Int j simul model 18 (2019) 3, 453-463
ISSN 1726-4529 Original scientific paper

METHODS FOR INCREASED EFFICIENCY OF FEM-BASED TOPOLOGY OPTIMIZATION

Glamsch, J.; Deese, K. & Rieg, F.
Chair of Engineering Design and CAD, University of Bayreuth, Germany
E-Mail: johannes.glamsch@uni-bayreuth.de, kevin.deese@uni-bayreuth.de

Abstract
Lightweight construction is playing an increasingly important role for a wide variety of reasons, such as improving energy efficiency. In addition to lightweight material construction, lightweight structure construction is gaining more and more influence, which is made possible due to topology optimization. The aim of topology optimization is to develop an optimal design proposal based on a construction space model and given boundary conditions (e.g. mechanical or thermal). The calculation of the structural response is often done using the time consuming finite element method (FEM). Since topology optimization is an iterative process, usually many finite element analyses (FEA) have to be performed, which results in high computing time. Therefore, this article presents different methods to minimize computing time by exploiting various special features that occur with FEA in the context of optimizations.
(Received in April 2019, accepted in July 2019. This paper was with the authors 2 weeks for 1 revision.)

Key Words: Structural Optimization, Topology Optimization, Computational Effort, Finite Element Method

1. INTRODUCTION

Lightweight construction is gaining in importance and is increasingly playing an important role in the modern development of new products. Great material savings and associated weight reduction lead to a more efficient use of resources, resulting in better environmental performance and lower operating costs. Topology optimization is an ideal tool for developing lightweight structures for various applications. Typical algorithms aim to optimize stiffness, strength or a combination of both while reducing the weight of the structure [1-4].

The problem with topology optimization is, however, that an optimization often takes several hours and depending on the problem size can take up to a few days. This is because the optimization usually is an iterative process, which uses the computationally expensive finite element method (FEM) in each iteration (see Fig. 1). Besides the typical topology optimization, which is designed for one optimization goal, other algorithms optimize several objectives at once. These multi-objective optimization algorithms often are population based and therefore need to conduct several calculations per iteration, which multiplies the effort [5-7]. This high time consumption often renders the topology optimization impractical and is one of the reasons that it is not yet widely used for commercial purposes.

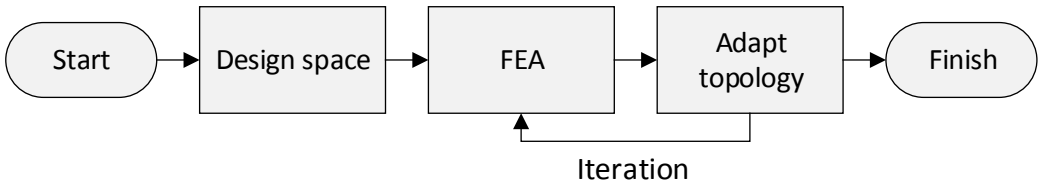

Figure 1: Typical topology optimization process.

One way to mitigate the problem of time consuming simulation runs is to reduce the size of the optimization problem. By reducing the number of the so-called design variables, respectively the number of the elements in the finite element mesh, the finite element analysis

(FEA) does not take as long and the whole topology optimization is faster. However, this procedure leads to a different problem. If the number of elements is reduced, the resolution of the optimized result is too low and it will be difficult for the user to interpret the result (see Fig. 2), even after smoothing the surface [8]. This also makes a parametric redesign more difficult, which in many cases is necessary for further steps, such as production. Another problem with reducing the size of the finite element mesh is that the optimizer may not have the necessary freedom to find optimum structures and create fine structures where they actually are required. Finally, a coarse mesh may lead to inaccurate results of the FEA and therefore to wrong optimization results [9, 10].

