tie models in reinforced concrete. The number of finite elements used for the discretization of the design domain was 75 by 47. ESO parameter of evolution ratio used `er` was 0.01. The optimized result shown in the picture was the one with the highest performance index with respect to final volume and set displacement constraint. The same case study is as also present in the book of Liang [4].

The solutions derived using the MATLAB code presented in this thesis used the following settings:

Solution in figure 3.45

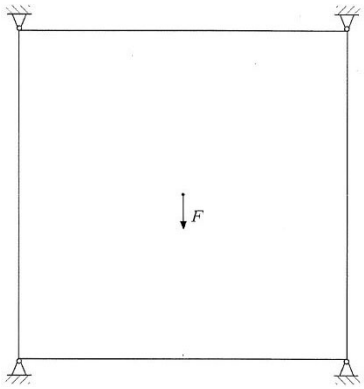


Figure 3.35: Case study n.8 - problem statement

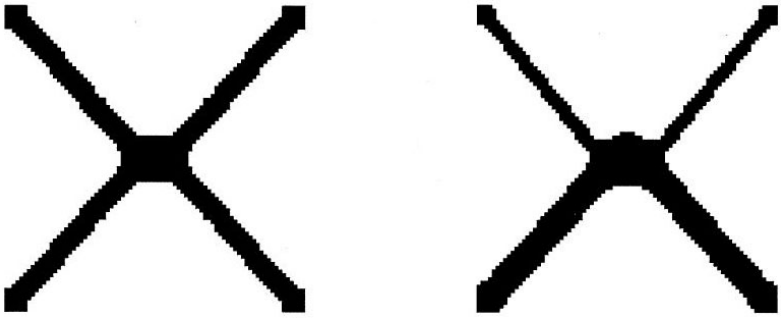


Figure 3.36: Case study n.8 - solution by Xie [19] with ESO stiffness / strain energy optimization with linear and material nonlinear (Drucker-Prager) FEA

(a) maintaining ext. geometry (b) optimizing ext. geometry

Figure 3.37: Case study n.8 - solution derived by the code developed in this thesis for reinforced concrete - no concrete cover

(a) maintaining ext. geometry (b) optimizing ext. geometry

Figure 3.38: Case study n.8 - solution derived by the code developed in this thesis for reinforced concrete - with concrete cover

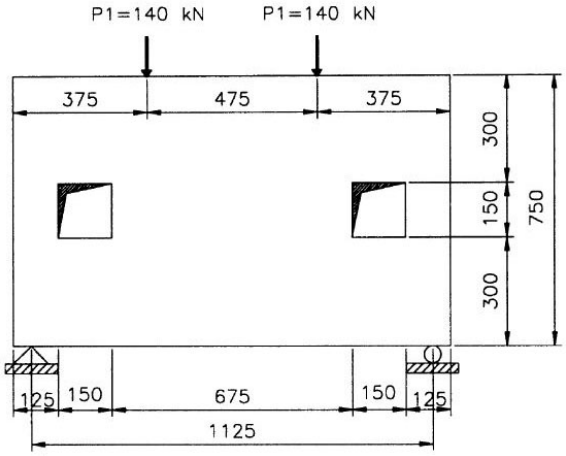


Figure 3.39: Case study n.9 - problem statement

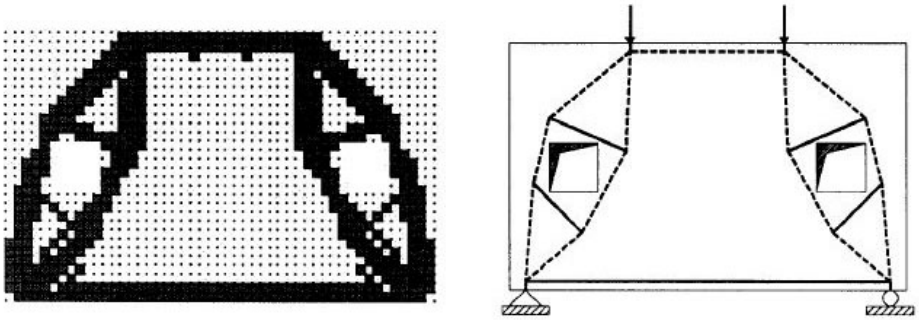
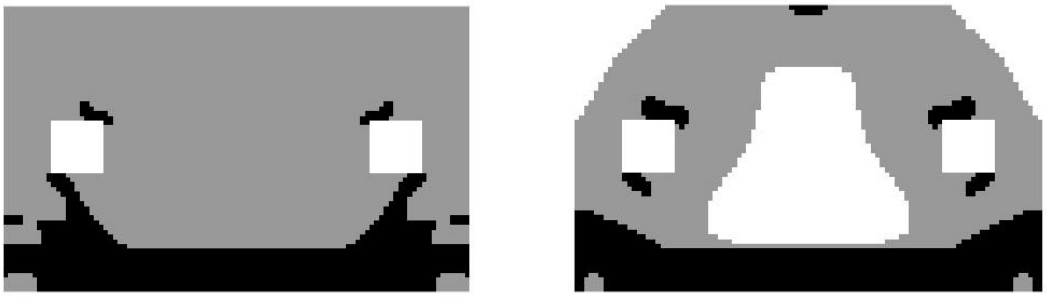
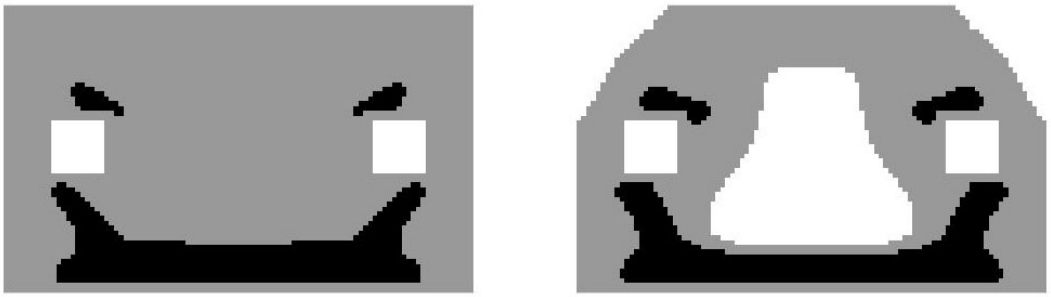


Figure 3.40: Case study n.9 - solution by Xie and Liang [20] with performance based ESO with displacement constraint and resulting strut and tie model



(a) maintaining external geometry (b) optimizing external geometry

Figure 3.41: Case study n.9 - solution derived by the code developed in this thesis for reinforced concrete - no concrete cover



(a) maintaining external geometry (b) optimizing external geometry

Figure 3.42: Case study n.9 - solution derived by the code developed in this thesis for reinforced concrete - with concrete cover

```
ELEM.nx = 150; ELEM.ny = 94;
SET.bctype=9;
SET.solidvoid=7;
SET.covertype=1;
ESO.volfrac=[0.60, 0.25, 0.10];
```

Solution in figure 3.46

```
ELEM.nx = 150; ELEM.ny = 94;
SET.bctype=9;
SET.solidvoid=7;
SET.covertype=7;
SET.coverthick=2;
ESO.volfrac=[0.60, 0.25, 0.10];
```

A second set of solutions developed with the MATLAB code of this thesis for this case study used instead the following settings:

Solution in figure 3.49

```
ELEM.nx = 150; ELEM.ny = 94;
SET.bctype=9;
SET.solidvoid=7;
SET.covertype=1;
ESO.volfrac=[0.76, 0.40, 0.20];
```

Solution in figure 3.50

```
ELEM.nx = 150; ELEM.ny = 94;
SET.bctype=9;
SET.solidvoid=7;
SET.covertype=7;
SET.coverthick=1;
ESO.volfrac=[0.76, 0.40, 0.20];
```

In `SET.bctype=9` the boundary conditions have been programmed for this case study and hold only for size of design domain of 150 by 94. With `SET.solidvoid=7` there are two square holes dimensioned and positioned exactly for a design domain of 150 by 94 to resemble the example from literature taken from the article of Xie and Liang [20].

3.11 Case study n.11

The case study n.11 is a loaded corbel jointed to a column. The problem statement is presented in figure 3.51. The boundary conditions in the model has been applied to five points in top part of colum and five points on bottom part, restraining both x and y directions.

The solution in figure 3.52 is taken from the article of Xie and Liang [20] and is derived with a performance based ESO optimization using a displacement constraint. Sensitivity numbers are defined to evaluate the influence of element removal to the change in the constrained displacement. It is neither a stress or stiffness-based optimization, but a method specially defined by Liang to generate strut and tie models in reinforced concrete. The number of finite elements used for the discretization of the design domain was 44 by 108. ESO parameter of evolution ratio used `er` was 0.01. The optimized result shown in the picture was the one with the highest performance index with respect to final volume and set displacement constraint. The same case study is as also present in the book of Liang [4].

The solutions derived using the MATLAB code presented in this thesis used the following settings:

Solution in figure 3.53

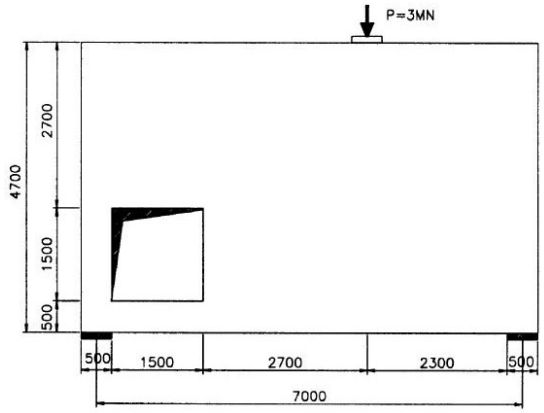


Figure 3.43: Case study n.10 - problem statement

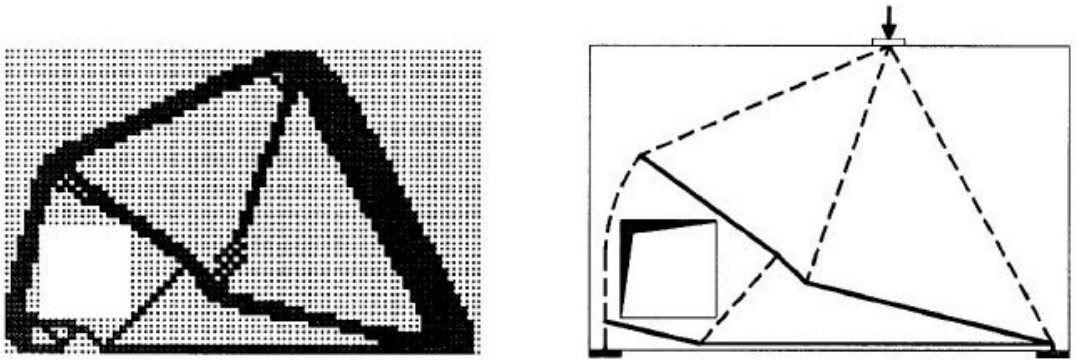


Figure 3.44: Case study n.10 - solution by Xie and Liang [20] with performance based ESO with displacement constraint and resulting strut and tie model

(a) maintaining external geometry (b) optimizing external geometry

Figure 3.45: Case study n.10 - solution derived by the code developed in this thesis for reinforced concrete - no concrete cover (volume fraction of first ESO loop smaller than 0.75)

(a) maintaining external geometry (b) optimizing external geometry

Figure 3.46: Case study n.10 - solution derived by the code developed in this thesis for reinforced concrete - with concrete cover (volume fraction of first ESO loop smaller than 0.75)

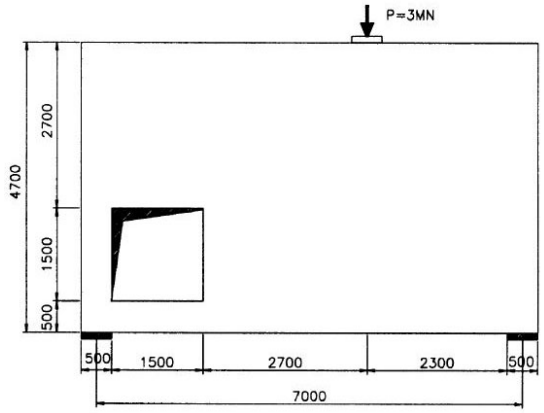


Figure 3.47: Case study n.10 - problem statement

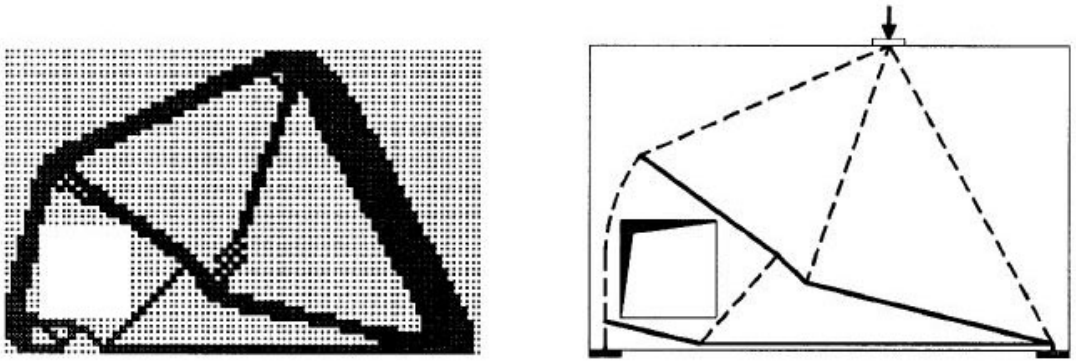
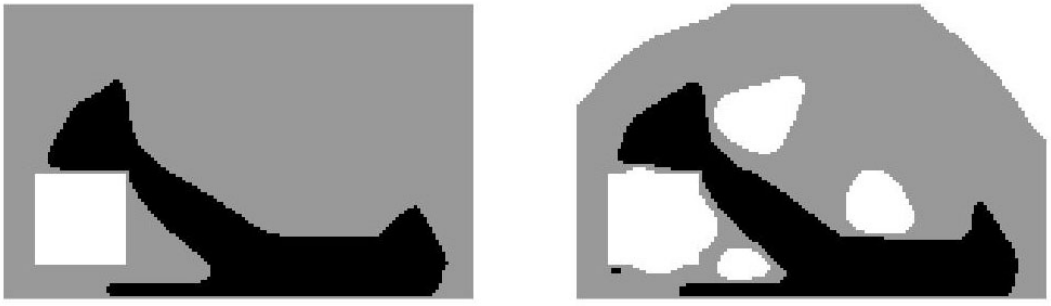


Figure 3.48: Case study n.10 - solution by Xie and Liang [20] with performance based ESO with displacement constraint and resulting strut and tie model

(a) maintaining external geometry (b) optimizing external geometry

Figure 3.49: Case study n.10 - solution derived by the code developed in this thesis for reinforced concrete - no concrete cover (volume fraction of first ESO loop bigger than 0.75)



(a) maintaining external geometry (b) optimizing external geometry

Figure 3.50: Case study n.10 - solution derived by the code developed in this thesis for reinforced concrete - with concrete cover (volume fraction of first ESO loop bigger than 0.75)

```
ELEM.nx = 88; ELEM.ny = 216;
SET.bctype=10;
SET.solidvoid=8;
SET.covertype=1;
ESO.volfrac=[0.50, 0.35, 0.15];
```

Solution in figure 3.54

```
ELEM.nx = 88; ELEM.ny = 216;
SET.bctype=10;
SET.solidvoid=8;
SET.covertype=6;
SET.coverthick=2;
ESO.volfrac=[0.50, 0.35, 0.15];
```

In `SET.bctype=10` the boundary conditions have been programmed for this case study and hold only for size of design domain of 88 by 216. With `SET.solidvoid=8` there are two rectangular holes dimensioned and positioned for a design domain of 88 by 216, to remove material from domain in top and bottom of the corbel.

3.12 Case study n.12

The case study n.12 is a corbel with a ledge support. The problem statement is presented in figure 3.55. The boundary conditions in the model has been applied to five points in top part of colum, restraining both x and y directions.

The solution in figure 3.56 is taken from the book of Liang [4] and is derived with a performance based ESO optimization using a displacement constraint. Sensitivity numbers are defined to evaluate the influence of element removal to the change in the constrained displacement. It is neither a stress or stiffness-based optimization, but a method specially defined by Liang to generate strut and tie models in reinforced concrete. The number of finite elements used for the discretization of the design domain was 44 by 44. ESO parameter of evolution ratio used `er` was 0.01. The optimized result shown in the picture was the one with the highest performance index with respect to final volume and set displacement constraint.

The solutions derived using the MATLAB code presented in this thesis used the following settings:

Solution in figure 3.57

```
ELEM.nx = 88; ELEM.ny = 88;
SET.bctype=11;
SET.solidvoid=9;
SET.covertype=1;
ESO.volfrac=[0.45, 0.30, 0.15];
```

Solution in figure 3.58

```
ELEM.nx = 88; ELEM.ny = 88;
SET.bctype=11;
SET.solidvoid=9;
SET.covertype=6;
SET.coverthick=2;
ESO.volfrac=[0.45, 0.30, 0.15];
```

In `SET.bctype=11` the boundary conditions have been programmed for this case study and hold only for size of design domain of 88 by 88. With `SET.solidvoid=9` there are two square holes dimensioned and positioned for a design domain of 88 by 88.

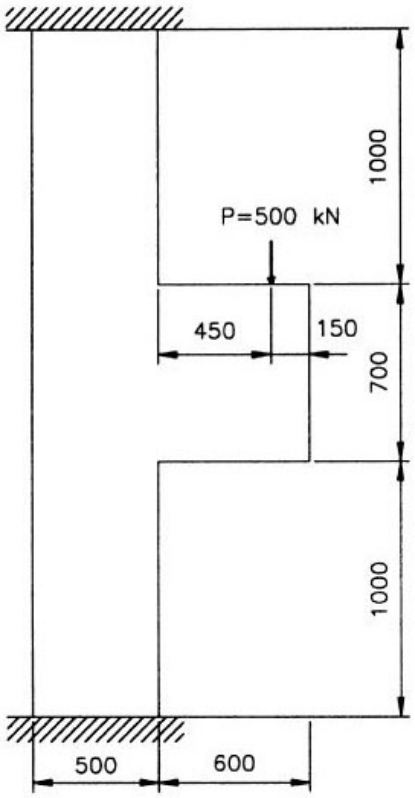


Figure 3.51: Case study n.11 - problem statement

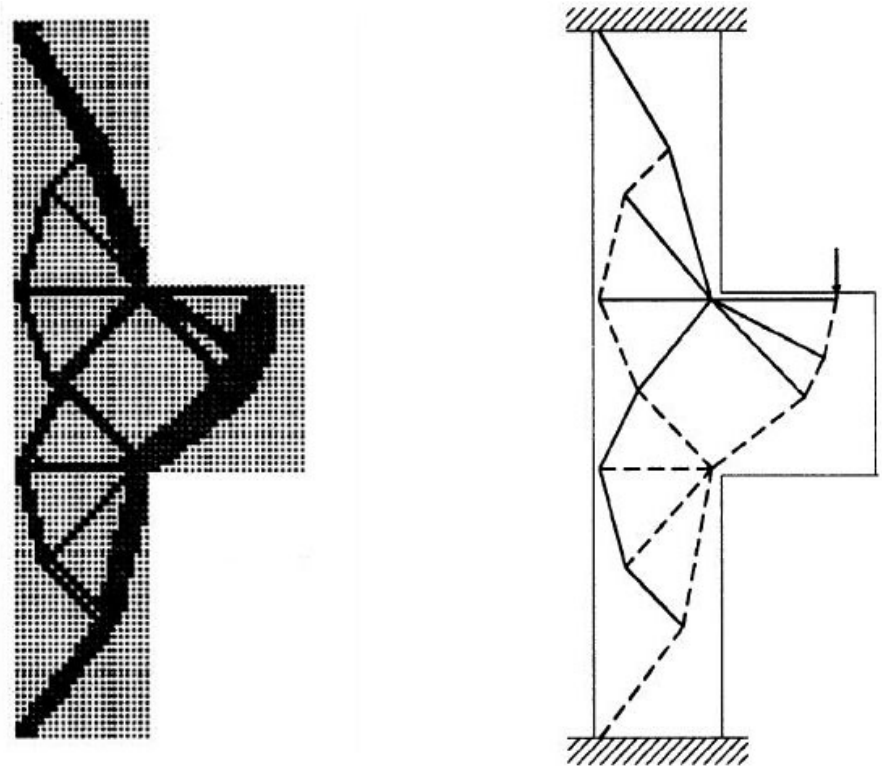
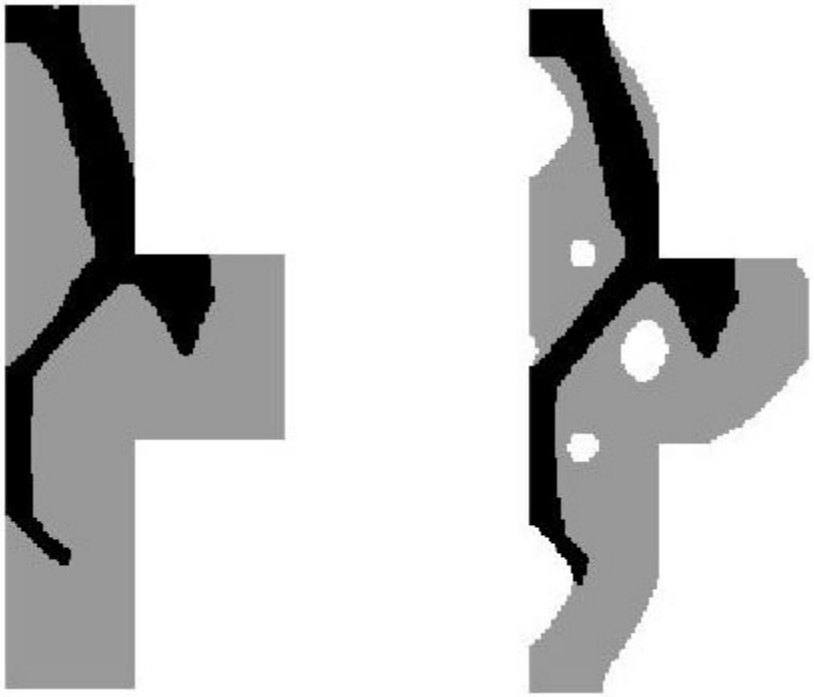


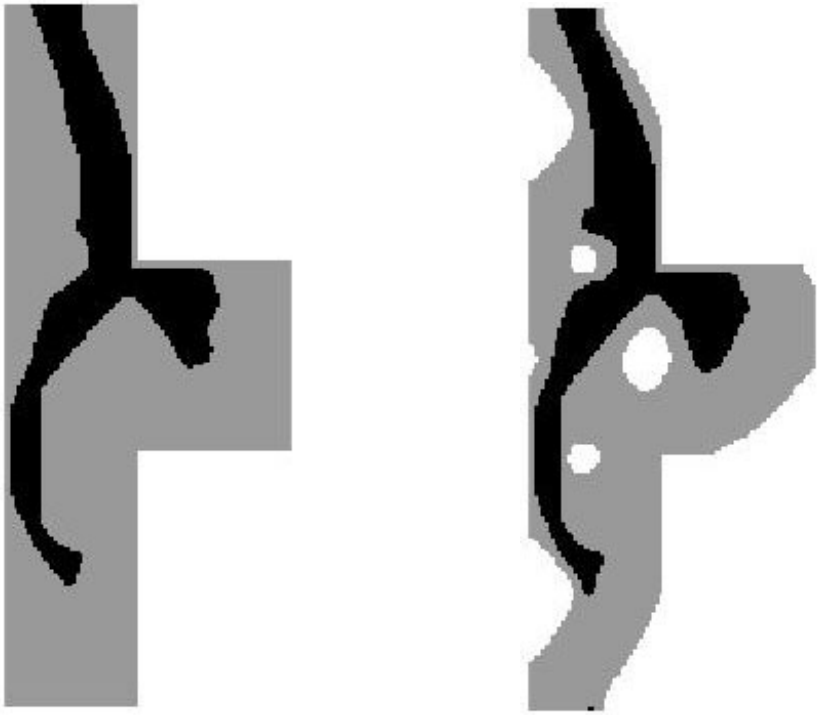
Figure 3.52: Case study n.11 - solution by Xie and Liang [20] with performance based ESO with displacement constraint and resulting strut and tie model



(a) maintaining external geometry

(b) optimizing external geometry

Figure 3.53: Case study n.11 - solution derived by the code developed in this thesis for reinforced concrete - no concrete cover



(a) maintaining external geometry

(b) optimizing external geometry

Figure 3.54: Case study n.11 - solution derived by the code developed in this thesis for reinforced concrete - with concrete cover

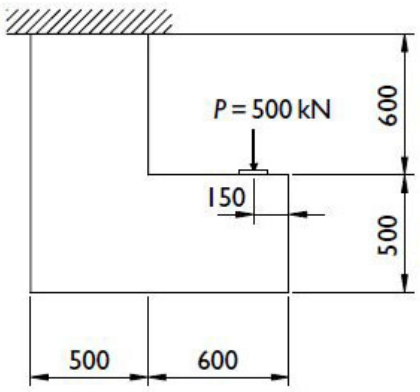


Figure 3.55: Case study n.12 - problem statement

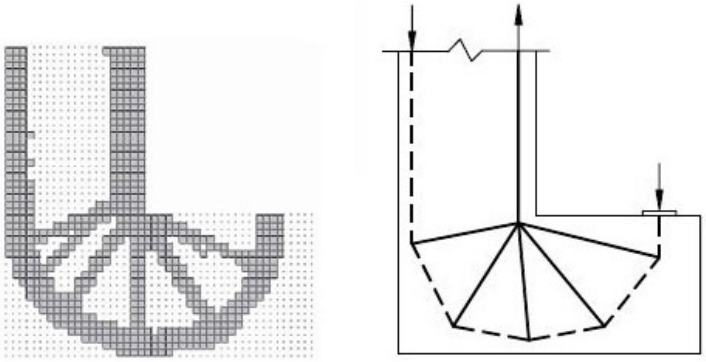


Figure 3.56: Case study n.12 - solution by Liang [4] with performance based ESO with displacement constraint and resulting strut and tie model

(a) maintaining ext. geometry (b) optimizing ext. geometry

Figure 3.57: Case study n.12 - solution derived by the code developed in this thesis for reinforced concrete - no concrete cover

(a) maintaining ext. geometry (b) optimizing ext. geometry

Figure 3.58: Case study n.12 - solution derived by the code developed in this thesis for reinforced concrete - with concrete cover

Chapter 4

Discussion

4.1 Discussion about chosen procedure

4.1.1 Choice of the structure of optimization algorithm

The peculiarities of the ESO applied to reinforced concrete include the following factors:

1. multiple materials are optimized
2. every material needs to be optimized with different criteria (steel reinforcement in tensile zones)
3. minimal sizes of members are different for each material (steel members can be much slender than concrete ones)

In literature, it is possible to find both multi-material optimization with same criteria in a single ESO loop [10] and single material multi-criteria optimization at once [13],[14], [15]. However, none of previous methods can be directly applied to reinforced concrete, since the methods found in literature respond to only one of above factors at a time. Therefore, it is needed to create a new variation of ESO procedure that could take into account all three factors simultaneously. This implies that the new ESO procedure has the following requirements:

1. more than one bi-phase material, and thus multiple matrices for densities x_i and sensitivities α_i, are needed
2. different sensitivity numbers definitions for α_i are required for each optimization criteria
3. different radiuses of sensitivity filter r_{min} are required for each material

One of most logic solutions is thus to have multiple optimization loops in series whereas in each loop there is a distinct bi-phase material, sensitivity number definition and radius of sensitivity filter. Other solutions may as well exist, however the choice of multiple ESO loops executed in series appears to be the most clear and intuitive.

4.1.2 Smooth change of sensitivity numbers

The choice of performing several optimization loops in series with different criteria, which has not been found by the author in the literature of ESO research, leads to a problem in convergence of the solution during the switch of optimality criteria.

During tests is has been noted that abruptly changing optimality criteria when a new ESO loop starts, the solution found in previous iteration can be lost, since many elements can be switched from one material to the other in the few first iterations with the new criteria, which may bring instabilities in the solution and sometimes even make it non-convergent.

It is in fact clear from previous researches that high number of elements change during an iteration could cause aforementioned problems. That is the reason why evolution ratio of volume is set generally low, between 1% and 5%. On the other hand, it is also noted that each element change during an

iteration influence the solution of next iteration, thus some mistaken additions/removals could lead the final solution to converge to a non optimal shape or not to converge at all.

Therefore, the concept of smoothing out the transition between two optimality criteria sounds quite logic and natural to try best to preserve the convergence of the solution.

It is found that a reasonable number of iterations for transition from one criteria to another is between 20 and 30. In case the number of iterations would be bigger that 30, the optimization loop could end before the final transition of optimality criteria is completed (since the second and third optimization loops of this thesis' code usually require between 30 and 70 iterations each to converge). The number of iterations for the transition in all case studies `SET.translenght` has been set to 20, to ensure that the transition to one criteria to another is carried on completely before convergence and as smoothly as possible.

4.1.3 Use of $mask$ and y density matrices

Together with the matrix $x\{i\}$ of the material densities, there are used two more parameters, $y\{i\}$ and $mask\{i\}$, where i is the ESO cycle number.

In the matrix $y\{i\}$ it is stored the density matrix from $x\{j\}$, where $j = 1, ..., n.cycles$, which describes the distribution between concrete and void for the solution.

In the matrix $mask\{i\}$ it is stored the density matrix from $x\{j\}$, where $j = 1, ..., n.cycles$, which contains the distribution between steel and concrete of previous ESO cycle, for influencing the results of current ESO cycle with previous ones.
Below here it is explained how they are used in the code.

Use of y matrix on stiffness

The ESO procedure does not eliminate completely the void elements from the mesh. In the first optimization loop the code simply sets its material properties, e.g. stiffness and Poisson's ratio to zero, since the bi-phase material of the first loop optimization considers void as one of the phases. In the next optimization loops the bi-phase material consider two non-void materials, therefore the void elements are calculated with the stiffness of the second phase of the bi-phase material (thus not zero) but penalized multiplying by the values contained in the matrix `y` which are 0.001 where void elements are located.

Use of y matrix on sensitivity

On the side of sensitivity numbers, the void elements calculated in the first ESO loop have density $x_{ij} = 0.001$ which diminishes the value of sensitivity considerably but not sets it to zero. This particular choice of not setting the sensitivity to zero, called soft-kill BESO, is discussed in the book of Xie [3] and has several advantages like the easier addition of previously removed material. Due to the non-zero value of sensitivity number of void elements, it may happen that void material elements could be converted to steel and thus a steel structure could be created outside of concrete, which is not desired in the case of reinforced concrete whereas the steel has to always remain inside the concrete. Therefore, it is enforced the boundary between concrete and void stored in the matrix `y` setting the sensitivity of void elements low as the minimum value of sensitivity. In the cases when it is used a concrete cover calculated in the function COVER (that is for `SET.covertype>=4`) the boundary between concrete and void is set inside the vector `ESO.cover` and enforced later in the ESOscript.

```
if SET.covertype>=4 %compute automatic cover for cases SET.covertype>=4
       [ESO.cover]=COVER(ELEM,SET,y,cycle);
else %outside y a no-material zone is enforced,
       %for SET.covertype>=4 it is done already in ESO.cover by COVER function
       ESO.alpha(y{cycle}~=1)=min(min(ESO.alpha));
end
```

In case of `SET.covertype>=4`, the boundary between concrete and void is applied with:

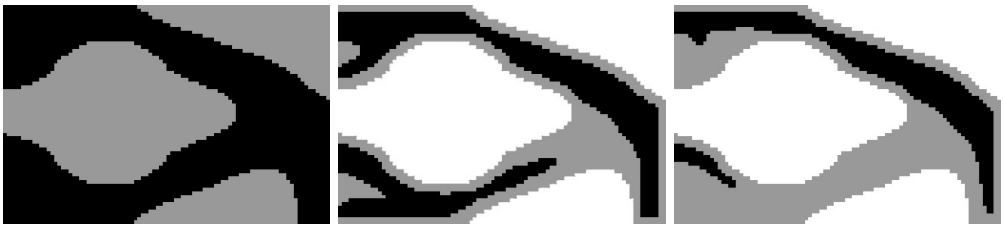
```
ESO.alpha(ESO.cover==1) = min(min(ESO.alpha));
```

Use of y matrix on representation

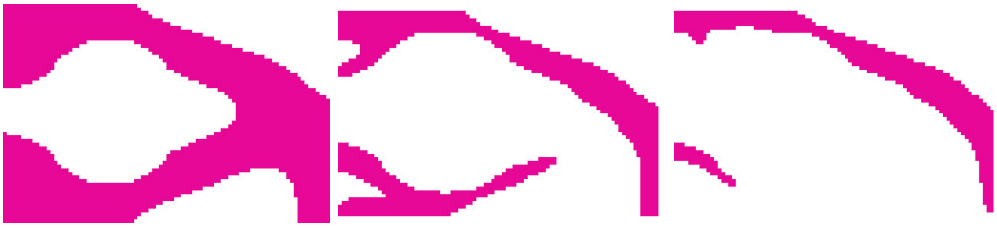
The values of `y` matrix are used to determine distribution between reinforced concrete and void in the solution, while the values of density matrix x define the distribution inside the reinforced concrete between concrete and steel. This is clearly shown in the print function from the file `ESOscript.m` reported here below:

```
colormap(gray); imagesc(-y{cycle}); hold on; fingeom=imagesc(-x{cycle}); hold off;
alpha(fingeom,.6); axis equal; axis tight; axis off;pause(1e-6);
```

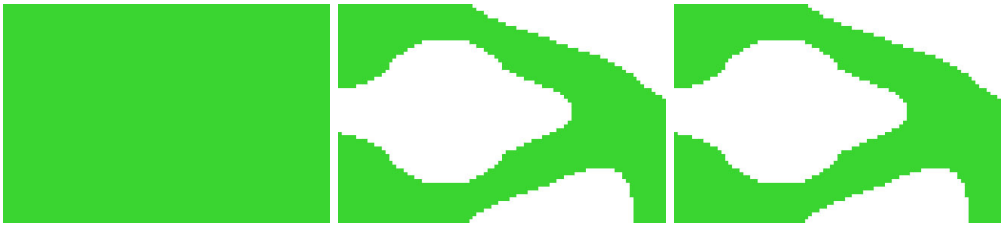
whereas with `imagesc(-y{cycle})` the void elements are colored in white and the reinforced concrete elements are colored (temporarily) in black, while with `fingeom=imagesc(-x{cycle})` and `alpha(fingeom,.6)` the concrete materials are colored with 60% opaque black, that results grey, and steel materials are left black.



(a) RC solution - end 1^{st} loop (b) RC solution - end 2^{nd} loop (c) RC solution - end 3^{st} loop



(d) x matrix - end 1^{st} loop (e) x matrix - end 2^{nd} loop (f) x matrix - end 3^{st} loop



(g) y matrix - end 1^{st} loop (h) y matrix - end 2^{nd} loop (i) y matrix - end 3^{st} loop

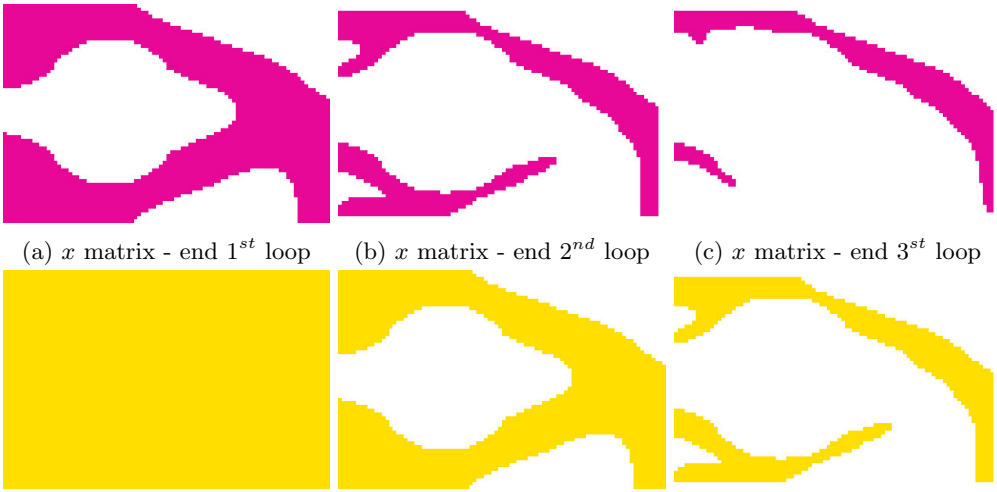
Figure 4.1: Case study n.5 - solutions and matrices x, y for each ESO loop

Use of *mask* matrix on sensitivity

The `mask` matrix is used to take into account of optimization history in the calculation of final solution. This means that the solution of ESO cycle n will be influenced by the solution of ESO cycle n-1 and so on. In fact, it is set:

```
update_mask=@(x0,x)    {x0,          x{1},           x{2}            };
...
mask=update_mask(x0,x);
```

This influence is exerted by multiplying the sensitivity numbers by $mask^{penalmask}$. The values of *mask* are 1 where in previous cycle there was the first phase material and 0.001 where there was the second phase material. Therefore, the elements that in previous iteration where turned from steel into concrete are penalized in their sensitivities to make them more likely to remain concrete, in other words not to let steel outside steel-concrete boundaries of previous iteration.



