




Investigating best algorithms for structural topology optimization

Sohayb Abdulkerim*¹ 

¹ Gaziantep University, Department of Aerospace Engineering, Türkiye, karim@gantep.edu.tr

Cite this study: Abdulkerim, S. (2024). Investigating best algorithms for structural topology optimization. Turkish Journal of Engineering, 8 (1), 116-126

Keywords

Interior Point Method
Topology Optimization
Finite Element
Plane Stress

Research Article

DOI: 10.31127/tuje.1298508

Received:17.05.2023

Revised: 19.06.2023

Accepted:21.06.2023

Published:03.01.2024



Abstract

This study investigates the topology optimization problem using various optimization approaches, taking inspiration from the 99-line MATLAB code developed by Sigmund. The educational MATLAB code is based on the Solid Isotropic Material with Penalization (SIMP) model of the artificial material density method. The objective is to minimize the compliance function with a weight constraint, with the design variables being the densities of all elements. The aim is to identify a more efficient optimization technique as an alternative to the commonly used optimality criteria algorithm provided by other MATLAB built-in tools. Two types of optimization algorithms are examined: gradient-based methods such as Interior-Point, Sequential Quadratic Programming (SQP), and Active-Set, as well as metaheuristic methods including the Genetic Algorithm. The results are verified and validated by comparing them with existing literature, demonstrating good agreement. Performance assessments are conducted to compare the results obtained from these algorithms in terms of quality and computational efficiency. The numerical findings indicate that the interior-point method outperforms the other investigated methods, although the optimality criteria algorithm remains the most efficient for solving topology optimization problems.

1. Introduction

Over the past two decades, the topology optimization problem has been investigated by many researchers. The purpose of the optimization, in general, is to find the optimum layout or the optimum distribution of the material to meet the design requirements, with minimum weight and cost. Therefore, research centers and industrial sectors are increasingly attracted to implementing this technique because it helps to find competitive constructions in various fields such as bridges, cars, and aerospace structures. Topology optimization expanded in several disciplines, including a combination of structures, acoustic, fluid flow, heat transfer, material design, and aero-elasticity [1].

The topology optimization aims at minimizing the structure objective function within constraints.

However, the type of the objective function and the constraints depends greatly on the type of the problem that needs solved. More specifically, in some applications, the weight is the most crucial parameter; therefore, the objective function focuses on minimizing the compliance within weight limits [2].

While in vibration problems, avoiding resonance is the most important problem. The objective function, in this case, is to adjust the natural frequency of the structure within an acceptable range [1].

Accordingly, in fixed-wing air vehicles, the wing flutter problem is always of big interest to aerospace designers. Increasing the vibration modes separation is another type of objective function. Constraints could be varied depending on the type of problem such as weight, strength, or heat [3].

Similarly, to the objective functions, constraints, and the boundary of the design variables; the optimization numerical approach is an important issue affecting the computational efficiency and the quality of the results.

Optimality Criteria (OC) is a classical optimization tool for solving topology problems, It was first used by the Australian Michel in 1904 [4]. Bendsoe and Sigmund offer a definitive review of the topology optimization and OC problem for newcomers to the topic [5].

Besides OC, so many different types of approaches are found in the literature. However, this paper is not focused on reviewing the topology optimization as many

works in this field have done, such as review papers by [2,4,6-9].

In brief, Rozvany [4], compared numerical methods of structural topology between Evolutionary Structural Optimization (ESO), (or Sequential Element Rejections and Admissions) and SIMP. He found that the latter is a rigorously derived gradient method and requires fewer iterations. Furthermore, there is a wide range of applications.

In addition, he reported that the disadvantage of SIMP is that the global optimum is not always found. While some recently published papers that focused on the ESO method found significant fundamental flaws, fully heuristic behavior, and computational inefficiency [4]. A different categorization of the topology optimization problem was made by Eschenauer et al. [7]. They included 425 references in their review and divided the topology problem into two different kinds of topology, namely the material or microstructure technique and the geometrical or macrostructure technique.

Rozvany [8] found that the OC is the oldest technique and most popular method. However, the disadvantage of OC is that may not yield a minimum weight design even for simple stress constraints. He examined and discussed alternative methods such as "hard-kill" methods, ESO, or the Adaptive Biological Growth (ABG) method as well as Generalized Stress Design (GSD) method [8].

He added that methods that rely on minimizing compliance may lead to a locally optimal solution. While Zero-order methods such as ESO are computationally efficient if a rapid design improvement is needed without necessarily finding the best solution [8]. Differently, the optimization problem methods were categorized into two types, analytical and numerical methods, by Zargham et al. [2] in their paper review. The numerical methods were also classified into three sub-classifications: direct methods, such as mathematical programming; indirect methods, such as optimality criteria; and lastly metaheuristic methods, such as Genetic Algorithms (GA). They compared various algorithms in vibration problems and discussed the performance of algorithms such as the Global Convergent Method of Moving Asymptotes (GCMMA) algorithm, which was employed simultaneously to optimize for static loads and random excitations. In addition, the level-set method, was first used in 2005, showed promise for future applications. The most popular method applied was the SIMP. Lastly, they found that GA is easy to use but computationally inefficient [2]. This agrees with the conclusion of Hajel and Lee 1995 [4].