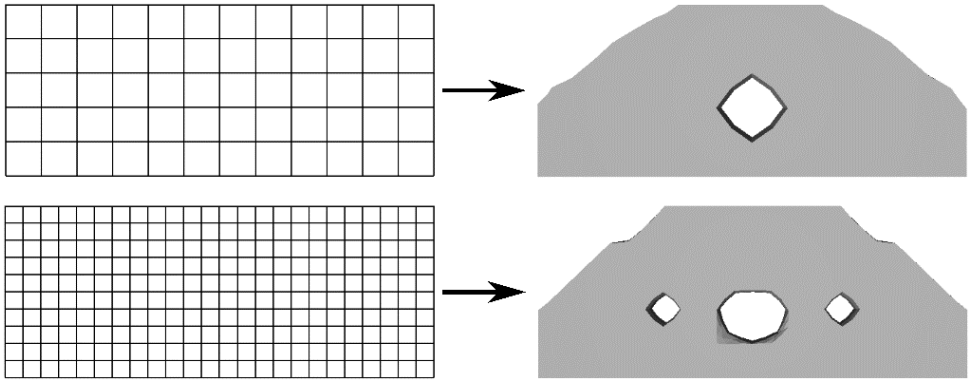

Figure 2: Low resolution (top) compared to high resolution (bottom) results of topology optimization.

This article shows different methods for speeding up the topology optimization by exploiting characteristic features of optimization problems. The main idea is not to change the optimization problem by reducing its complexity but to change the way the FEM is implemented. The reason for this is that the FEA is the most time consuming part of the optimization process and therefore bears the highest potential for speeding up the process. The authors' experience indicates that the FEA makes up about 90 % to 95 % of the overall time.

2. THEORETICAL BACKGROUND

In the following, the theoretical basics of the finite element method and topology optimization are presented. Based on these fundamentals, different optimization potentials to accelerate the FEA by exploiting different characteristics of topology optimization problems are shown in the next section.

2.1 Finite Element Method

The FEM is a mathematical method for solving differential equations. In mechanical engineering applications, the method is mainly used for calculating the mechanical behaviour of structures, e.g. displacements, stresses and strains. The usual application involves load cases that only lead to small displacements and therefore simplify the FEM. Further applications, such as contact simulations or calculations with highly nonlinear material behaviour require a more complex formulation of the method and will not be considered here [9-12].

A FEA consists of the following steps [9-11]:

- discretization of the geometry,
- calculation of the element stiffness matrices,
- compilation of the global stiffness matrix,
- imposition of the boundary conditions,

- solving the linear equation system,
- generation of additional output fields, e.g. stresses and strains.

Geometry Discretization

As first step of the FEA, the structure must be discretized, i.e. be broken down into individual elements. Static mechanical analyses mainly use tetrahedral and hexahedral elements for three dimensional structures, which can be distinguished according to the polynomial degree of their shape functions [10].

The quality of the results of a FEA depends largely on the mesh, in particular the mesh size. In general, mesh generation can be divided into structured and unstructured mesh generation [11].

Element stiffness matrices

The element stiffness matrix $\boldsymbol{K}_e$ of an individual element can be interpreted as an extension of the Hooke's Law to multiple dimensions. Therefore, the nodal displacement vector u_e of each element can be retrieved by solving the linear equation system $\boldsymbol{K}_e u_e = f_e$, where f_e describes the element force vector, which contains the nodal forces. The element stiffness matrices are formed by calculating the following volume integral [10]:

$$\boldsymbol{K}_e = \iiint_V \boldsymbol{B}^T \boldsymbol{C} \boldsymbol{B} \, dV \tag{1}$$

The volume of each element is described by V. The matrix $\boldsymbol{B}$ contains derivatives of the shape functions and depends on the geometry of the finite element. Last, the matrix $\boldsymbol{C}$ is the material matrix and depends—for isotropic materials—on the Young's modulus E and the Poisson ratio ν [9, 10].