(a) x matrix - end 1^{st} loop (b) x matrix - end 2^{nd} loop (c) x matrix - end 3^{st} loop

(d) *mask* matrix - end 1^{st} loop (e) *mask* matrix - end 2^{nd} loop (f) *mask* matrix - end 3^{st} loop

Figure 4.2: Case study n.5 - matrices x, *mask* for each ESO loop

In the thesis the exponent *penalmask* has been chosen as 2, equal as the penalty exponent for x, that is $(penal - 1) = 3 - 1 = 2$. This exponent could be as well modified by the user manually changing in ESOscript the parameter `ESO.penalmask` with the desired amount. Setting a new exponent between 0 and 2, the penalization effect of the *mask* can be softened, whereas with exponent equal to 0 the effect of the mask would not be effective at all.

The solutions calculated with the aid of the *mask* usually perform more stably and tend to preserve small steel members which would otherwise be removed. Therefore, the use of the *mask* in most cases is advisable. However, much attention has to be taken if some needed tensile members are removed during first iterations due to their too small thickness. In case of the *mask* is applied in computation, this means that such lost members cannot be restored for the steel material, even in case the concrete design boundary is not modified. That is because during the optimization process the optimized geometry of the *mask* would have lost that member and carry this information to influence next ESO loops. Such issue is encountered in one of the case studies of this thesis and further discussed in the section "Discussion about solutions for case studies".

With the implementation of the *mask* it is also possible to steer the solution of the optimization process with other user defined matrices, for example taken from engineering practice or literature case studies. It is just needed to create, modify or define through an algorithm a matrix of the same size of the mesh and insert it in the preferred position in the array of masks: `update_mask=@(x0,x){x0, x{1}, x{2}};` according to which cycle it is wished to influence.

4.1.4 Order of materials in optimization loops

In the first ESO loop, logically, it should be needed to define the distribution of two materials - concrete and void. Consequentially steel would be added in the second and successive ESO loops, to define distribution of material between steel and concrete. However, such procedure determines instabilities in the solution of the optimization problem due to the fact that in the second loop it is introduced a more rigid material in the mix (bigger E modulus). This complies with the choice in the article of Xie [10] to rank materials by stiffness and start optimization from the stiffest one.

Therefore, the working solution is to use steel and void in the first ESO loop and in the second loop introduce the concrete. This means that the steel is substituted in second loop with concrete in the parts where it is not needed to have high strength at first (with Von Mises criterion) and secondly (with transition) where not needed to have tensile strength (with Drucker-Prager criterion). During third ESO loop, the steel continues to be substituted with concrete where it is not needed to have tensile strength (with Drucker-Prager criterion) and slowly transitioning in all other compression zones (with tensile stress

criterion).

4.1.5 Different materials in finite element analysis

In the code used in the thesis, the multiple materials in the finite element analysis are computed with their elastic modulus and Poisson's ratio as distinct materials. Therefore, the stresses are "continuous" in the change of materials while the strains are not. This means that the two materials in each bi-phase material does not fully work in a composite way. In analogy to a standard reinforced concrete FEM analysis, this corresponds to the case when the steel and concrete are "allowed" to slip between each other. This is just the result of the choice to compute concrete and steel with different stiffnesses in the finite element analysis and has to be kept in mind when analyzing the solutions.

4.1.6 Choice of sensitivity numbers formulations

In ESO literature, Von Mises and tensile stress optimizations [17] have been addressed with first generation ESO only (replaced by BESO method in 2007), which performed only element removal based on a rejection ratio and not employing sensitivity numbers at all. On another side, a Drucker-Prager type optimization can be found in more recent ESO literature [19], addressed with current BESO method, but derived through nonlinear FEA optimization using sensitivity numbers related to elastic and plastic energy. It is therefore found by the author that currently in ESO literature there are no "standard" formulas for sensitivity numbers for linear FEA optimization for Von Mises, Drucker-Prager and tensile stress BESO optimization.

In the current thesis, sensitivity numbers for stress-based optimization (for linear FEA and Bi-directional ESO) have been chosen in an heuristic way, as described here below. It has been chosen to have a multiplicator of x_{ij}^{p-1} inside all sensitivity numbers, in a similar way as it is in the sensitivity number for stiffness optimization.
Regarding Von Mises optimization, it has been chosen to have alpha to be proportional to the Von Mises stress $\sigma_{vm} = \sqrt{3J_2}$ and therefore the following definition for alpha has been used (formula 2.24):

$$\alpha_i = \sqrt{J_2}\, x_i^{p-1}$$

Regarding Drucker-Prager optimization, a definition of a Drucker-Prager stress does not exist, therefore, it has been chosen to use instead the formula of the yield criterion in the following way for alpha (formula 2.27):

$$\alpha_i = (\aleph I_1 + \sqrt{J_2})\, x_i^{p-1}$$

This formulation is very convenient since, in case of a material equally resistant in tension and in compression, the parameter $\aleph$ turns to be equal to zero and therefore the sensitivity number for Drucker-Prager optimization becomes the sensitivity number for Von Mises optimization. The definitions of the sensitivities for the two criteria are then strongly related.
Regarding tensile stress optimization, it has been chosen to have alpha proportional to the sum of the principal stresses $\sigma_{i,11} + \sigma_{i,22}$ (similarly as for first generation ESO method in the paper [17]), which results in the following definition for alpha (formula 2.28):

$$\alpha_i = \Big(\sigma_{1,i} + \sigma_{2,i} + \big|\min(\sigma_1)\big| + \big|\min(\sigma_2)\big|\Big)\, x_i^{p-1}$$

It has been chosen to "shift" all values of the principal stresses (summing $|\min(\sigma_{i,11})|$ and $|\min(\sigma_{i,22})|$), so that first phase material elements with $\sigma_{11} \neq 0$, $\sigma_{22} \neq 0$: $\sigma_{11} + \sigma_{22} \simeq 0$ wouldn't have sensitivity numbers similar to second phase material elements with $\sigma_{11} \neq 0$, $\sigma_{22} \neq 0$: $(\sigma_{11} + \sigma_{22})x^{p-1} = (\sigma_{11} + \sigma_{22})0.001^{p-1} \simeq 0$. If there wouldn't be a "shift" of $|\min(\sigma_{i,11})| + |\min(\sigma_{i,22})|$ in the definition of α, both cases described above would have $\alpha = 0$. This could cause mistaken element removals and/or recovers. So, thanks to the "shift" in the modified definition of alpha, the elements with $\alpha \simeq 0$ are all and only the elements of second phase material. And a correct definition of alpha should indeed be able to separate through the values of alpha the "penalized" second phase material elements from the others of the first phase. The formula 2.28 is able to do that and has shown to perform well along all case studies in the thesis.

4.2 Discussion about solutions for case studies

4.2.1 Target volume of reinfocred concrete

Regulating the target volume of reinforced concrete (that is changing the parameter `ESO.volfrac` for the first ESO loop), it is possible to influence the number and size of holes in the structure. It is observed that setting a target volume lower than 90%-85%, holes start to be created inside the structure. When the target volume is set approximately lower than 55%-50%, the resulting optimized structure starts resembling a lattice structure comprised by tensile and compressive members. In case the target volume is set too low, relevant but thin members are eventually removed and the structure turns to a highly stressed non-optimal solution.

The effect of various reinforced concrete target volumes is shown in figure 4.4 for case study n.1, solved with concrete cover and with the following target volumes for first ESO loop: 95%, 90%, 80%, 70%, 55%, 35%. Together with the target volume of the reinforced concrete (`ESO.volfrac` for first ESO loop), also the intermediate and final target volume of steel (`ESO.volfrac` for second and third ESO loop) have been changed to maintain a similar proportion as in solution of case study n.1 from chapter 3. All values of `ESO.volfrac` are reported under each corresponding image. In figures 4.4(e) and 4.4(f), the structure becomes respectively a lowly and highly sub-optimal solution, considering the actual slenderness of the beam, with respect to the solution of 4.4(d).

For non-slender structures (that is when $0.3 \div 0.5 < height/lenght < 2 \div 3$) it is usually possible to have a smaller target volume than for slender structures. In fact, slender structures tend to have a higher number of members than non-slender ones and therefore need to have a proportionally higher volume requirement to preserve the optimum solution.

In this context, it is useful to remember that the minimum size of members is related to `ESO.rmin`, the radius used in the filtering of sensitivities. When r_{min} is increased then also the minimum "allowed" size of members is increased. The members with thickness smaller that the resulting minimum are "penalized" in the sensitivity numbers of their elements and therefore removed from the design.

To get an optimal solution in terms of both structural efficiency and manufacturability it is needed therefore to choose a target volume and a r_{min} for the first ESO loop not too small, in order to avoid small-volume-non-optimal solutions and to keep the resulting members thick enough to avoid problems of concrete cover and instability. Moreover, in some cases it is preferable to use an even higher target volume than required, so to limit the number of holes in the structure and mantain a simple and cost-effective geometry. In the solution of case studies in the thesis it has been used for the first ESO loop the parameters: $r_{min} = 8$ and target volume $= 0.50 \div 0.60 \div 0.70$.

4.2.2 Target volume of steel

In general, it is observed that it is needed to set a relatively high target volume of steel in the algorithm, i.e. 10%, 15%, 20% (instead of around 1% what usually employed in reality) to show solutions of interests. In fact, it has been noted that if the ESO procedure for steel is taken to target volumes smaller than 10%, the solution is comprised by only a member in the highest tensile zone while the steel members on the secondary tensile zones vanish. In this case the solution becomes trivial and fails to address the more complete solution.

An example of this effect is shown on the case study n.1 in figure 4.6. In such study, it has been kept a fixed target volume of 0.70 for first ESO loop and of 0.35 for second loop. The target volume of third ESO loop has been varied choosing between the following values: 0.20 - 0.15 - 0.10 - 0.05. As it is possible to see in the pictures in figure 4.6, after the target steel volume of 0.20, the solution start to become trivial, losing steel members in tensile zones where needed. In fact, in the solution 4.6(b), corresponding to target volume of 0.15, the upper part of a vertical tensile member is lost, while in pictures 4.6(c) and 4.6(d) the solution excludes completely all vertical tensile members.

To choose the smallest yet still useful target volume for steel, the author advices to set a small target volume, e.g. 0.10, and observe the evolution of the steel distribution on the plotted image on screen and the relative value of volume, printed in the MATLAB command window while the script execution is still on. Then, with engineering judgement, it is needed to choose a target volume corresponding to one of last non-trivial solutions (before relevant members are removed) and re-run the script with the adjusted target volume.

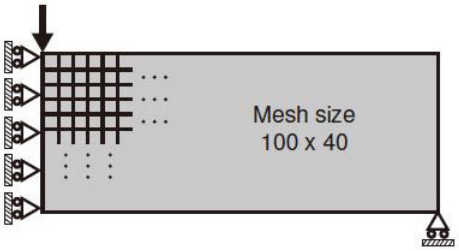
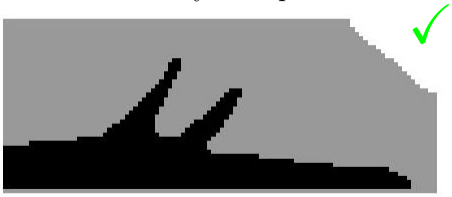
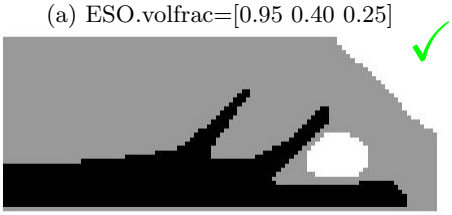


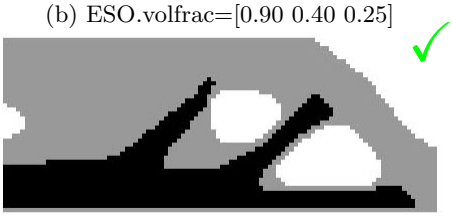
Figure 4.3: Case study n.1 - problem statement



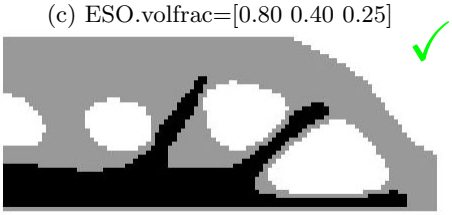
(a) ESO.volfrac=[0.95 0.40 0.25]



(b) ESO.volfrac=[0.90 0.40 0.25]

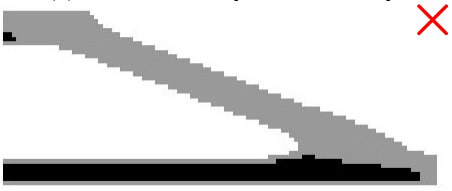


(c) ESO.volfrac=[0.80 0.40 0.25]



(d) ESO.volfrac=[0.70 0.35 0.20]

(e) ESO.volfrac=[0.55 0.30 0.15]



(f) ESO.volfrac=[0.35 0.20 0.10]

Figure 4.4: Solutions for case study n.1 with different reinforced concrete target volumes, with concrete cover

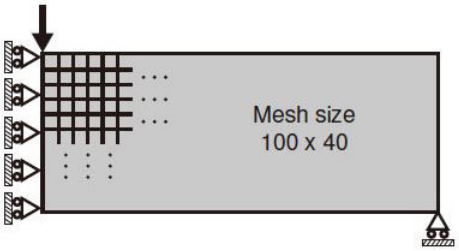


Figure 4.5: Case study n.1 - problem statement

(a) steel target volume = 0.20

(b) steel target volume = 0.15

(c) steel target volume = 0.10

(d) steel target volume = 0.05

Figure 4.6: Solutions for case study n.1 with different steel target volumes, with concrete cover

It has been observed that, depending on the case study, the lowest useful values for target volume of steel for the current optimization code are usually ranging in the interval 15-20%. This results in bulky but lowly stressed zones of steel, that could not be used as reference in practice directly but need to be interpreted to create the resulting reinforcement layouts. It is then more appropriate to refer to the black zones in the solutions from the ESO procedure, not as steel but as zones where steel reinforcement is desired, and provide a successive interpretation making use of principal stress plots.

4.2.3 Target volume of steel in intermediate ESO loop

Target volumes for first and third ESO loop have been addressed in previous sections, respectively in 4.2.1 and 4.2.2. Regarding target volume of second ESO loop, the following recommendations have been formulated after experiments during the solutions of the thesis case studies.

Since in the second loop there is the transition from Von Mises to Drucker-Prager optimization criteria,

the resulting steel area will be both in tensile and compressive zones, with members of higher thickness in tensile zones. Therefore, the presence of compressive members (to be removed in third ESO loop) are to be accounted for in the target volume of second ESO loop too. It is required then to choose a target volume not too close to one of the third ESO loop. In the examples of the thesis, it has been used the following range:

$$\text{ESO.volfrac}\,(2) \approx 1.5 \div 2\ \ \text{ESO.volfrac}\,(3)$$

On the other hand, in the second ESO loop it is desirable to reduce the thickness of the members in compressive zones as much as possible to ease their removal in the third ESO loop. Therefore, the target volume for the second loop should be as small as possible but should still be able to retain the structural scheme obtained in the first ESO loop. During the solution of case studies in the thesis, it has been used the following range:

$$\text{ESO.volfrac}\,(2) \approx 0.5 \div 0.7\ \ \text{ESO.volfrac}\,(1)$$

4.2.4 Effect of stress concentrations

It has been noted that steel members that have ends on edges of internal holes are more stable and "resistant" to removal (meaning that useful thin members remain even at small target volumes). This effects of "resistance" to removal of steel members anchored to holes, seems related to the presence of stress concentration around openings, which makes the tensile elements positioned right on the edge of the hole harder to eliminate with tensile stress criterion.

When the steel members not "anchored" to holes, at a certain point in the ESO loop, "retract" and get shorter than engineering judgement would suggest. This effect is specially displayed in solutions where concrete boundary is kept rectangular, i.e. not optimized, where the vertical steel members seem shorter and/or less in number and/or removed, with respect to the solutions with the optimized concrete boundary. This can be clearly seen in the examples of case studies n.2, 6, 12 reported in figure 4.7.

(a) Solution for case study n.2 - original external geometry

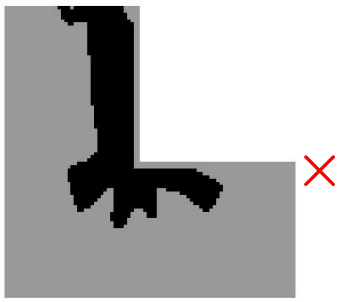
VS

(b) Solution for case study n.2 - optimized external geometry

(c) Solution for case study n.6 - original external geometry

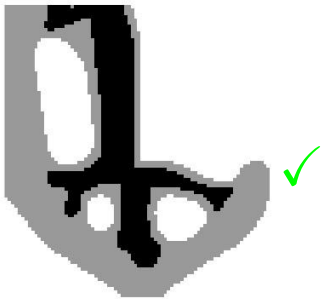
VS

(d) Solution for case study n.6 - optimized external geometry



(e) Solution for case study n.1 - original external geometry

VS

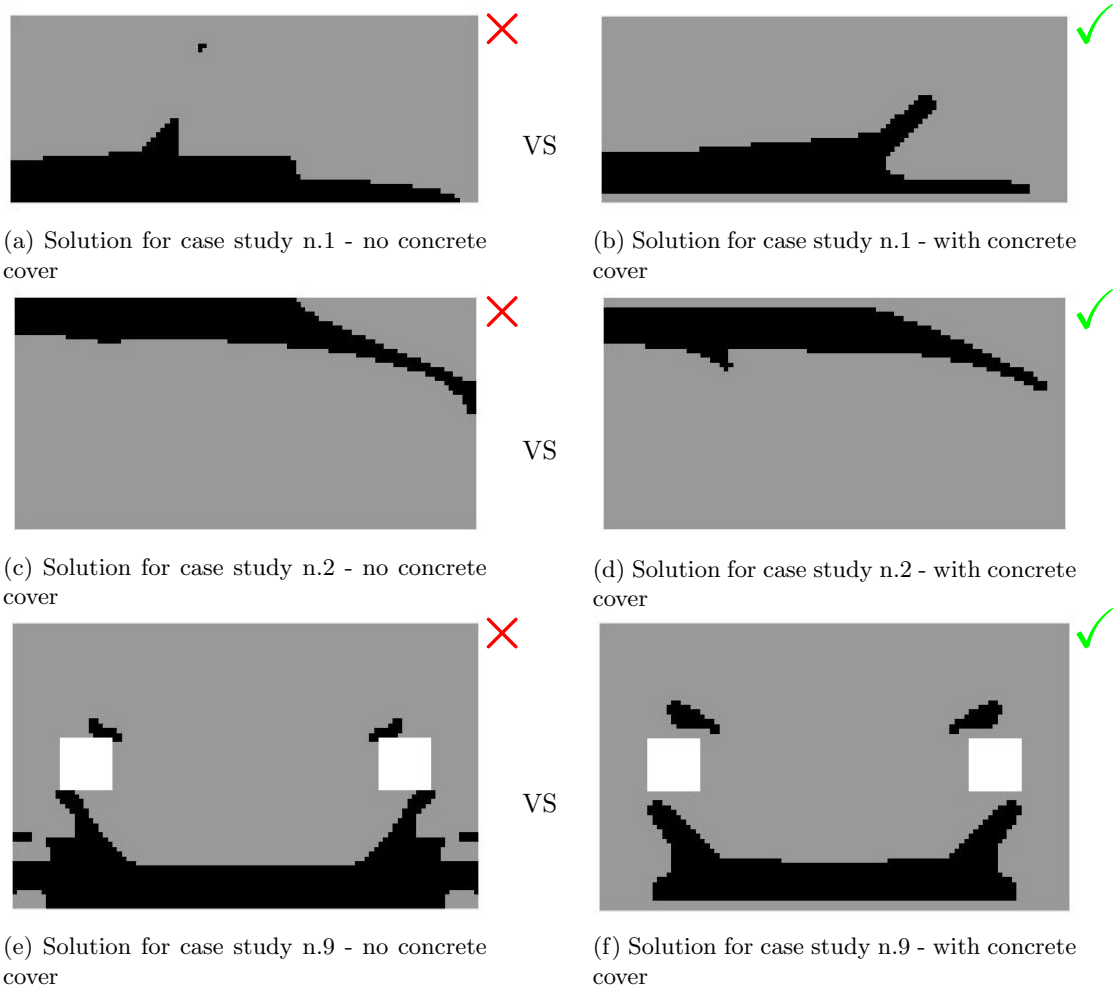


(f) Solution for case study n.12 - optimized external geometry

Figure 4.7: Comparation of solutions for case studies with and without "anchoring" effect of steel to holes

4.2.5 Effect of concrete cover

Solutions calculated with the concrete cover enforcement in most of the cases deliver a better solution, since it removes steel from the design boundary and around holes in the geometry, therefore reducing the bulkiness of the steel areas and therefore the better steel distribution for a given steel volume, with respect to the solutions without concrete cover. This is specially shown in the examples in figure 4.8.



(a) Solution for case study n.1 - no concrete cover

(b) Solution for case study n.1 - with concrete cover

(c) Solution for case study n.2 - no concrete cover

(d) Solution for case study n.2 - with concrete cover

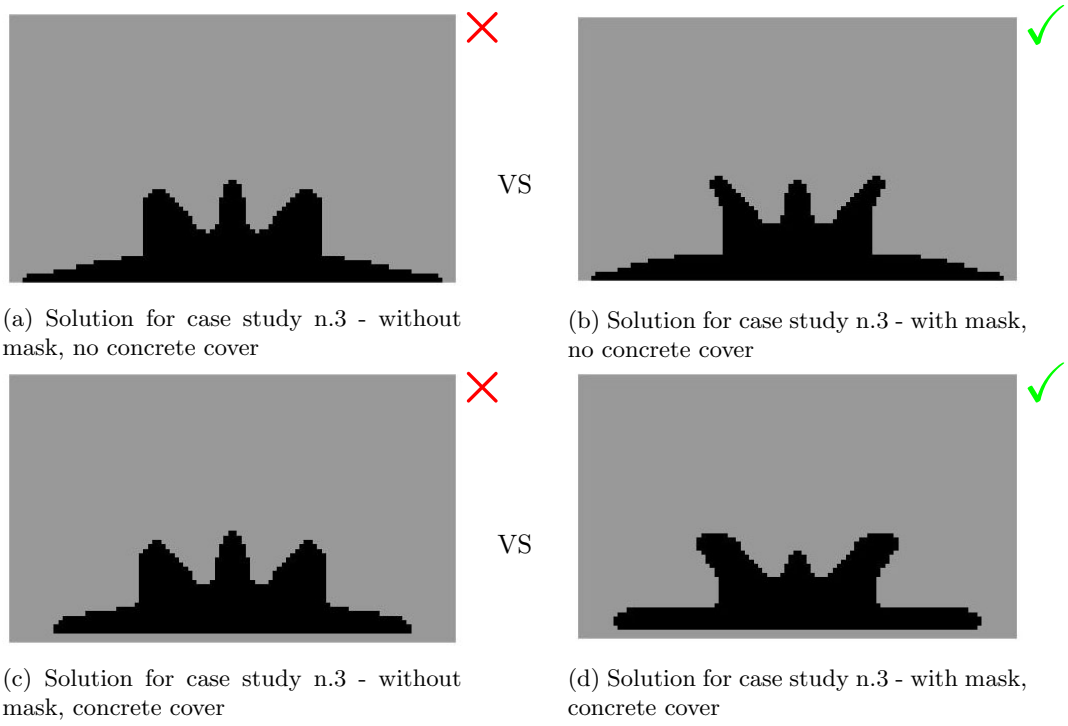
(e) Solution for case study n.9 - no concrete cover

(f) Solution for case study n.9 - with concrete cover

Figure 4.8: Comparation of solutions for case studies with and without concrete cover

4.2.6 Effect of mask

The effect of mask on solutions is possible to see in the examples in figure 4.9, calculated for case study n.3 with and without mask. In the case calculated without mask it has been set `ESO.penalmask=0`, whereas in the case calculated with mask the settings used were `ESO.penalmask=2`. Setting the penalty exponent of the mask set to zero, it completely nullifies the effect of the mask because $mask^{penalmask} = mask^0 = 1$. Therefore, the sensitivity numbers are multiplied by 1 no matter what the values in the *mask* matrix. For having a better understanding of the effect of the *mask*, without other influences, it has been set `SET.covertype=2`, because such concrete cover has no effect over the zones affected by the *mask*. In fact, concrete covers `SET.covertype=4,5,6,7,8` would have brought an additional effect due to the information of holes from previous ESO loops inside the `ESO.cover` matrix.
The images shown in figure 4.9 demonstrate that the application of the mask preserves the 45 degrees direction of the left and right steel members to counter the shear stresses, while in the solution without mask such members change angle to 90 degrees which results in a less efficient solution.



(a) Solution for case study n.3 - without mask, no concrete cover

(b) Solution for case study n.3 - with mask, no concrete cover

(c) Solution for case study n.3 - without mask, concrete cover

(d) Solution for case study n.3 - with mask, concrete cover

Figure 4.9: Comparation of solutions for case studies with and without mask

There is an example in the case studies of the thesis, where the application of the *mask* results in an inefficient solution. Such issue occurs in combination with the problem of removal of thin members in ESO optimization. The latter problem has been demonstrated with a famous case study under the name of the Zhou-Rozvany problem, depicted in figure 4.10 and found cited in many articles in literature. In the article of Xie [11] a brief but well explanatory summary of the problem is presented. In such problem it has been demonstrated that the optimal solution found by ESO algorithm could be only a local optimum, far more inefficient than the real optimum.

In fact, the ESO solution for the Zhou-Rozvany problem would at first remove the tensile vertical member because highly stressed. However, as result it would create a simple cantilever with much more stressed elements. The general optimal solution would be to not remove any element from the tensile vertical member but only in the horizontal beam.

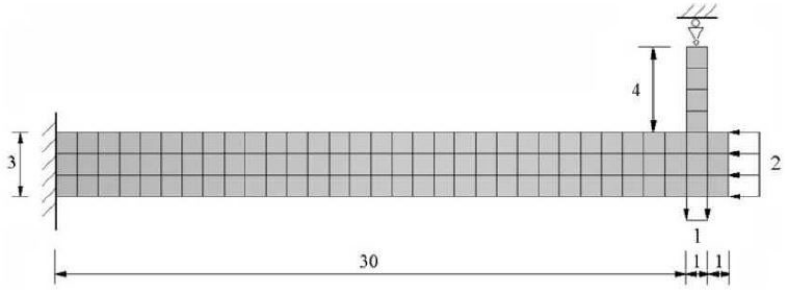


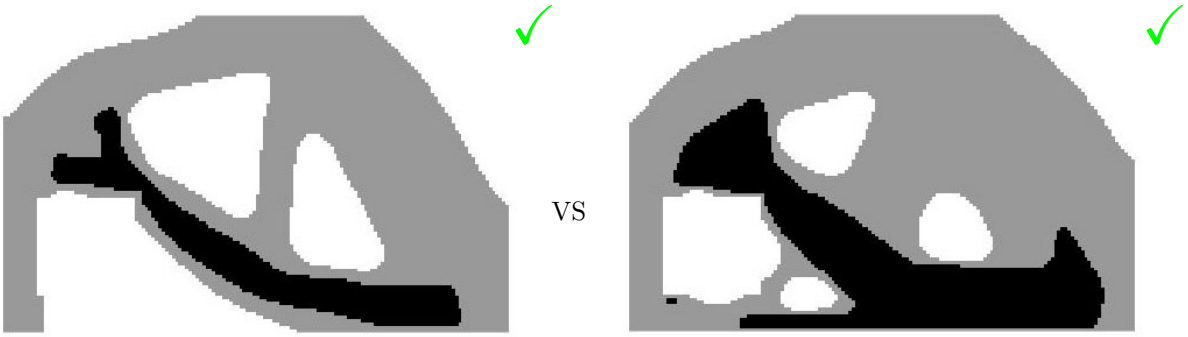
Figure 4.10: An illustration of the Zhou-Rozvany problem

Similar type of non-optimal element removal is encountered in case study n.10 where the lower left member close to the square hole is too small and is removed during first ESO loop thus generating an inefficient local optimum solution, as shown in figure 4.11(a). Moreover, due to the application of the *mask* in the calculation of the solution in figure 4.11(c), the thin member removal from the first ESO loop, noticeable in figure 4.11(a), causes the impossibility to restore a tensile member in the lower left part of the beam. This is because the information of the first ESO loop is stored inside the ma-

trix *mask* and used to drive the solution of second ESO loop inside those boundaries. Therefore, the wrong removal of a thin member during the first ESO loop combined with the application of *mask* for the solution of the problem with original concrete boundaries, shown in figure 4.11(c), does not work well.

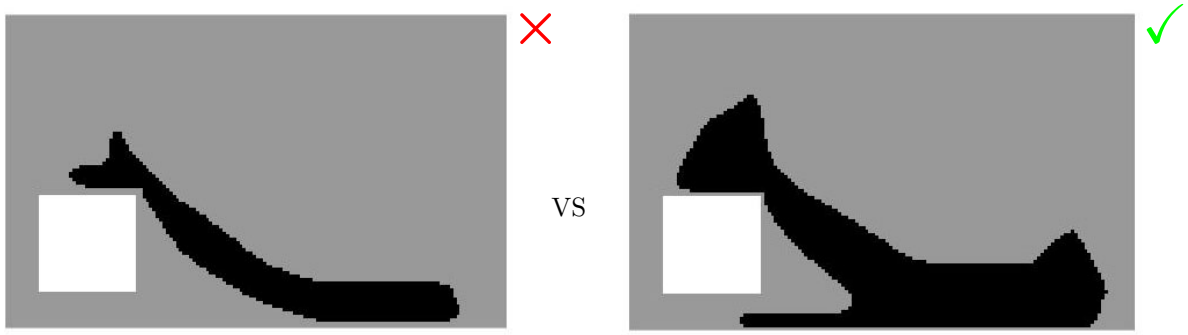
It is noted therefore that to apply the *mask* it is necessary to monitor intermediate solutions during the ESO process, to make sure that potentially useful members are not removed, and this holds true specially in the case when the original rectangular concrete boundary needs to be preserved.

For the case study n.10, by trial and error it has been found that for volume fractions of first ESO loop bigger than 0.75 the lower left member is preserved while for values of `volfrac` smaller than 0.75 it is removed. The problem of work element removal + application of mask is resolved in figure4.11(d), that is a case study calculated with first `volfrac=0.76`.



(a) Case study n.10, first loop target volume=0.60

(b) Case study n.10, first loop target volume=0.76



(c) Case study n.10, first loop target volume=0.60

(d) Case study n.10, first loop target volume=0.76

Figure 4.11: Comparation of solutions for case study n.10 with different first loop target volume - with concrete cover

4.3 Evaluation and interpretation of solutions for case studies

In the following section, the solutions for all case studies with concrete cover are shown along with the plots of the principal stresses. For each case study, three set of solutions are shown:

- original geometry, no reinforcement;
- original geometry with optimized reinforcement;
- optimized geometry and reinforcement.

An example interpretation of the solutions from the ESO procedure is given for finding resulting steel layouts.

4.3.1 Case study n.1

Case study n.1 - original geometry, no reinforcement

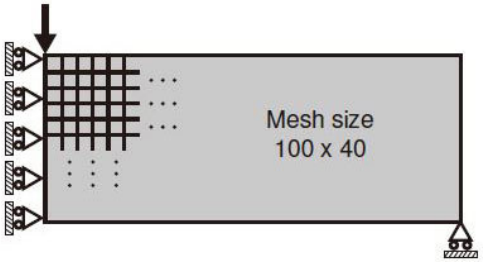
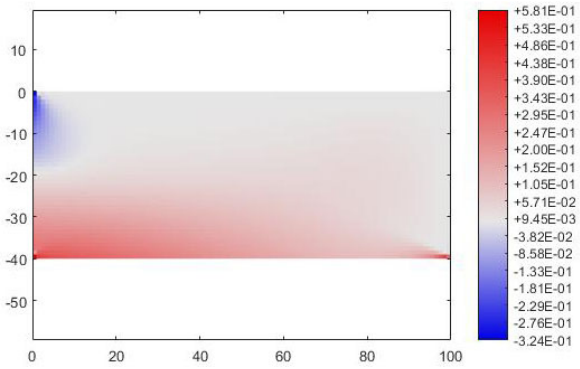
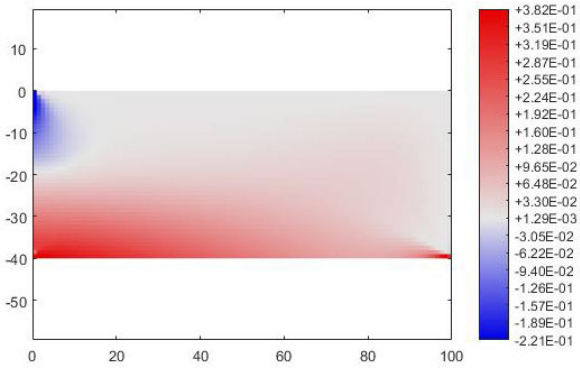


Figure 4.12: Case study n.1 - problem statement

Figure 4.13: Case study n.1 - geometry without reinforcement and optimization

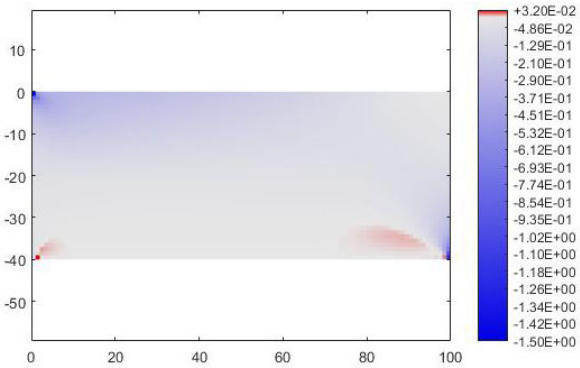


(a) full range of stresses

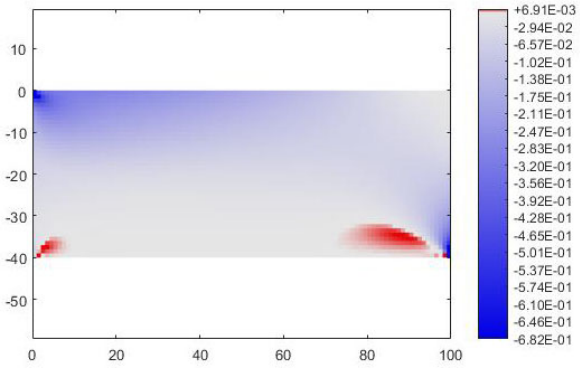


(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.14: Case study n.1 - plots of principal stress sigma 1



(a) full range of stresses



(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.15: Case study n.1 - plots of principal stress sigma 2

Case study n.1 - original geometry with optimized reinforcement

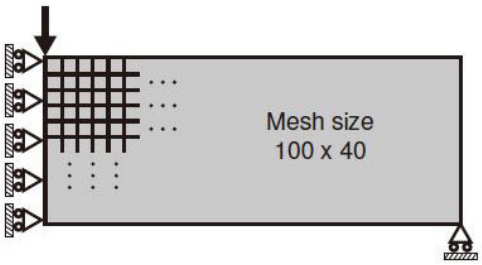
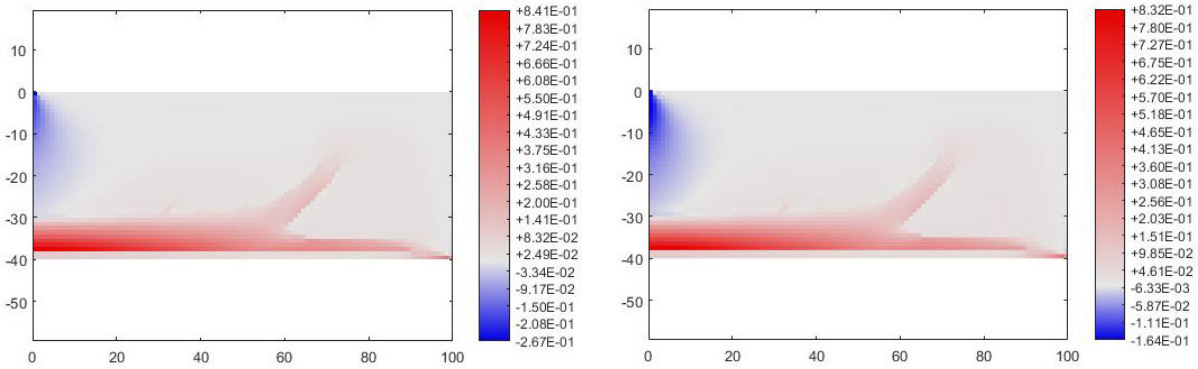


Figure 4.16: Case study n.1 - problem statement

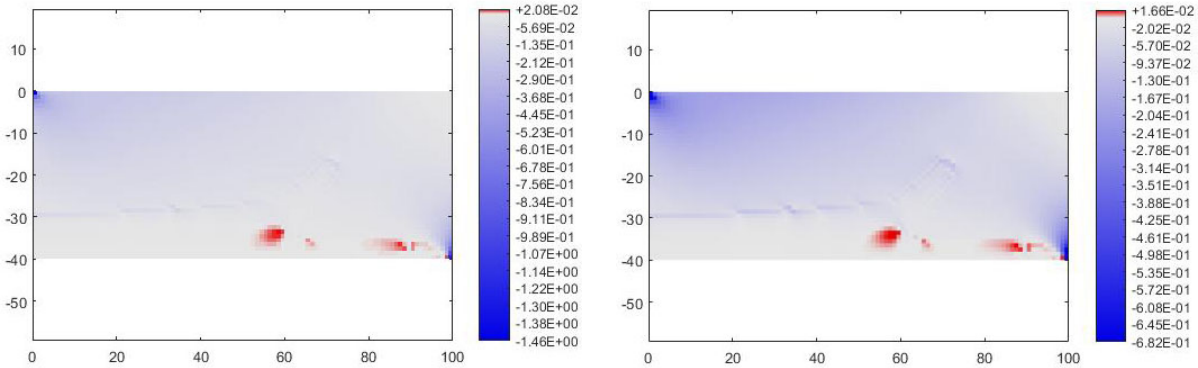
Figure 4.17: Case study n.1 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.18: Case study n.1 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.19: Case study n.1 - plots of principal stress sigma 2

Case study n.1 - optimized geometry and reinforcement

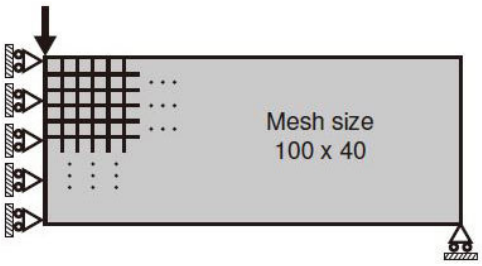
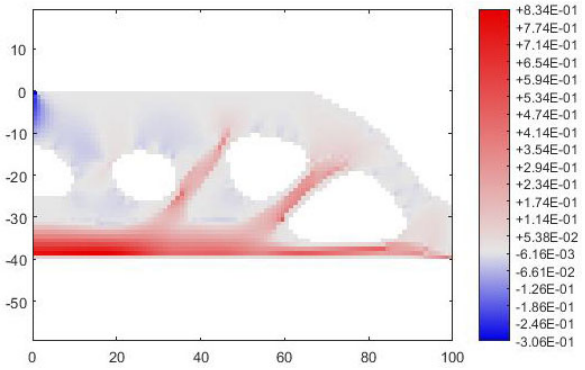
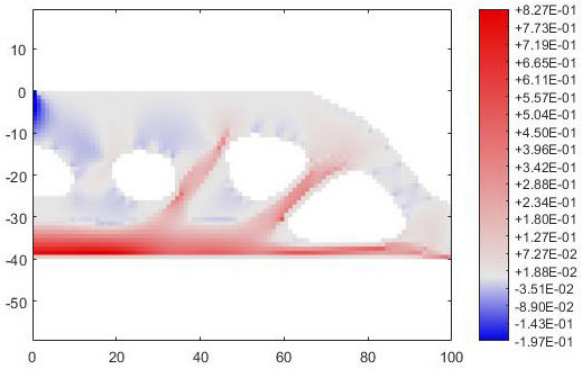


Figure 4.20: Case study n.1 - problem statement

Figure 4.21: Case study n.1 - solution by the code of this thesis

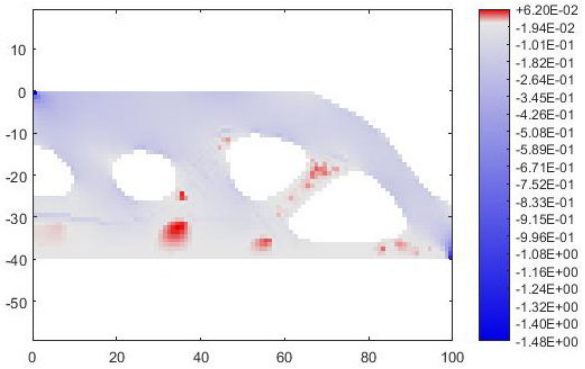


(a) full range of stresses

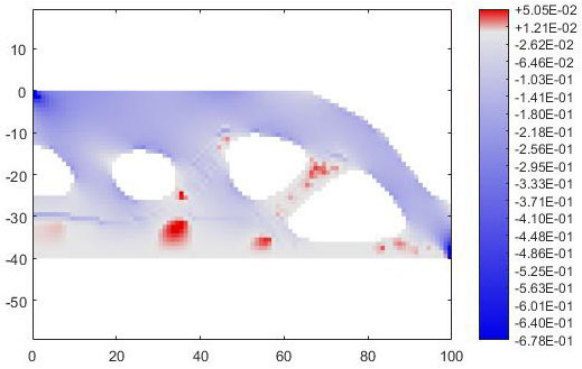


(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.22: Case study n.1 - plots of principal stress sigma 1



(a) full range of stresses



(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.23: Case study n.1 - plots of principal stress sigma 2

Case study n.1 - results interpretation

The case study n.1 corresponds to the standard benchmark test of the MBB beam, a beam simply supported with a load in the middle. In the ESO code it is modelled only its half, and the final result is mirrored.