GA was also implemented by Cardillo et al. [10] to solve multi-objective topology optimization using hybridization of partial solutions and also was compared with the OC algorithm. It is reported that optimization based on OC is more efficient than GA from a computational point of view in spite of the fact that GA has a higher capability in finding the global optimal solution.

Literature regarding topology optimization, highlighting the development from 2000 to 2012, was surveyed by Joshua et al. [1]. Who also divided the problem into four categories as follows [1]: density-based method (SIMP), Hard-kill methods evolutionary

ESO, Boundary variation methods (Level set and phase field), and then finally a new biologically inspired method based on cellular division rules.

Sivapuram and Picelli [11] demonstrated the possibility of using the Integer Linear Programming (ILB) method to solve the topology optimization problems using binary variables, the constraints, and objective functions are linearized using Taylor's first-order approximation.

In the same subject, Hassani and Hinton [6] focused their review on topology optimization homogenization theory.

Educationally, Challis et al. [12] introduced a simple MATLAB code, 129 lines, inspired by the educational paper 99-line-code work by Sigmund [13]. The code implements the level set method. The educational report 88 line code by Andreassen et al. [14] which also was inspired by the 99-line code by Sigmund [13] where the improvement was made by pre-allocating arrays and vectoring loops. A Benchmark case study for 7500 elements was introduced. In their code, the Heaviside filter was used.

Talischi et al. [12] used a MATLAB tool to solve topology optimization. The finite element mesh was established using unstructured polygonal finite element meshes.

In order to reduce the computational time, Parallel programming was used by several researchers and found an efficient solution [1].

Wang et al. [15] used an enhanced the Genetic Algorithm (GA) to solve topology optimization using discrete variable density void/solid elements. The enhancement depended on the knowledge of the topology optimization problem. In order to solve element connectivity problems, an image-processing-based connectivity analysis was developed and implemented.

Wang et al. [15] also introduced a comparison between the enhanced GA with the SIMP method. They concluded that GA generates better solutions but with higher computational costs.

Another optimization method is the Conservative Convex Separable Approximations (CCSA). Introduced by Svanberg [16], it is capable of solving nonlinear inequality constraints. It was applied to a very large number of design variables. Talischi et al. [12] implemented a MATLAB code to solve topology optimization using unstructured polygonal finite element meshes. Technical Description of the Method of Moving Asymptotes (MMA) first described and implemented by [17]. The SIMP method is first introduced by [5,18]. Several different approaches such as OC methods, Sequential Linear Programming (SLP) methods or the MMA by Svanberg [17] and others.

It was found that involving methods, such as Interior-Point, Sequential Quadratic Programming, SQP, or Active-Set method, as an optimization scheme is possible to solve topology optimization.

Based on the literature review, it is worthwhile to test the efficiency and compare these methods in solving the topology optimization problem. Therefore, in the first section, a statement of the standard topology optimization problem will be defined. Then, a brief definition of the used scheme is introduced. Definitions

of two case studies found in the literature will be illustrated with graphical results. The next chapter presents the numerical analysis of those two cases using the tested schemes coded by MATLAB and followed by discussions and conclusions.

2. Statement of the standard topology optimization problem:

Consider domain structure modeled using the well-known finite element method as shown in Equation 1:

$$\{F\} = [K]\{U\} \tag{1}$$

where, $[K]$ is the global stiffness matrix, $\{U\}$ is nodal displacement vector, $\{F\}$ is nodal applied forces. After applying the boundary conditions, the displacement vector can be calculated as shown in Equation 2:

$$\{U\} = [K]^{-1}\{F\} \tag{2}$$

Assuming x_e is the density of the element (e), and then the compliance function $C(x)$ as a function of the densities penalized by the power law can be calculated as shown in Equation 3:

$$C(x) = \sum_{e=1}^N x_e^p U_e^T k_e U_e \tag{3}$$

where, N is the total number of elements, p is the penalty parameter, and equal to 3 as a typical value used by others, U_e is the nodal displacements vector of the element (e), k_e is the local stiffness matrix of the element (e).

The total volume of the structure as a function of the artificial densities of the elements can be calculated as shown in Equation 4:

$$V(x) = \sum_{e=1}^N x_e \cdot v_e \tag{4}$$

Thus, the standard topology optimization using the SIMP method can be determined as shown in Equation 5:

$$\left[\begin{array}{l} \text{minimize } C(x) \\ \text{subject to } \left\{ \begin{array}{l} \sum_{e=1}^N x_e \cdot v_e \geq fV_o \\ LB \leq X \leq UB \end{array} \right. \end{array} \right] \tag{5}$$

where, V_o is the base volume, f the fraction of the weight (volumetric fraction), LB and UB are the lower and upper bound of the densities respectively; LB values are normally taken as 0.003 instead of zero values in order to avoid singularity in the calculation of the global stiffness matrix while UB values are unities. Obviously, the objective function is nonlinear. While the constraints are linear inequalities.