The integration of $\boldsymbol{K}_e$ is done via Gauss-Legendre quadrature by first transforming the finite element onto the space $[-1, +1]^3$ with the coordinate axes r, s and t. The integral can then be approximately calculated by weighted sums, where the weights depend on the number of integration points used for the quadrature. It should be noted that $dV = \det \boldsymbol{J} \, dr \, ds \, dt$, where $\boldsymbol{J}$ describes the Jacobi determinant. This leads to the following approximation of $\boldsymbol{K}_e$ [10]:

$$\boldsymbol{K}_e \approx \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} \sum_{k=1}^{n_t} \alpha_i \alpha_j \alpha_k \, \boldsymbol{B}^T(r_i, s_j, t_k) \, \boldsymbol{C} \, \boldsymbol{B}(r_i, s_j, t_k) \det \boldsymbol{J}(r_i, s_j, t_k) \tag{2}$$

The values n_r, n_s and n_t describe the integration order. For hexahedra, the integration order $n = 2$ or $n = 3$ is often assumed which leads to 8 respectively 27 integration points [10].

Global stiffness matrix compilation and boundary condition imposition

The global stiffness matrix $\boldsymbol{K}$ and global external force vector f are built by compiling the element stiffness matrices $\boldsymbol{K}_e$ respectively the element force vectors f_e in respect to the nodal numeration [9]:

$$\boldsymbol{K} = \sum_i \boldsymbol{K}_{e,i} \qquad\qquad f = \sum_i f_{e,i} \tag{3}$$

This leads to the following linear equation system for linear static mechanical analyses, which can be solved for the global displacement vector u [9]:

$$\boldsymbol{K} u = f \tag{4}$$

After compiling the global equation system, the boundary conditions like external forces or displacements can be imposed. The reader is referred to the literature for further details, e.g. [9, 10, 13].

Solving and output field generation

The next step is solving the resulting equation system $\boldsymbol{K}u = f$. The methods for solving these equation systems can be distinguished between direct and iterative methods in principle. Common examples of these methods are, for example, the Cholesky or LU decomposition respectively the conjugate gradient method [10].

Based on the displacement results, further output fields can be determined. For example, the stress at a specific point in an element is obtained by the following equation [10]:

$$\sigma = \boldsymbol{C}\boldsymbol{B}u_e \tag{5}$$

2.2 Topology optimization

Topology optimization can be described as a product development tool, which iteratively leads to a new product design that optimally satisfies a given goal, the so-called target function. Additionally, there may be further constraints to the resulting design. Typical goals are maximum stiffness or minimum deflection, while typical constraints can be a maximum stress value or a maximum weight of the structure [1, 2, 14-17].

Usually, the process for topology optimization is as follows (see Fig. 3): First, the designer creates a rough concept of the new product, i.e. he describes the design space that can be used and defines interfaces to surrounding other components. Based on this design space, a FEA provides structural responses, such as stress values. The optimization algorithm uses these responses to adapt the structure, which usually happens by adapting material properties of the finite elements. A new FEA now provides responses for the structure with the new properties, which in turn are used for a further iteration in the optimization algorithm. This iterative process repeats itself until certain abort criteria are met [3].

In a typical topology optimization, the elements' geometry does not change during the optimization but only the respective material properties. Besides other aspects, this characteristic can be used to reduce the number of mathematical operations per FEA and therefore reduce the time consumption of the whole topology optimization.