Figure 4.24: Case study n.1 - final solution - original structure boundaries

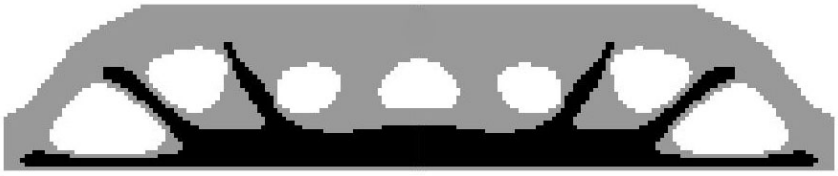


Figure 4.25: Case study n.1 - final solution - optimized structure boundaries

It is possible to observe the presence of steel in the bottom part of the beam to absorb tensile stresses generated by bending. On the sides, diagonal steel members are present to resist shear stresses in the beam. In the case of original beam geometry, the shear members have angle measured to be around 43°. In the case the optimized structure geometry, there are two couples of shear members, the ones closer to the support with angle around 43° and the others with angle of approximately 54°. The interpretation of the solution is the following:

Figure 4.26: Case study n.1 - example interpretation of solution - original structure boundaries

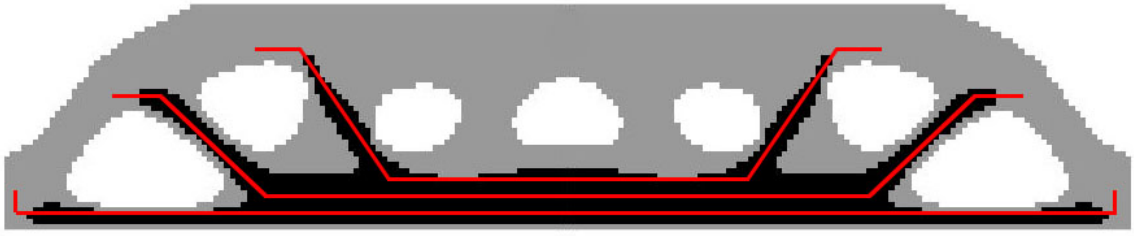


Figure 4.27: Case study n.1 - example interpretation of solution - optimized structure boundaries

In both solutions, from the analysis of principal stresses plots, the only remaining tensile zones in concrete are on supports. Steel reinforcement in those zones has to be manually added by the user in phase of detailing of the reinforcement layout.

4.3.2 Case study n.2

Case study n.2 - original geometry, no reinforcement

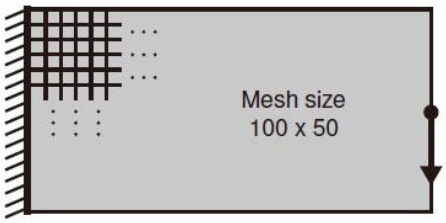
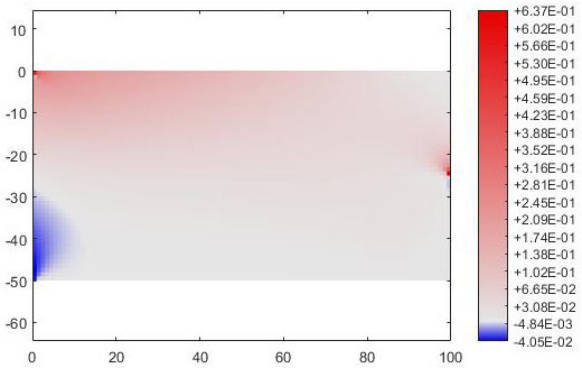
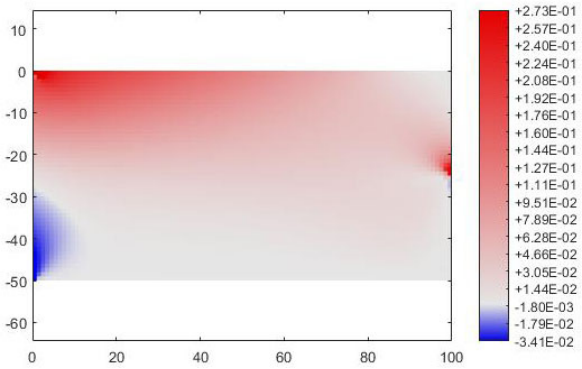


Figure 4.28: Case study n.2 - problem statement

Figure 4.29: Case study n.2 - geometry without reinforcement and optimization

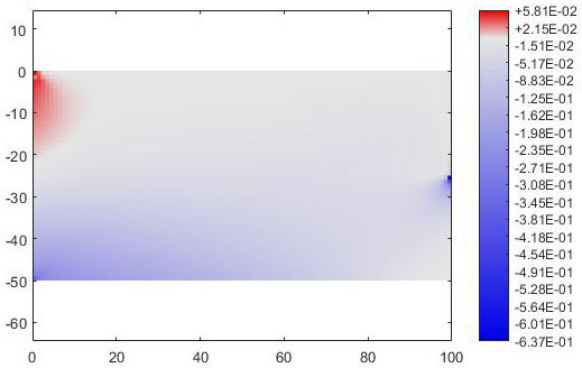


(a) full range of stresses

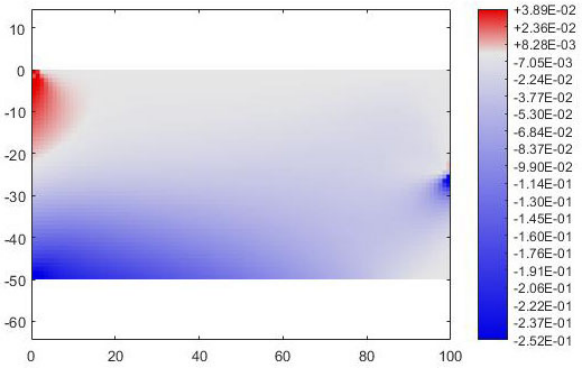


(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.30: Case study n.2 - plots of principal stress sigma 1



(a) full range of stresses



(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.31: Case study n.2 - plots of principal stress sigma 2

Case study n.2 - original geometry with optimized reinforcement

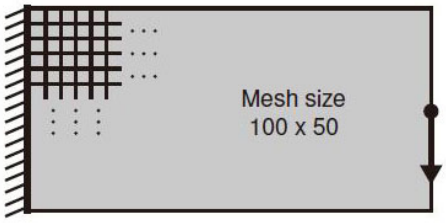
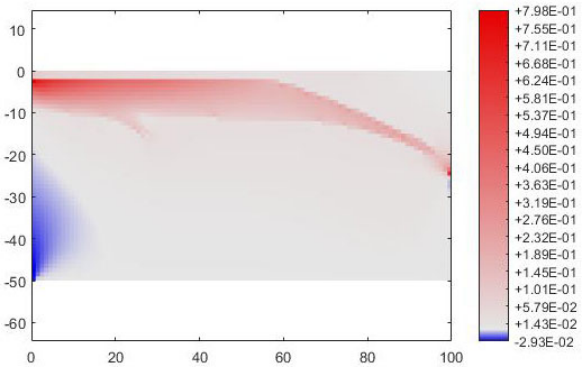
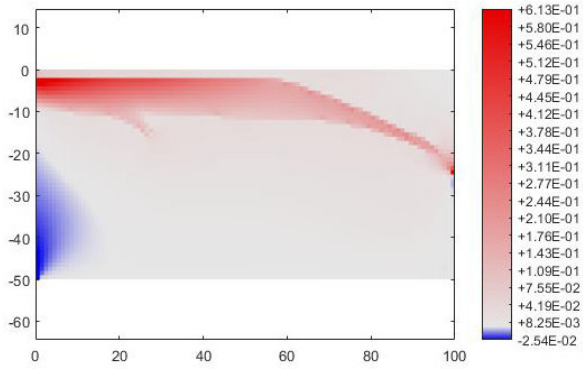


Figure 4.32: Case study n.2 - problem statement

Figure 4.33: Case study n.2 - solution by the code of this thesis

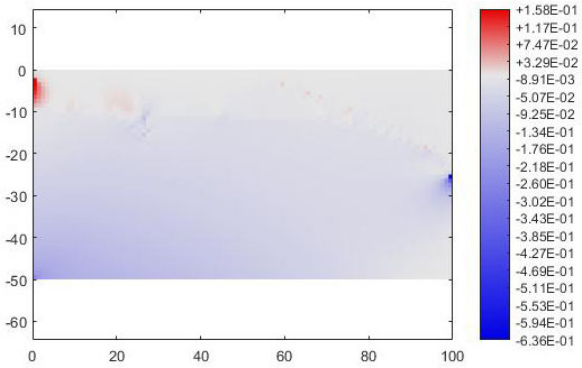


(a) full range of stresses

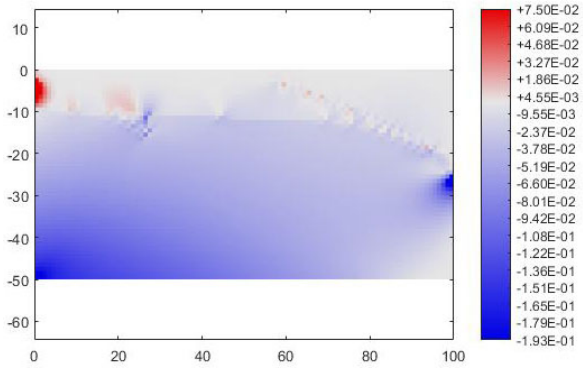


(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.34: Case study n.2 - plots of principal stress sigma 1



(a) full range of stresses



(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.35: Case study n.2 - plots of principal stress sigma 2

Case study n.2 - optimized geometry and reinforcement

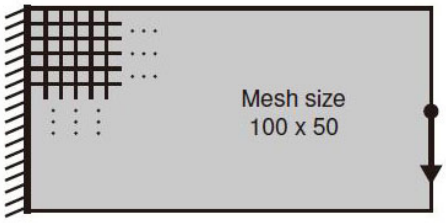
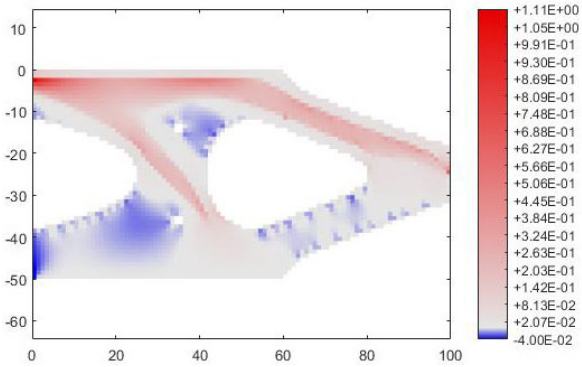
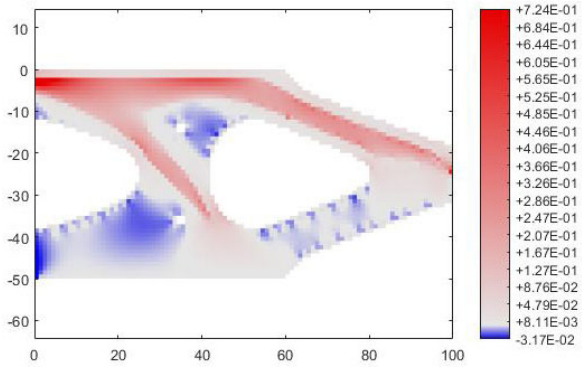


Figure 4.36: Case study n.2 - problem statement

Figure 4.37: Case study n.2 - solution by the code of this thesis

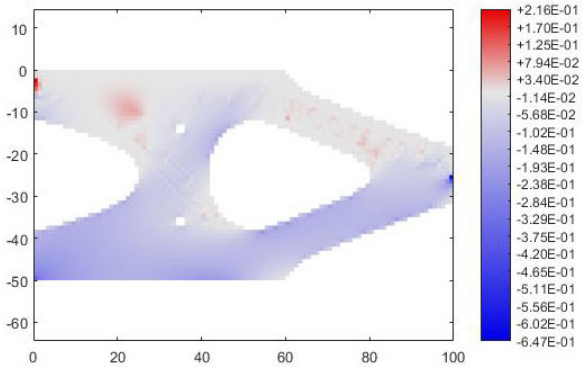


(a) full range of stresses

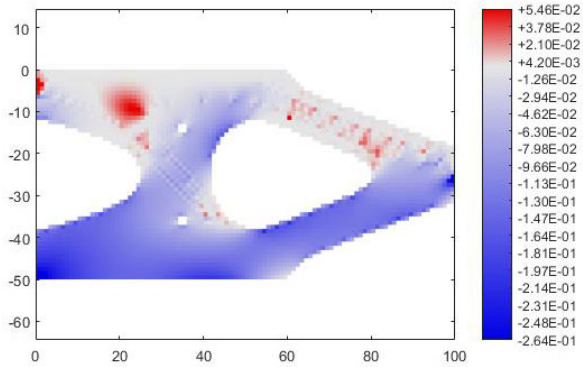


(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.38: Case study n.2 - plots of principal stress sigma 1



(a) full range of stresses



(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.39: Case study n.2 - plots of principal stress sigma 2

Case study n.2 - results interpretation

The case study n.2 corresponds a short cantilever with lenght = 2 x width, with centered force on the right edge. In the solution of the ESO procedure two tensile members are recognizable, the "external" and longer to counter the bending forces and the one closer to the constraints to resist the shear forces. In the solution with the original rectangular boundaries, the shear steel member gets very short during the optimization procedure but still can be seen and interpreted. The steel member closer to the support has an angle of approximately 49° in case the reinforced concrete geometry is optimized, and of 35° circa, in the case the geometry is kept rectangular. In both cases, the longest steel member, the one close to the load, has inclination around 22°.

The interpretation of the solution is the following:

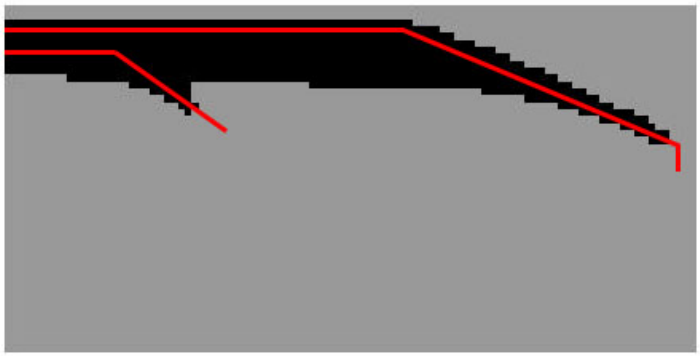


Figure 4.40: Case study n.2 - example interpretation of solution - original structure boundaries

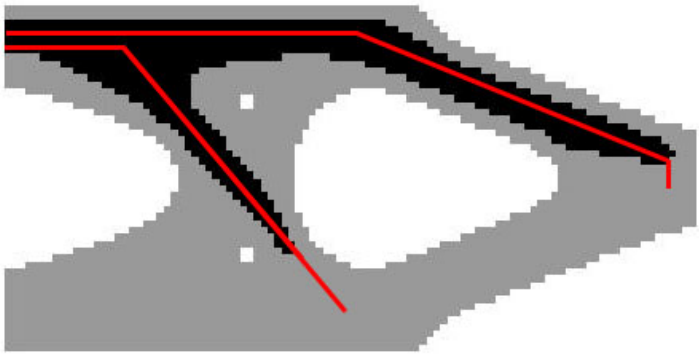


Figure 4.41: Case study n.2 - example interpretation of solution - optimized structure boundaries

From analysis of principal stress plots, in the case the reinforced concrete boundaries are optimized, a relatively high tensile concrete zone remains in the bottom part of the beam, the zone neighboring the left corner of the hole and below the end of the central oblique steel member. In the example interpretation above, the steel member has been elongated to take on such tensile stresses. The reinforcement close to the load is also elongated, to improve the load introduction in the structure.

4.3.3 Case study n.3

Case study n.3 - original geometry, no reinforcement

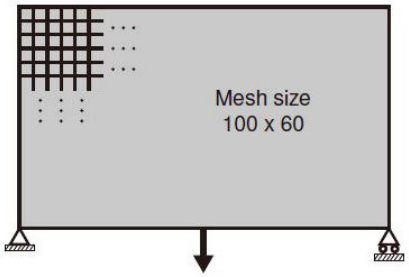
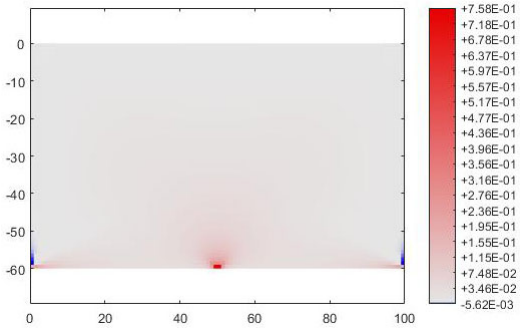
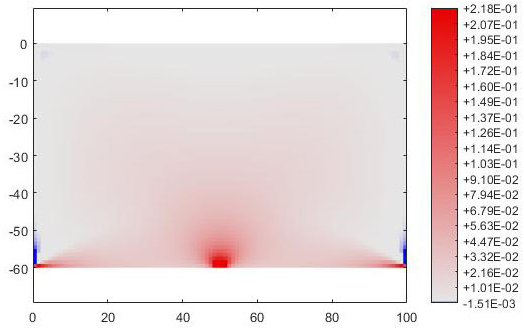


Figure 4.42: Case study n.3 - problem statement

Figure 4.43: Case study n.3 - geometry without reinforcement and optimization

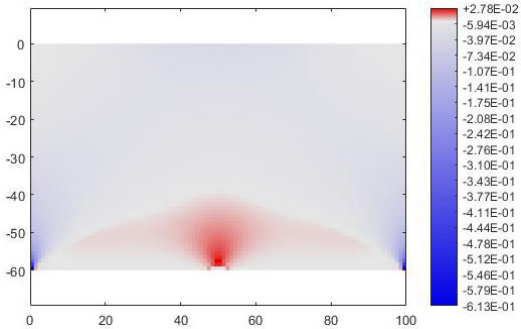


(a) full range of stresses

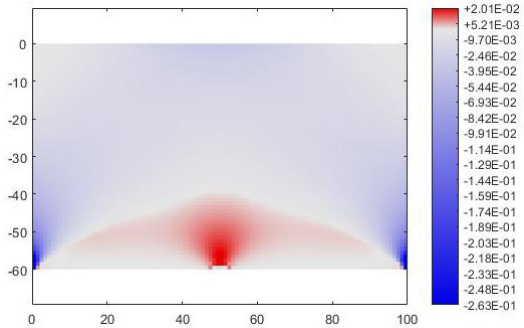


(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.44: Case study n.3 - plots of principal stress sigma 1



(a) full range of stresses



(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.45: Case study n.3 - plots of principal stress sigma 2

Case study n.3 - original geometry with optimized reinforcement

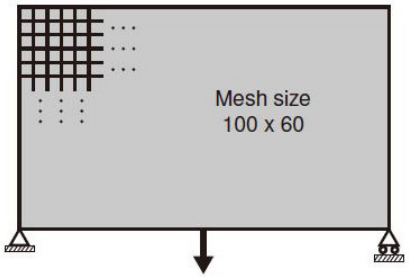
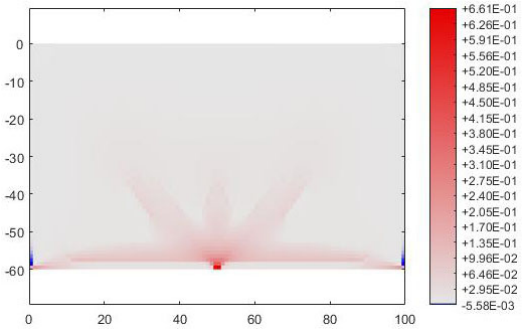
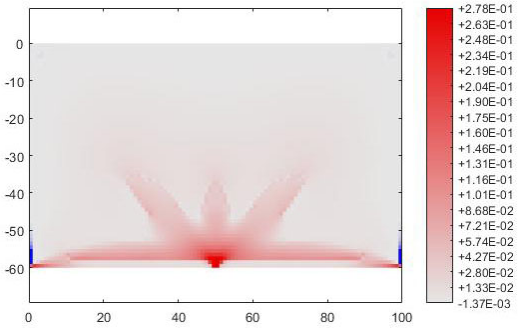


Figure 4.46: Case study n.3 - problem statement

Figure 4.47: Case study n.3 - solution by the code of this thesis

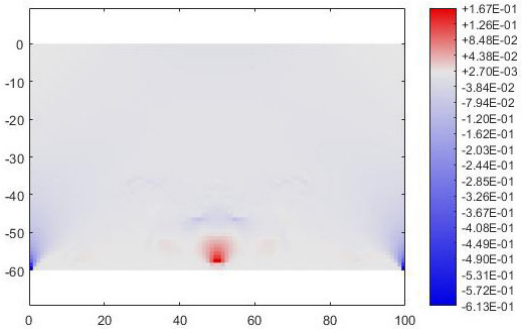


(a) full range of stresses

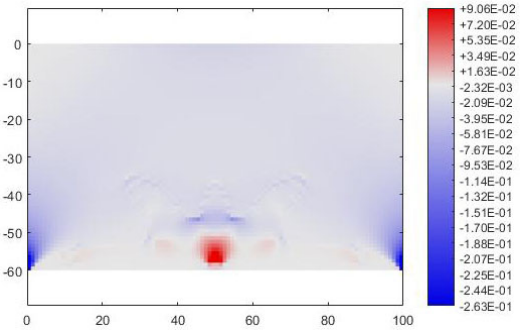


(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.48: Case study n.3 - plots of principal stress sigma 1



(a) full range of stresses



(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.49: Case study n.3 - plots of principal stress sigma 2

Case study n.3 - optimized geometry and reinforcement

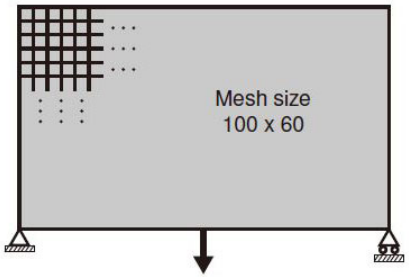
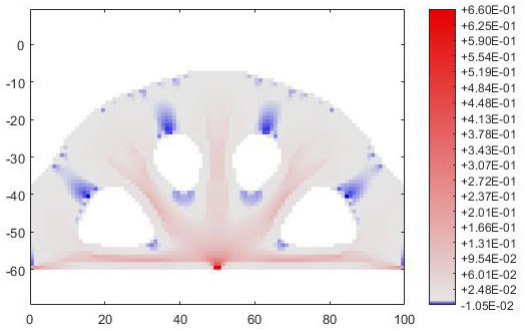
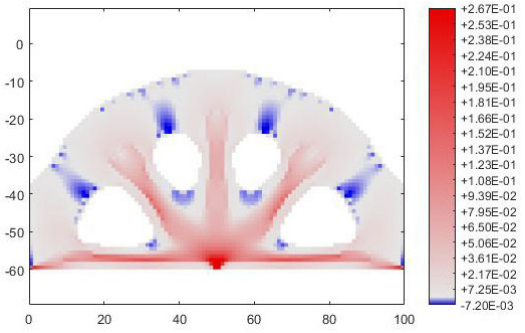


Figure 4.50: Case study n.3 - problem statement

Figure 4.51: Case study n.3 - solution by the code of this thesis

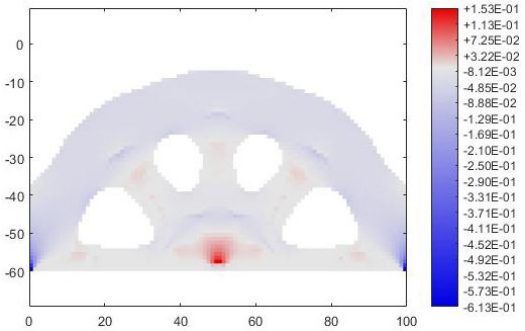


(a) full range of stresses

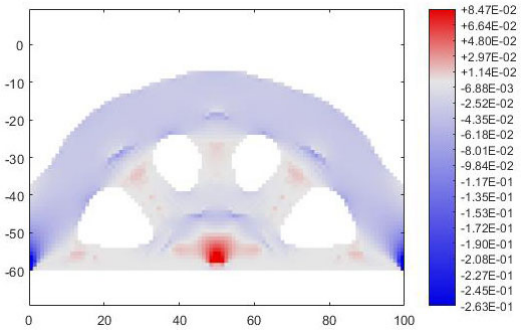


(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.52: Case study n.3 - plots of principal stress sigma 1



(a) full range of stresses



(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.53: Case study n.3 - plots of principal stress sigma 2

Case study n.3 - results interpretation

The case study n.3 corresponds to a simply supported beam with load in the center of bottom edge. In both solutions, there is one long horizontal steel member for "bending" and three vertical and oblique members for "shear". In the case the structure geometry is kept unmodified rectangular, the latter members are shorter. By trial and error on the ESO procedure, if the target volume is decreased, the only result is the shortening of such members until their complete disappearance. In both solutions, the inclination of the oblique steel members is measured to be around 49°.

The interpretation of the solution is the following:

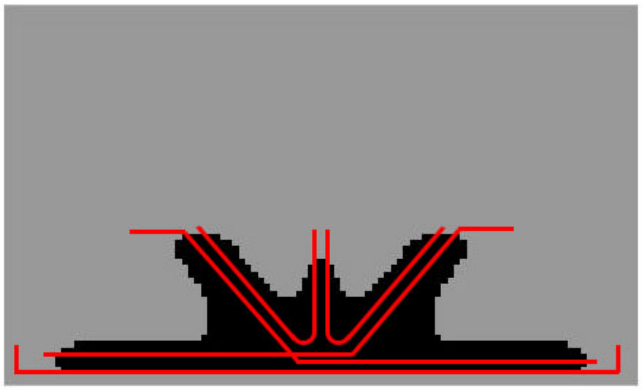


Figure 4.54: Case study n.3 - example interpretation of solution - original structure boundaries

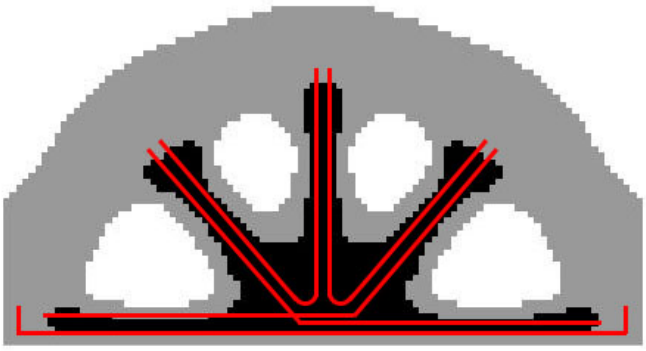


Figure 4.55: Case study n.3 - example interpretation of solution - optimized structure boundaries

In both solutions, from the analysis of principal stresses plots, the only remaining tensile zones in concrete are on the supports and right on the point of application of the load, in the center of bottom edge.

4.3.4 Case study n.4

Case study n.4 - original geometry, no reinforcement

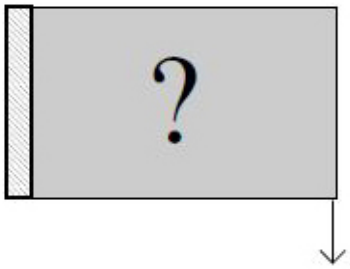
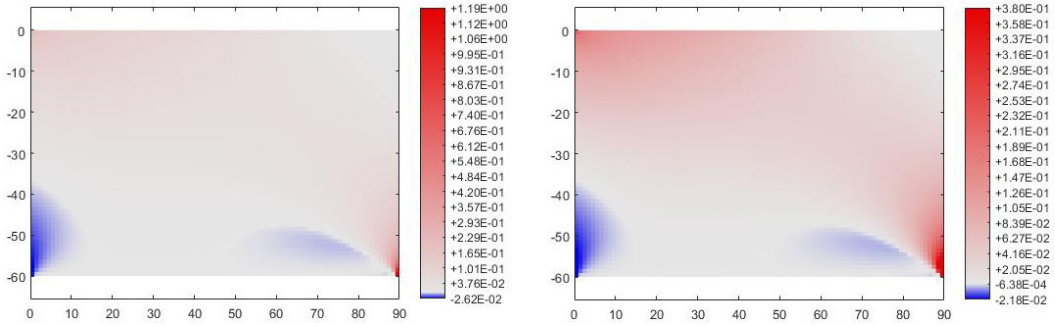


Figure 4.56: Case study n.4 - problem statement

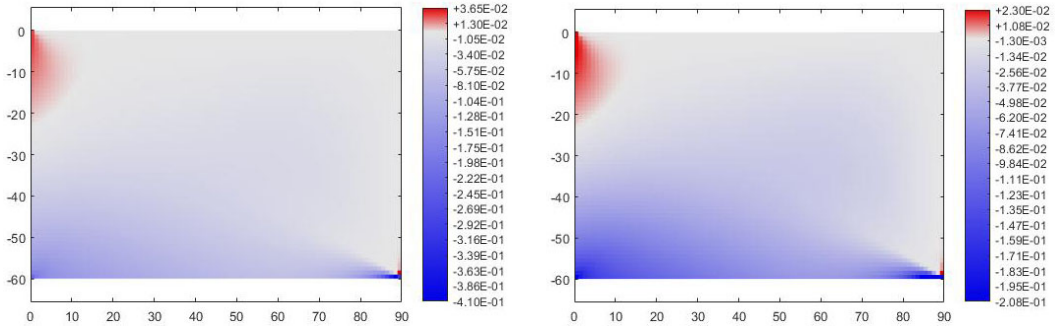
Figure 4.57: Case study n.4 - geometry without reinforcement and optimization



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.58: Case study n.4 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.59: Case study n.4 - plots of principal stress sigma 2

Case study n.4 - original geometry with optimized reinforcement

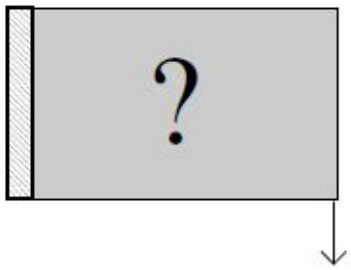
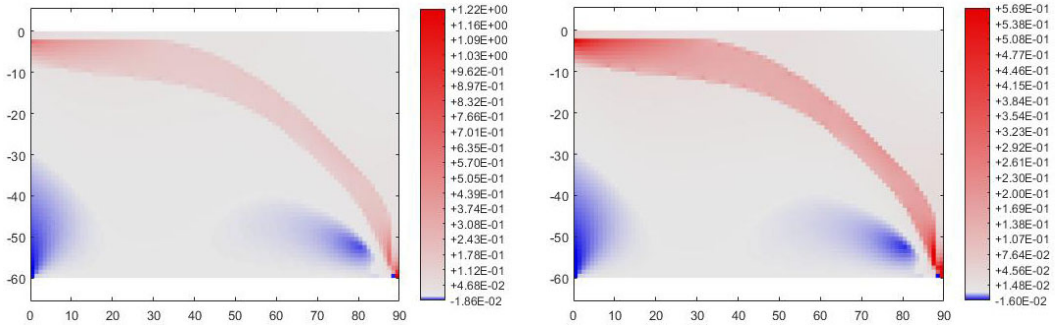


Figure 4.60: Case study n.4 - problem statement

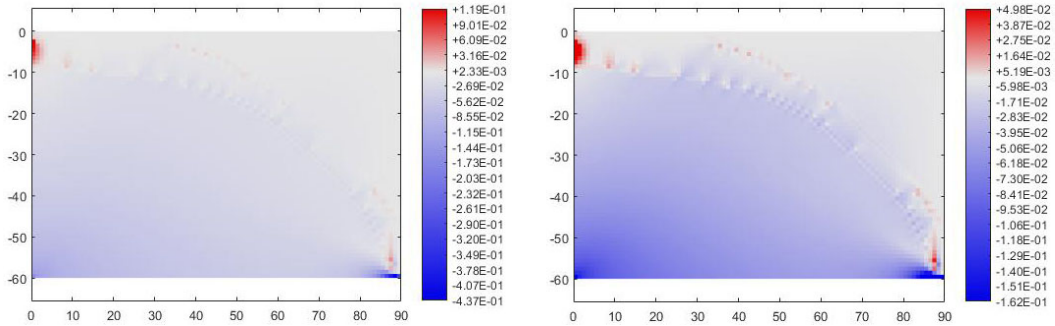
Figure 4.61: Case study n.4 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.62: Case study n.4 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.63: Case study n.4 - plots of principal stress sigma 2

Case study n.4 - optimized geometry and reinforcement

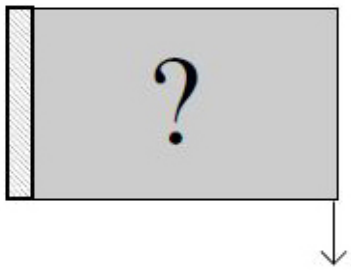
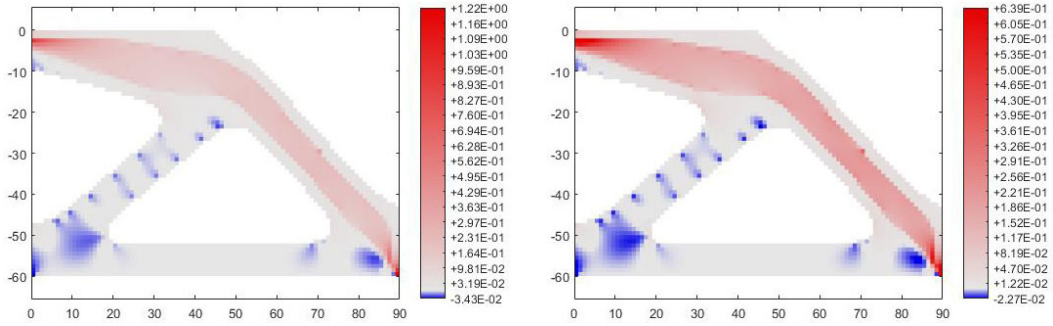


Figure 4.64: Case study n.4 - problem statement

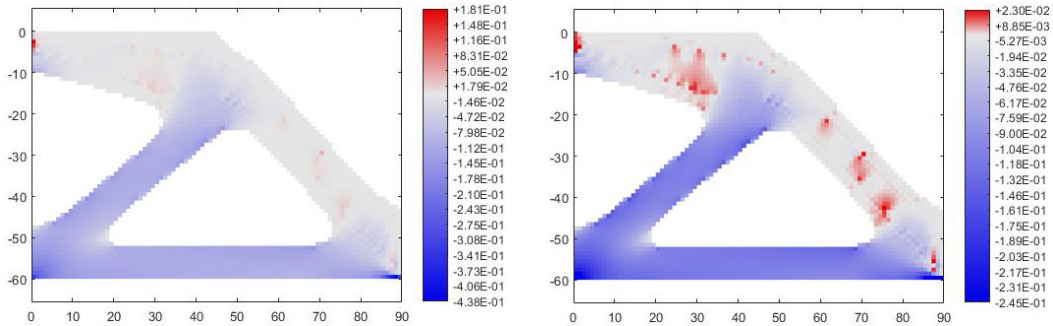
Figure 4.65: Case study n.4 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.66: Case study n.4 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.67: Case study n.4 - plots of principal stress sigma 2

Case study n.4 - results interpretation

The case study n.4 corresponds to a short cantilever with load in the bottom furthest point from the support. In case the structure is kept rectangular, the steel members derived by the ESO procedure assume a curved shape, starting from the top-left corner (on the support) to the bottom-right corner (on the load). In the example interpretation, the resulting steel layout for this case is represented by a four segments polyline, with angles of around 27° and 47°. In case the structure geometry is optimized, the steel members are comprised by a horizontal member close to the support, and two oblique members respectively with inclinations of 13° and 46°.