The gradient function can be calculated as shown in Equation 6:

$$\frac{\partial C(x)}{\partial x} = \sum_{e=1}^N p \cdot x_e^{p-1} U_e^T k_e U_e \tag{6}$$

Some schemes can perform better if the hessian function is provided, which can be derived from the gradient function by derivative respect to density variables as shown in Equation 7:

$$\frac{\partial^2 C(x)}{\partial x^2} = \sum_{e=1}^N (p - 1) \cdot x_e^{p-2} U_e^T k_e U_e \tag{7}$$

3. Optimization algorithms

In order to compare the efficiency of different optimization schemes, this study investigated two categories of optimizers: gradient-based algorithms and heuristic methods. The first category, gradient-based algorithms, includes 'Interior-point', 'SQP' (Sequential Quadratic Programming), 'SQP-legacy', and 'active-set'. These algorithms utilize gradient information to guide the optimization process towards the optimal solution. On the other hand, the second category comprises heuristic methods or non-gradient-based approaches, with the genetic algorithm being one of the prominent examples. Heuristic methods employ techniques inspired by natural processes or behaviors to explore the solution space without relying on explicit gradient information. By examining both gradient-based and heuristic methods, this study aims to provide insights into their respective strengths and weaknesses in solving topology optimization problems.

It is important to give a brief description of these schemes. The 'Interior-point' algorithm is an optimization algorithm used to solve nonlinear convex problems that are subject to nonlinear inequality constraints. It was first utilized by John von Neumann in 1948 for linear programming [19]. During the 1960s, it gained popularity for solving nonlinear constrained objective functions and was pioneered by Fiacco and McCormick [19]. Over the years, from 1979 to 2003, the method underwent significant development and became the most efficient and reliable method for solving large-scale problems. The 'Interior-point' algorithm is widely available in software packages such as MATLAB, among others. Notably, it has demonstrated high efficiency in handling spar matrix problems [20–22].

To enable the implementation of various optimization algorithms, a general MATLAB subroutine was developed. The subroutine is structured as follows:

```
function [X,FVAL,EXITFLAG,OUTPUT,LAMBDA,GRAD,
elapsedTime]=runfmincon(strMethod)
global nelx nely volfrac
% linear constraints
A = []; b = [];
Aeq = []; beq = [];
% initial guess
```

```

x0(1:nely*nex) = volfrac;
lb(1:nely*nex)=0.001;
ub(1:nely*nex)=1.0;
% nonlinear constraints
nonlincon = @nlcon;
options =
optimoptions('fmincon','Algorithm','strMethod ...
    , 'ConstraintTolerance',1e-6 ...
    , 'StepTolerance',1e-6 ...
    , 'FunctionTolerance',1e-6 ...
    , 'OptimalityTolerance',1e-10 ...
    , 'MaxFunctionEvaluations',1000000 ...
    , 'MaxIterations',1000 ...
    , 'SpecifyObjectiveGradient',false ...
    , 'SpecifyConstraintGradient',false ...
);

tic;
% Call fmincon to run the optimization using the specified
algorithm
[X,FVAL,EXITFLAG,OUTPUT,LAMBDA,GRAD,
elapsedTime] = ...

fmincon(@objectiveMAC,x0,A,b,Aeq,beq,lb,ub,nonlincon,o
ptions);
elapsedTime= toc;
x = reshape(X,[nely,nex]);
end

```

The code provides a general MATLAB subroutine for implementing the optimization algorithms. The subroutine sets up the necessary constraints, initializes the variables, and defines the options for the optimization algorithm using `optimoptions`. The `fmincon` function is then called with the objective function `objectiveMAC`, the initial guess `x0`, and other parameters. Finally, the solution is reshaped into a matrix `x` based on the dimensions specified by `nely` and `nex`.

To use the routine, the value of the `strMethod` parameter has to be set to one of the available options: 'Interior-Point', 'sqp', 'sqp-Legacy', or 'Active-Set', depending on the desired optimization method. For example; the following two lines:

```

strMethod='SQP';
[X,FVAL,EXITFLAG,OUTPUT,LAMBDA,GRAD]=runfmincon
(strMethod);

```

is used to utilize the SQP algorithm. By incorporating those lines of code, users can harness the capabilities of those algorithms to optimize their objective functions, subject to the specified constraints. The subroutine encapsulates the necessary functionalities, including constraint handling and option settings, to facilitate the application of the SQP algorithm in MATLAB.

3.1. Sequential quadratic programming (SQP)

Sequential Quadratic Programming (SQP) is a popular optimization algorithm widely used to solve optimization problems where the objective function needs to be minimized or maximized, subject to

nonlinear inequality constraints. It is considered one of the most effective approaches in optimization due to its ability to handle a wide range of problems efficiently [22-24]. The SQP algorithm operates by iteratively solving a series of quadratic sub-problems that approximate the original nonlinear optimization problem. At each iteration, the algorithm constructs a quadratic model of the objective function and the nonlinear constraints around the current iterate. The quadratic model is then minimized or maximized to obtain a new iterate, which is expected to improve the objective function value while satisfying the constraints [22-24].

The quadratic sub-problems in SQP involve solving a quadratic programming (QP) sub-problem at each iteration, which is a mathematical programming problem with a quadratic objective function and linear constraints. These QP sub-problems are typically easier to solve compared to the original nonlinear problem, allowing for efficient convergence towards the optimal solution [22-24].

One of the advantages of SQP is its ability to handle both small and large-scale optimization problems. It is particularly suitable for problems with nonlinear inequality constraints, where it can effectively handle the nonlinearity and provide feasible solutions that satisfy the constraints [22-24].

Overall, SQP is a powerful optimization algorithm that combines the benefits of quadratic programming and sequential approximation techniques. Its iterative nature, quadratic model generation, and effective handling of nonlinear constraints make it a widely used method for solving optimization problems in various fields, including engineering, economics, finance, and operations research [22-24]. A detailed explanation of the SQP algorithm can be found in the textbook by Fletcher [23,24].