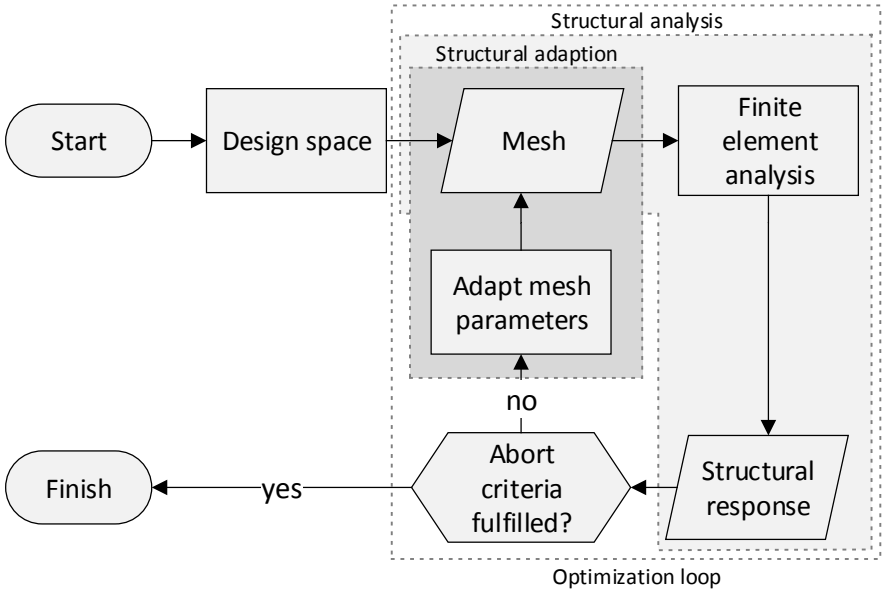

Figure 3: A detailed topology optimization process using the finite element method for calculating structural responses.

As an example of a topology optimization algorithm, the soft kill option method (SKO) strives to maximize the strength of a new part while at the same time reducing the weight. For this, it uses the stress response out of the FEA and adapts the Young's modulus of each finite element in every iteration according to the following formula [18, 19]:

$$E_i^{(k+1)} = E_i^k + s \cdot (\sigma_i^k - \sigma_{ref}) \tag{6}$$

Here, i stands for the i^{th} finite element, k is the current iteration, σ_{ref} a user-defined reference stress, σ the stress response out of the FEA, s a user-defined scaling factor and E the Young's modulus. If the current stress level in an element is higher than the reference stress, then the Young's modulus will be increased, which will render the material stiffer. Vice versa, if the stress level lies below the reference stress, then the material will be made softer by decreasing the Young's modulus. A softer material in return approaches the behaviour of a hole in the structure.

This approach leads to a homogeneous surface stress level over the whole structure by strengthening highly loaded components and by removing parts that are unnecessary for carrying the load. This in turn reduces the weight of the structure.

3. METHODS FOR INCREASED EFFICIENCY

In the following, various methods for increasing the efficiency of the topology optimization process are presented and discussed. As already mentioned, both the FEM and the optimization algorithms are already highly optimized. Since FEM accounts for the largest share of the total computing time, it poses the greatest potential for improvement. The approaches presented are using special features of the model design for optimizations.

3.1 Adapting software architecture

The first aspect to be addressed is the software architecture of topology optimization suites.

State of the art

Current topology optimization software suites mostly consist of two separate programs [19]. On the one hand, this is due to the fact that the topology optimization would also be possible with another method for calculating the structure response instead of the FEM. On the other hand, many FEM solvers like NASTRAN, Abaqus, ANSYS or Z88 existed before the common use of topology optimization. Another reason for this development is that an optimizer could work with different solvers.

The input data for the most common FEM solvers are text based input files, e.g. INP files for Abaqus. Therefore, the two programs must communicate via a text file based interface (see Fig. 4). This is the standard procedure for many topology optimization software suites like Tosca/Abaqus [20] and Z88Arion [21, 22].

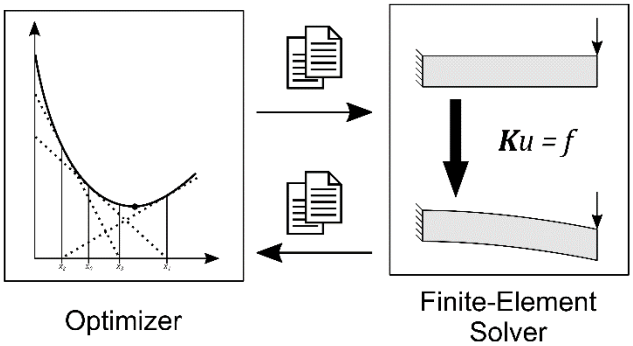

Figure 4: Typical software architecture of topology optimization suites.