The interpretation of the solution is the following:

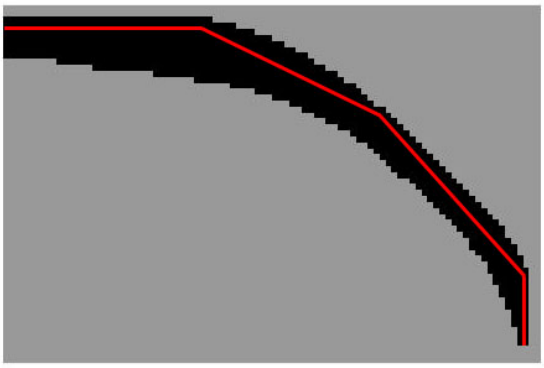


Figure 4.68: Case study n.4 - example interpretation of solution - original structure boundaries

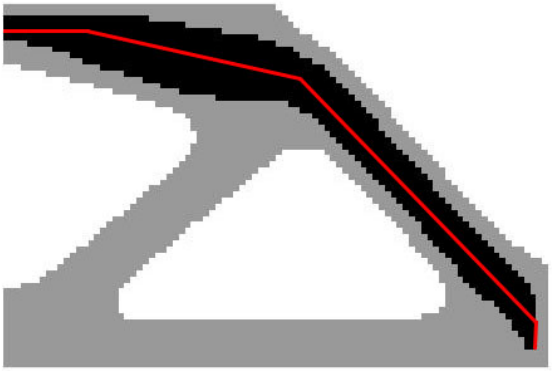


Figure 4.69: Case study n.4 - example interpretation of solution - optimized structure boundaries

In both cases, from the analysis of principal stresses plots, no remaining tensile zones appear in concrete, other than the few elements of concrete right on the load application point.

4.3.5 Case study n.5

Case study n.5 - original geometry, no reinforcement

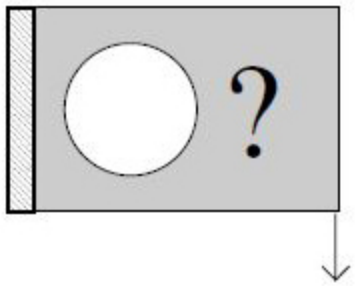
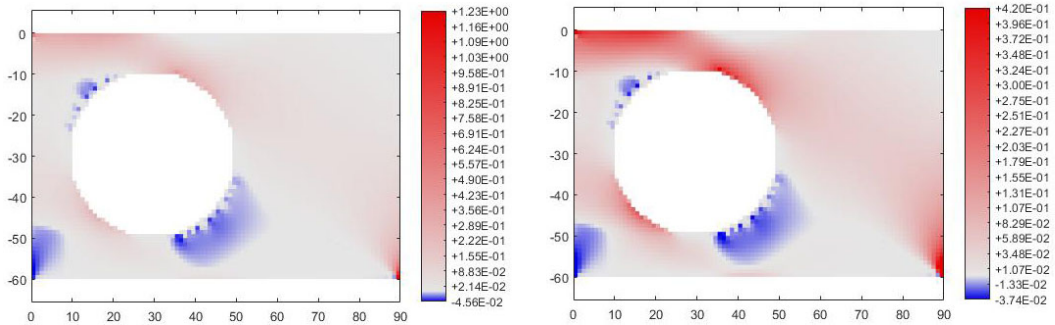


Figure 4.70: Case study n.5 - problem statement

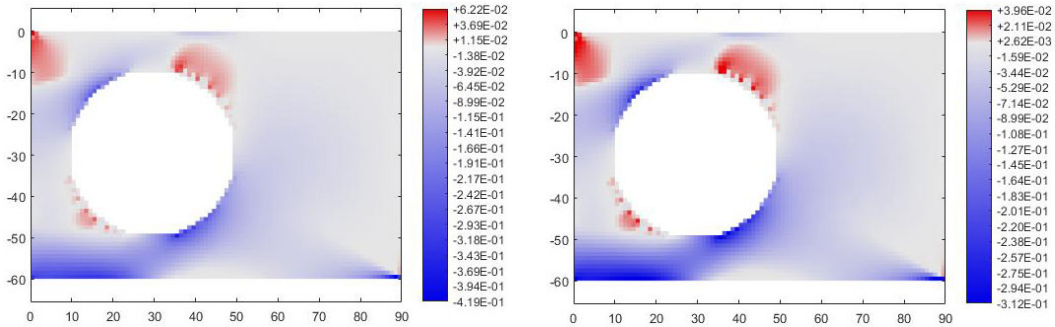
Figure 4.71: Case study n.5 - geometry without reinforcement and optimization



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.72: Case study n.5 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.73: Case study n.5 - plots of principal stress sigma 2

Case study n.5 - original geometry with optimized reinforcement

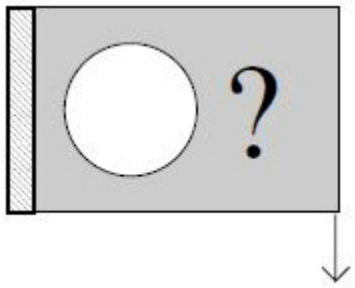
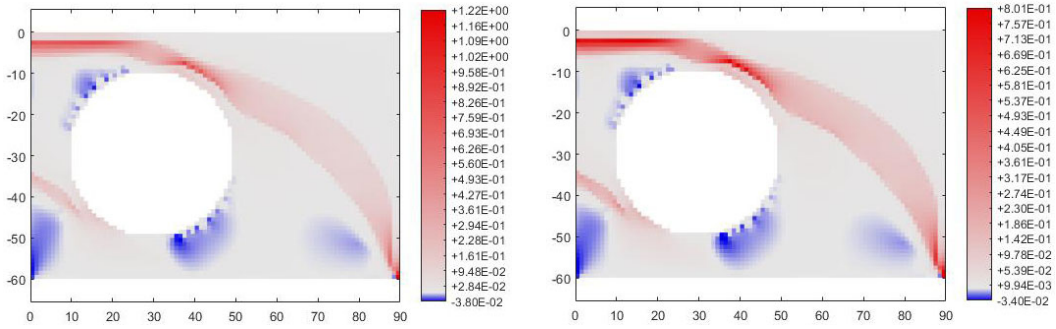


Figure 4.74: Case study n.5 - problem statement

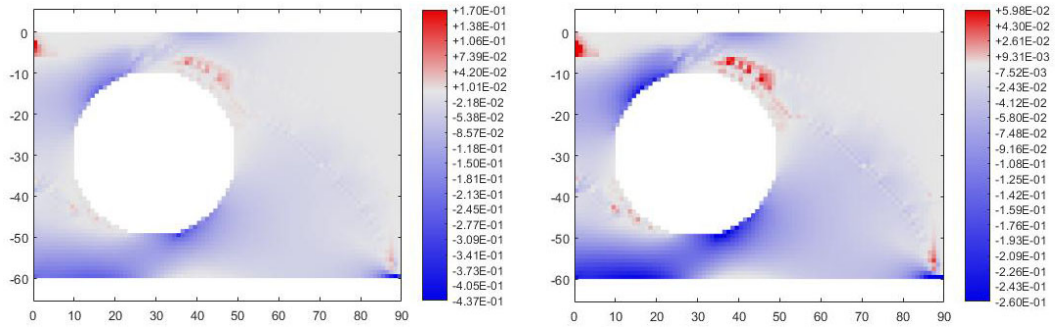
Figure 4.75: Case study n.5 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.76: Case study n.5 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.77: Case study n.5 - plots of principal stress sigma 2

Case study n.5 - optimized geometry and reinforcement

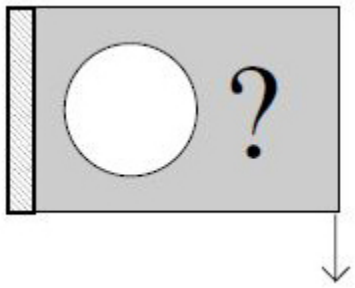
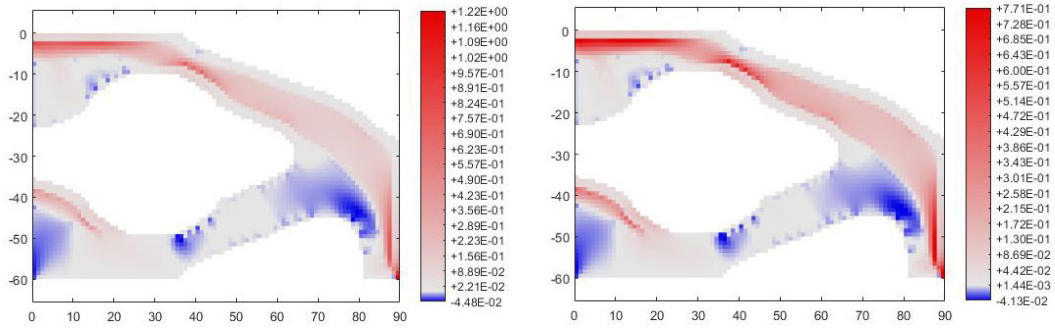


Figure 4.78: Case study n.5 - problem statement

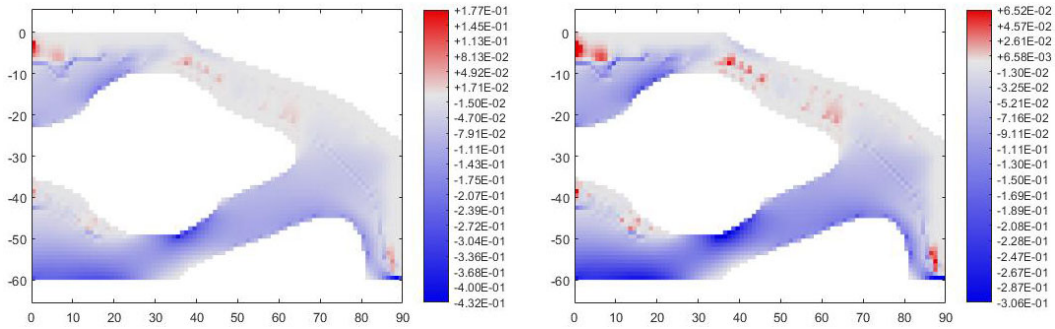
Figure 4.79: Case study n.5 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.80: Case study n.5 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.81: Case study n.5 - plots of principal stress sigma 2

Case study n.5 - results interpretation

The case study n.5 corresponds to a short cantilever with load in the bottom furthest point from the support and with a big circular hole close to the support. The overall geometry and boundary conditions are the same as for case study n.4, except for the hole. In case the structure is kept rectangular, the main steel member starts with a straight shape close to the support followed by a curved shape connecting it to the load. In the example interpretation, the chosen resulting shape is a four segments polyline, with angles of around 20° and 47° for the inner segments. Under the bottom-left corner of the hole a steel member appears to counter the tensile stresses generated by the presence of the hole. The resulting layout is interpreted by a straight line and an oblique line inclined with an angle of around 45°. Such line has to be extended to absorb remaining tensile stresses present in the concrete zones around and below it.

In case the structure geometry is optimized, a total of three resulting steel members can be identified. The principal steel member, can be interpreted as a three-segment polyline with inclination of the central segment of 27°. The second steel member appears under the bottom-left part of the hole, as in the case with original boundaries, but with different angles, assuming an inclination of around 36°. A third steel member appears in the top-left corner, just under the principal one, to take on tensile stresses generated in that area due to the structure's optimized geometry around the imposed circular hole and below it. A resulting inclination of 46° can be interpreted for the third steel member.

The interpretation of the solution is the following:

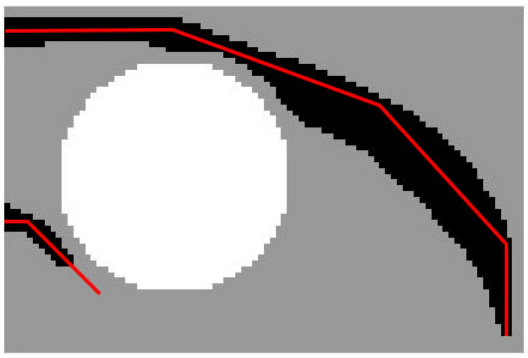


Figure 4.82: Case study n.5 - example interpretation of solution - original structure boundaries

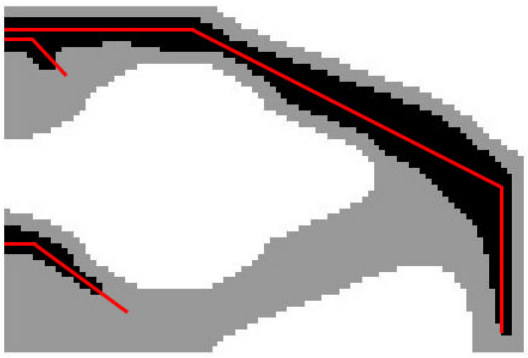


Figure 4.83: Case study n.5 - example interpretation of solution - optimized structure boundaries

In both cases, from the analysis of principal stresses plots, a remaining tensile zone in concrete appears under the bottom-left corner of the hole and refers to the stress concentrations generated by the presence of the imposed hole. The interpreted steel member in that area is elongated in both cases to take on the remaining tensile stresses.

4.3.6 Case study n.6

Case study n.6 - original geometry, no reinforcement

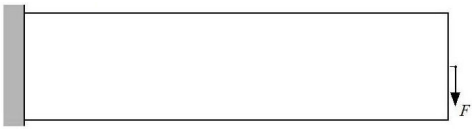
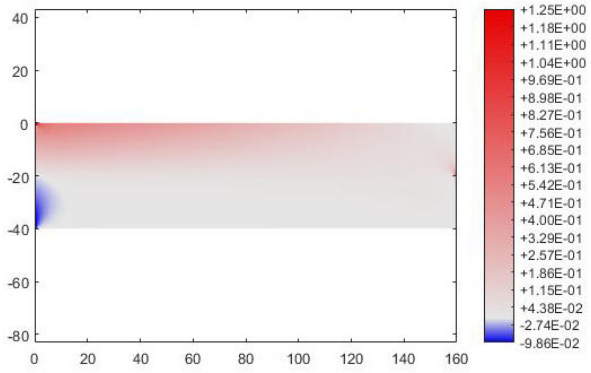
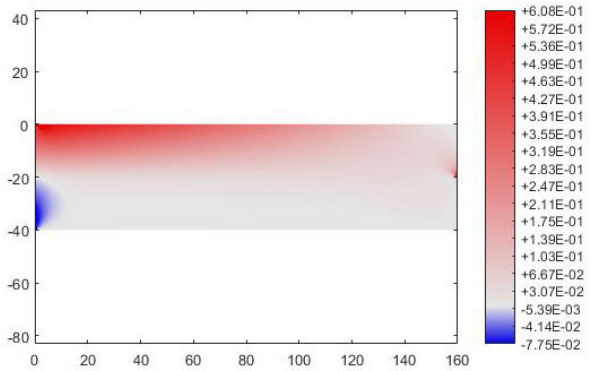


Figure 4.84: Case study n.6 - problem statement

Figure 4.85: Case study n.6 - geometry without reinforcement and optimization

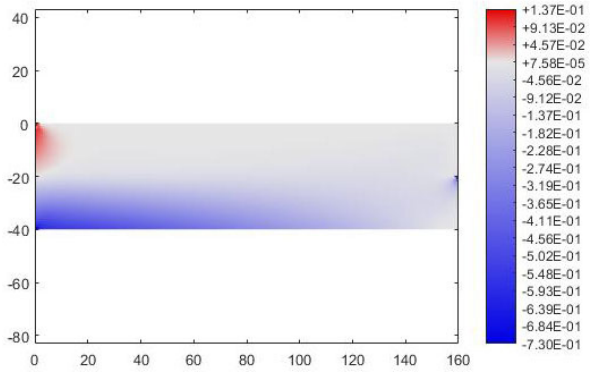


(a) full range of stresses

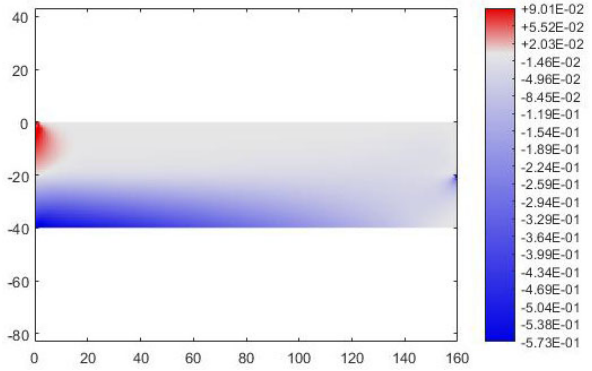


(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.86: Case study n.6 - plots of principal stress sigma 1



(a) full range of stresses



(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.87: Case study n.6 - plots of principal stress sigma 2

Case study n.6 - original geometry with optimized reinforcement

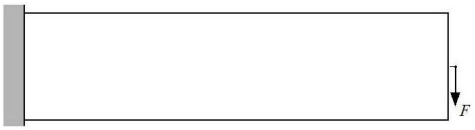
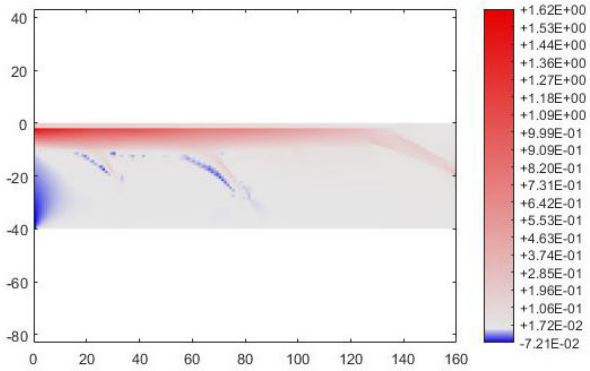
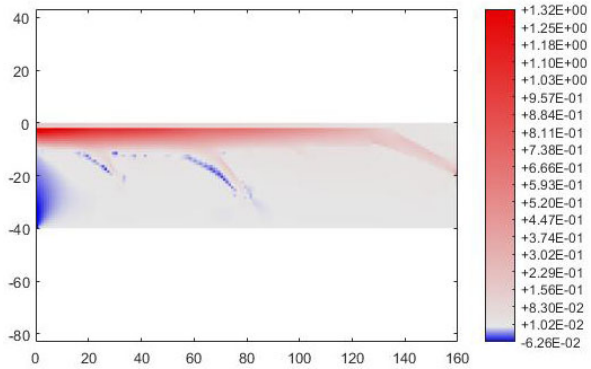


Figure 4.88: Case study n.6 - problem statement

Figure 4.89: Case study n.6 - solution by the code of this thesis

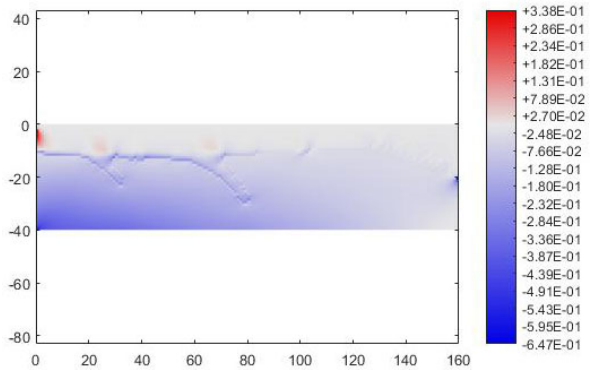


(a) full range of stresses

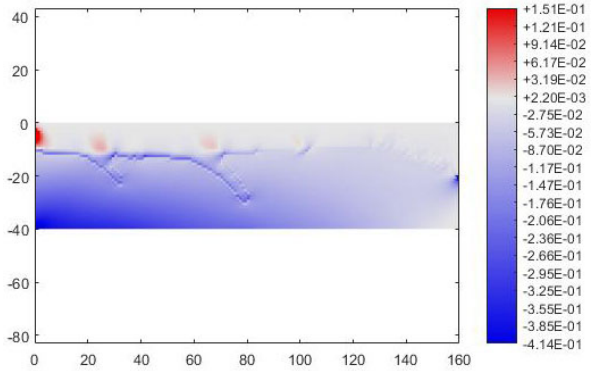


(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.90: Case study n.6 - plots of principal stress sigma 1



(a) full range of stresses



(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.91: Case study n.6 - plots of principal stress sigma 2

Case study n.6 - optimized geometry and reinforcement

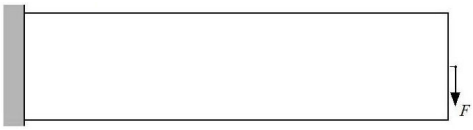
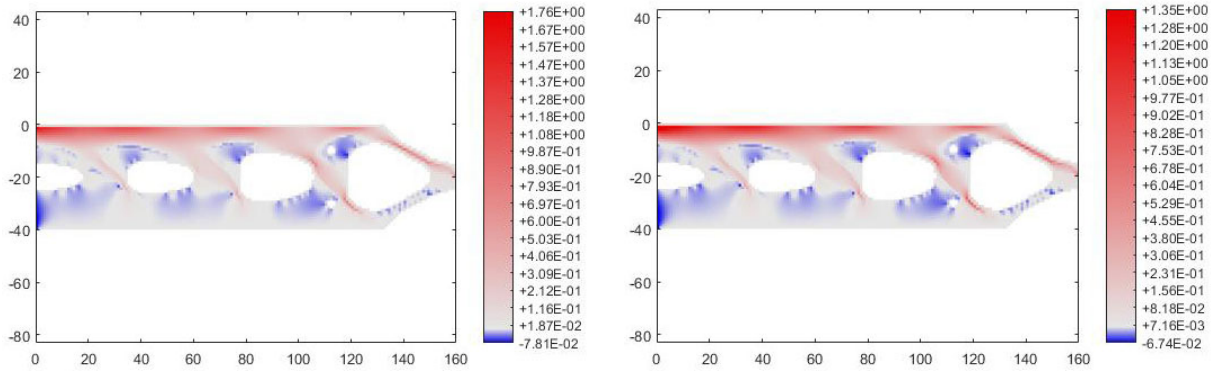


Figure 4.92: Case study n.6 - problem statement

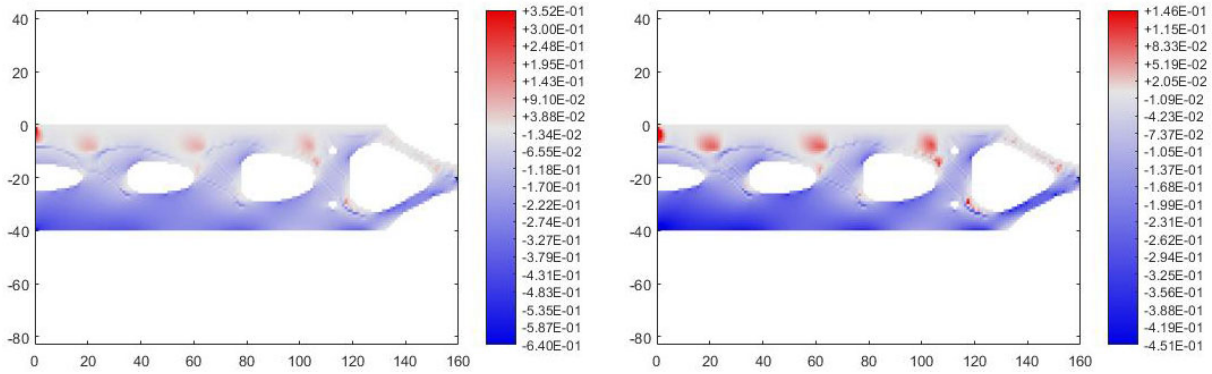
Figure 4.93: Case study n.6 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.94: Case study n.6 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.95: Case study n.6 - plots of principal stress sigma 2

Case study n.6 - results interpretation

The case study n.6 corresponds a long cantilever with lenght = 4 x width, twice the lenght of case study n.2, with centered force on the right edge. In the solution with the original rectangular boundaries, a horizontal steel member is found on almost the whole lenght of the beam in the top edge to counter the tensile stresses due to the bending in the beam. The main horizontal steel member changes direction in its end to connect with the load application point with an inclined member of around 28°. Two additional oblique members are found close to midspan with inclination of around 47°.

In the solution with the optimized structure boundaries, the main horizontal steel member has an oblique end with angle approximately equal to 30°. The "shear" member in that case are three, instead of two, and have angles respectively equal to 45°, 45° and 53° (from left to right). The third shear member, the one closer to the end of the beam, is present in the case of optimized structure, while is almost absent in case of original rectangular boundaries, as it gets removed in the ESO procedure quite early due to its relatively low tensile stress.

The interpretation of the solution is the following:

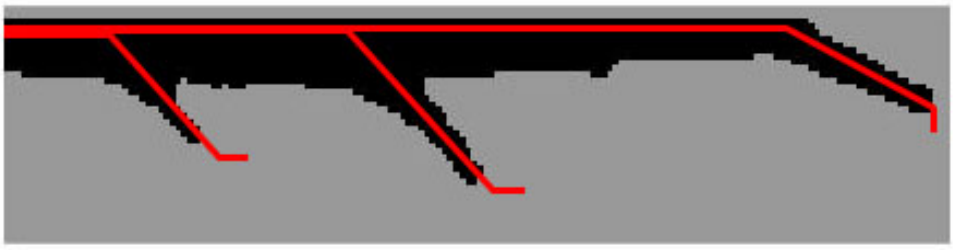


Figure 4.96: Case study n.6 - example interpretation of solution - original structure boundaries

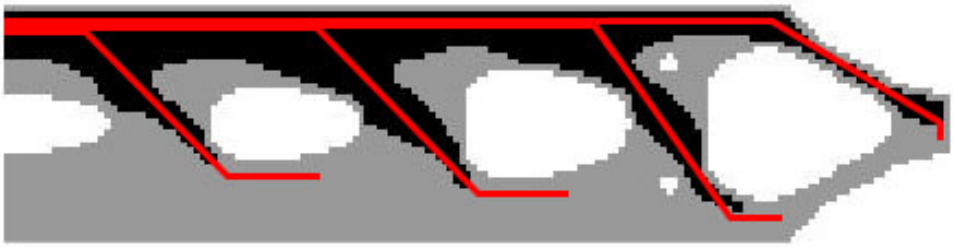


Figure 4.97: Case study n.6 - example interpretation of solution - optimized structure boundaries

From analysis of principal stress plots, no remaining tensile zones are found in the concrete for both solutions. The zones neighboring the bottom-left corners of the holes are characterized by the presence of tensile stress concentrations, which are addressed, in the example interpretation above, elongating the "shear" members by adding horizontal segments to their ends. The latter horizontal members are also extended, outside above mentioned tensile zones, to meet the compressive concrete struts coming from the other side of the holes.

4.3.7 Case study n.7

Case study n.7 - original geometry, no reinforcement

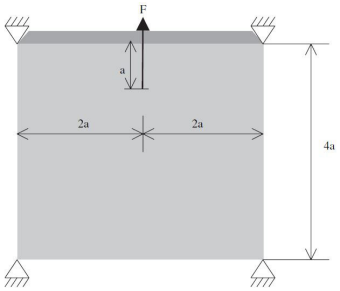


Figure 4.98: Case study n.7 - problem statement

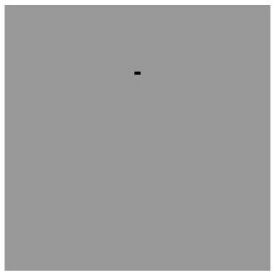
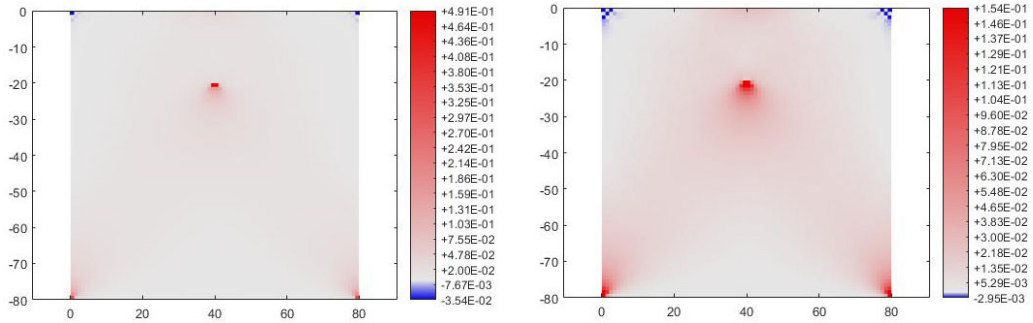


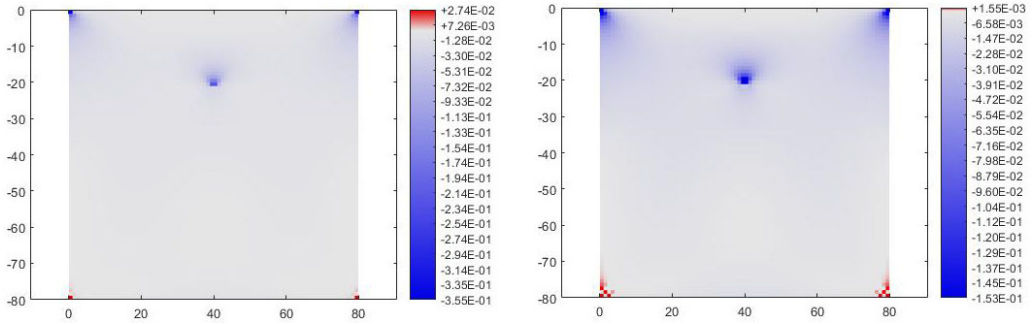
Figure 4.99: Case study n.7 - geometry without reinforcement and optimization



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.100: Case study n.7 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.101: Case study n.7 - plots of principal stress sigma 2

Case study n.7 - original geometry with optimized reinforcement

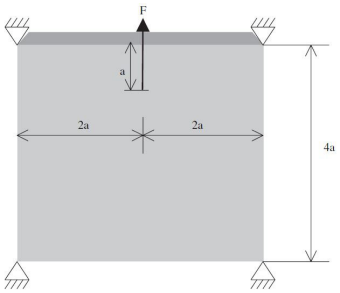
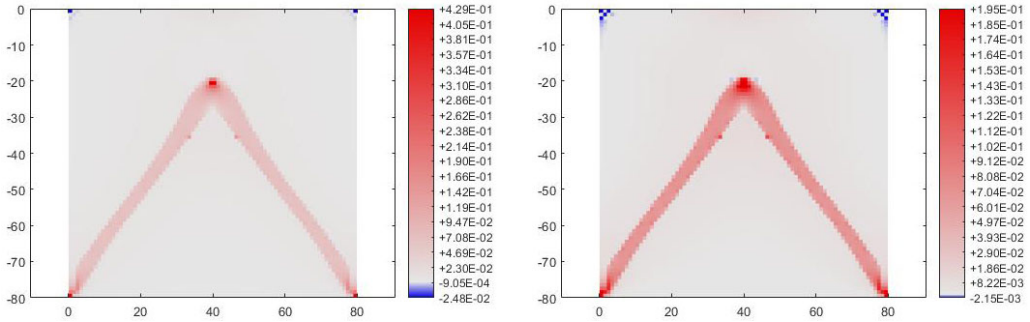


Figure 4.102: Case study n.7 - problem statement

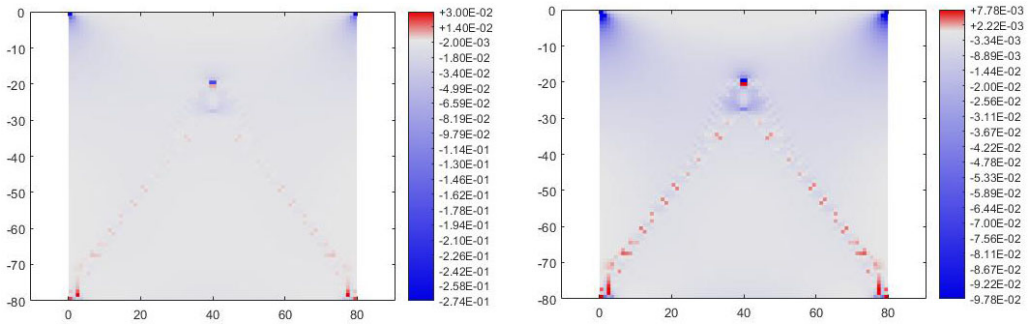
Figure 4.103: Case study n.7 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.104: Case study n.7 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.105: Case study n.7 - plots of principal stress sigma 2

Case study n.7 - optimized geometry and reinforcement

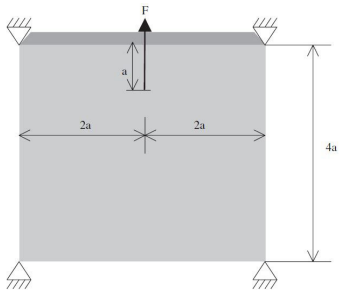
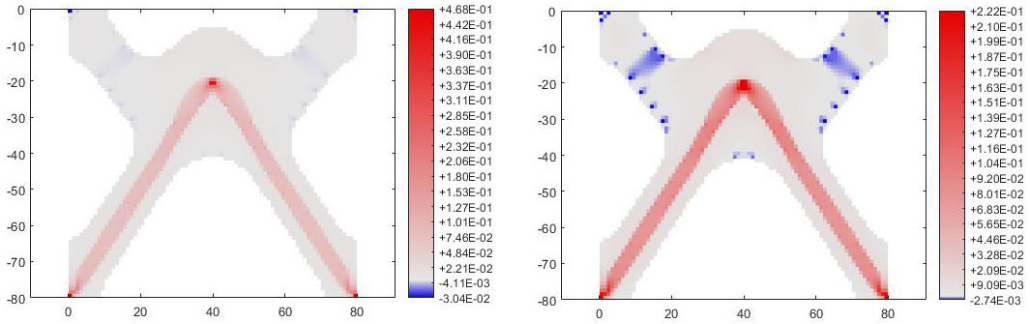


Figure 4.106: Case study n.7 - problem statement

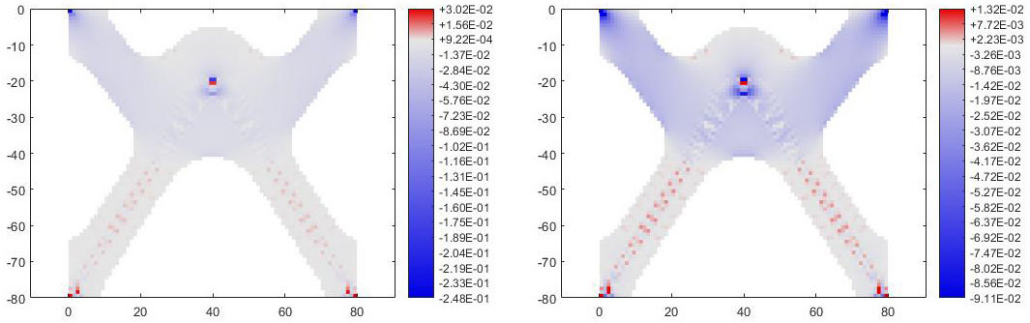
Figure 4.107: Case study n.7 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.108: Case study n.7 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.109: Case study n.7 - plots of principal stress sigma 2

Case study n.7 - results interpretation

The case study n.7 corresponds to a square deep beam loaded in the center of its upper half. In all solutions two inclined tensile steel members can be identified connecting the load application point to the two lower supports. In case the structure is kept rectangular, the angle between the two steel members is around 72°, while in the case the structure boundaries are optimized such angle results approximately equal to 65°. The difference in inclination between the steel members is to be mainly attributed to the form variation of upper half of the deep beam (optimized or original), which contributes to the load bearing capacity through compression.