3.2. SQP-Legacy

SQP-Legacy is a variant of the Sequential Quadratic Programming (SQP) algorithm. It shares similarities with the SQP algorithm in terms of its approach and methodology. However, SQP-Legacy is typically characterized by longer computation times and higher memory requirements compared to the standard SQP algorithm. Despite these considerations, SQP-Legacy can still be a valuable optimization approach in certain scenarios [25]. For example, it may be preferred when compatibility with legacy code or systems is a priority. Additionally, in cases where the problem size is relatively small or the computational resources are not a major constraint, SQP-Legacy can provide satisfactory results [25].

It's important to note that SQP-Legacy should be used judiciously, considering the specific requirements and constraints of the optimization problem at hand. If computational efficiency and faster convergence are critical, it may be worthwhile to explore alternative optimization algorithms or more recent versions of SQP. Nonetheless, SQP-Legacy remains a viable option that can be employed effectively in appropriate situations, leveraging its established methodology and capabilities

for solving optimization problems with nonlinear inequality constraints [25].

3.3. Active-Set

Active-Set is an optimization algorithm used to solve the optimization problem within inequality constraints. If an objective function is subject to inequality constraints, $L_i(x) \geq 0$, given a point x in the solution space, if any constraint satisfies an equality, $L_i(x) = 0$, it is called active at x . Thus the solution can be developed as follows: first solve the equality constraint defined by the active-set, then compute the Lagrange multipliers, then constraints with negative Lagrange multipliers are removed, then search for infeasible constraints, repeat the previous steps until enough approaching the final solution with negligible change [21].

The Active-Set algorithm effectively combines the concepts of feasibility and optimality to find the optimal solution within the inequality constraints. By iteratively adjusting the active set based on the current point's active constraints and Lagrange multipliers, the algorithm seeks to optimize the objective function while satisfying the inequality constraints [21]. The textbook Numerical Optimization by Jorge Nocedal and Stephen J. Wright 2006 explained this technique in more details [21].

3.4. Genetic Algorithm

The Genetic Algorithm (GA) is a heuristic optimization algorithm based on the principles of evolutionary theory, specifically the concepts of natural selection and genetic inheritance [26]. In this method, a population of random solutions, often referred to as individuals, is generated and evaluated based on their fitness or objective function value. Each individual represents a potential solution to the optimization problem [26].

The GA mimics the process of biological evolution by allowing the individuals to evolve and reproduce through a selection process. This selection process involves competition among individuals, where the fittest individuals have a higher chance of reproducing and passing on their genetic information to the next generation. This concept is commonly referred to as the "survival of the fittest." [26].

The Genetic Algorithm was first introduced by John Holland in 1970 and has since become a widely used optimization technique. In the book "Genetic Algorithm" by Kramer [27], detailed explanations of the algorithm's principles and implementation are provided [26].

In this work, GA programming is tested using the MATLAB built-in function, gamultiobj. This tool allows for the optimization of a user-defined objective function. Both nonlinear equality and non-equality constraints are allowed. Additionally, upper, and lower bounds are possible to set for the state variables.

4. Code Validation

4.1. Case Study 1

For comparison with the existing literature, two case studies were conducted in this report. The first case study, originally reported by Sigmund [13], involved a cantilever beam modeled using finite element plane stress analysis. The beam was clamped from the left side, as shown in Figure 1. The mesh used for this analysis consisted of 32 elements horizontally and 20 elements vertically. In Sigmund's study, a volume fraction of 0.4, a power penalty of 3, and a filter radius size of 1.2 were employed. The optimization algorithm used in this case study was coded in MATLAB following Sigmund's 99-line code. The results obtained from this code are presented in Figure 1.

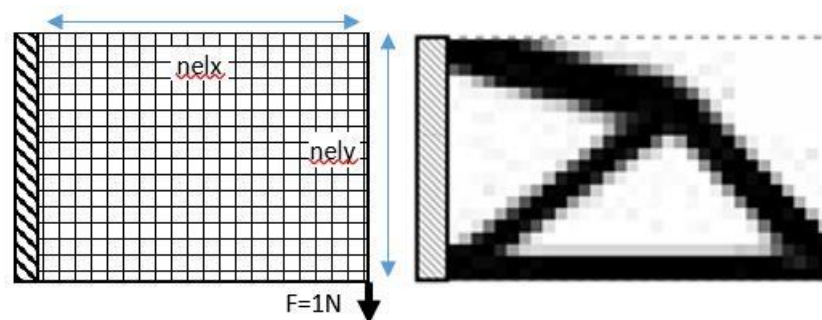


Figure 1. Case study 1, the topology optimization results of a cantilever beam and concentrated load by Sigmund [10]

Additionally, Sigmund [13], suggested other possible optimization methods such as Sequential Linear Programming (SLP) and the Method of Moving Asymptotes (MMA). However, in this work, a different family of optimization algorithms was implemented using the MATLAB package and the 'fmincon' function. These algorithms are gradient-based schemes and have been briefly described in the introduction chapter. The purpose of using these algorithms was to verify their

performance and identify the most efficient schemes for the two case studies conducted in this report.