For this reason, the FEM solver must re-read all structure data (nodes and elements), boundary conditions, material definitions for each element and other control parameters in each iteration. Furthermore, the optimizing software must read the whole output data of the solver, which can already include large amounts of data even for relatively small models.

Although this text-based interface offers advantages, such as the possibility to develop optimizers and solvers independently of one another, they slow down the process considerably. For example, the parsing of an Abaqus INP input file with 119925 nodes and 112480 hexahedron elements requires about 7 seconds for a total calculation time of 33 seconds (on an eight core CPU), which is a fraction of over 20 %. This parsing has to be done in each iteration of the optimization.

Proposal

Our proposal is to combine the optimizer and FEM solver, either through a monolithic development or through including the FEM solver as a program library as shown in Fig. 5.

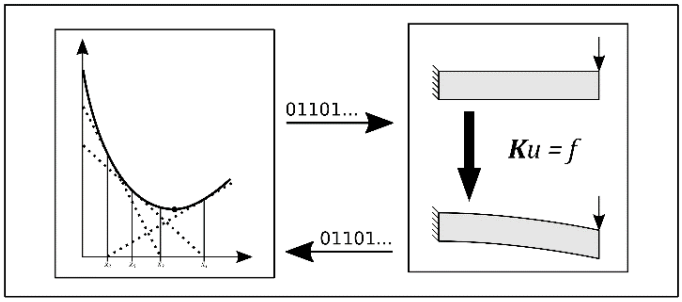

Figure 5: Proposed software architecture.

This combination leads to two main advantages:

- The slow text-based interface is avoided (parsing and reading/writing, which is mainly limited due to the speed of current mass storage devices).
- The structure (nodes and elements) and the boundary conditions do not change during a topology optimization. Therefore, in each iteration, only one vector of material properties needs to be exchanged between the FEM solver and the optimizer.

This proposal has a high potential to speed up the optimization. As stated, the parsing of an input file can make up to about 20 % of the whole calculation time. This slowdown can be completely avoided by changing the software architecture.

3.2 Caching of element stiffness matrices

The combination of the optimizer and FEM solver (see section 3.1) leads to possibilities to increase the speed of topology optimizations, which are presented in this chapter.

State of the art

As already mentioned, the structure (nodes, elements and boundary conditions) of the FE model does not change during the individual iterations of topology optimization. The only change lies in the material data of each finite element. This fact is often not exploited in current topology optimization software suites. Therefore, in each iteration, the global stiffness matrix $\boldsymbol{K}$ is rebuilt by calculating each element stiffness matrix $\boldsymbol{K}_e$ and compiling the global matrix.

Proposal

We suggest that the element stiffness matrices are not recalculated in each iteration but cached. In case of a linear elastic FEA, the element stiffness depends linearly on the Young's

modulus of the corresponding element [19]. For this reason, the Young's modulus E can be pulled in front of the numerical quadrature of the stiffness matrix:

$$\begin{aligned} \boldsymbol{K}_e &= E \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} \sum_{k=1}^{n_t} \alpha_i \alpha_j \alpha_j \, \boldsymbol{B}^T(r_i, s_j, t_k) \, \boldsymbol{C}' \boldsymbol{B}(r_i, s_j, t_k) \det \boldsymbol{J}(r_i, s_j, t_k) \\ &= E \, \boldsymbol{K}'_e \quad \text{with } \boldsymbol{C}' = \frac{1}{E} \boldsymbol{C} \end{aligned} \tag{7}$$

With this method, the computationally very expensive integration of the element stiffness matrices has to be carried out only at the beginning of the topology optimization in the first iteration. In each successive iteration, the matrix $\boldsymbol{K}'_{\mathrm{e}}$ only needs to be multiplied by the current Young's modulus E of the element. In addition, the element shape functions of the element must only be evaluated in the first iteration of the optimization.