The interpretation of the solution is the following:

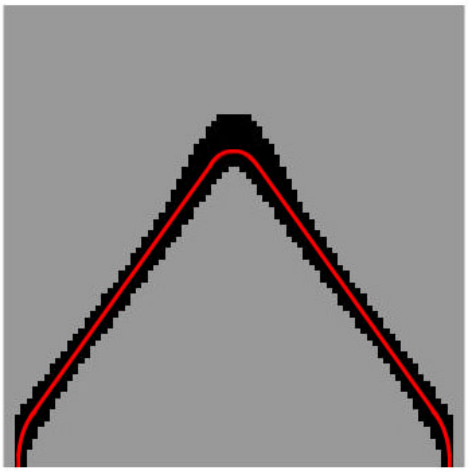


Figure 4.110: Case study n.7 - example interpretation of solution - original structure boundaries

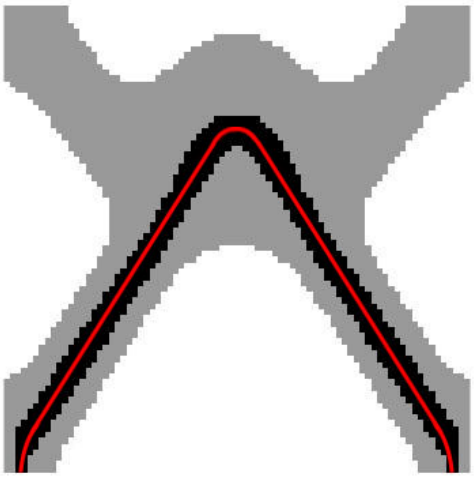


Figure 4.111: Case study n.7 - example interpretation of solution - optimized structure boundaries

From the analysis of principal stresses plots, no remaining tensile zones appear in concrete for both cases.

4.3.8 Case study n.8

Case study n.8 - original geometry, no reinforcement

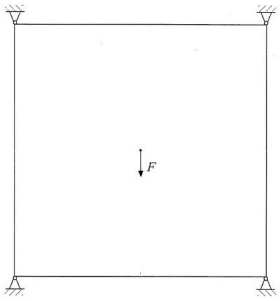


Figure 4.112: Case study n.8 - problem statement

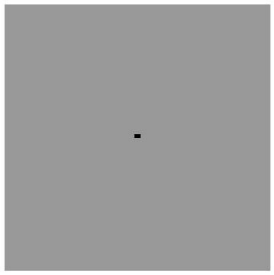
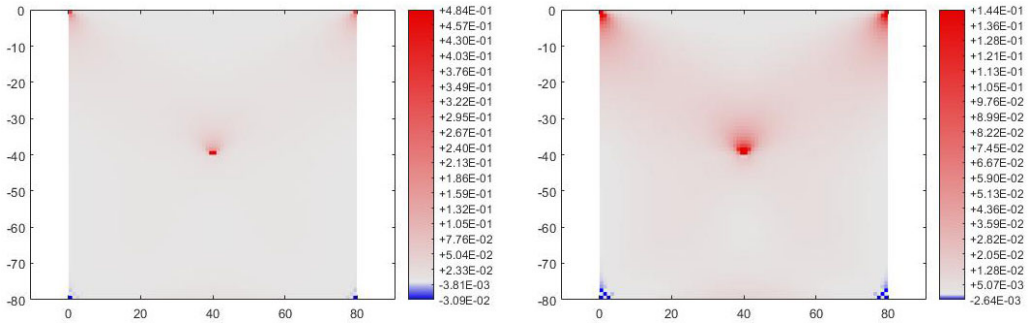


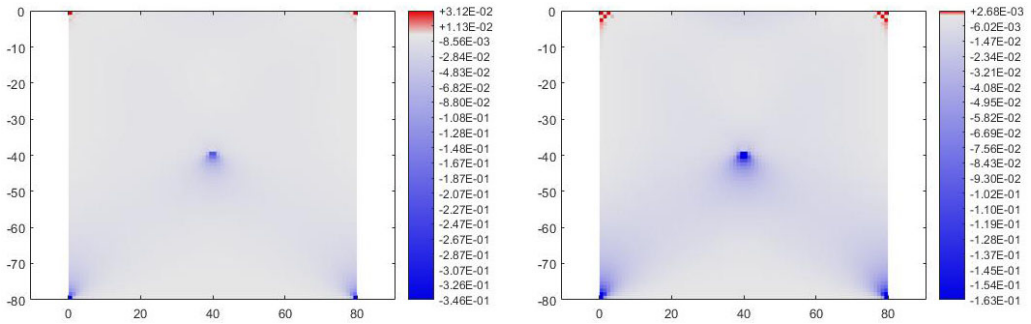
Figure 4.113: Case study n.8 - geometry without reinforcement and optimization



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.114: Case study n.8 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.115: Case study n.8 - plots of principal stress sigma 2

Case study n.8 - original geometry with optimized reinforcement

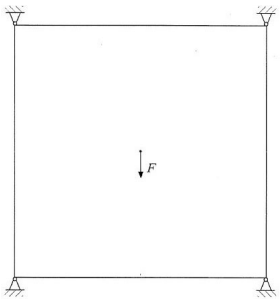
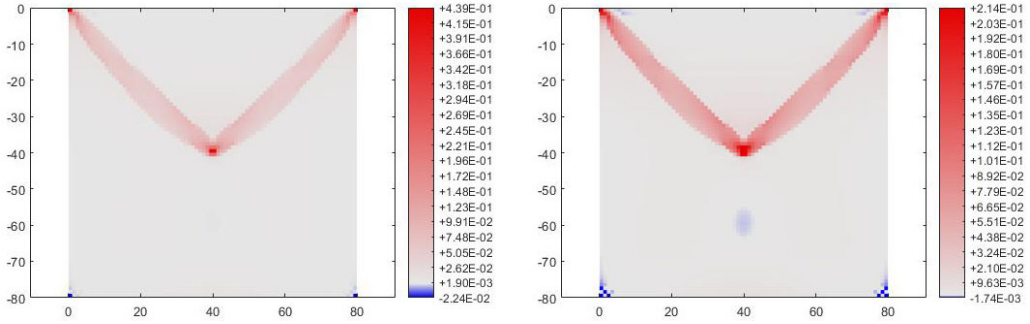


Figure 4.116: Case study n.8 - problem statement

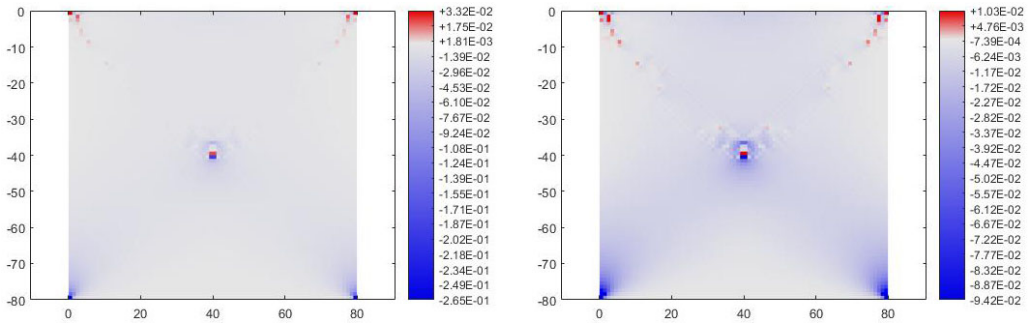
Figure 4.117: Case study n.8 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.118: Case study n.8 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.119: Case study n.8 - plots of principal stress sigma 2

Case study n.8 - optimized geometry and reinforcement

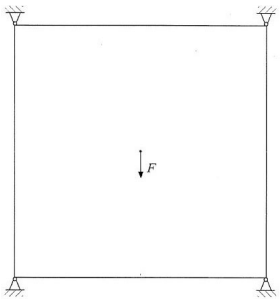
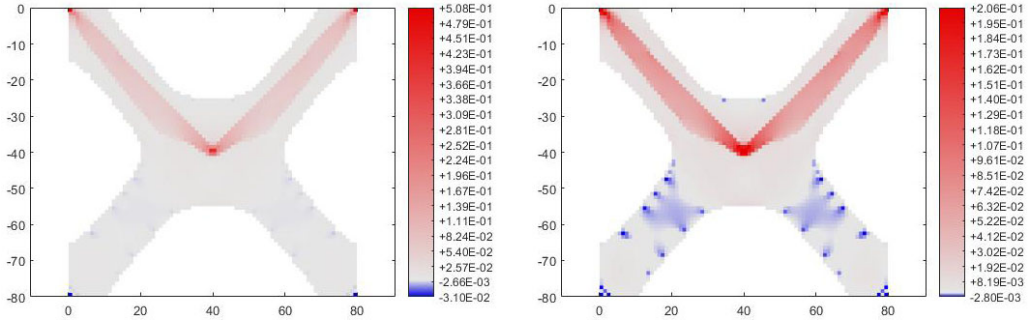


Figure 4.120: Case study n.8 - problem statement

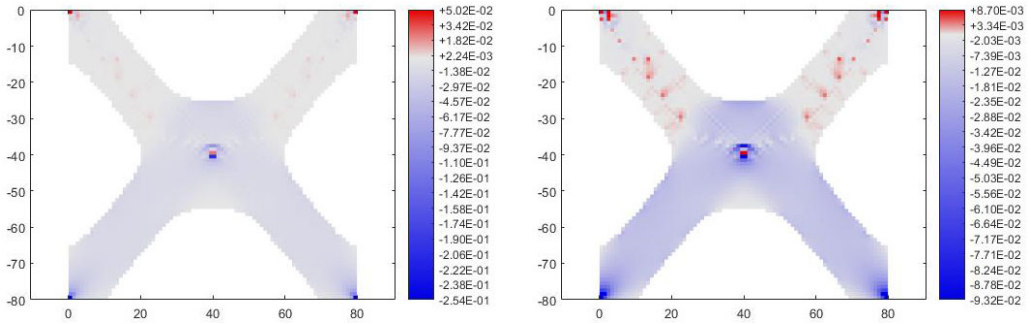
Figure 4.121: Case study n.8 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.122: Case study n.8 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.123: Case study n.8 - plots of principal stress sigma 2

Case study n.8 - results interpretation

The case study n.8 corresponds to a square deep beam loaded in its center. In all solutions two inclined tensile steel members can be identified connecting the load application point to the two upper supports. In case the structure is kept rectangular, the angle between the two steel members is around 92°, while in the case the structure boundaries are optimized such angle results approximately equal to 88°. The difference in inclination between the steel members is to be mainly attributed to the variation in overall structure geometry (optimized or original), which contributes to the load bearing capacity also by compression in its lower part.

The interpretation of the solution is the following:

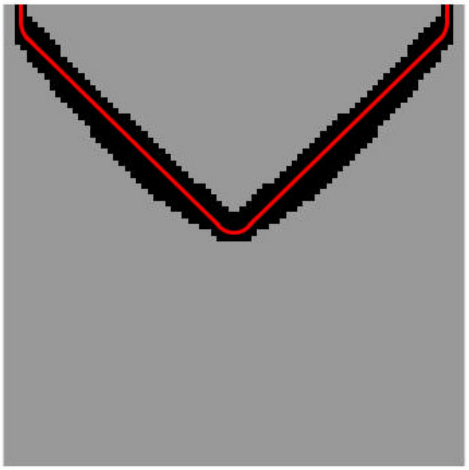


Figure 4.124: Case study n.8 - example interpretation of solution - original structure boundaries

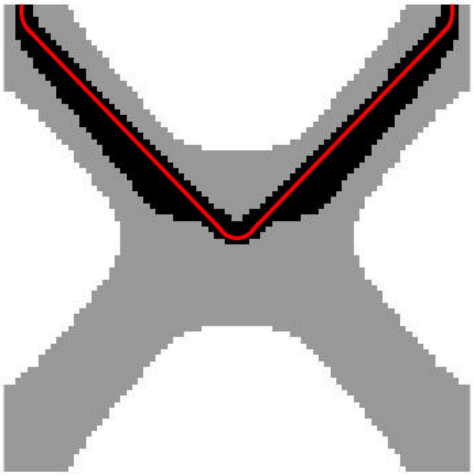


Figure 4.125: Case study n.8 - example interpretation of solution - optimized structure boundaries

From the analysis of principal stresses plots, no remaining tensile zones appear in concrete for both cases.

4.3.9 Case study n.9

Case study n.9 - original geometry, no reinforcement

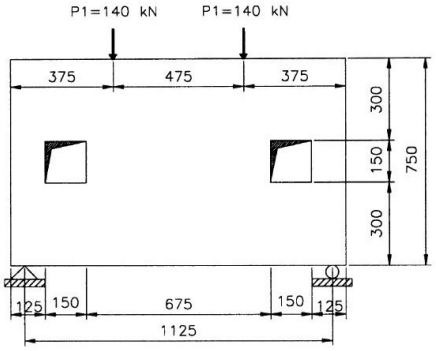


Figure 4.126: Case study n.9 - problem statement

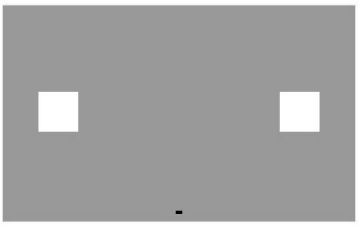
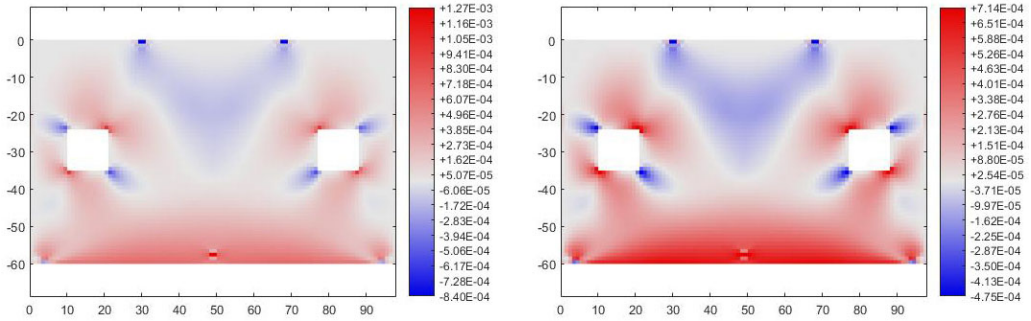


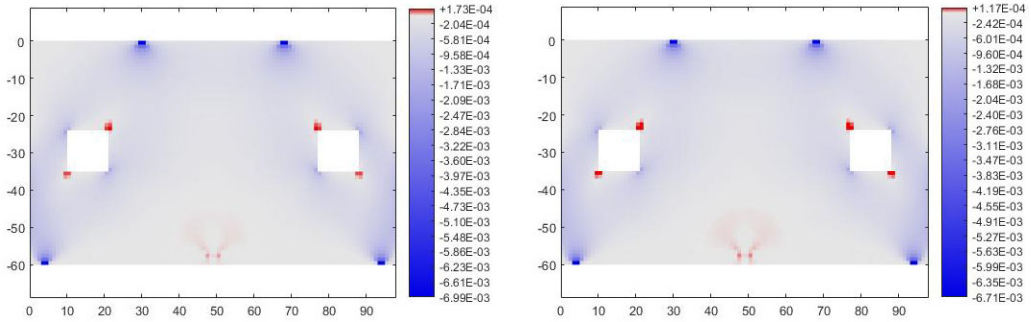
Figure 4.127: Case study n.9 - geometry without reinforcement and optimization



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.128: Case study n.9 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.129: Case study n.9 - plots of principal stress sigma 2

Case study n.9 - original geometry with optimized reinforcement

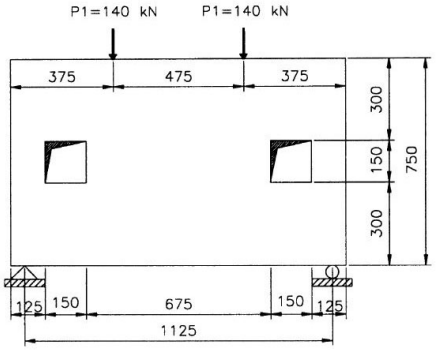
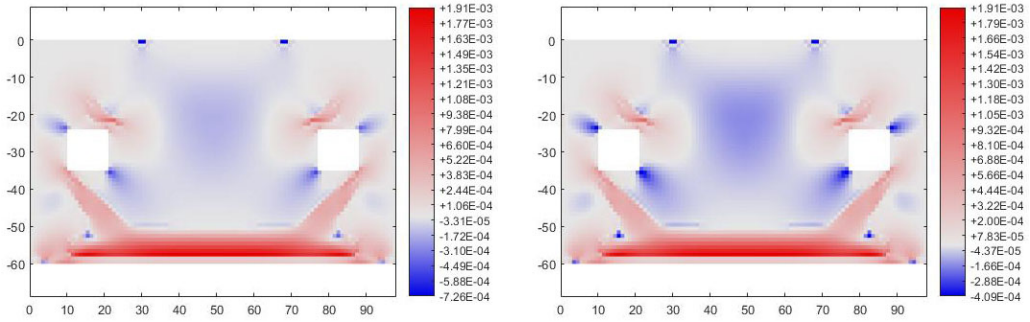


Figure 4.130: Case study n.9 - problem statement

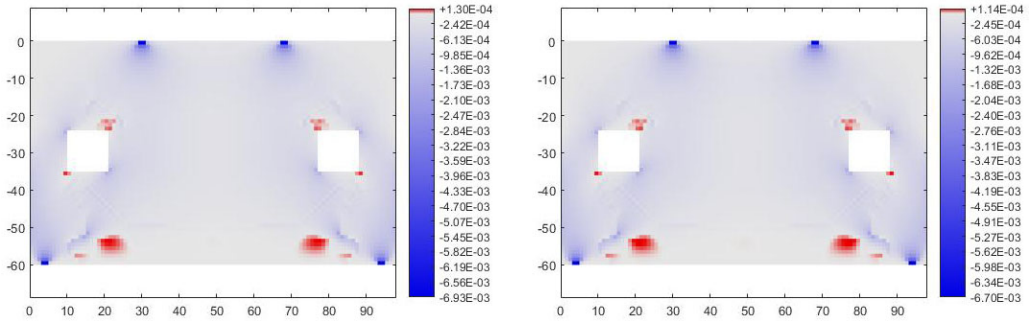
Figure 4.131: Case study n.9 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.132: Case study n.9 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.133: Case study n.9 - plots of principal stress sigma 2

Case study n.9 - optimized geometry and reinforcement

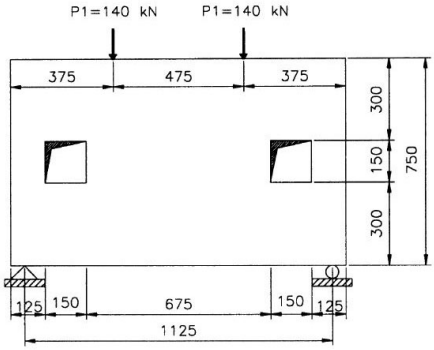


Figure 4.134: Case study n.9 - problem statement

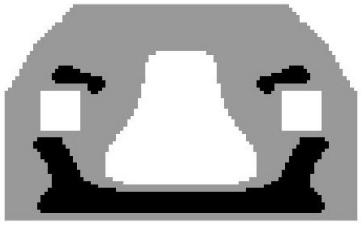
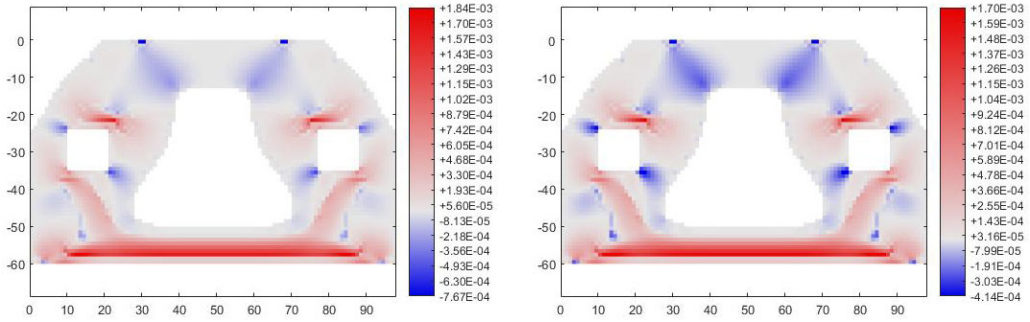


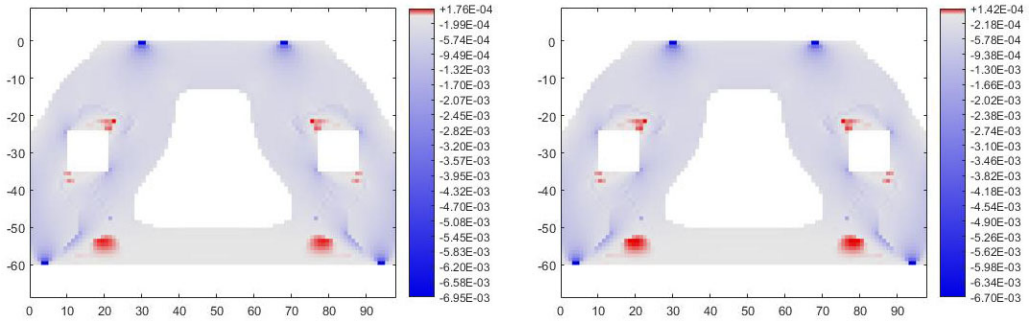
Figure 4.135: Case study n.9 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.136: Case study n.9 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.137: Case study n.9 - plots of principal stress sigma 2

Case study n.9 - results interpretation

The case study n.9 corresponds to a simply supported deep beam with two symmetric square openings. Resulting steel layouts are similar in both solutions (optimized and original boundaries). However, angles differ between the two cases. Three types of steel members can be interpreted from both solutions. A main horizontal steel member is present in the bottom edge of the beam, to take on the bending. Right above it, two symmetrical inclined members rise from the bottom of the beam to the bottom of the square openings, to counter shear and stress concentrations across the openings. The third type of steel members appears on top of the square openings on the side closer to the center of the beam, to take on the tensile stress concentrations on top of the square holes.

In the solution with the original rectangular boundaries, the inclination of steel members under the holes is found to be approximately 51°. The inclination of the members on top of the holes is of circa 19°.

In the solution with the optimized structure boundaries, the inclination of steel members under the holes is measured to be around 60°. The members on top of the holes are inclined by around 13°.

The interpretation of the solution is the following:

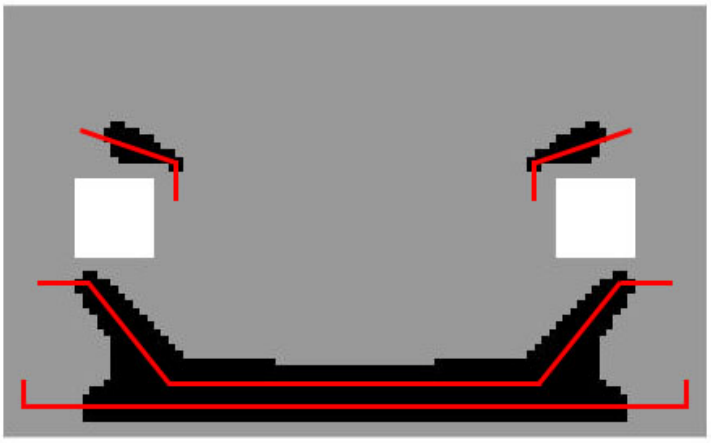


Figure 4.138: Case study n.9 - example interpretation of solution - original structure boundaries

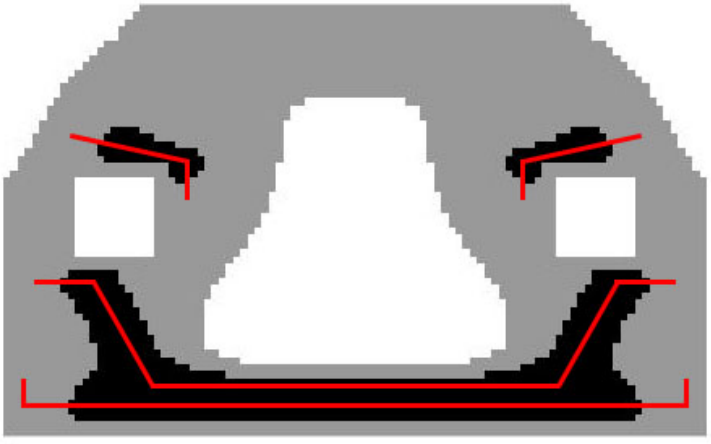


Figure 4.139: Case study n.9 - example interpretation of solution - optimized structure boundaries

In both cases, from analysis of principal stress plots, some tensile zones remain in the concrete right on top of the supports and under the external lower corner of the square openings. In that zones, additional steel members are drawn manually in the example interpretation of the solutions.

4.3.10 Case study n.10

Case study n.10 - original geometry, no reinforcement

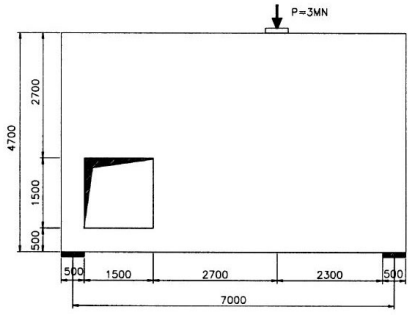
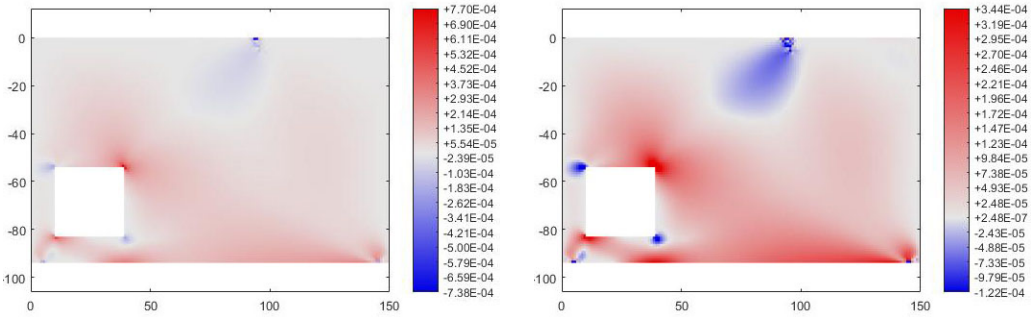


Figure 4.140: Case study n.10 - problem statement

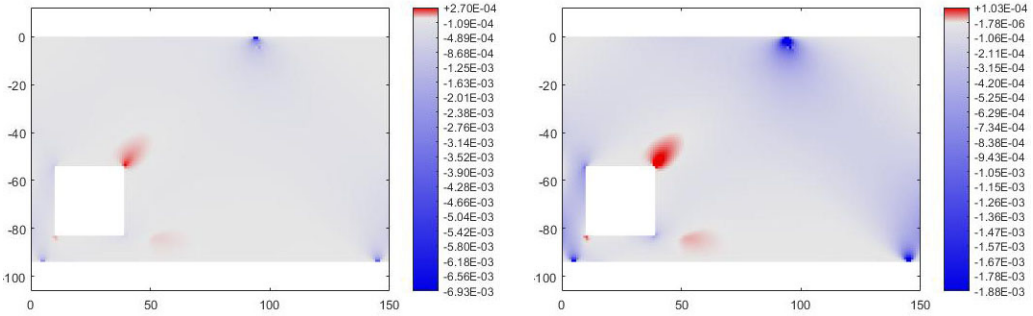
Figure 4.141: Case study n.10 - geometry without reinforcement and optimization



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.142: Case study n.10 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.143: Case study n.10 - plots of principal stress sigma 2

Case study n.10 - original geometry with optimized reinforcement

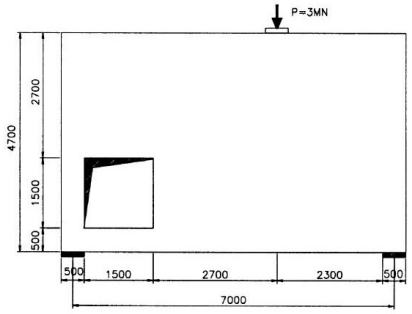
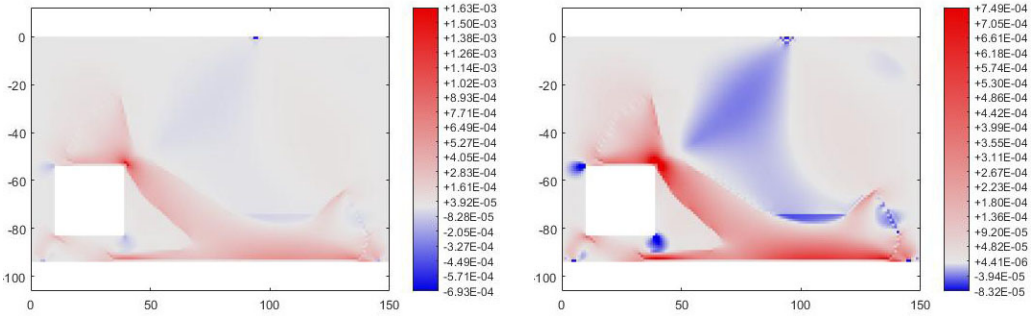


Figure 4.144: Case study n.10 - problem statement

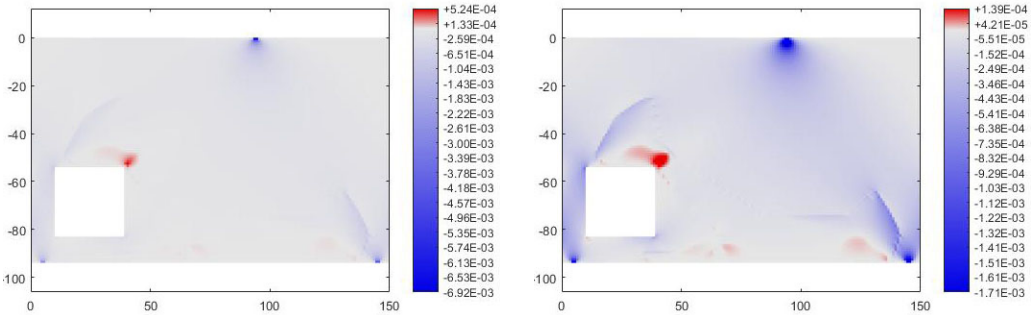
Figure 4.145: Case study n.10 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.146: Case study n.10 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.147: Case study n.10 - plots of principal stress sigma 2

Case study n.10 - optimized geometry and reinforcement

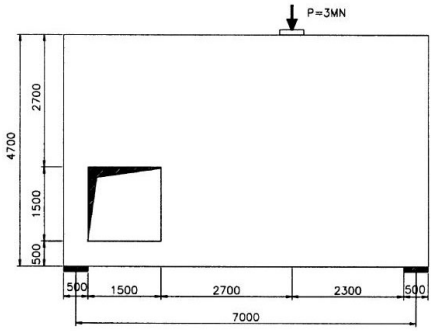


Figure 4.148: Case study n.10 - problem statement

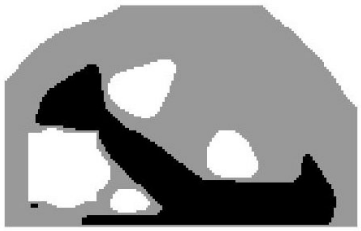
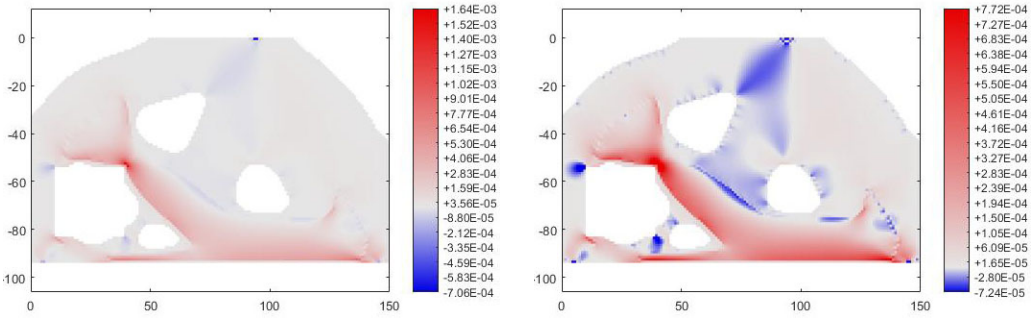


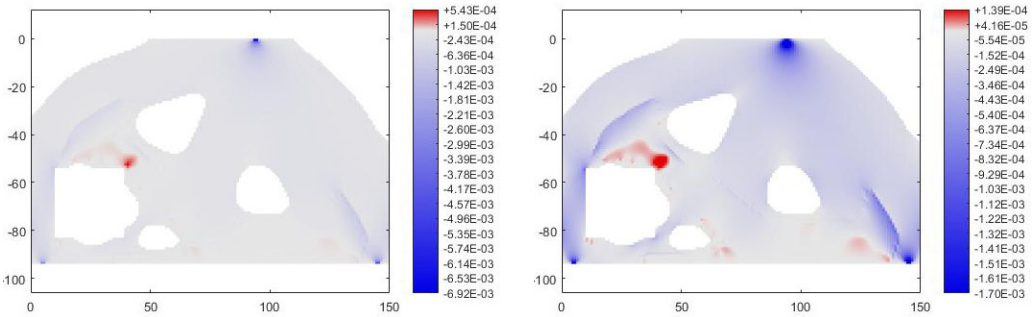
Figure 4.149: Case study n.10 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.150: Case study n.10 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.151: Case study n.10 - plots of principal stress sigma 2

Case study n.10 - results interpretation

The case study n.10 corresponds to a simply supported deep beam with asymmetric square opening and loading. Through the combined interpretation of steel zones from ESO procedure and principal stress plots, three main steel members are found. One horizontal member is found on the bottom edge of the beam. Most probably, it is due to the small thickness of the structure under the square hole that the steel member on the left support gets eliminated during the ESO procedure (approximately when steel volume trespasses the value of 25%). The interpreted steel member is elongated manually into that zone. The second steel member starts horizontally from the top of the square hole, gets inclined by an angle of 48° and gets horizontal again when it reaches the bottom edge of the beam. The third main steel member lays on top of second one, as an offset of it in its central part, and gets inclined on both ends to take the shear in the beam. In the ends of the third member, it is possible to recognize two triangular "steel" zones, each composed by two "ties" and one "strut" (from the principal stress plots). The two "ties" are respectively the ends of the second and third steel member described above. The ends of the third member have different angles in the solutions with optimized and original geometry. In case the geometry is kept original, the angles are 77° on the left and 45° on the right. In case the geometry is optimized, the angles are approximately 82° on the left and 47° on the right. The difference in angles between the two solutions is to be imputed to the variation of the internal load transfer due to the presence or absence of the holes in the structure.

The interpretation of the solution is the following:

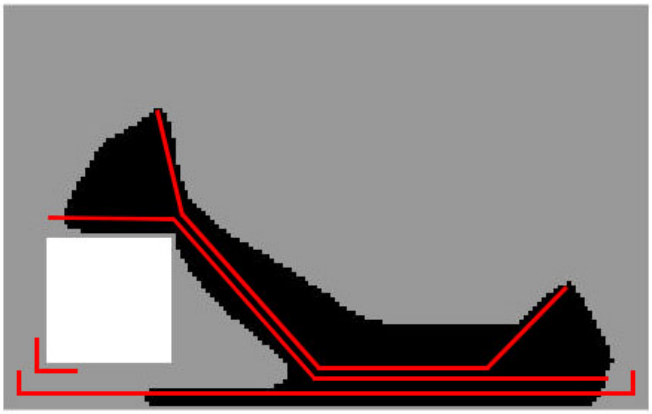


Figure 4.152: Case study n.10 - example interpretation of solution - original structure boundaries

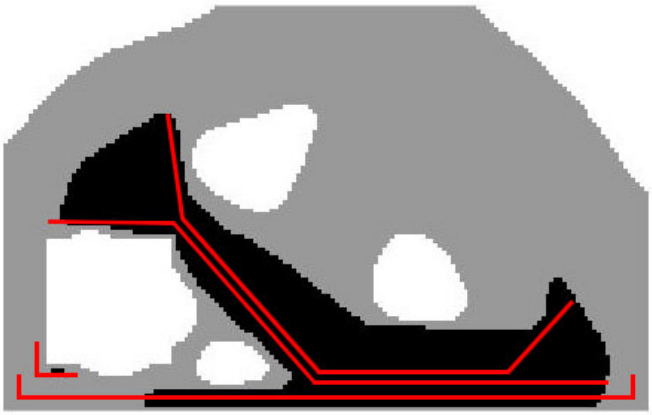


Figure 4.153: Case study n.10 - example interpretation of solution - optimized structure boundaries

In both cases, from analysis of principal stress plots, a tensile zone remains in the concrete on top of the left support and around the lower-left corner of the square opening. In that zone, additional steel members are drawn manually in the example interpretation of the solutions.