Aguilar et al. [28] conducted a similar case study as depicted in Figure 2. The study involved a plate with dimensions of 40 x 30 cm and a meshing density of 32x24 using isoperimetric plane stress elements. The optimization method employed in their study was the Genetic Algorithm (GA). The results obtained using the GA for this case study are also presented in Figure 2.

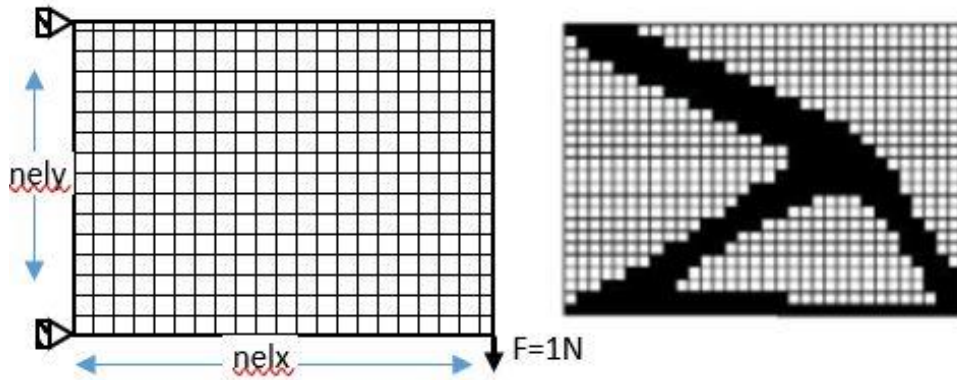


Figure 2. Case study 1 for the beam that introduced by Aguilar et al. [25], [26].

4.2. Case Study 2

The second case study, as presented by Sigmund [13] and Andreassen et al. [14] is illustrated in Figure 3. The figure displays the right-hand half of a simply supported beam subjected to a downward load at the top-central point. Due to the symmetry around the vertical axis at the

central line, only one half of the beam was analyzed to obtain the results. The density distributions depicted in the figure were generated using the following parameters: a mesh with 60 elements, consisting of 20 elements in the horizontal direction and 20 elements in the vertical direction; a filter radius of $r_{min} = 1.5$; and the volumetric ratio was 0.5.

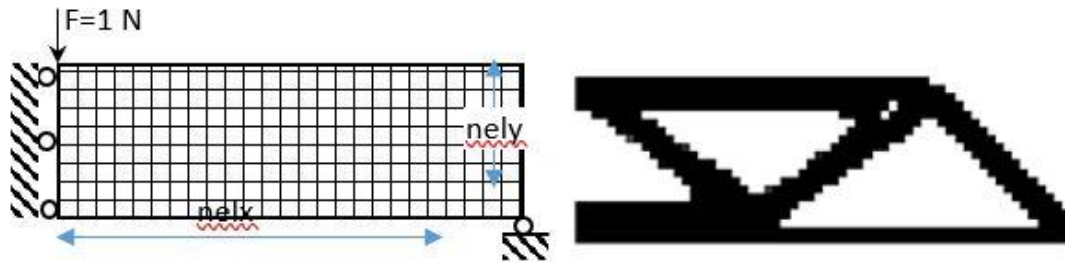


Figure 3. Case study 2, MATLAB 88-lines code case study results by Andreassen et al. [11] and by Sigmund [10].

5. Results

The numerical results of the two case studies were replicated in this study using six different algorithms implemented in MATLAB code, as described in detail in Appendix A. The computational costs, measured in terms of computation elapsed time, were compared to assess the quality of the results. The quality of the results was evaluated using grayscale graphs, where higher density regions were represented by black color.

For the Genetic Algorithm (GA), the following parameter values were chosen: a Crossover Fraction of 0.8000, a Population Size of 200, and Uniform Mutation with a value of 0.0500. The fitness function utilized in the GA was the same as the one employed in all the other algorithms. It is worth noting that, unlike other routines, this particular routine does not require a gradient function.

5.1. Case Study 1

The results of Case Study 1, which focused on a beam modeled using the finite element method with plane stress state, are presented in Figure 4. The beam was subjected to specific boundary conditions, where all nodes along the left-hand edges were assigned constrained degrees of freedom. Furthermore, a unity vertical force was applied to the bottom right node to induce loading.

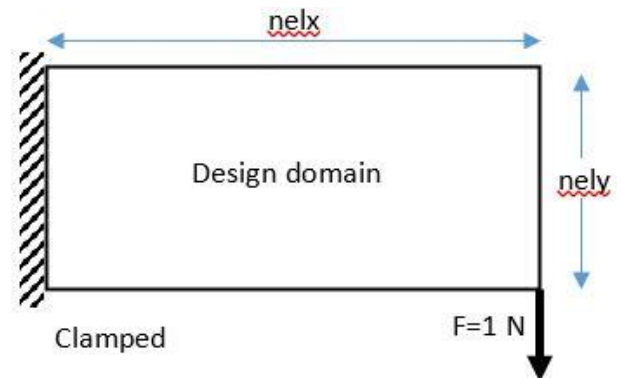


Figure 4. Base structure of a clamped beam subjected to load at the tip.