This method also provides speed advantages in the compilation of the global stiffness matrix $\boldsymbol{K} = \sum_i \boldsymbol{K}_{e,i}$. The global stiffness matrix $\boldsymbol{K}$ is usually stored as a sparse matrix instead of a dense matrix since it contains only a small fraction of non-zero entries, which reduces the memory requirement significantly [10]. The sparse matrix can be stored in various formats, for example the CSR format [23]. The compilation requires a symbolic compilation beforehand, i.e. the location of the non-zero elements of the matrix must be determined [10]. Due to the unchanged structure of the FE model, the location of the non-zero entries does not change and the time consuming symbolic compilation has to be done only once at the beginning of the calculation.

3.3 Exploiting mesh characteristics

As said, certain properties of the finite element meshes used for topology optimizations can be exploited to speed up the optimization. These approaches are further described in this section.

State of the art

According to the previous section, in each iteration of the optimization the global stiffness matrix $\boldsymbol{K}$ must be compiled from the element stiffness matrices $\boldsymbol{K}_e$. This process can be accelerated by caching the material independent matrices $\boldsymbol{K}'_e$, since no complete recalculation of $\boldsymbol{K}_e$ is necessary. Nevertheless, in the first iteration the computational expensive calculation of all matrices $\boldsymbol{K}'_e$ is necessary.

Proposal

The basis for topology optimization is usually a design space model. This space is often geometrically simpler than the final component. If the user puts effort in the meshing process – for example by partitioning the design space into geometric primitives like rectangular cuboids, pyramids or even cylinders and using a structured mesh – a large proportion of elements of the same size and orientation can be generated. These elements have the same element stiffness matrix $\boldsymbol{K}_e$ respectively $\boldsymbol{K}'_e$, since the stiffness of an element is independent of its position in space and is only described relatively between its nodes. For this reason, the element stiffness matrix does not have to be calculated for each element, but can be grouped together for the same elements and only has to be performed once. The information about the same elements can hereby be taken directly from the meshing software. If this is not possible, the same elements must be identified before the first iteration. Moreover, this approach can not only be applied to the same elements, but also be extended to rotated and/or scaled elements, since $\boldsymbol{K}_e$ can be transformed with transformation matrices, which is more efficient than the integration of each element stiffness matrix.

3.4 Reducing model size

It was mentioned in the introduction that the mesh must not be too coarse when running a topology optimization; otherwise, the optimizer does not have enough freedom to determine an optimal structure. This section describes how the size of the finite element model respectively the global stiffness matrix $\boldsymbol{K}$ can still be reduced to speed up the calculation.

State of the art

Currently, the entire finite element model is calculated during an optimization. However, some of the elements are often contained in so-called fix sets or frozen regions. The Young's modulus of these elements is not varied during an optimization, but is fixed to the initial value. This serves to avoid optimizing certain sections of the model, for example in areas of load application, constraints or other structurally important areas such as contact surfaces [20, 22]. These elements must be considered in each iteration, although their Young's modulus is not a design variable, which slows down both the generation of the equation system $\boldsymbol{K}u = f$ and its solution. Depending on the model, the fix sets can include a large portion of the elements.

Proposal

One possibility to reduce the number of degrees of freedom is the so-called substructure technique. Parts of the structure are condensed in advance, so that all internal nodes of the substructure can be eliminated. Our proposal is to apply this technique automatically to all fix sets before the first iteration. Only the nodes which are connected to the rest of the structure remain. A typical method for the static condensation is the so called Guyan reduction [24-26]. The equation system $\boldsymbol{K}u = f$ is rearranged, whereas the indices e respectively i denote external (remaining) and internal (eliminated) degrees of freedom [24, 25]:

$$\begin{pmatrix} \boldsymbol{K}_{ee} & \boldsymbol{K}_{ei} \\ \boldsymbol{K}_{ie} & \boldsymbol{K}_{ii} \end{pmatrix} \begin{pmatrix} u_e \\ u_i \end{pmatrix} = \begin{pmatrix} f_e \\ f_i \end{pmatrix} \tag{8}$$