4.3.11 Case study n.11

Case study n.11 - original geometry, no reinforcement

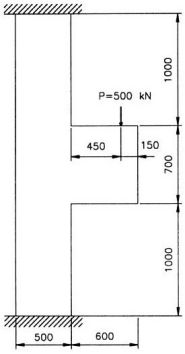
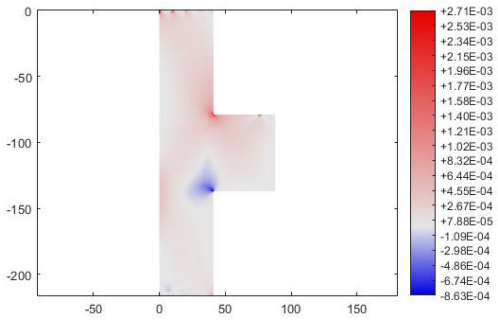
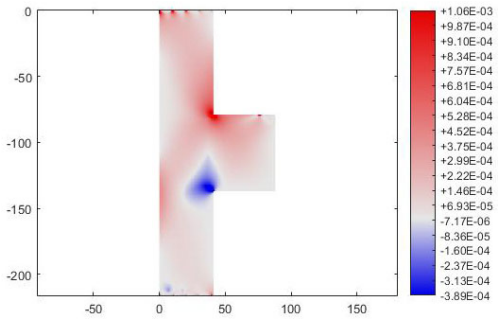


Figure 4.154: Case study n.11 - problem statement

Figure 4.155: Case study n.11 - geometry without reinforcement and optimization

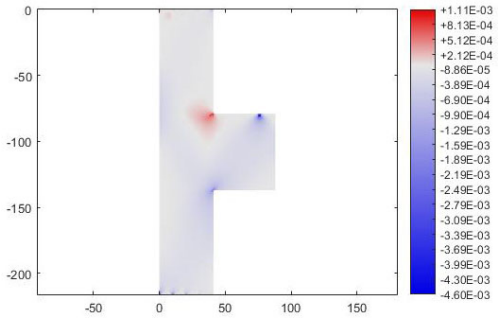


(a) full range of stresses

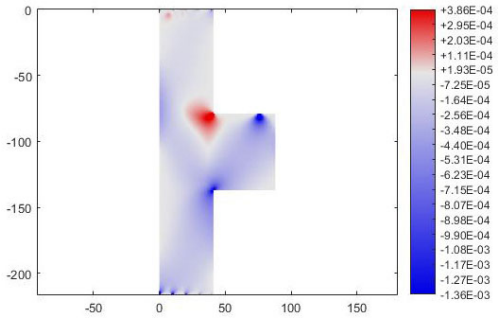


(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.156: Case study n.11 - plots of principal stress sigma 1



(a) full range of stresses



(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.157: Case study n.11 - plots of principal stress sigma 2

Case study n.11 - original geometry with optimized reinforcement

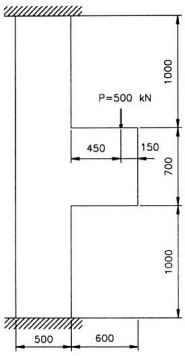
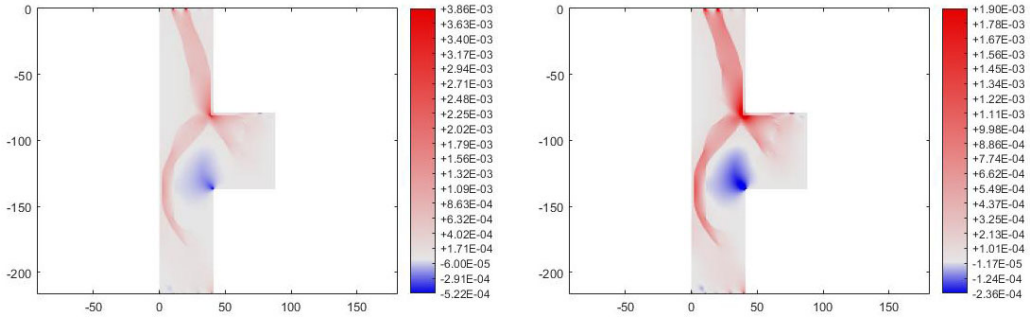


Figure 4.158: Case study n.11 - problem statement

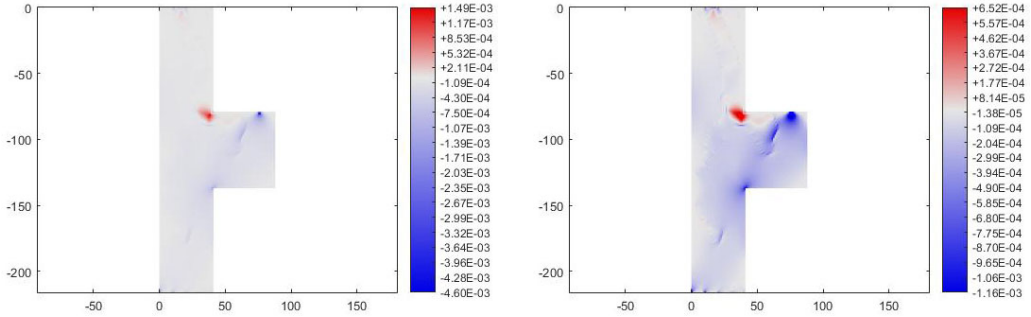
Figure 4.159: Case study n.11 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.160: Case study n.11 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.161: Case study n.11 - plots of principal stress sigma 2

Case study n.11 - optimized geometry and reinforcement

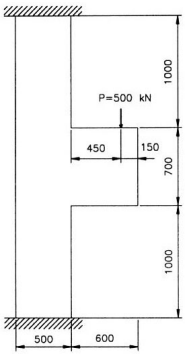
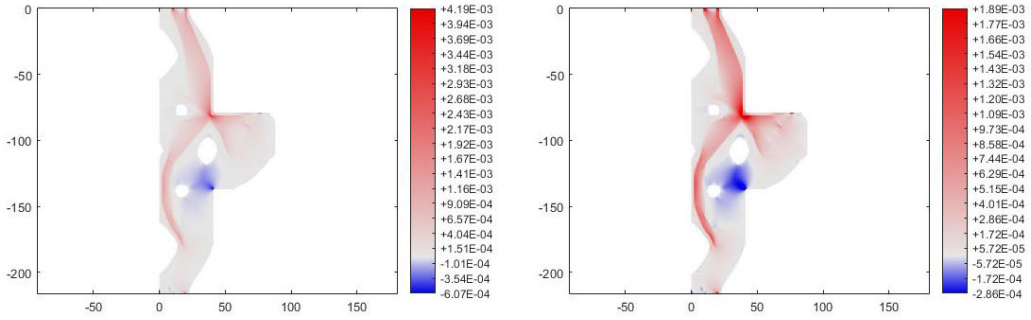


Figure 4.162: Case study n.11 - problem statement

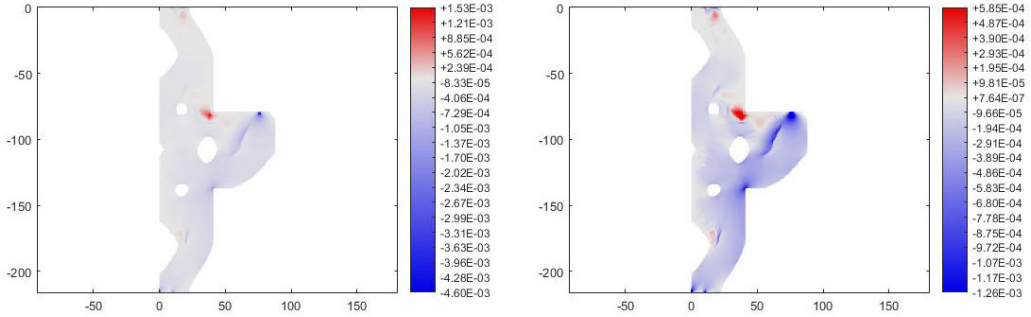
Figure 4.163: Case study n.11 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.164: Case study n.11 - plots of principal stress sigma 1



(a) full range of stresses

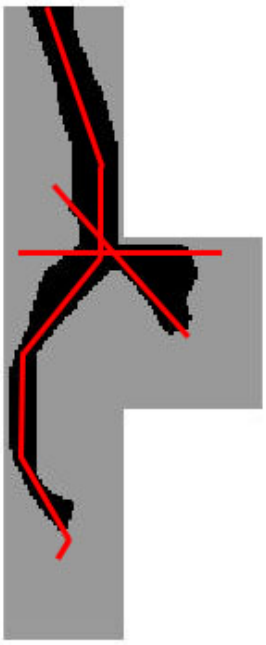
(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.165: Case study n.11 - plots of principal stress sigma 2

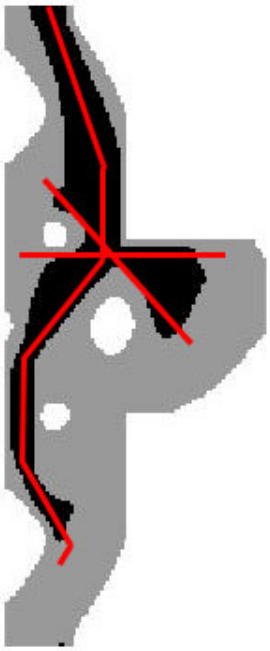
Case study n.11 - results interpretation

The case study n.11 corresponds to a corbel jointed to a column. The steel distribution found with the ESO procedure is similar for both solutions. A tensile member appears on the right side in the top part of the column and on the left side in the bottom part of the column, to take on the moment generated in the column by the load on the corbel. A horizontal steel member can be recognized in the top edge of the corbel to take on the tensile stresses generated by the bending in the corbel. A third steel member can be identified in an inclined direction of 48° circa, to take on the shear force in the corbel. The interpretation of the triangular "steel" area in the corbel into two steel members is supported also by the presence, in the second principal stress plot, of a recognizable compressive strut connecting the two resulting steel members.

The interpretation of the solution is the following:



(a) Case study n.11 - example interpretation of solution - original structure boundaries



(b) Case study n.11 - example interpretation of solution - optimized structure boundaries

4.3.12 Case study n.12

Case study n.12 - original geometry, no reinforcement

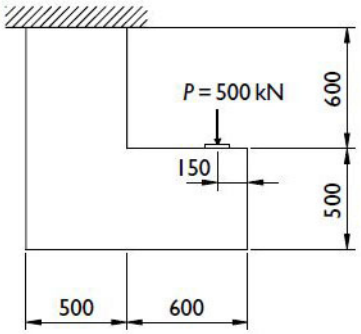
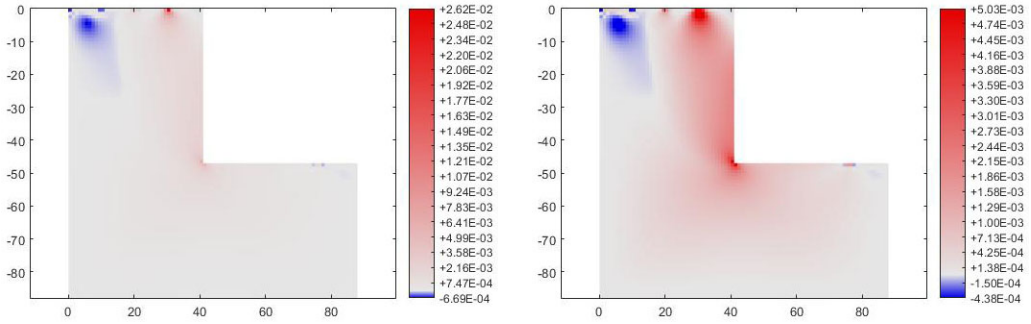


Figure 4.167: Case study n.12 - problem statement

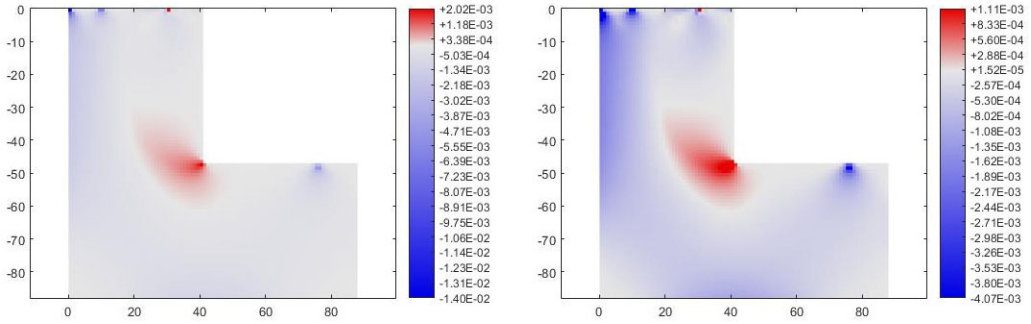
Figure 4.168: Case study n.12 - geometry without reinforcement and optimization



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.169: Case study n.12 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.170: Case study n.12 - plots of principal stress sigma 2

Case study n.12 - original geometry with optimized reinforcement

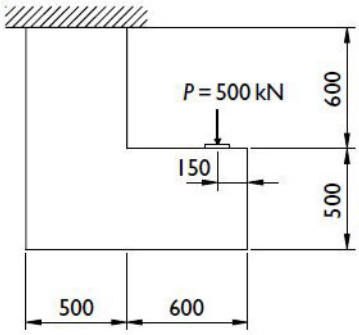
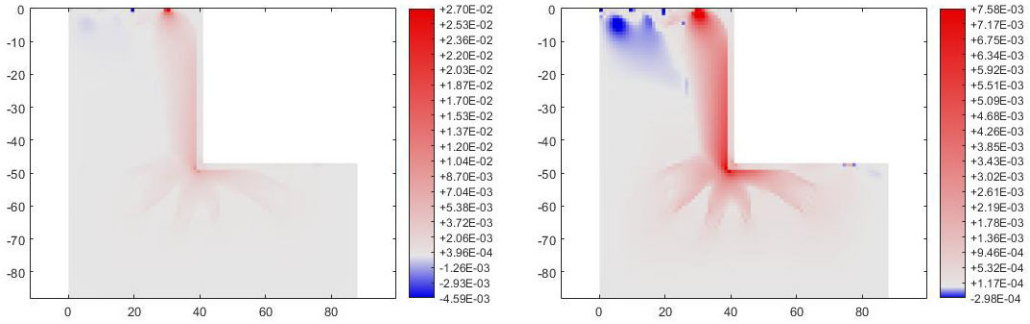


Figure 4.171: Case study n.12 - problem statement

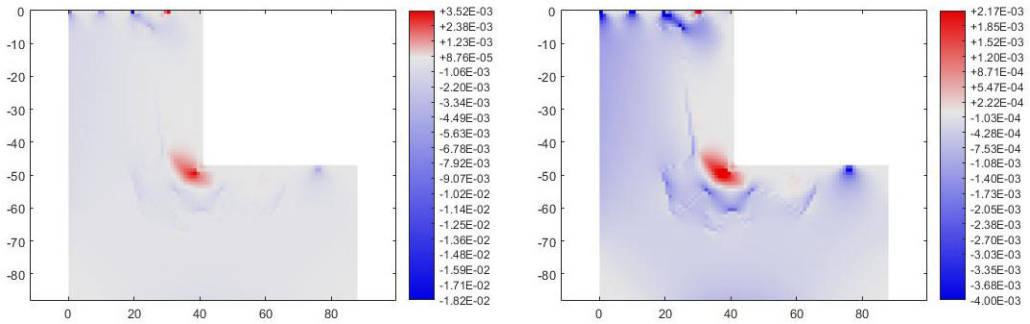
Figure 4.172: Case study n.12 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.173: Case study n.12 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.174: Case study n.12 - plots of principal stress sigma 2

Case study n.12 - optimized geometry and reinforcement

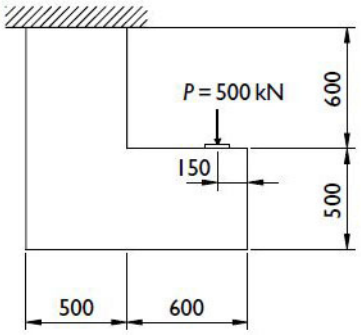
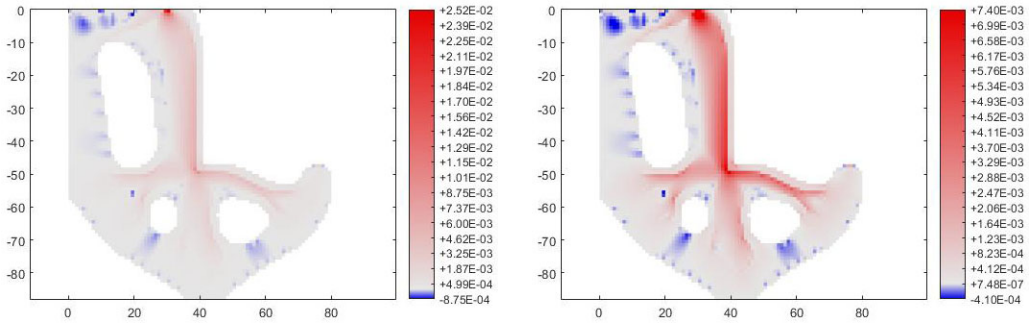


Figure 4.175: Case study n.12 - problem statement

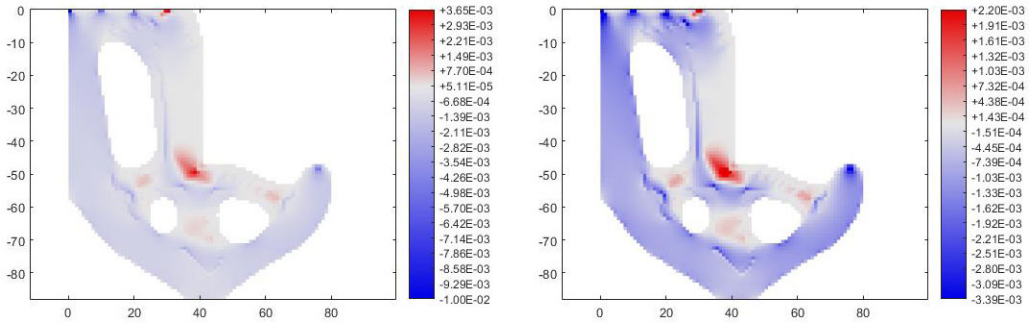
Figure 4.176: Case study n.12 - solution by the code of this thesis



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.177: Case study n.12 - plots of principal stress sigma 1



(a) full range of stresses

(b) stress concentrations removed: 0.5% highest tensile and 0.5% compressive elements

Figure 4.178: Case study n.12 - plots of principal stress sigma 2

Case study n.12 - results interpretation

The case study n.12 corresponds to a corbel with a ledge support. Through combined interpretation of steel distribution from ESO procedure and principal stress plots, it is derived a resulting steel layout similar for both solutions (original and optimized structure boundaries). Two vertical steel members descending directly from the support, transfer the tensile stresses to the boundary conditions. Two horizontal and inclined steel members take on the remaining tensile stresses in the corbel. The latter members have an inclination of approximately 57° on the left and 27° on the right in case the structure is optimized, while they have angles of around 45° on the left and 30° in case the structure geometry is kept original. In case the structure is optimized, from the solution of ESO process, it is possible to identify an additional short steel member between the support and the top left edge of the upper hole, with an inclination of circa 20°. This member takes on the stress concentrations around the hole and the shear force close to the support.

The interpretation of the solution is the following:

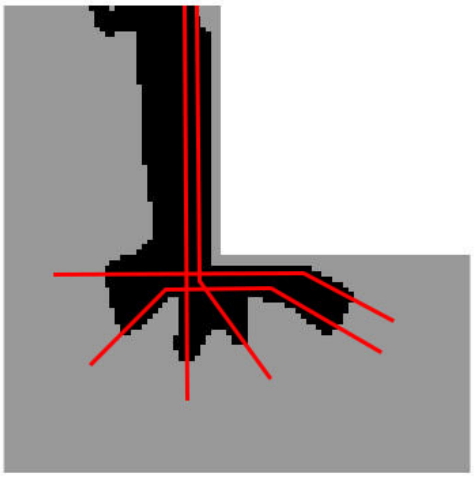


Figure 4.179: Case study n.11 - example interpretation of solution - original structure boundaries

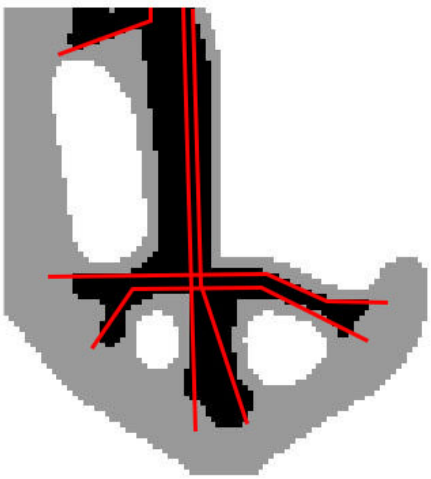


Figure 4.180: Case study n.11 - example interpretation of solution - optimized structure boundaries

Chapter 5

Conclusion

5.1 Conclusions

At this point, it is useful to recall the objective and the scope set for this thesis (as written in chapter 1 "Introduction"), and answer each of them along with the considerations and remarks from the main results of the thesis.

The objective of the thesis was to:

"Write a MATLAB code that is able to perform automatic preliminary design of reinforced concrete structures, through an Evolutionary Structural Optimization process."

In other words:

"Given a starting geometry, a set of boundary conditions (constraints and applied forces & displacements), and a set of target volumes for concrete and steel, the code should be able to define the optimal shape of the structure, and the optimal topology & distribution between concrete and steel, also taking concrete cover requirements into consideration.

The conclusions regarding the objective of the thesis are the following:

- It is possible to observe that Evolutionary Structural Optimization is well suited approach for the design process of RC structures, since the very purpose of the design is to place material in the most effective zones, remove material from zones where not needed, and use the given materials in the best way. This agrees with the basic principle of ESO, that is to remove material in ineffective zones to reach for the optimum.
- The first design issue, that is about finding the optimal outer shape of the RC structure, is possible through only one - the first - ESO loop, as there is no distinction between the concrete and the steel. Therefore, all results and all possible applications from ESO literature can be applied, simply changing the definition of the sensitivity number for the first ESO loop. Then the code can be easily extended to optimize the outer reinforced concrete structure with other criteria from literature.
  The second design issue, that is about finding the optimal topology & distribution between steel and concrete inside the structure, requires additional ESO loops. In the thesis, it has been found that two additional ESO loops would suffice to give a first approximation solution, in particular, when making use of transition of optimality criteria in sensitivity analysis from Von Mises to Drucker-Prager in one loop, and transition from Drucker-Prager to tensile stress criteria in the second loop. Summarizing, to find the optimal shape of the structure, and the optimal topology & (first approximation) distribution between concrete and steel, a total of three ESO loops are required to be executed in series.
- Regulating the target volume of reinforced concrete and the parameter r_{min} (radius of filtering for sensitivity numbers), it is possible to influence the number and size of holes in the RC structure.

It is also possible to favor a structure with thick members by setting a high r_{min} or getting a lattice-type structure by setting a target volume generally lower than 55%-50%. It is found that very small target volumes can remarkably simplify the structure at the expense of degenerating the solution to a local optimum, generally obtaining a sub-optimal, highly stressed and inefficient solution.

- With the current code, the required target volume of steel to obtain a useful and informative steel layout is relatively high, i.e. 10%, 15%, 20%, instead of around 1% what is required in real construction. In fact, setting in the algorithm steel target volumes smaller than 10%, steel members in secondary tensile zones vanish, making the solution degenerate to a trivial one, usually comprised only by a member in the highest tensile zone. To have solutions of interest, it is therefore required to set a significantly higher than normal target steel volume in the optimization procedure, which results in solutions with bulky, but lowly stressed, zones of steel. The resulting "black" areas in the solutions from the ESO procedure have then to be interpreted not directly as steel, but as zones where steel reinforcement is desired. Therefore, solutions have to be interpreted in combination with principal stresses plots, to create resulting informative steel reinforcement layouts.

- Stress concentrations have a relevant influence on the optimization's results. Particularly, the presence of steel members nearby stress concentrations results favored, due to the high stresses near singularities, and such elements present a stronger "resistance" to removal. This influence is not necessarily negative and results even beneficial in some cases. In fact, in presence of internal holes in the structure (and therefore in presence of stress concentrations around them), steel members in tensile zones result "anchored" to the holes, getting thinner and thinner but not being removed during the ESO process, and remain in the solution also for relatively small target volumes of steel. This results in more meaningful steel layouts when the reinforced concrete final structure contains holes (i.e. is optimized), compared to when it remains without holes (i.e. non-optimized).

- The application of concrete cover inside the ESO procedure has been proven possible, also for the case of automatic cover around holes with not previously known positions (i.e. the holes determined during the optimization process). Moreover, the concrete cover enforcement on the solution reduces the bulkiness of the steel members on the edges of the structure and so it makes possible to have a better steel distribution for a given steel target volume. On the other hand, the application of concrete cover in presence of small members in the structure can cause the disappearance of steel members with size approximately equal to double the concrete cover thickness. In cases when the presence of thin members in the structure cannot be avoided, the cover thickness must be reduced accordingly, so that at least a minimal steel thickness is preserved inside thin tensile members, through the optimization process.

- A novel concept of "mask" has been introduced in the thesis. In simple words, the *mask* is a chosen density matrix of an intermediate ESO loop that is stored and multiplied by the sensitivity numbers, to influence the resulting distribution of steel. The most relevant effects of the *mask* take place in the third ESO loop, where the *mask* makes possible to "average" the solution driven by tensile stress criterion, with historical result of the structure optimized with Drucker-Prager criterion from the second loop. In few case studies, it has been observed that the *mask* contributed to preserve the optimal angles for tensile members through the whole optimization process, especially for steel members resisting shear stresses. In fact, without the effect of the mask, the orientation of steel members has been observed to change in the third ESO loop, as result of steel removal from all compressive zones.

- Special attention has to be payed when optimizing internal distribution between steel and concrete without optimizing structure's outer boundaries. In fact, it may happen that, in intermediate ESO loops, some thin yet useful members are removed and, due to the application of the *mask* and/or of the automatic concrete cover, such members cannot be restored in later loops. Both *mask* and automatic concrete cover store information of material density from intermediate ESO loops to drive results of subsequent loops, as a result making more difficult to restore a wrongly removed element during intermediate loops. Therefore, both *mask* and automatic concrete cover must be deployed paying attention to the whole optimization process evolution. Whenever needed, the *mask* can

easily be deactivated setting `ESO.penalmask=0`, while non-automatic concrete covers can be chosen setting one of the cases from `SET.covertype=1,2,3`.

On a wider level, the scope of this thesis, as stated in the end of chapter 1, was to extend the ESO method for the particular case of:

"Topological optimization of composite materials with macro structure,
+ with different optimization criteria applied to the component materials,
+ with one component material that has different resistance in tension and compression (i.e. concrete),
+ with geometrical constraints for one component material (i.e. concrete cover applied on steel)."

The conclusions and remarks regarding the scope of the thesis are the following:

- The code developed in the thesis was indeed able to address all the above requirements and specificities. However, in order to do so, it was needed to use multiple ESO loops executed in series and apply different requirements in various phases of optimization. In each cycle, the optimization of distribution between two materials only is carried on and, for optimizing more than two materials, it must be done in a series of ESO loops, each of them with a couple of materials to be chosen by the user.

- The novel approach of linear interpolating two optimization criteria during an ESO process used in this thesis could turn useful in other ESO applications. Moreover, if the interpolation is not executed completely, for example if the first criteria is reduced from 100% and stopped to 40% and the second criteria is increased from 0% and stopped to 60%, it is possible to obtain a gradual and partial combination of multiple optimization criteria, different from the methods found in literature.

- The algorithm developed does not check if the resistance of the material is being reached, in other words, the limits for the stresses in the elements are not implemented. Even in the case when different behavior under tensile and compressive stresses is taken into account, it is considered only through the ratio between the two resistances and not considering their magnitude. This characteristic of the code is to be taken into consideration to understand the limits of application of the results obtained with it. Therefore, the solutions obtained in this thesis are valid for optimum topology pre-failure, while for post-failure, the optimum solution could have different shapes as well. To find an optimal solution for post-failure behavior (e.g. to optimize ductility), the code of this thesis should most probably be completely revised, introducing: resistance limits for materials, appropriate material models for post-failure behavior and a Non-Linear Finite Element Analysis solver or a Sequentially Linear Analysis procedure.

5.2 Directions for further development

5.2.1 Problems with self-weight

Even though it is beyond the scope of this thesis, the MATLAB code developed already includes the parameters to take self-weight into account, both in optimization and finite element analysis codes. In section 1.3 of the code `ESOscript.m`, in the point where material input data is inserted, it is possible to define the material density and the gravity acceleration vector to compute the self-weight:

```
gravity =  [ 0.0    %gravity acceleration x-dir
            -1.0];  %gravity acceleration y-dir
VOID.density= 0.0; %material density (rho)
VOID.bodyforce = VOID.density*gravity;  %specific weight(gamma=rho*g)
STEEL.density= 0.0;
STEEL.bodyforce = STEEL.density*gravity;
CONCR.density= 0.0;
CONCR.bodyforce = CONCR.density*gravity;
```

The function `FEM.m` is already fit to include self-weight into the finite element analysis as well. The forces applied to the finite elements due to self-weight are calculated for the two materials of the bi-phase material:

```
Nmat = zeros(2, 8); % form N matrix
Nmat(1, 1:2:end) = N(:)';
Nmat(2, 2:2:end) = N(:)';
f_int1 = Nmat'*MAT1.bodyforce*dX*GEOM.thickness;
f_e1 = f_e1 + f_int1;
f_int2 = Nmat'*MAT2.bodyforce*dX*GEOM.thickness;
f_e2 = f_e2 + f_int2;
```

and later on, they are assembled in the `f` vector, keeping into account the material distribution of the bi-phase material with the density matrices `x` and `y`:

```
edof = DOF.n_mat(iel,:);
f(edof) = f(edof) + (f_e1 *x(iel)^penal + f_e2 *(1-x(iel)^penal))*y(iel)^penal
```

However, it is not only necessary to specify gravity vector and material densities, but to adapt sensitivity numbers definitions as well. This has not been done in the current thesis and is left to the reader.

To illustrate the extents of required adaptations to sensitivity numbers for self-weight problems, it is reported here below an example found from literature [21] which treats stiffness ESO optimization with design-dependent loads. In the article [21], the following material interpolation scheme had been chosen:

$$\begin{cases} \rho_i = x_i \rho \\ E(x_i) = \frac{x_i}{1-q(1-x_i)} E \end{cases} \qquad \forall i = 1, ..., N \tag{5.1}$$

with q a penalty factor (in the article set as $q = 5$). It is useful to remind that for problems *without* self-weight the material interpolation scheme usually chosen is:

$$E(x_i) = E x_i^p \qquad \forall i = 1, ..., N$$

After having defined a proper material interpolation scheme, the procedure to derive the sensitivity numbers is, as usual:

- determine the objective function C (in the case from literature, it is the compliance);
- compute the derivative of the objective function $\frac{\partial C}{\partial x_i}$;
- substitute the material interpolation scheme into $\frac{\partial C}{\partial x_i}$;
- determine a convenient definition of the sensitivity number α_i (linear proportional to $\frac{\partial C}{\partial x_i}$);
- substitute the final expression of $\frac{\partial C}{\partial x_i}$ obtained into α_i.

It is to be noted that also the derivative of the objective function differs between problems with and without self-weight. In the article [21], the following derivative of the objective function is used for problems with self-weight:

$$\frac{\partial C}{\partial x_i} = \frac{\partial f_i^T}{\partial x_i} u_i - \frac{1}{2} u_i^T \frac{\partial K_i}{\partial x_i} u_i \qquad \forall i = 1, ..., N \tag{5.2}$$

It is useful, once again, to compare it to the one used in problems *without* self weight:

$$\frac{\partial C}{\partial x_i} = -\frac{1}{2} u_i^T \frac{\partial K_i}{\partial x_i} u_i \qquad \forall i = 1, ..., N$$

In problems with self-weight, the term $\frac{\partial f_i^T}{\partial x_i}$ is not zero since the self-weight, which is accounted for into the vector f, is dependent to the material density distribution x_i, that changes along the optimization process.

The definition of sensitivity number is also slightly different between problems with and without self-weight. In the article [21], it is used a sensitivity number definition:

$$\alpha_i = -\frac{1}{q+1}\frac{\partial C}{\partial x_i} \tag{5.3}$$

compared to the definition of sensitivity number used in problems of stiffness optimization *without* self-weight:

$$\alpha_i = -\frac{1}{p}\frac{\partial C}{\partial x_i}$$

Further elaborations of the above formulas to derive the final expression for the sensitivity number for stiffness-based ESO with self-weight is not reported in this thesis but may be found on the article of Xie [21].

It is relevant to note that the formulations of sensitivities found in literature for stiffness optimization with self-weight cannot be directly applied to stress-based optimization.

At the same time, the effect of self-weight must be taken into account into the sensitivity numbers and not only in the FEM analysis. Otherwise the ESO optimization would not work correctly, rising problems of divergence, giving incorrect results, etc. Several tests have been performed by the author and not once the ESO process would work correctly when the self-weight was included only in FEM and not in the sensitivities. Therefore, it appears extended research is needed to find out suitable formulations for sensitivity numbers for stress-based ESO with self-weight. This work is not carried out in the present thesis and is left to readers as a suggestion for further research.

5.2.2 Improvement of steel layout optimization

Even though steel layouts in optimized solutions of case studies, computed with the current code, appear to be in the right positions, the steel members are too bulky and cannot be reduced in thickness enough through diminishing the steel target volume. This issue has already been presented and discussed in section 4.2.2 “Target volume of steel”. It is therefore desirable to improve the present code to enable the decrease of current limits in steel target volume without getting trivial solutions.

The author believes it is possible to address this problem using a similar approach as the one used to create the automatic concrete cover in the file `COVER.m`. The main idea is to run a `for` cycle on all the steel elements and, for each element, check its neighboring elements within a defined radius, check a predefined rule and override the material phase setting for the central element (depending if the predefined rule is true or false). In such a way, the author believes that it is possible to program a minimum and maximum size for steel members and define a minimum size of concrete between two separate steel members. Therefore, in the resulting steel layouts, the bulky and lowly stressed steel members obtained by the present version of code, could be transformed to multiple and higher stressed steel members, with minimum spacing and maximum size requirements. At the same time, the algorithm should be written in such a way that thin members are preserved during this “rationalization” process. Following this approach, the added code would most probably assume the form of a new function, to be executed during current third ESO loop, or of an additional fourth ESO loop.