The following parameters were utilized for Case Study 1: a tolerance of $1e-6$ for constraints, a tolerance of $1e-10$ for optimality, and a total of $(nel * nely) = (32 \times 20 = 512)$ four-node elements in the plane stress analysis. The boundary condition was specified as $fixeddofs = [1:2*(nely+1)]$, and the applied force was identified as $F(2*(nelx+1)*(nely+1),1)=-1.0$. The volume fraction was set to 0.4. It is worth noting that the absolute values of dimensions and loads did not affect the density distributions, so unity dimensions and loads were considered. Table 1 presents the results obtained from different optimization algorithms for Case Study 1. The

first column indicates the name of the algorithm, the second column displays the computational elapsed time using a desktop computer (the same computer for all cases), the third column denotes the number of iterations, and the fourth column shows the value of the

objective function. The fifth column provides the count of objective function calculations. Specifically, Table 2 lists the results of the genetic algorithm scheme, including the number of generations and the count of objective function calculations.

Table 1. The analysis results of case study 1 using different optimization algorithms.

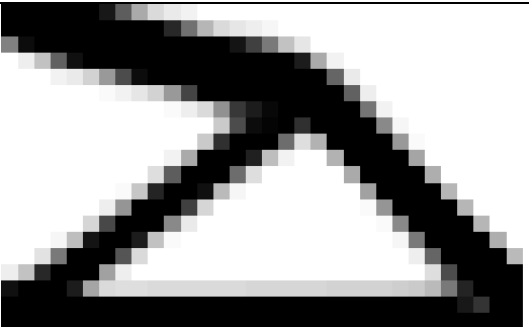
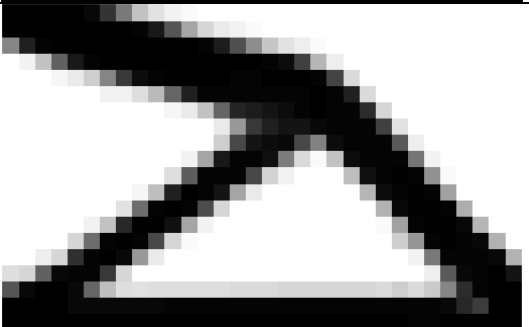
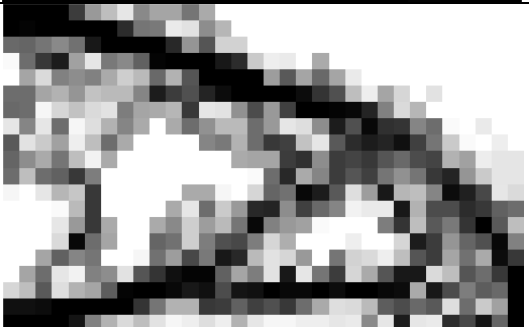


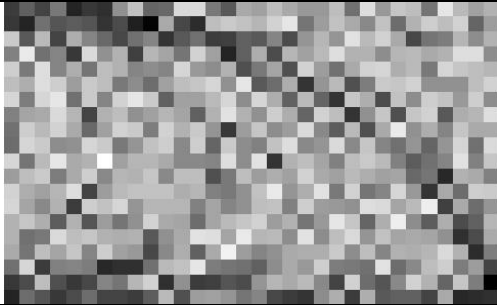
Method	Time [sec]	Iteration	Objective Function	Function Count	Results
Optimality criteria by 99-line code Sigmund [10]	2.14	71	57.35	71	
Interior Point	10	48	59.9	30801	
Sequential quadratic programming SQP	51	17	106	10984	
SQP-Legacy	63	18	104	11630	
Active-set	4420	74	263	47627	

Table 2. The analysis results of case study 1 using genetic algorithm.

Method	Time [sec]	Generations	Objective Function	Function Count	Results
Genetic Algorithm	3060	663	230	132801	

By inspecting the graphical results of the density distribution in the domain, it is evident that there is a good agreement with the results shown by others, which validates the MATLAB code and the implementation of the optimization tools. However, some methods did not perform as well due to the nature of this type of application. A comparative analysis of the results can be seen in Table 1 and Table 2.

The interior-point method stands out due to its superior performance in terms of both result quality and computational efficiency. It achieves a low objective function value of 59.9, which is very close to the value calculated by the optimal configuration (OC) as 57. Additionally, the elapsed time for the interior-point method is the lowest at 10 seconds. The SQP method also yields relatively close results, with an elapsed time of 51 seconds and an objective function value of 106. However, the objective function value is approximately 60% higher compared to the interior-point method. On the other hand, the results of the Active-Set and Genetic Algorithm methods are significantly poorer.

For instance, the genetic algorithm takes 3060 seconds to reach an objective function value of 230, which is much higher (worse) than the minimum achieved by the interior-point method. Similarly, the Active-Set method performs poorly in terms of computational costs and reaching a minimum. To facilitate further comparisons, another case will be analyzed and presented next.

5.2 Case Study 2

The results of Case 2, depicted in Figure 3, are presented here, involving a simply supported beam that is symmetric about a vertical axis at the midpoint and loaded downward at the central point-top. MATLAB optimization tools were used with the following data: a mesh of (nelxnely) = (60x20 = 1200) elements in the horizontal and vertical directions, a filter radius of rmin = 1.5, and a volumetric ratio of 0.5. The boundary conditions were defined as fixeddofs = union([1:2:2(nely+1)], [2*(nelx+1)*(nely+1)]), and the applied force was given as F(2,1) = -1.