Annex

The MATLAB code is comprised by 5 files for the main code:

- Evolutionary Structural Optimization script - file *"ESOscript.m"*
- Finite Element Analysis function - file *"FEM.m"*
- Sensitivity Analysis function - file *"SENS.m"*
- Post-processing function - file *"POST.m"*
- Automatic concrete cover function - file *"COVER.m"*

and one file to plot results:

- Plot field (defined on element-level) script - file *"plotELEMfield.m"*

Evolutionary Structural Optimization script

```
% This Matlab code was written by Andrea De Marco, for the scope of his MSc
% thesis dissertation at Delft University of Technology, The Netherlands,
% held in June 2018.
%
% The code has been elaborated from the algorithm published in "Bi-directional
% Evolutionary Structural Optimization on Advanced Structures and Materials:
% A Comprehensive Review" by L.Xia, Q.Xia, X.Huang, Y.M.Xie, in 2016.
%
% The code is intended for educational purposes. Details about the code and
% the procedure can be found in the MSc thesis "Application of Evolutionary
% Structural Optimization to Reinforced Concrete Structures" by Andrea De Marco
% downloadable at http://repository.tudelft.nl/
%
% Disclaimer:
% This code is provided by the author "as is" without any warranties of any
% sort. In no event shall the author be liable for any damage arising by the
% use of this program.
%
%-0---COMMENTS---------------------------------------------------------------
% This script is programmed with bilinear 4 noded elements numerically or
% analytically integrated with 2x2 gauss points, in condition of plane stress
% The design area is drawn in a rectangular domain with square regular elements
tic
%-1---INPUT------------------------------------------------------------------
%-1.1---INPUT-GEOMETRY-------------------------
ELEM.nx = 100; ELEM.ny = 50; %number of elements used to discretize design area
GEOM.lenght = ELEM.nx; GEOM.height = ELEM.ny;  %length and height of design area
GEOM.thickness = 1;
%-1.2---INPUT-SETTINGS-------------------------
SET.inttype=1; %1=2x2integration, 2=analytical
SET.bctype=2; %1=manual, 2=cantilever with central force, 3=half MBB, 4=half wheel,
              %5=cantilever with bottom force ... see section 1.5 for others
SET.solidvoid=1; %1=no solids and voids, 2=round hole, 3=square hole, 4=rhombus hole,
                 %5=rectangular hole ... see section 1.6 for others
SET.boundaries=2; %1=keep original boundaries, 2=optimize boundaries
SET.covertype=8; %1=no concrete cover, 2=top&bottom, 3=left&right, 4=automatic cover,
                 %5=automatic cover + top&bottom, 6=automatic cover + left&right
                 %7=automatic cover + top&bottom + left&right
                 %8=automatic cover + top&bottom + right
SET.coverthick=2; %concrete cover thickness (expressed in number of elements)
SET.translenght=20; %n. of iterations for transition between two optimization criteria
%FOR SWITCHING BETWEEN CASE STUDIES OF THE THESIS IS IN NOT NEEDED TO
%MODIFY PARAMETERS UNDER THIS LINE, WITH THE EXEPTION OF ESO.volfrac
%IN SECTION 1.4.2
%-1.3---INPUT-MATERIAL-DATA--------------------
VOID.E  = 0;
VOID.nu = 0;
VOID.lambda = 0;
VOID.mu     = 0;
STEEL.E  = 10.0;
STEEL.nu = 0.3;
STEEL.lambda = STEEL.nu*STEEL.E/( (1+STEEL.nu)*(1-2*STEEL.nu) );
STEEL.mu     = STEEL.E/( 2*(1+STEEL.nu) );
CONCR.E  = 1.0;
CONCR.nu = 0.2;
CONCR.lambda = CONCR.nu*CONCR.E/( (1+CONCR.nu)*(1-2*CONCR.nu) );
CONCR.mu     = CONCR.E/( 2*(1+CONCR.nu) );
% factors for drucker prager criterion
sigmac=1;
```

```
sigmat=0.2;
DPalpha=(sigmac-sigmat)/(sqrt(3)*(sigmac+sigmat));
clear sigmac sigmat
% for selfweight problems only
gravity =  [ 0.0    %gravity acceleration x-dir
            -1.0];  %gravity acceleration y-dir
VOID.density= 0.0; %material density (rho)
VOID.bodyforce = VOID.density*gravity;  %specific weight(gamma=rho*g)
STEEL.density= 0.0;
STEEL.bodyforce = STEEL.density*gravity;
CONCR.density= 0.0;
CONCR.bodyforce = CONCR.density*gravity;
clear gravity
%-1.4---INPUT-ESO-PARAMETERS--------------------------
%-1.4.1--INITALIZATION-ESO-PARAMETERS----
x0=ones(ELEM.ny,ELEM.nx); %start geometry
ESO.cycles=3; %determine the number of cycles
x=cell(1,ESO.cycles); %initialize eso density matrix
for cycle=1:ESO.cycles; x{cycle}(1:ELEM.ny,1:ELEM.nx)=1.; end
ESO.penal = 3.; %penalty exponent applied to x
ESO.penalmask = 2.; %penalty exponent applied to mask
%set ESO.penalmask=2 to have same effect as of ESO.penal on x
%set ESO.penalmask=0 to nullify effect of mask (works only for SET.covertype=1,2,3)
%-1.4.2--INPUT-VARIABLE-ESO-PARAMETERS----------------------
%ESO settings (1 column = 1 cycle) to be changed for each case study
ESO.volfrac=          [0.60,        0.30,           0.15            ]; % <-------
%-1.4.3--INPUT-FIXED-ESO-PARAMETERS-----------------------------------
%ESO settings (1 column = 1 cycle) usually not needed to be changed
ESO.conv_err=         [0.001        0.001           0.001           ]; %allowable conv. error
ESO.er=               [0.04,        0.03,           0.02            ];
ESO.rmin=             [8,           3,              2               ];
ESO.criteria=         [2,           5,              7               ];
MAT1=                 {STEEL,       STEEL,          STEEL           };
MAT2=                 {VOID,        CONCR,          CONCR           };
ESO.startvol=         [1,           ESO.volfrac(1), ESO.volfrac(2)  ];
update_startx=@(x0,x) {x0,          x{1},           x{2}            }; %values of ESO.start
update_mask=@(x0,x)   {x0,          x{1},           x{2}            }; %values of mask
switch SET.boundaries   % set values of y (concrete boundaries)
case 1   % preserve original boundaries
update_y=@(x0,x)      {x0,          x0,             x0              };
case 2   % optimize boundaries
update_y=@(x0,x)      {x0,          x{1},           x{1}            };
end
% startx, y and mask are set in the beginning and updated inside the eso loops
% at end of each main-iteration because x{1}, x{2} etc are calculated inside the
% eso loops, to do this MATLAB function handles are used
startx=update_startx(x0,x);
y=update_y(x0,x);
mask=update_mask(x0,x);
% In y it is stored the information about concrete boundaries
%-1.5---INPUT-BOUNDARY-CONDITIONS--------------
% GEOM.bc=[# # # #]'=[node n., bctype, dof, bcvalue]'
% (bctype=0 for force, bctype=1 for displacement)
switch (SET.bctype)
    % 1-manual
    case 1
      GEOM.bc = [
        1      1   1    0.0 %example restrained displacement x-dir
        1      1   2    0.0 %example restrained displacement y-dir
        683    0   1    -10.0 %example applied force x-dir
        683    0   2    -10.0 %example applied force y-dir
```

```
      3      1   1    -5.0 %example applied displacement x-dir (settlement)
      3      1   2    -5.0 %example applied displacement y-dir (settlement)
    ]';  %note that the matrix is transposed
% 2-cantilever with central force
case 2
   GEOM.bc=zeros(4,2*(ELEM.ny+1)+1);
   for ibc=1:ELEM.ny+1
       GEOM.bc (:,ibc)= [ibc, 1, 1, 0.0]; %x-dir
       GEOM.bc (:,ibc+ELEM.ny+1)= [ibc, 1, 2, 0.0]; %y-dir
   end
   GEOM.bc (:,2*(ELEM.ny+1)+1)= [(ELEM.nx+1)*(ELEM.ny+1)-ELEM.ny/2, 0, 2, -1.0]; %force
   clear ibc
% 3-half MBB beam
case 3
    GEOM.bc=zeros(4,(ELEM.ny+1)+2);
   for ibc=1:ELEM.ny+1
       GEOM.bc (:,ibc)= [ibc, 1, 1, 0.0]; %x-dir
   end
   GEOM.bc (:,(ELEM.ny+1)+1)= [(ELEM.nx+1)*(ELEM.ny+1), 1, 2, 0.0]; %support
   GEOM.bc (:,(ELEM.ny+1)+2)= [1, 0, 2, -1.0]; %force
   clear ibc
% 4-half wheel design
case 4
   GEOM.bc=zeros(4,4);
   GEOM.bc (:,1)= [(ELEM.ny+1), 1, 1, 0.0]; %x-dir
   GEOM.bc (:,2)= [(ELEM.ny+1), 1, 2, 0.0]; %y-dir
   GEOM.bc (:,3)= [(ELEM.nx+1)*(ELEM.ny+1), 1, 2, 0.0]; %support
   GEOM.bc (:,4)= [(ELEM.nx/2+1)*(ELEM.ny+1), 0, 2, -1.0]; %force
% 5-cantilever with bottom force
case 5
   GEOM.bc=zeros(4,2*(ELEM.ny+1)+1);
   for ibc=1:ELEM.ny+1
       GEOM.bc (:,ibc)= [ibc, 1, 1, 0.0]; %x-dir
       GEOM.bc (:,ibc+ELEM.ny+1)= [ibc, 1, 2, 0.0]; %y-dir
   end
   GEOM.bc (:,2*(ELEM.ny+1)+1)= [(ELEM.nx+1)*(ELEM.ny+1), 0, 2, -1.0]; %force
   clear ibc
% 6-plate fixed on edges with up force on upper quarter in the middle
case 6
   GEOM.bc=zeros(4,9);
   GEOM.bc (:,1)= [1, 1, 1, 0.0]; %x-dir
   GEOM.bc (:,2)= [1, 1, 2, 0.0]; %y-dir
   GEOM.bc (:,3)= [(ELEM.ny+1), 1, 1, 0.0]; %x-dir
   GEOM.bc (:,4)= [(ELEM.ny+1), 1, 2, 0.0]; %y-dir
   GEOM.bc (:,5)= [(ELEM.nx+1)*(ELEM.ny+1)-ELEM.ny, 1, 1, 0.0]; %x-dir
   GEOM.bc (:,6)= [(ELEM.nx+1)*(ELEM.ny+1)-ELEM.ny, 1, 2, 0.0]; %y-dir
   GEOM.bc (:,7)= [(ELEM.nx+1)*(ELEM.ny+1), 1, 1, 0.0]; %x-dir
   GEOM.bc (:,8)= [(ELEM.nx+1)*(ELEM.ny+1), 1, 2, 0.0]; %y-dir
   GEOM.bc (:,9)= [(ELEM.nx/2+1)*(ELEM.ny+1)-3/4*ELEM.ny, 0, 2, 1.0]; %force
% 7-plate fixed on edges with down force in the middle
case 7
   GEOM.bc=zeros(4,9);
   GEOM.bc (:,1)= [1, 1, 1, 0.0]; %x-dir
   GEOM.bc (:,2)= [1, 1, 2, 0.0]; %y-dir
   GEOM.bc (:,3)= [(ELEM.ny+1), 1, 1, 0.0]; %x-dir
   GEOM.bc (:,4)= [(ELEM.ny+1), 1, 2, 0.0]; %y-dir
   GEOM.bc (:,5)= [(ELEM.nx+1)*(ELEM.ny+1)-ELEM.ny, 1, 1, 0.0]; %x-dir
   GEOM.bc (:,6)= [(ELEM.nx+1)*(ELEM.ny+1)-ELEM.ny, 1, 2, 0.0]; %y-dir
   GEOM.bc (:,7)= [(ELEM.nx+1)*(ELEM.ny+1), 1, 1, 0.0]; %x-dir
   GEOM.bc (:,8)= [(ELEM.nx+1)*(ELEM.ny+1), 1, 2, 0.0]; %y-dir
   GEOM.bc (:,9)= [(ELEM.nx/2+1)*(ELEM.ny+1)-0.5*ELEM.ny, 0, 2, -1.0]; %force
```

```
   % 8-wall with 2 square holes case study 98 x 60 only (not scalable)
   case 8
     GEOM.bc=zeros(4,5);
     GEOM.bc (:,1)= [305, 1, 1, 0.0]; %x-dir
     GEOM.bc (:,2)= [305, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,3)= [5795, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,4)= [1831,  0, 2, -0.010]; %force 1
     GEOM.bc (:,5)= [4149,  0, 2, -0.010]; %force 2
   % 9-wall with 1 square holes case study 150 x 94 only (not scalable)
   case 9
     GEOM.bc=zeros(4,4);
     GEOM.bc (:,1)= [570, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,2)= [13870, 1, 1, 0.0]; %x-dir
     GEOM.bc (:,3)= [13870, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,4)= [8931,  0, 2, -0.010]; %force
   % 10-column with corbel case study 88 x 216 only (not scalable)
   case 10
     GEOM.bc=zeros(4,21);
     GEOM.bc (:,1)= [1, 1, 1, 0.0]; %x-dir
     GEOM.bc (:,2)= [1, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,3)= [217, 1, 1, 0.0]; %x-dir
     GEOM.bc (:,4)= [217, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,5)= [2171, 1, 1, 0.0]; %x-dir
     GEOM.bc (:,6)= [2171, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,7)= [2387, 1, 1, 0.0]; %x-dir
     GEOM.bc (:,8)= [2387, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,9)= [4341, 1, 1, 0.0]; %x-dir
     GEOM.bc (:,10)= [4341, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,11)= [4557, 1, 1, 0.0]; %x-dir
     GEOM.bc (:,12)= [4557, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,13)= [6511, 1, 1, 0.0]; %x-dir
     GEOM.bc (:,14)= [6511, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,15)= [6727, 1, 1, 0.0]; %x-dir
     GEOM.bc (:,16)= [6727, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,17)= [8681, 1, 1, 0.0]; %x-dir
     GEOM.bc (:,18)= [8681, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,19)= [8897, 1, 1, 0.0]; %x-dir
     GEOM.bc (:,20)= [8897, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,21)= [16573,  0, 2, -0.010]; %force
   % 11-corbel with ledge support case study 88 x 88 only (not scalable)
   case 11
     GEOM.bc=zeros(4,10);
     GEOM.bc (:,1)= [1, 1, 1, 0.0]; %x-dir
     GEOM.bc (:,2)= [1, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,3)= [891, 1, 1, 0.0]; %x-dir
     GEOM.bc (:,4)= [891, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,5)= [1781, 1, 1, 0.0]; %x-dir
     GEOM.bc (:,6)= [1781, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,7)= [2671, 1, 1, 0.0]; %x-dir
     GEOM.bc (:,8)= [2671, 1, 2, 0.0]; %y-dir
     GEOM.bc (:,9)= [3561, 1, 1, 0.0]; %x-dir
     GEOM.bc (:,10)= [6813,  0, 2, -0.010]; %force
end
%-1.6---VOIDS-AND-SOLIDS-IN-GEOMETRY-----------
%set ESO.passive = 1 (holes) or 2 (solids)
switch (SET.solidvoid)
    % 1-no void or solids
    case 1
        ESO.passive = zeros(ELEM.ny,ELEM.nx);
    % 2-circular hole
    case 2
```

```
    ESO.passive = zeros(ELEM.ny,ELEM.nx);
    for ii = 1:ELEM.nx
      for jj = 1:ELEM.ny
          %sqrt((jj - center_y)^2+(ii- center_x)^2) < radius
        if sqrt((jj-ELEM.ny/2)^2+(ii-ELEM.nx/3)^2) < ELEM.ny/3
        ESO.passive(jj,ii) = 1;
        end
      end
    end
% 3-square hole
case 3
    ESO.passive = zeros(ELEM.ny,ELEM.nx);
    for ii = 1:ELEM.nx
      for jj = 1:ELEM.ny
          %max([abs(jj - center_y),abs(ii- center_x)] < half length
        if max([abs(jj-ELEM.ny/2),abs(ii-ELEM.nx/3)]) < ELEM.ny/3
        ESO.passive(jj,ii) = 1;
        end
      end
    end
% 4-rhombus hole
case 4
    ESO.passive = zeros(ELEM.ny,ELEM.nx);
    for ii = 1:ELEM.nx
      for jj = 1:ELEM.ny
          %abs((jj - center_y)+abs(ii- center_x) < half length
        if abs(jj-ELEM.ny/2)+abs(ii-ELEM.nx/3) < ELEM.ny/3
        ESO.passive(jj,ii) = 1;
        end
      end
    end
% 5-rectangular hole
case 5
    ESO.passive = zeros(ELEM.ny,ELEM.nx);
    for ii = 1:ELEM.nx
      for jj = 1:ELEM.ny
          %max([abs(jj - center_y)/y_scale,abs(ii- center_x)/x_scale] < half length
        if max([abs(jj-ELEM.ny/2)/1,abs(ii-ELEM.nx/3)/1.5]) < ELEM.ny/3
        ESO.passive(jj,ii) = 1;
        end
      end
    end
 % 6-two rectangular holes for wall with two forces SET.bctype=8
 case 6
    ESO.passive = zeros(ELEM.ny,ELEM.nx);
    for ii = 1:ELEM.nx
      for jj = 1:ELEM.ny
          %max([abs(jj - center_y)/y_scale,abs(ii- center_x)/x_scale] < half length
        if max([abs(jj-30),abs(ii-16)]) < 6
        ESO.passive(jj,ii) = 1;
        end
        if max([abs(jj-30),abs(ii-83)]) < 6
        ESO.passive(jj,ii) = 1;
        end
      end
    end
 % 7-rectangular hole for wall with one force SET.bctype=9
 case 7
    ESO.passive = zeros(ELEM.ny,ELEM.nx);
    for ii = 1:ELEM.nx
      for jj = 1:ELEM.ny
```

```
            %max([abs(jj - center_y)/y_scale,abs(ii- center_x)/x_scale] < half length
          if max([abs(jj-69)/1,abs(ii-25)/1]) < 15
          ESO.passive(jj,ii) = 1;
          end
        end
      end
    % 8-rectangular hole for column and corbel case study SET.bctype=10
    case 8
      ESO.passive = zeros(ELEM.ny,ELEM.nx);
      for ii = 1:ELEM.nx
        for jj = 1:ELEM.ny
            %max([abs(jj - center_y)/y_scale,abs(ii- center_x)/x_scale] < half length
          if max([abs(jj-40)/4,abs(ii-65)/2.4]) < 10
          ESO.passive(jj,ii) = 1;
          end
          if max([abs(jj-177)/4,abs(ii-65)/2.4]) < 10
          ESO.passive(jj,ii) = 1;
          end
        end
      end
    % 9-rectangular hole for column and corbel case study SET.bctype=11
    case 9
      ESO.passive = zeros(ELEM.ny,ELEM.nx);
      for ii = 1:ELEM.nx
        for jj = 1:ELEM.ny
            %max([abs(jj - center_y),abs(ii- center_x)] < half length
          if max([abs(jj-24),abs(ii-65)]) < 24
          ESO.passive(jj,ii) = 1;
          end
        end
      end
end
%-1.7---CONCRETE-COVER-------------------------
switch (SET.covertype)
    % 1-no concrete cover imposed
    case 1
        ESO.cover = zeros(ELEM.ny,ELEM.nx);
    % 2-top and bottom cover
    case 2
        ESO.cover = zeros(ELEM.ny,ELEM.nx);
        for ii = 1:ELEM.nx
          for jj = 1:ELEM.ny
            %max([abs(jj - center_y)/y_scale,abs(ii- center_x)/x_scale] < half length
            if max([abs(jj-1),abs(ii-ELEM.nx/2)/ELEM.nx]) < SET.coverthick
            ESO.cover(jj,ii) = 1;
            end
            if max([abs(jj-ELEM.ny),abs(ii-ELEM.nx/2)/ELEM.nx]) < SET.coverthick
            ESO.cover(jj,ii) = 1;
            end
          end
        end
    % 3-left and right cover
    case 3
        ESO.cover = zeros(ELEM.ny,ELEM.nx);
        for ii = 1:ELEM.nx
          for jj = 1:ELEM.ny
            %max([abs(jj - center_y)/y_scale,abs(ii- center_x)/x_scale] < half length
            if max([abs(jj-ELEM.ny/2)/ELEM.ny,abs(ii-1)]) < SET.coverthick
            ESO.cover(jj,ii) = 1;
            end
            if max([abs(jj-ELEM.ny/2)/ELEM.ny,abs(ii-ELEM.nx)]) < SET.coverthick
```

```
            ESO.cover(jj,ii) = 1;
            end
          end
        end
    % 4-automatic cover around void elements
    case 4
        ESO.cover = zeros(ELEM.ny,ELEM.nx);
        %function to fill ESO.cover is called inside the ESO code
    % 5-automatic cover around void elements + top and bottom cover
    case 5
        ESO.cover = zeros(ELEM.ny,ELEM.nx);
        %function to fill ESO.cover is called inside the ESO code
    % 6-automatic cover around void elements + left and right cover
    case 6
        ESO.cover = zeros(ELEM.ny,ELEM.nx);
        %function to fill ESO.cover is called inside the ESO code
    % 7-automatic cover around void elements + top and bottom + left and right cover
    case 7
        ESO.cover = zeros(ELEM.ny,ELEM.nx);
        %function to fill ESO.cover is called inside the ESO code
    % 8-automatic cover around void elements + top and bottom + right cover
    case 8
        ESO.cover = zeros(ELEM.ny,ELEM.nx);
        %function to fill ESO.cover is called inside the ESO code
end
%-2---INITIALIZATION-----------------------------------------------------
%-2.1---INITIALIZATION-GEOMETRY------------------
% creation of node and dof numbers
% - matrix of nodes numbers
NODE.n_mat = reshape(1:(1+ELEM.nx)*(1+ELEM.ny),1+ELEM.ny,1+ELEM.nx);
% - node number on bottom right corner
DOF.n_mat_col_1 = reshape(2*NODE.n_mat(1:end-1,1:end-1)+1,ELEM.nx*ELEM.ny,1);
% - matrix of dof numbers in the element
DOF.n_mat = repmat(DOF.n_mat_col_1,1,8)+repmat([0 1 2*ELEM.ny+[2 3 0 1] -2 -1],ELEM.nx*ELEM.ny,1);
% creation of matrix of nodes coordinates (order top to bottom, right to left)
[NODE.x_mat,NODE.y_mat] = ...
                    meshgrid(linspace(0,GEOM.lenght,ELEM.nx+1),linspace(0,-GEOM.height,ELEM.ny+1));
NODE.xy = [NODE.x_mat(:) NODE.y_mat(:)];
NODE.n = size(NODE.xy, 1);
% creation of connections of nodes for members (order top to bottom, right to left)
NODE.connect = [(DOF.n_mat(:,2)/2) (DOF.n_mat(:,4)/2) (DOF.n_mat(:,6)/2) (DOF.n_mat(:,8)/2)];
ELEM.n = size(NODE.connect, 1);
%-2.2---INITIALIZATION-INTEGRATION-SCHEME---------
% 2x2 point Gauss integr
GP.def = zeros(3,4);
GP.def(1,:) = 1.0; %weight of gp,
GP.def(2,1) = -1.0/sqrt(3.0); GP.def(3,1) = -1.0/sqrt(3.0); %eta and xi coord of gp
GP.def(2,2) =  1.0/sqrt(3.0); GP.def(3,2) = -1.0/sqrt(3.0);
GP.def(2,3) =  1.0/sqrt(3.0); GP.def(3,3) =  1.0/sqrt(3.0);
GP.def(2,4) = -1.0/sqrt(3.0); GP.def(3,4) =  1.0/sqrt(3.0);
%-2.3---INITIALIZATION-MESH-INDEPENDENCY-FILTER-----
for cycle=1:ESO.cycles
iH = ones(ELEM.nx*ELEM.ny*(2*(ceil(ESO.rmin(ESO.cycles))-1)+1)^2,1);
jH = ones(size(iH)); sH = zeros(size(iH)); k = 0;
for i1 = 1:ELEM.nx
  for j1 = 1:ELEM.ny
    e1 = (i1-1)*ELEM.ny+j1;
    for i2 = max(i1-(ceil(ESO.rmin(cycle))-1),1):min(i1+(ceil(ESO.rmin(cycle))-1),ELEM.nx)
      for j2 = max(j1-(ceil(ESO.rmin(cycle))-1),1):min(j1+(ceil(ESO.rmin(cycle))-1),ELEM.ny)
        e2 = (i2-1)*ELEM.ny+j2; k = k+1;
        iH(k) = e1; jH(k) = e2;
```

```
        sH(k) = max(0,ESO.rmin(cycle)-sqrt((i1-i2)^2+(j1-j2)^2));
      end
    end
  end
end
ESO.H{cycle} = sparse(iH,jH,sH);
ESO.Hs{cycle} = sum(ESO.H{cycle},2); %the arrays needed for filtering are ESO.H and ESO.Hs
clear e1 e2 i1 i2 j1 j2 iH jH sH k cycle
end
%-3---OPTIMIZATION-LOOP-------------------------------------------------
for cycle=1:ESO.cycles
  %-3.1--INITIALIZATION i-(th) ESO PROCEDURE-------------
  iter = 0; change = 1.;
  x{cycle}=startx{cycle};
  vol=ESO.startvol(cycle);
  %-3.2--START ITERATIONS FOR i-(th) ESO PROCEDURE-------
  while change > ESO.conv_err(cycle)
    %-3.2.1--volume and historical data
    iter = iter + 1;
    if iter >1
       ESO.oldapha = ESO.alpha;
    end
    if cycle==2 && iter==1 && SET.covertype>1
       %in first iter when it is applied cover don't reduce volume by ER
    else
       vol = max(vol*(1-ESO.er(cycle)),ESO.volfrac(cycle));
    end
    %-3.2.2--fem analysis
    [DOF.K, DOF.f, DOF.u, ELEM.KE1, ELEM.KE2]=...
        FEM(ELEM,NODE,DOF,GEOM,GP,SET,MAT1{cycle},MAT2{cycle},x{cycle},y{cycle},ESO.penal);
    %-3.2.3--postprocess for stress based eso criteria
    if ESO.criteria(cycle) ~= 1
    [GP.strains, GP.stresses, GP.xy]=POST(ELEM,NODE,DOF,GP, MAT1{cycle}, MAT2{cycle},x{cycle},ESO.penal);
    end
    %-3.2.4--sensitivity analysis
    [ESO.alpha, SET] = ...
        SENS(ELEM,DOF,GP,SET,ESO.criteria(cycle),x{cycle},mask{cycle},ESO.penal,ESO.penalmask,DPalpha,iter);
    %calculate total objective function
    ESO.obj(iter) = sum(sum(ESO.alpha));
    %filtering sensitivities
    ESO.alpha(:) = ESO.H{cycle}*ESO.alpha(:)./ESO.Hs{cycle};
    %stabilizing sensitivities
    if iter > 1
        ESO.alpha = (ESO.alpha+ESO.oldapha)/2.;
    end
    %-3.2.5--enforce voids and solids
    ESO.alpha(ESO.passive==1) = min(min(ESO.alpha));
    ESO.alpha(ESO.passive==2) = max(max(ESO.alpha));
    %for cases when concrete boundaries are not optimized (SET.boundaries==1)
    %enforce voids and solids directly in x0
    if iter==1 && SET.boundaries==1
    x0(ESO.passive==1) = 0.001; %enforced voids by codes in section 1.6
    x0(ESO.passive==2) = 1;     %enforced solids by codes in section 1.6
    end
    %-3.2.6--enforce concrete cover
    if cycle>1
       if SET.covertype>=4
           %compute automatic concrete cover (for cases SET.covertype>=4)
           %and apply steel-free zone outside concrete boundaries (done inside function COVER)
           [ESO.cover]=COVER(ELEM,SET,y,cycle);
       else
```

```
          %concrete cover already set before (for cases SET.covertype<4)
          %then only apply steel-free zone outside concrete boundaries
          ESO.alpha(y{cycle}~=1)=min(min(ESO.alpha));
       end
      if cycle==2 && iter==1 %only the first iteration on 2 cycle it adjusts volume
          removedsteel=0;
          for indx=1:ELEM.n
          %if there is steel inside concrete boundary and it needs to be
          %deleted due to application of concrete cover in first iteration
              if y{cycle}(indx)==ESO.cover(indx)
                 removedsteel=removedsteel+1;
              end
          end
          %modify volume
          vol=vol-removedsteel/ELEM.n;
       end
       %concrete cover is enforced overriding ESO.alpha
       ESO.alpha(ESO.cover==1) = min(min(ESO.alpha));
    end
    %-3.2.7--beso design update
    l1 = min(ESO.alpha(:)); l2 = max(ESO.alpha(:));
    while abs((l2-l1)/(l1+l2)) > 1.0e-9
        th = (l1+l2)/2.0;
        x{cycle} = max(0.001,sign(ESO.alpha-th)) ;
        if mean(x{cycle}(:)) - vol > 0
            l1 = th;
        else
            l2 = th;
        end
    end
    clear l1 l2 th
    %-3.2.8--output results
    %print results
    if iter>10 && vol == ESO.volfrac(cycle)
       %if minimum 10 iterations are performed and target volume has been reached
        change=abs(sum(ESO.obj(iter-9:iter-5))-sum(ESO.obj(iter-4:iter)))/sum(ESO.obj(iter-4:iter));
    end
    disp([' It.: ' sprintf('%4i',iter) ' Obj.: ' sprintf('%10.4f',ESO.obj(iter)) ...
           ' Vol.: ' sprintf('%6.3f',sum(sum(x{cycle}))/(ELEM.nx*ELEM.ny)) ...
            ' ch.: ' sprintf('%6.3f',change )])
    %plot design
    if cycle ==1
    colormap(gray); imagesc(-x{cycle}); axis equal; axis tight; axis off;pause(1e-6);
    else
    colormap(gray); imagesc(-y{cycle}); hold on; fingeom=imagesc(-x{cycle}); hold off;
    alpha(fingeom,.6); axis equal; axis tight; axis off;pause(1e-6);
    end
  end   %end of while loop of i-(th) ESO procedure
  clear vol change iter
  %-3.3---UPDATE-x-y-mask-WITH-i-(th)-ITERATION-CALCULATIONS-
  %it is needed because x{1} x{2} etc. are calculated during the loops
  startx=update_startx(x0,x);
  y=update_y(x0,x);
  mask=update_mask(x0,x);
end  %end of for loop of the series of ESO optimizations
%-4--POST-PROCESSING-------------------------------------------------------
[GP.strains, GP.stresses, GP.xy]=POST(ELEM,NODE,DOF,GP, MAT1{cycle}, MAT2{cycle},x{cycle},ESO.penal);
% assign nan value to void elements for not printing stresses and strains of void parts
gp_id=0;
for i_element=1:ELEM.n
    for gauss_point = GP.def
```

```
        gp_id = gp_id +1;
        if y{cycle}(i_element)~= 1 %get the boundaries of the concrete
            GP.stresses(gp_id,:) = nan;
            GP.strains(gp_id,:) = nan;
        end
    end
end
clear DPalpha ii jj indx removedsteel gp_id cycle gauss_point i_element fingeom x0
% extract ux and uy in separate vectors, needed for deformed plots
NODE.ux = DOF.u(1:2:end);
NODE.uy = DOF.u(2:2:end);
% extract epsxx, epsyy and epsxy in separate vectors
GP.epsxx=GP.strains(:,1);
GP.epsyy=GP.strains(:,2);
GP.epsxy=GP.strains(:,3);
% extract sigmaxx, sigmayy and sigmaxy in separate vectors
GP.sigmaxx=GP.stresses(:,1);
GP.sigmayy=GP.stresses(:,2);
GP.sigmaxy=GP.stresses(:,3);
% compute principal stresses and von mises stress
GP.toumax = +sqrt((0.5*(GP.stresses(:,1)-GP.stresses(:,2))).^2+GP.stresses(:,3).^2);
GP.sigma1 = 0.5*(GP.stresses(:,1)+GP.stresses(:,2))+GP.toumax ;
GP.toumin = -sqrt((0.5*(GP.stresses(:,1)-GP.stresses(:,2))).^2+GP.stresses(:,3).^2);
GP.sigma2 = 0.5*(GP.stresses(:,1)+GP.stresses(:,2))+GP.toumin ;
GP.vonmises = sqrt(0.5*((GP.sigma1-GP.sigma2).^2+GP.sigma2.^2+GP.sigma1.^2));
% for plots with element averaged stresses
ELEM.epsxx = sum(reshape(GP.epsxx,4,ELEM.n))/4;
ELEM.epsyy = sum(reshape(GP.epsyy,4,ELEM.n))/4;
ELEM.epsxy = sum(reshape(GP.epsxy,4,ELEM.n))/4;
ELEM.sigmaxx = sum(reshape(GP.sigmaxx,4,ELEM.n))/4;
ELEM.sigmayy = sum(reshape(GP.sigmayy,4,ELEM.n))/4;
ELEM.sigmaxy = sum(reshape(GP.sigmaxy,4,ELEM.n))/4;
ELEM.sigma1 = sum(reshape(GP.sigma1,4,ELEM.n))/4;
ELEM.sigma2 = sum(reshape(GP.sigma2,4,ELEM.n))/4;
ELEM.vonmises = sum(reshape(GP.vonmises,4,ELEM.n))/4;
toc
```

Finite Element Analysis function

```
% function to execute the finite element analysis and calculate displacements
function [K, f, u, KE1, KE2] = FEM(ELEM,NODE,DOF,GEOM,GP,SET,MAT1,MAT2,x,y,penal)
%-1---ASSEMBLE-K,f-MATRICES-----------------------------------------------
% preallocate element matrices
K_e1 = zeros(8); K_e2 = zeros(8);
KEx = zeros(8,8,ELEM.n); KE1 = zeros(8,8,ELEM.n); KE2 = zeros(8,8,ELEM.n);
f_e1 = zeros(8,1); f_e2 = zeros(8,1);
% preallocate global vectors
f = sparse(NODE.n*2,1);
u = zeros(NODE.n*2,1);
% loop over all elements to get necessary data to assemble stiffness matrix
for iel=1:ELEM.n
  % extract coordinates of the nodes of element considered
  xy_el = NODE.xy(NODE.connect(iel,:),:);
  % assemble
  switch (SET.inttype)
    % 1 numerical integration
    case 1
    % loop over integration points to assemble element matrix
    for gauss_point = GP.def
        % shape functions for 4-noded quad
        N = zeros(4,1); dN = zeros(4,2);
        a(1) =-1.0; a(2) = 1.0; a(3) = 1.0; a(4) =-1.0; % x locations of nodes
        b(1) =-1.0; b(2) =-1.0; b(3) = 1.0; b(4) = 1.0; % y locations of nodes
        xi  = gauss_point(2);
        eta = gauss_point(3);
        N(:)    = 0.25*(1.0 + a(:)*xi + b(:)*eta + a(:).*b(:)*xi*eta);
        dN(:,1) = 0.25*(a(:) + a(:).*b(:)*eta);
        dN(:,2) = 0.25*(b(:) + a(:).*b(:)*xi);
        % transform derivatives from isoparametric configuration to global config.
        J = xy_el' * dN;  % Note that J is the inverse of the usual J
        jacobian = det(J); % determinant of Jacobian matrix
        dN = dN/J; % transform derivatives to global system (=dN*inv(J))
        % form B matrix
        B = zeros(3, 8);  %(8 = 2dof x 4nodes)
        B(1, 1:2:end) = dN(:,1)';
        B(2, 2:2:end) = dN(:,2)';
        B(3, 1:2:end) = dN(:,2)';
        B(3, 2:2:end) = dN(:,1)';
        % form D matrix for material 1
        D1 = zeros(3);
        D1(1,1) = MAT1.lambda + 2*MAT1.mu; D1(2,2) = MAT1.lambda + 2*MAT1.mu;
        D1(3,3) = MAT1.mu;  D1(1,2) = MAT1.lambda;  D1(2,1) = MAT1.lambda;
        % form D matrix for material 2
        D2 = zeros(3);
        D2(1,1) = MAT2.lambda + 2*MAT2.mu; D2(2,2) = MAT2.lambda + 2*MAT2.mu;
        D2(3,3) = MAT2.mu; D2(1,2) = MAT2.lambda; D2(2,1) = MAT2.lambda;
        % compute matrix K_int and assemble K_e
        dX = gauss_point(1)*jacobian;
        K_int1 = B'*D1*B*dX*GEOM.thickness;
K_e1 = K_e1 + K_int1;
        K_int2 = B'*D2*B*dX*GEOM.thickness;
K_e2 = K_e2 + K_int2;
        % only for self weight problems, assemble f vector
        Nmat = zeros(2, 8); % form N matrix
        Nmat(1, 1:2:end) = N(:)';
        Nmat(2, 2:2:end) = N(:)';
        f_int1 = Nmat'*MAT1.bodyforce*dX*GEOM.thickness;
        f_e1 = f_e1 + f_int1;
```

```
        f_int2 = Nmat'*MAT2.bodyforce*dX*GEOM.thickness;
        f_e2 = f_e2 + f_int2;
    end
    % 2 analytical integration
    case 2
    k1=[ 1/2-MAT1.nu/6   1/8+MAT1.nu/8 -1/4-MAT1.nu/12 -1/8+3*MAT1.nu/8 ...
        -1/4+MAT1.nu/12 -1/8-MAT1.nu/8  MAT1.nu/6       1/8-3*MAT1.nu/8];
    K_e1 = MAT1.E/(1-MAT1.nu^2)* [k1(1) k1(2) k1(3) k1(4) k1(5) k1(6) k1(7) k1(8)
                                  k1(2) k1(1) k1(8) k1(7) k1(6) k1(5) k1(4) k1(3)
                                  k1(3) k1(8) k1(1) k1(6) k1(7) k1(4) k1(5) k1(2)
                                  k1(4) k1(7) k1(6) k1(1) k1(8) k1(3) k1(2) k1(5)
                                  k1(5) k1(6) k1(7) k1(8) k1(1) k1(2) k1(3) k1(4)
                                  k1(6) k1(5) k1(4) k1(3) k1(2) k1(1) k1(8) k1(7)
                                  k1(7) k1(4) k1(5) k1(2) k1(3) k1(8) k1(1) k1(6)
                                  k1(8) k1(3) k1(2) k1(5) k1(4) k1(7) k1(6) k1(1)];
    k2=[ 1/2-MAT2.nu/6   1/8+MAT2.nu/8 -1/4-MAT2.nu/12 -1/8+3*MAT2.nu/8 ...
        -1/4+MAT2.nu/12 -1/8-MAT2.nu/8  MAT2.nu/6       1/8-3*MAT2.nu/8];
    K_e2 = MAT2.E/(1-MAT2.nu^2)* [k2(1) k2(2) k2(3) k2(4) k2(5) k2(6) k2(7) k2(8)
                                  k2(2) k2(1) k2(8) k2(7) k2(6) k2(5) k2(4) k2(3)
                                  k2(3) k2(8) k2(1) k2(6) k2(7) k2(4) k2(5) k2(2)
                                  k2(4) k2(7) k2(6) k2(1) k2(8) k2(3) k2(2) k2(5)
                                  k2(5) k2(6) k2(7) k2(8) k2(1) k2(2) k2(3) k2(4)
                                  k2(6) k2(5) k2(4) k2(3) k2(2) k2(1) k2(8) k2(7)
                                  k2(7) k2(4) k2(5) k2(2) k2(3) k2(8) k2(1) k2(6)
                                  k2(8) k2(3) k2(2) k2(5) k2(4) k2(7) k2(6) k2(1)];
  end
  % assemble f vector, only for selfweight problems
  edof = DOF.n_mat(iel,:);
  f(edof) = f(edof) + (f_e1 *x(iel)^penal + f_e2 *(1-x(iel)^penal))*y(iel)^penal;
  % store K_e matrix of (iel)-th element on KE tensor
  KEx(:,:,iel)=(K_e1(:,:)*x(iel)^penal+K_e2(:,:)*(1-x(iel)^penal))*y(iel)^penal;
  KE1(:,:,iel)=K_e1(:,:); % used for sensitivity only
  KE2(:,:,iel)=K_e2(:,:); % used for sensitivity only
  % reset element matrix
  K_e1(:) = 0.0; K_e2(:) = 0.0;
  f_e1 = 0.0; f_e2 = 0.0; % only for selfweight problems
end
% assemble stiffness matrix
iK = reshape(kron(DOF.n_mat,ones(8,1))',64*ELEM.nx*ELEM.ny,1);
jK = reshape(kron(DOF.n_mat,ones(1,8))',64*ELEM.nx*ELEM.ny,1);
sK = reshape(KEx,64*ELEM.nx*ELEM.ny,1);
K = sparse(iK,jK,sK);
%-2---APPLY-BOUNDARY-CONDITIONS-------------------------------------------
fixeddofs=zeros(1,size(GEOM.bc,2));
i_dof=0;
for bc = GEOM.bc
    if bc(2) == 0  % Neumann boundary conditions, forces
        dof = 2*(bc(1)-1)+bc(3); % multiplied by 2 because 2 dof per node
        f(dof) = f(dof)+bc(4); % it is added to selfweight if present
    elseif bc(2) == 1  %Dirichlet boundary conditions, displacements
        i_dof=i_dof+1;
        dof = 2*(bc(1)-1)+bc(3); % multiplied by 2 because 2 dof per node
        fixeddofs(1,i_dof)=dof;
        u(dof) = bc(4); % assign value of displacement to fixed dof
    else
    disp('Unkown boundary condition code')
    end
end
%-3---SOLUTION-OF-SYSTEM--------------------------------------------------
alldofs = (1:NODE.n*2);
freedofs = setdiff(alldofs,fixeddofs);
```

```
%solve systems for free dof only; fixed dof are set when applying bc
u(freedofs) = K(freedofs,freedofs)\f(freedofs);
```