The results obtained through optimization using the same set of techniques are summarized in Table 3 and Table 4. The quality of the algorithm results is illustrated using grayscale graphs, with the highest density represented as black. The tables include the name of the

optimization algorithm in the first column, the computational elapsed time on a normal desktop computer in the second column, the number of iterations in the third column, and the value of the objective function in the fourth column. The fifth column indicates the number of objective function calculations. The results of the genetic algorithm are presented separately in Table 2.

Expectedly, the numerical results for Case Study 2 once again demonstrate that the Interior Point algorithm outperformed the other tested algorithms in terms of computational efficiency and achieving global optimization results. However, it is worth noting that the OC method is still faster and produces better results.

6. Conclusion

In conclusion, this work focused on solving the topology optimization problem for domain structures using MATLAB and built-in optimization tools. The approach followed the structural modeling similar to Sigmund’s 99-line code (2001), utilizing the finite element plane-stress method and compliance as the objective function. The densities of the elements were considered as design variables, and the power law was used to weigh the densities.

Several optimization algorithms, including Interior-Point, SQP, SQP-legacy, Active-Set, and Genetic Algorithm, were implemented in MATLAB for comparison and validation. Two case studies from the literature were reproduced using these methods, and the results were compared with published results. Overall, the obtained results showed good agreement with the literature.

Among the tested algorithms, the Interior-Point method demonstrated superior performance in terms of both computational efficiency and result quality. It achieved acceptable results with reasonable computational effort required. On the other hand, the Genetic Algorithm exhibited low computational efficiency, and the Active-Set method yielded the worst results both in terms of computational expense and result quality.

It was observed that providing analytical gradient and Hessian functions reduced the number of iterations required for convergence. However, it is important to note that some methods, especially heuristic approaches like the Genetic Algorithm, do not rely on gradient

functions. Therefore, for problems where gradient functions are unavailable, it is advisable to use heuristic approaches.

In summary, this study successfully implemented and validated various optimization algorithms for topology optimization of domain structures. The Interior-Point method emerged as the most efficient and effective approach, while the Genetic Algorithm and Active-Set method showed limitations. The findings highlight the

importance of considering algorithm selection and availability of gradient functions when solving topology optimization problems.

Conflicts of interest

The authors declare no conflicts of interest.

Table 3. The analysis results of case study 2 using different optimization algorithms.

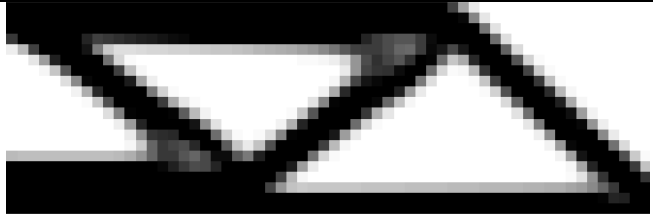
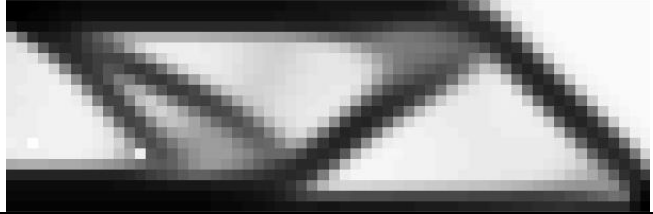



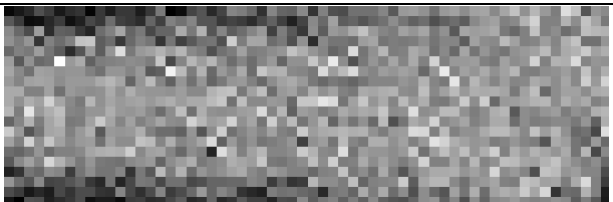
Method	Time [sec]	Iteration	Objective Function	Function Count	Results
Optimality criteria by 99-line code Sigmund [10]	7	94	203	94	
Interior Point	31.6	41	288	49318	
Sequential quadratic programming SQP	341	7	416	8463	
SQP-Legacy	258	7	416	8463	
Active-set	2037	190	345	57585	

Table 4. The analysis results of case study 2 using genetic algorithm.

Method	Time [sec]	Generations	Objective Function	Function Count	Results
Genetic Algorithm	10004	688	562	137801	

References

1. Deaton, J. D., & Grandhi, R. V. (2014). A survey of structural and multidisciplinary continuum topology optimization: post 2000. *Structural and Multidisciplinary Optimization*, 49, 1-38. <https://doi.org/10.1007/s00158-013-0956-z>
2. Zargham, S., Ward, T. A., Ramli, R., & Badruddin, I. A. (2016). Topology optimization: a review for structural designs under vibration problems. *Structural and Multidisciplinary Optimization*, 53, 1157-1177. <https://doi.org/10.1007/s00158-015-1370-5>
3. Bendsoe, M. P. (1989). Optimal shape design as a material distribution problem. *Structural optimization*, 1, 193-202. <https://doi.org/10.1007/BF01650949>
4. Rozvany, G. I. (2009). A critical review of established methods of structural topology optimization. *Structural and multidisciplinary optimization*, 37, 217-237. <https://doi.org/10.1007/s00158-007-0217-0>
5. Bendsoe, M. P., & Sigmund, O. (2003). *Topology optimization: theory, methods, and applications*. Springer Science & Business Media.
6. Hassani, B., & Hinton, E. (1998). A review of homogenization and topology optimization I—homogenization theory for media with periodic structure. *Computers & Structures*, 69(6), 707-717. [https://doi.org/10.1016/S0045-7949\(98\)00131-X](https://doi.org/10.1016/S0045-7949(98)00131-X)
7. Eschenauer, H. A., & Olhoff, N. (2001). Topology optimization of continuum structures: a review. *Applied Mechanics Reviews*, 54(4), 331-390. <https://doi.org/10.1115/1.1388075>
8. Rozvany, G. I. N. (2001). Stress ratio and compliance-based methods in topology optimization—a critical review. *Structural and Multidisciplinary Optimization*, 21, 109-119. <https://doi.org/10.1007/s001580050175>
9. Tiismus, H., Kallaste, A., Vaimann, T., & Rassõlkin, A. (2022). State of the art of additively manufactured electromagnetic materials for topology optimized electrical machines. *Additive Manufacturing*, 55, 102778. <https://doi.org/10.1016/j.addma.2022.102778>
10. Cardillo, A., Cascini, G., Frillici, F. S., & Rotini, F. (2013). Multi-objective topology optimization through GA-based hybridization of partial solutions. *Engineering with Computers*, 29, 287-306. <https://doi.org/10.1007/s00366-012-0272-z>
11. Sivapuram, R., & Picelli, R. (2018). Topology optimization of binary structures using integer linear programming. *Finite Elements in Analysis and Design*, 139, 49-61. <https://doi.org/10.1016/j.finel.2017.10.006>
12. Talischi, C., Paulino, G. H., Pereira, A., & Menezes, I. F. (2012). PolyTop: a Matlab implementation of a general topology optimization framework using unstructured polygonal finite element meshes. *Structural and Multidisciplinary Optimization*, 45, 329-357. <https://doi.org/10.1007/s00158-011-0696-x>
13. Sigmund, O. (2001). A 99 line topology optimization code written in Matlab. *Structural and multidisciplinary optimization*, 21, 120-127. <https://doi.org/10.1007/s001580050176>
14. Andreassen, E., Clausen, A., Schevenels, M., Lazarov, B. S., & Sigmund, O. (2011). Efficient topology optimization in MATLAB using 88 lines of code. *Structural and Multidisciplinary Optimization*, 43, 1-16. <https://doi.org/10.1007/s00158-010-0594-7>
15. Wang, S. Y., Tai, K., & Wang, M. Y. (2006). An enhanced genetic algorithm for structural topology optimization. *International Journal for Numerical Methods in Engineering*, 65(1), 18-44. <https://doi.org/10.1002/nme.1435>
16. Svanberg, K. (2002). A class of globally convergent optimization methods based on conservative convex separable approximations. *SIAM Journal on Optimization*, 12(2), 555-573. <https://doi.org/10.1137/S1052623499362822>
17. Svanberg, K. (1987). The method of moving asymptotes—a new method for structural optimization. *International Journal for Numerical Methods in Engineering*, 24(2), 359-373. <https://doi.org/10.1002/nme.1620240207>
18. Bendsoe, M. P., & Sigmund, O. (1999). Material interpolation schemes in topology optimization. *Archive of Applied Mechanics*, 69, 635-654. <https://doi.org/10.1007/s004190050248>
19. Dantzig, G. B., & Thapa, M. N. (2003). *Linear programming: Theory and extensions (Vol. 2)*. New York: Springer.
20. Jansen, B. (2013). *Interior point techniques in optimization: Complementarity, sensitivity and algorithms (Vol. 6)*. Springer Science & Business Media.
21. Nocedal, J., & Wright, S. J. (Eds.). (1999). *Numerical optimization*. New York, NY: Springer New York.
22. Forsgren, A., Gill, P. E., & Wright, M. H. (2002). Interior methods for nonlinear optimization. *SIAM Review*, 44(4), 525-597. <https://doi.org/10.1137/S0036144502414942>
23. Dubois, T. (2013). EASA Requires A380 Structural Inspection for Cracking. <https://www.ainonline.com/aviation-news/air-transport/2013-11-11/easa-requires-a380-structural-inspection-cracking>
24. Singal, R. K., Gorman, D. J., & Forgues, S. A. (1992). A comprehensive analytical solution for free vibration of rectangular plates with classical edge conditions: Experimental verification. *Experimental Mechanics*, 32, 21-23. <https://doi.org/10.1007/BF02317979>
25. Hauser, F., Häberle, M., Merling, D., Lindner, S., Gurevich, V., Zeiger, F., Frank, R., & Menth, M. (2023). A survey on data plane programming with p4: Fundamentals, advances, and applied research. *Journal of Network and Computer Applications*, 212, 103561. <https://doi.org/10.1016/j.jnca.2022.103561>
26. Kramer, O. (2017). *Genetic Algorithm Essentials*, Springer International Publishing, Cham.

27. Kramer, O. (2017). Genetic algorithms (pp. 11-19). Springer International Publishing. repairing. Structural and Multidisciplinary Optimization, 32, 31-39.
28. Madeira, J. A., Rodrigues, H. C., & Pina, H. (2006). Multiobjective topology optimization of structures using genetic algorithms with chromosome <https://doi.org/10.1007/s00158-006-0007-0>



© Author(s) 2024. This work is distributed under <https://creativecommons.org/licenses/by-sa/4.0/>