Sensitivity Analysis function

```
% function used to get sensitivity numbers
function [alphamat,SET] = SENS(ELEM,DOF,GP,SET,criteria,x,mask,penal,penalmask,DPalpha,iter)
switch (criteria)
  % 1-stiffness criteria
  case 1
    alphavec=zeros(1,ELEM.n);
    for iiel=1:ELEM.n
        alphavec(iiel)=0.5 *  x(iiel)^(penal-1) *  mask(iiel)^(penalmask)* ...
        ( dot(DOF.u(DOF.n_mat(iiel,:))'*ELEM.KE1(:,:,iiel) , DOF.u(DOF.n_mat(iiel,:)))-...
          dot(DOF.u(DOF.n_mat(iiel,:))'*ELEM.KE2(:,:,iiel) , DOF.u(DOF.n_mat(iiel,:))) );
    end
    alphamat=reshape(alphavec,ELEM.ny,ELEM.nx);
  % 2-von mises criteria
  case 2
    W=[1,1,0]';
    V=[1,-0.5,0;    -0.5,1,0;    0,0,3];
    GP.I1=GP.stresses*W;
    GP.J2=1/3*sum(GP.stresses*V.*GP.stresses,2);
    % calculate average stress on elem from values on gauss points
    ELEM.J2vec = sum(reshape(GP.J2,4,ELEM.n))/4;
    ELEM.J2mat = reshape(ELEM.J2vec,ELEM.ny,ELEM.nx);
    alphamat=(sqrt(ELEM.J2mat)).*x.^(penal-1).*mask.^(penalmask);
  % 3-drucker prager criteria
  case 3
    W=[1,1,0]';
    V=[1,-0.5,0;    -0.5,1,0;    0,0,3];
    GP.I1=GP.stresses*W;
    GP.J2=1/3*sum(GP.stresses*V.*GP.stresses,2);
    % calculate average stress on elem from values on gauss points
    ELEM.I1vec = sum(reshape(GP.I1,4,ELEM.n))/4;
    ELEM.I1mat = reshape(ELEM.I1vec,ELEM.ny,ELEM.nx);
    ELEM.J2vec = sum(reshape(GP.J2,4,ELEM.n))/4;
    ELEM.J2mat = reshape(ELEM.J2vec,ELEM.ny,ELEM.nx);
    alphamat=(DPalpha*ELEM.I1mat+sqrt(ELEM.J2mat)).*x.^(penal-1).*mask.^(penalmask);
  % 4-tensile stress criteria
  case 4
    % calculate principal stresses on gauss points
    GP.toumax = +sqrt((0.5*(GP.stresses(:,1)-GP.stresses(:,2))).^2+GP.stresses(:,3).^2);
    GP.sigma1 = 0.5*(GP.stresses(:,1)+GP.stresses(:,2))+GP.toumax ;
    GP.toumin = -sqrt((0.5*(GP.stresses(:,1)-GP.stresses(:,2))).^2+GP.stresses(:,3).^2);
    GP.sigma2 = 0.5*(GP.stresses(:,1)+GP.stresses(:,2))+GP.toumin ;
    % calculate average stress on elem from values on gauss points
    ELEM.sigma1vec = sum(reshape(GP.sigma1,4,ELEM.n))/4;
    ELEM.sigma1mat = reshape(ELEM.sigma1vec,ELEM.ny,ELEM.nx);
    ELEM.sigma2vec = sum(reshape(GP.sigma2,4,ELEM.n))/4;
    ELEM.sigma2mat = reshape(ELEM.sigma2vec,ELEM.ny,ELEM.nx);
    alphamat=( ELEM.sigma1mat+ELEM.sigma2mat...
        +abs(min(min(ELEM.sigma1mat)))+abs(min(min(ELEM.sigma2mat))) )...
        .*x.^(penal).*mask.^(penalmask);
  % 5-transition from von mises to drucker prager criteria
  case 5
    W=[1,1,0]';
    V=[1,-0.5,0;    -0.5,1,0;    0,0,3];
    GP.I1=GP.stresses*W;
    GP.J2=1/3*sum(GP.stresses*V.*GP.stresses,2);
    % calculate average stress on elem
    ELEM.I1vec = sum(reshape(GP.I1,4,ELEM.n))/4;
    ELEM.I1mat = reshape(ELEM.I1vec,ELEM.ny,ELEM.nx);
    ELEM.J2vec = sum(reshape(GP.J2,4,ELEM.n))/4;
```

```
    ELEM.J2mat = reshape(ELEM.J2vec,ELEM.ny,ELEM.nx);
    if iter<=SET.translenght; SET.trans=iter; end
    alphamat=(SET.trans/SET.translenght*DPalpha*ELEM.I1mat+sqrt(ELEM.J2mat))...
        .*x.^(penal-1).*mask.^(penalmask);
  % 6-transition from von mises to tensile criteria
  case 6
    W=[1,1,0]';
    V=[1,-0.5,0;    -0.5,1,0;    0,0,3];
    GP.I1=GP.stresses*W;
    GP.J2=1/3*sum(GP.stresses*V.*GP.stresses,2);
    % calculate principal stresses on gauss points
    GP.toumax = +sqrt((0.5*(GP.stresses(:,1)-GP.stresses(:,2))).^2+GP.stresses(:,3).^2);
    GP.sigma1 = 0.5*(GP.stresses(:,1)+GP.stresses(:,2))+GP.toumax ;
    GP.toumin = -sqrt((0.5*(GP.stresses(:,1)-GP.stresses(:,2))).^2+GP.stresses(:,3).^2);
    GP.sigma2 = 0.5*(GP.stresses(:,1)+GP.stresses(:,2))+GP.toumin ;
    % calculate average stress on elem from values on gauss points
    ELEM.I1vec = sum(reshape(GP.I1,4,ELEM.n))/4;
    ELEM.I1mat = reshape(ELEM.I1vec,ELEM.ny,ELEM.nx);
    ELEM.J2vec = sum(reshape(GP.J2,4,ELEM.n))/4;
    ELEM.J2mat = reshape(ELEM.J2vec,ELEM.ny,ELEM.nx);
    ELEM.sigma1vec = sum(reshape(GP.sigma1,4,ELEM.n))/4;
    ELEM.sigma1mat = reshape(ELEM.sigma1vec,ELEM.ny,ELEM.nx);
    ELEM.sigma2vec = sum(reshape(GP.sigma2,4,ELEM.n))/4;
    ELEM.sigma2mat = reshape(ELEM.sigma2vec,ELEM.ny,ELEM.nx);
    if iter<=SET.translenght; SET.trans=iter; end
    alphamat=((1-SET.trans/SET.translenght)*(sqrt(ELEM.J2mat))...
        +(SET.trans/SET.translenght)*( ELEM.sigma1mat+ELEM.sigma2mat...
        +abs(min(min(ELEM.sigma1mat)))+abs(min(min(ELEM.sigma2mat))) ) )...
        .*x.^(penal-1).*mask.^(penalmask);
  % 7-transition from drucker prager to tensile criteria
  case 7
    W=[1,1,0]';
    V=[1,-0.5,0;    -0.5,1,0;    0,0,3];
    GP.I1=GP.stresses*W;
    GP.J2=1/3*sum(GP.stresses*V.*GP.stresses,2);
    % calculate principal stresses on gauss points
    GP.toumax = +sqrt((0.5*(GP.stresses(:,1)-GP.stresses(:,2))).^2+GP.stresses(:,3).^2);
    GP.sigma1 = 0.5*(GP.stresses(:,1)+GP.stresses(:,2))+GP.toumax ;
    GP.toumin = -sqrt((0.5*(GP.stresses(:,1)-GP.stresses(:,2))).^2+GP.stresses(:,3).^2);
    GP.sigma2 = 0.5*(GP.stresses(:,1)+GP.stresses(:,2))+GP.toumin ;
    % calculate average stress on elem from values on gauss points
    ELEM.I1vec = sum(reshape(GP.I1,4,ELEM.n))/4;
    ELEM.I1mat = reshape(ELEM.I1vec,ELEM.ny,ELEM.nx);
    ELEM.J2vec = sum(reshape(GP.J2,4,ELEM.n))/4;
    ELEM.J2mat = reshape(ELEM.J2vec,ELEM.ny,ELEM.nx);
    ELEM.sigma1vec = sum(reshape(GP.sigma1,4,ELEM.n))/4;
    ELEM.sigma1mat = reshape(ELEM.sigma1vec,ELEM.ny,ELEM.nx);
    ELEM.sigma2vec = sum(reshape(GP.sigma2,4,ELEM.n))/4;
    ELEM.sigma2mat = reshape(ELEM.sigma2vec,ELEM.ny,ELEM.nx);
    if iter<=SET.translenght; SET.trans=iter; end
    alphamat=((1-SET.trans/SET.translenght)*(DPalpha*ELEM.I1mat+sqrt(ELEM.J2mat))...
        +(SET.trans/SET.translenght)*( ELEM.sigma1mat+ELEM.sigma2mat...
        +abs(min(min(ELEM.sigma1mat)))+abs(min(min(ELEM.sigma2mat))) ) )...
        .*x.^(penal-1).*mask.^(penalmask);
end
```

Post-processing function

```
% function used to get stresses, strains at integration points
function [strainsGP, stressesGP, xyGP]=POST(ELEM,NODE,DOF,GP, MAT1, MAT2,x,penal)
strainsGP = zeros(4*ELEM.n,3) ;     % strains at GP (e_xx, e_yy, e_xy)
stressesGP = zeros(4*ELEM.n,3) ;    % stresses at GP (s_xx, s_yy, s_xy)
xyGP = zeros(4*ELEM.n,2) ;          % location of GP
% GP are ordered in a single column from first to fourth, element after element
% loop over all elements to assemble stiffness matrix
gp_id=0;
for i_element=1:ELEM.n
    % extract coordinates of the nodes of element considered
    x_element = NODE.xy(NODE.connect(i_element,:),:);
    % loop over integration points to assemble element matrix
    for gauss_point = GP.def
        gp_id = gp_id +1;
        % shape functions for 4-noded quad
        N = zeros(4,1); dN = zeros(4,2);
        a(1) =-1.0; a(2) = 1.0; a(3) = 1.0; a(4) =-1.0; % x locations of nodes
        b(1) =-1.0; b(2) =-1.0; b(3) = 1.0; b(4) = 1.0; % y locations of nodes
        xi  = gauss_point(2);
        eta = gauss_point(3);
        N(:)    = 0.25*(1.0 + a(:)*xi + b(:)*eta + a(:).*b(:)*xi*eta);
        dN(:,1) = 0.25*(a(:) + a(:).*b(:)*eta);
        dN(:,2) = 0.25*(b(:) + a(:).*b(:)*xi);
        % transform derivatives from isoparametric configuration to global config.
        J = x_element.' * dN;  % Note that J is the inverse of the usual J
        dN = dN/J; % transform derivatives to global system (=dN*inv(J))
        % form B matrix
        B = zeros(3, 8);  %(8 = 2dof x 4nodes)
        B(1, 1:2:end) = dN(:,1)';
        B(2, 2:2:end) = dN(:,2)';
        B(3, 1:2:end) = dN(:,2)';
        B(3, 2:2:end) = dN(:,1)';
        % form D matrix for material 1
        D1 = zeros(3);
        D1(1,1) = MAT1.lambda + 2*MAT1.mu; D1(2,2) = MAT1.lambda + 2*MAT1.mu;
        D1(3,3) = MAT1.mu;  D1(1,2) = MAT1.lambda;  D1(2,1) = MAT1.lambda;
        % form D matrix for material 2
        D2 = zeros(3);
        D2(1,1) = MAT2.lambda + 2*MAT2.mu; D2(2,2) = MAT2.lambda + 2*MAT2.mu;
        D2(3,3) = MAT2.mu; D2(1,2) = MAT2.lambda; D2(2,1) = MAT2.lambda;
        % compute locations of GP
        xyGP(gp_id,:)=N'*NODE.xy(NODE.connect(i_element,:),:);
        % compute strains on GP
        strainsGP(gp_id,:)=B*DOF.u(DOF.n_mat(i_element,:));
        % compute stresses on GP
        stressesGP(gp_id,:)=D1*B*DOF.u(DOF.n_mat(i_element,:))*x(i_element)^penal+...
            D2*B*DOF.u(DOF.n_mat(i_element,:))*(1-x(i_element)^penal);
    end
end
```

Automatic concrete cover function

```
% function to create automatic cover around empty elements
% It is needed to do it in separate function because it must be called
% inside the ESO loop and not before it, like other concrete cover cases
%ESO.cover is created based on positions of void determined in the first ESO loop so with x{1}
function [cover] = COVER(ELEM,SET,y,cycle)
switch (SET.covertype)
    % 4-automatic cover around void elements
    case 4
    cover = zeros(ELEM.ny,ELEM.nx);
        %for every element:
        for ii = 1:ELEM.nx
          for jj = 1:ELEM.ny
            %check a close range of surrounding elements with radius = cover distance
            covercheck1=0;
            covercheck2=0;
            for hh = 1:ELEM.nx
                for kk = 1:ELEM.ny
                   %sqrt((kk - center_y)^2+(hh- center_x)^2) < half length
                    if sqrt((kk-jj)^2+(hh-ii)^2) < SET.coverthick +1
                        covercheck1=covercheck1+1; %total max area elements in close range
                        covercheck2=covercheck2+y{cycle}(kk,hh); %total real area in close range
                    end
                end
            end
            %check if there are void elements in close range
            if covercheck1==covercheck2
                cover(jj,ii) = 0;
            else
                cover(jj,ii) = 1;
            end
          end
         end
    % 5-automatic cover around void elements + top and bottom cover
    case 5
       cover = zeros(ELEM.ny,ELEM.nx);
        %automatic cover around void elements
        %for every element:
        for ii = 1:ELEM.nx
          for jj = 1:ELEM.ny
            %check a close range of surrounding elements with radius = cover distance
            covercheck1=0;
            covercheck2=0;
            for hh = 1:ELEM.nx
                for kk = 1:ELEM.ny
                   %sqrt((kk - center_y)^2+(hh- center_x)^2) < half length
                    if sqrt((kk-jj)^2+(hh-ii)^2) < SET.coverthick + 1
                        covercheck1=covercheck1+1; %total max area elements in close range
                        covercheck2=covercheck2+y{cycle}(kk,hh); %total real area in close range
                    end
                end
            end
            %check if there are void elements in close range
            if covercheck1==covercheck2
                cover(jj,ii) = 0;
            else
                cover(jj,ii) = 1;
            end
          end
        end
```

```
    %top and bottom
    for ii = 1:ELEM.nx
      for jj = 1:ELEM.ny
        %max([abs(jj - center_y)/y_scale,abs(ii- center_x)/x_scale] < half length
        if max([abs(jj-1),abs(ii-ELEM.nx/2)/ELEM.nx]) < SET.coverthick
        cover(jj,ii) = 1;
        end
        if max([abs(jj-ELEM.ny),abs(ii-ELEM.nx/2)/ELEM.nx]) < SET.coverthick
        cover(jj,ii) = 1;
        end
      end
    end
% 6-automatic cover around void elements + left and right cover
case 6
   cover = zeros(ELEM.ny,ELEM.nx);
    %automatic cover around void elements
    %for every element:
    for ii = 1:ELEM.nx
      for jj = 1:ELEM.ny
        %check a close range of surrounding elements with radius = cover distance
        covercheck1=0;
        covercheck2=0;
        for hh = 1:ELEM.nx
            for kk = 1:ELEM.ny
               %sqrt((kk - center_y)^2+(hh- center_x)^2) < half length
                if sqrt((kk-jj)^2+(hh-ii)^2) < SET.coverthick + 1
                    covercheck1=covercheck1+1; %total max area elements in close range
                    covercheck2=covercheck2+y{cycle}(kk,hh); %total real area in close range
                end
            end
        end
        %check if there are void elements in close range
        if covercheck1==covercheck2
            cover(jj,ii) = 0;
        else
            cover(jj,ii) = 1;
        end
      end
    end
    %left and right
    for ii = 1:ELEM.nx
      for jj = 1:ELEM.ny
        %max([abs(jj - center_y)/y_scale,abs(ii- center_x)/x_scale] < half length
        if max([abs(jj-ELEM.ny/2)/ELEM.ny,abs(ii-1)]) < SET.coverthick
        cover(jj,ii) = 1;
        end
        if max([abs(jj-ELEM.ny/2)/ELEM.ny,abs(ii-ELEM.nx)]) < SET.coverthick
        cover(jj,ii) = 1;
        end
      end
    end
% 7-automatic cover around void elements + top and bottom + left and right cover
case 7
   cover = zeros(ELEM.ny,ELEM.nx);
    %automatic cover around void elements
    %for every element:
    for ii = 1:ELEM.nx
      for jj = 1:ELEM.ny
        %check a close range of surrounding elements with radius = cover distance
        covercheck1=0;
        covercheck2=0;
```

```
        for hh = 1:ELEM.nx
            for kk = 1:ELEM.ny
               %sqrt((kk - center_y)^2+(hh- center_x)^2) < half length
                if sqrt((kk-jj)^2+(hh-ii)^2) < SET.coverthick + 1
                    covercheck1=covercheck1+1; %total max area elements in close range
                    covercheck2=covercheck2+y{cycle}(kk,hh); %total real area in close range
                end
            end
        end
        %check if there are void elements in close range
        if covercheck1==covercheck2
            cover(jj,ii) = 0;
        else
            cover(jj,ii) = 1;
        end
      end
    end
    %top and bottom
    for ii = 1:ELEM.nx
      for jj = 1:ELEM.ny
        %max([abs(jj - center_y)/y_scale,abs(ii- center_x)/x_scale] < half length
        if max([abs(jj-1),abs(ii-ELEM.nx/2)/ELEM.nx]) < SET.coverthick
        cover(jj,ii) = 1;
        end
        if max([abs(jj-ELEM.ny),abs(ii-ELEM.nx/2)/ELEM.nx]) < SET.coverthick
        cover(jj,ii) = 1;
        end
      end
    end
    %left and right
    for ii = 1:ELEM.nx
      for jj = 1:ELEM.ny
        %max([abs(jj - center_y)/y_scale,abs(ii- center_x)/x_scale] < half length
        if max([abs(jj-ELEM.ny/2)/ELEM.ny,abs(ii-1)]) < SET.coverthick
        cover(jj,ii) = 1;
        end
        if max([abs(jj-ELEM.ny/2)/ELEM.ny,abs(ii-ELEM.nx)]) < SET.coverthick
        cover(jj,ii) = 1;
        end
      end
    end
% 8-automatic cover around void elements + top and bottom + right cover
case 8
   cover = zeros(ELEM.ny,ELEM.nx);
    %automatic cover around void elements
    %for every element:
    for ii = 1:ELEM.nx
      for jj = 1:ELEM.ny
        %check a close range of surrounding elements with radius = cover distance
        covercheck1=0;
        covercheck2=0;
        for hh = 1:ELEM.nx
            for kk = 1:ELEM.ny
               %sqrt((kk - center_y)^2+(hh- center_x)^2) < half length
                if sqrt((kk-jj)^2+(hh-ii)^2) < SET.coverthick + 1
                    covercheck1=covercheck1+1; %total max area elements in close range
                    covercheck2=covercheck2+y{cycle}(kk,hh); %total real area in close range
                end
            end
        end
        %check if there are void elements in close range
```

```
            if covercheck1==covercheck2
                cover(jj,ii) = 0;
            else
                cover(jj,ii) = 1;
            end
          end
        end
        %top and bottom
        for ii = 1:ELEM.nx
          for jj = 1:ELEM.ny
            %max([abs(jj - center_y)/y_scale,abs(ii- center_x)/x_scale] < half length
            if max([abs(jj-1),abs(ii-ELEM.nx/2)/ELEM.nx]) < SET.coverthick
            cover(jj,ii) = 1;
            end
            if max([abs(jj-ELEM.ny),abs(ii-ELEM.nx/2)/ELEM.nx]) < SET.coverthick
            cover(jj,ii) = 1;
            end
          end
        end
        %right
        for ii = 1:ELEM.nx
          for jj = 1:ELEM.ny
            %max([abs(jj - center_y)/y_scale,abs(ii- center_x)/x_scale] < half length
            if max([abs(jj-ELEM.ny/2)/ELEM.ny,abs(ii-ELEM.nx)]) < SET.coverthick
            cover(jj,ii) = 1;
            end
          end
        end
end
```

Plot field (defined on element-level) script

```
% script to plot deformed mesh and a color plot linearly interpolated
% from values at the nodes
% -> component vector must be defined on elements!
% -> select factor=0 to plot undeformed mesh
% -> possible to set colorbar:
% setbar=1 - jet colormap - plot all range of values of field
% setbar=2 - jet colormap - remove stress concentrations (0.5% positive, 0.5% negative)
% setbar=3 - jet colormap -  remove stress concentrations (0.5% positive)
% setbar=4 - jet colormap -  remove stress concentrations (0.5% negative)
% setbar=5 - red/blue colormap - plot all range of values of field
% setbar=6 - red/blue colormap - remove stress concentrations (0.5% positive, 0.5% negative)
% setbar=7 - red/blue colormap -  remove stress concentrations (0.5% positive)
% setbar=8 - red/blue colormap -  remove stress concentrations (0.5% negative)
function plotELEMfield(ELEM,NODE,factor,component,setbar)
X = zeros(4,ELEM.n); UX = zeros(4,ELEM.n);
Y = zeros(4,ELEM.n); UY = zeros(4,ELEM.n);
profile = zeros(4,ELEM.n); %the vector used to assign colors
component = repmat(component,4,1); %information from element is copied to the 4 nodes
for iel=1:ELEM.n            % for (iel)-th element:
    nd=NODE.connect(iel,:); % - extract numbers of connected nodes
    X(:,iel)=NODE.xy(nd,1); % - extract x values of the nodes
    Y(:,iel)=NODE.xy(nd,2); % - extract x values of the nodes
    UX(:,iel)=NODE.ux(nd'); % - extract ux values for the nodes
    UY(:,iel)=NODE.uy(nd'); % - extract uy values for the nodes
    profile(:,iel) = component(:,iel);
end
defoX = X+factor*UX;
defoY = Y+factor*UY;
% plotting the profile of a property on the deformed mesh
    figure
    plot(defoX,defoY,'k')
    colors=fill(defoX,defoY,profile);
    title('');
    axis equal ;
    set(colors,'EdgeColor','none');
%---set-colormap-----------------------------------------------------------
switch(setbar)
    % 1 - jet colormap - plot all range of values of field
    case 1
        colormap(jet(16));
        numpts = 17 ;    % Number of points to be displayed on colorbar
        cmax=max(max(profile));
        cmin=min(min(profile));
        caxis([cmin cmax]);
    % 2 - jet colormap - remove stress concentrations (0.5% positive, 0.5% negative)
    case 2
        colormap(jet(16));
        numpts = 17 ;    % Number of points to be displayed on colorbar
        % exclude highest values of stress concentrations from colorbar and colorplot
        maxnum=round(ELEM.n*0.005); %number of highest value to skip (0.5%)
        minnum=round(ELEM.n*0.005); %number of highest value to skip (0.5%)
        profileclear = sort(profile(~isnan(profile))); %remove nan from vector and sort
        cmax=max(max(profile))
        cmax2 = profileclear(end-maxnum+1)
        cmin=min(min(profile))
        cmin2 = profileclear(minnum)
        caxis([cmin2 cmax2]);
    % 3 - jet colormap -  remove stress concentrations (0.5% positive)
    case 3
```

```
    colormap(jet(16));
    numpts = 17 ;    % Number of points to be displayed on colorbar
    % exclude highest values of stress concentrations from colorbar and colorplot
    maxnum=round(ELEM.n*0.005); %number of highest value to skip (0.5%)
    profileclear = sort(profile(~isnan(profile))); %remove nan from vector and sort
    cmax=max(max(profile))
    cmax2 = profileclear(end-maxnum+1)
    cmin=min(min(profile));
    caxis([cmin cmax2]);
% 4 - jet colormap -  remove stress concentrations (0.5% negative)
case 4
    colormap(jet(16));
    numpts = 17 ;    % Number of points to be displayed on colorbar
    % exclude highest values of stress concentrations from colorbar and colorplot
    minnum=round(ELEM.n*0.005); %number of highest value to skip (0.5%)
    profileclear = sort(profile(~isnan(profile))); %remove nan from vector and sort
    cmax=max(max(profile));
    cmin=min(min(profile))
    cmin2 = profileclear(minnum)
    caxis([cmin2 cmax]);
% 5 - red/blue colormap - plot all range of values of field
case 5
    numpts = 20 ;    % Number of points to be displayed on colorbar
    cmax=max(max(profile));
    cmin=min(min(profile));
    caxis([cmin cmax]);
    s=sign(profile);
    n_pos=sum(s(:)==1)/4;
    n_neg=sum(s(:)==-1)/4;
    i1val=0;
    i2val=0;
    c_pos=round(abs(cmax)/(abs(cmax)+abs(cmin))*ELEM.n);
    c_neg=round(abs(cmin)/(abs(cmax)+abs(cmin))*ELEM.n);
    redmap=zeros(c_pos,3);
    bluemap=zeros(c_neg,3);
    for i1=1:c_pos
        i1val=i1val+1;
        redmap(i1,1)=0.9;
        redmap(i1,2)=i1val/c_pos*0.9;
        redmap(i1,3)=i1val/c_pos*0.9;
    end
    for i2=1:c_neg
        i2val=i2val+1;
        bluemap(i2,1)=i2val/c_neg*0.9;
        bluemap(i2,2)=i2val/c_neg*0.9;
        bluemap(i2,3)=0.9;
    end
    redmapflip = flipud(redmap);
    redplusbluemap=cat(1,bluemap,redmapflip);
    colormap(redplusbluemap);
% 6 - red/blue colormap - remove stress concentrations (0.5% positive, 0.5% negative)
case 6
    numpts = 20 ;    % Number of points to be displayed on colorbar
    % exclude highest values of stress concentrations from colorbar and colorplot
    maxnum=round(ELEM.n*0.005); %number of highest value to skip (0.5%)
    minnum=round(ELEM.n*0.005); %number of highest value to skip (0.5%)
    profileclear = sort(profile(~isnan(profile))); %remove nan from vector and sort
    cmax=max(max(profile))
    cmax2 = profileclear(end-maxnum+1)
    cmin=min(min(profile))
    cmin2 = profileclear(minnum)
```

```
    caxis([cmin2 cmax2]);
    s=sign(profile);
    n_pos=sum(s(:)==1)/4;
    n_neg=sum(s(:)==-1)/4;
    i1val=0;
    i2val=0;
    c_pos=round(abs(cmax2)/(abs(cmax2)+abs(cmin2))*ELEM.n);
    c_neg=round(abs(cmin2)/(abs(cmax2)+abs(cmin2))*ELEM.n);
    redmap=zeros(c_pos,3);
    bluemap=zeros(c_neg,3);
    for i1=1:c_pos
        i1val=i1val+1;
        redmap(i1,1)=0.9;
        redmap(i1,2)=i1val/c_pos*0.9;
        redmap(i1,3)=i1val/c_pos*0.9;
    end
    for i2=1:c_neg
        i2val=i2val+1;
        bluemap(i2,1)=i2val/c_neg*0.9;
        bluemap(i2,2)=i2val/c_neg*0.9;
        bluemap(i2,3)=0.9;
    end
    redmapflip = flipud(redmap);
    redplusbluemap=cat(1,bluemap,redmapflip);
    colormap(redplusbluemap);
% 7 - red/blue colormap -  remove stress concentrations (0.5% positive)
    case 7
    numpts = 20 ;    % Number of points to be displayed on colorbar
    % exclude highest values of stress concentrations from colorbar and colorplot
    maxnum=round(ELEM.n*0.005); %number of highest value to skip (0.5%)
    profileclear = sort(profile(~isnan(profile))); %remove nan from vector and sort
    cmax=max(max(profile))
    cmax2 = profileclear(end-maxnum+1)
    cmin=min(min(profile));
    caxis([cmin cmax2]);
    s=sign(profile);
    n_pos=sum(s(:)==1)/4;
    n_neg=sum(s(:)==-1)/4;
    i1val=0;
    i2val=0;
    c_pos=round(abs(cmax2)/(abs(cmax2)+abs(cmin))*ELEM.n);
    c_neg=round(abs(cmin)/(abs(cmax2)+abs(cmin))*ELEM.n);
    redmap=zeros(c_pos,3);
    bluemap=zeros(c_neg,3);
    for i1=1:c_pos
        i1val=i1val+1;
        redmap(i1,1)=0.9;
        redmap(i1,2)=i1val/c_pos*0.9;
        redmap(i1,3)=i1val/c_pos*0.9;
    end
    for i2=1:c_neg
        i2val=i2val+1;
        bluemap(i2,1)=i2val/c_neg*0.9;
        bluemap(i2,2)=i2val/c_neg*0.9;
        bluemap(i2,3)=0.9;
    end
    redmapflip = flipud(redmap);
    redplusbluemap=cat(1,bluemap,redmapflip);
    colormap(redplusbluemap);
% 8 - red/blue colormap -  remove stress concentrations (0.5% negative)
    case 8
```

```
        numpts = 20 ;    % Number of points to be displayed on colorbar
        % exclude highest values of stress concentrations from colorbar and colorplot
        minnum=round(ELEM.n*0.005); %number of highest value to skip (0.5%)
        profileclear = sort(profile(~isnan(profile))); %remove nan from vector and sort
        cmax=max(max(profile));
        cmin=min(min(profile))
        cmin2 = profileclear(minnum)
        caxis([cmin2 cmax]);
        s=sign(profile);
        n_pos=sum(s(:)==1)/4;
        n_neg=sum(s(:)==-1)/4;
        i1val=0;
        i2val=0;
        c_pos=round(abs(cmax)/(abs(cmax)+abs(cmin2))*ELEM.n);
        c_neg=round(abs(cmin2)/(abs(cmax)+abs(cmin2))*ELEM.n);
        redmap=zeros(c_pos,3);
        bluemap=zeros(c_neg,3);
        for i1=1:c_pos
            i1val=i1val+1;
            redmap(i1,1)=0.9;
            redmap(i1,2)=i1val/c_pos*0.9;
            redmap(i1,3)=i1val/c_pos*0.9;
        end
        for i2=1:c_neg
            i2val=i2val+1;
            bluemap(i2,1)=i2val/c_neg*0.9;
            bluemap(i2,2)=i2val/c_neg*0.9;
            bluemap(i2,3)=0.9;
        end
        redmapflip = flipud(redmap);
        redplusbluemap=cat(1,bluemap,redmapflip);
        colormap(redplusbluemap);
end
%---colorbar-setting-----------------------------------------------------------
cbar = colorbar;
% title of the colorbar
set(get(cbar,'title'),'string','');
% setting the values on colorbar
% get the color limits
clim = caxis;
ylim(cbar,[clim(1) clim(2)]);
int = linspace(clim(1),clim(2),numpts);
set(cbar,'YtickMode','manual','YTick',int); % Set the tickmode to manual
for i = 1:numpts
    imep = num2str(int(i),'%+3.2E');
    vasu(i) = {imep} ;
end
set(cbar,'YTickLabel',vasu(1:numpts),'fontsize',9);
%---other---------------------------------------------------------------------
%{
% uncomment to plot nodes and elements numbers
for i1 = 1:ELEM.n
  for i2 = 1:4
    text(defoX(i2,i1),defoY(i2,i1), int2str(NODE.connect(i1,i2)),...
                'fontsize',8,'color','b','HorizontalAlignment','center');
  end
    text(sum(defoX(:,i1))/4,sum(defoY(:,i1))/4,int2str(i1),...
                'fontsize',10,'color','k','HorizontalAlignment','center') ;
end
%}
```

Bibliography

Books

[1] Bendsoe M.P., Sigmund O., *"Topology Optimization: Theory, Methods and Applications"*, Springer-Verlag Berlin Heidelberg, 2003.

[2] Steven G.P., Xie Y.M., *"Evolutionary Structural Optimization"*, Springer-Verlag London Limited, 1997.

[3] Steven G.P., Xie Y.M., *"Evolutionary Topology Optimization of Continuum Structures: Methods and Applications"*, John Wiley & Sons, 2010.

[4] Liang Q.Q., *"Performance-Based Optimization of Structures: Theory and Applications"*, Spon Press, 2005.

[5] Sasaki M., *"Flux Structure"*, Toto Publishers, 2005.

Articles

[6] Steven G. P., Xie Y.M., *"A Simple Evolutionary Procedure for Structural Optimization"*, Computers & Structures, Volume 49, Issue 5, 885-401, 1993.

[7] Sigmund, O., *"A 99 line topology optimization code written in Matlab"*, Structural and Multidisciplinary Optimization, Volume 21, Issue 2, 120-127, 2001.

[8] Andreassen E., Clausen A., Schevenels M., Lazarov B. S., Sigmund O., *"Efficient topology optimization in MATLAB using 88 lines of code"*, Structural and Multidisciplinary Optimization, Volume 43, Issue 1, 1-16, 2011.

[9] Huang X., Xie Y.M., *"Convergent and mesh-independent solutions for the bi-directional evolutionary structural optimization method"* Finite Elements in Analysis and Design, Volume 43, Issue 14 1039-1049, 2007.

[10] Huang X., Xie Y.M., *"Bi-directional evolutionary topology optimization of continuum structures with one or multiple materials"*, Computational Mechanics, Volume 43, 393-401, 2009.

[11] Huang X., Xia L., Xia Q., Xie Y.M., *"Bi-directional Evolutionary Structural Optimization on Advanced Structures and Materials: A Comprehensive Review"*, Archives of Computational Methods in Engineering, 1-42, 2016.

[12] Guan H., Steven G.P., Xie Y.M., *"Evolutionary structural optimization incorporating tension and compression materials"*, Advances in Structural Engineering, Volume 2, Issue 4, 273-288, 1999.

[13] Proos K.A., Querin O.M., Steven G.P., Xie Y.M., *"Stiffness and inertia multicriteria evolutionary structural optimization"*, Engineering Computations, Volume 18, Issue 7, 1031-1054, 2001.

[14] Proos K.A., Querin O.M., Steven G.P., Xie Y.M., *"Multicriterion Evolutionary Structural Optimization Using the Weighting and the Global Criterion Methods"*, American Institute of Aeronautics and Astronautics Journal, Volume 39, Issue 10, 2006-2012, 2001.

[15] Li Q., Steven G.P., Xie Y.M., *"Multicriteria optimization that minimizes maximum stress and maximizes stiffness"*, Computers & Structures, Volume 80, 2433-2448, 2002.

[16] Kang Z., Luo Y., *"Topology optimization of continuum structures with Drucker-Prager yield stress constraints"*, Computers & Structures, Volume 90, Issue 91, 65-75, 2012.

[17] Burry M.C., Felicetti P., Tang J.W., Xie Y.M., *"Form finding for complex structures using evolutionary structural optimization method"*, Design Studies, Volume 26, Issue 1, 55-72, 2005.

[18] Huang X., Xie Y.M., *"A further review of ESO type methods for topology optimization"*, Structural and Multidisciplinary Optimization, Volume 41, Issue 5, 671-683, 2010.

[19] Huang X., Xie Y.M., *"Evolutionary topology optimization of geometrically and materially nonlinear structures"*, Structural Engineering and Mechanics, Volume 34, Issue 5, 581-595, 2010.

[20] Liang Q.Q., Steven G.P., Xie Y.M., *"Topology Optimization of Strut-and-Tie Models in Reinforced Concrete Structures Using an Evolutionary Procedure"*, ACI Structural Journal, Volume 97, Issue 36, 322-332, 2000.

[21] Huang X., Xie Y.M., *"Evolutionary topology optimization of continuum structures including design-dependent self-weight loads"*, Finite Elements in Analysis and Design, Volume 47, Issue 8, 942-948, 2011.

[22] Banachowicz M., Januszkiewicz K., *"Nonlinear Shaping Architecture Designed with Using Evolutionary Structural Optimization Tools"*, IOP Conference Series: Materials Science and Engineering, Volume 245, Article 082042, 2017.

[23] Ohmori H., *"Computational Morphogenesis - Its Current State and Possibility for the Future"*, Proceedings of the 6th International Conference on Computation of Shell and Spatial Structures, IASS-IACM 2008: "Spanning Nano to Mega", Cornell University, Ithaca, NY, USA, 28-31 May 2008.

[24] Ohmori H., Futai H., Iijima T., Muto A., Hasegawa Y., *"Computational Morphogenesis and its Application to Structural Design, "*, Proceedings of International Symposium on Shell and Spatial Structures, Theory, Technique, Valuation, Maintenance, Bucharest Poiana Brasov, Romania, 13-20, 2005.

[25] Ohmori H., Futai H., Iijima T., Muto A., Hasegawa Y., *"Structural Design of Office Building by Computational Morphogenesis"*, AIJ Journal of Technology and Design Volume 10, Issue 20, 77-82, 2004.

[26] Cui C., Ohmori H., Sasaki M., *"Computational Morphogenesis of 3D Structures by Extended ESO Method"*, Journal of the International Association for Shell and Spatial Structures (IASS) Volume 44, Issue 1, 51-61, 2003.

Internet resources

[27] Berg K., *"Optimoinnin historiaa"*, Aalto University - Courses materials, `http://salserver.org.aalto.fi/vanhat_sivut/Opinnot/Mat-2.3139/`

[28] Ohmori H., *"Computational Morphogenesis - Plenary lecture at IASS-IACM2008 Cornell University, Ithaca, NY, USA 28-31 May 2008"*, Nagoya University - Courses materials, `http://ocw.nagoya-u.jp/files/399/Slide_IASS-IACM_2008.pdf`

[29] Qatar National Convention Centre, *"QNCC Exterior at Dusk"*, `http://www.qncc.qa/about-qncc/gallery`

[30] Arata Isozaki & Associates, *"Qatar National Convention Center, Doha, Qatar, 2004"*, `http://www.isozaki.co.jp/`