The elimination of the internal degrees of freedom leads to the following equation [25]:

$$u_i = \boldsymbol{K}_{ii}^{-1}(f_i - \boldsymbol{K}_{ie}u_e) \tag{9}$$

Resubstituting yields the following reduced linear equation system [25]:

$$(\boldsymbol{K}_{ee} - \boldsymbol{K}_{ei}\boldsymbol{K}_{ii}^{-1}\boldsymbol{K}_{ie})u_e = f_e - \boldsymbol{K}_{ei}\boldsymbol{K}_{ii}^{-1}f_i \tag{10}$$

$$\boldsymbol{K}_{red}u_e = f_{red} \tag{11}$$

The reduction of the fix sets only has to be done once at the beginning of the optimization and is about as computationally expensive as the one-time solution of the equation system of the substructure. The main effort consists in the inversion of the matrix $\boldsymbol{K}_{ii}$, which has – like the global stiffness matrix $\boldsymbol{K}$ – only a small amount of non-zero entries and therefore can be stored in a memory efficient sparse storage format, e.g. CSR. This calculation step can be solved efficiently by not inverting the matrix directly, but by decomposing the matrix and solving the equation system by forward elimination and back substitution. For this step, highly optimized algorithms are available, which are presented in the following section. By applying this technique, the number of degrees of freedom respectively nodes can be significantly reduced depending on the model, which accelerates each individual iteration of the optimization.

An example is shown in Fig. 6 (left). All of the light grey elements are in a fix set and therefore their Young's modulus is not a design variable. Like stated, the stiffness of the fixed sets can be condensed to all interface nodes (see Fig. 6 (right)), which leads to a much smaller model size and therefore higher calculation speed in each optimization iteration.

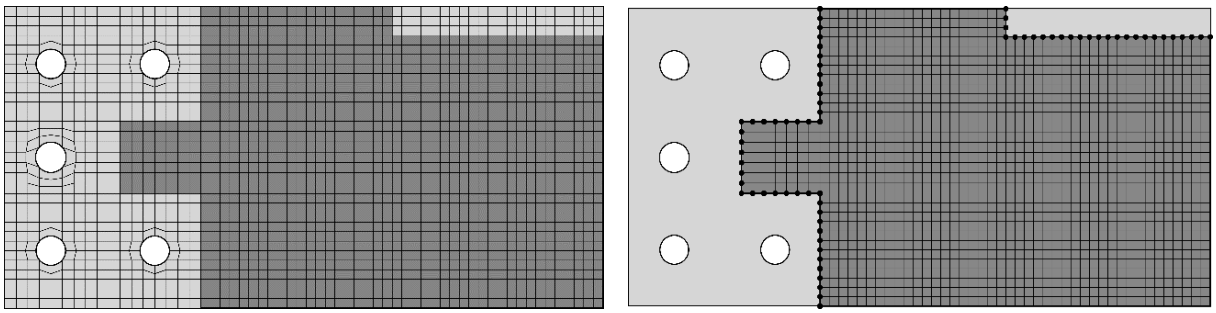

Figure 6: Reduction of model size due to substructures (light grey: fixed sets, dark grey: design space).

3.5 Exploiting matrix characteristics

In this section, further possibilities of how the iterations of the topology optimization can be accelerated shall be described. Therefore, the fact that not the whole structure but only the material of the individual elements changes is exploited.

State of the art

As described, the global stiffness matrix $\boldsymbol{K}$ usually contains only few entries and is therefore stored as a sparse matrix. This sparsity pattern of the matrix does not change due to the same model structure in each iteration and hence can only be calculated once for all iterations. The resulting linear equation system $\boldsymbol{K}u = f$ can then be solved using an equation solver, either a direct or iterative one. Due to their numerical superiority, however, direct equation solvers are mostly used, which are based on a decomposition of the matrix (e.g. $\boldsymbol{K} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$ when using a Cholesky factorization) and subsequently solve the equation system by forward elimination and back substitution. Since solving the equation system requires most of the computing time in each iteration, it is important to optimize this step [10].

Proposal

Most equation system solvers are highly optimized, which means that the basic algorithm cannot be further accelerated. However, it can be exploited that the sparsity pattern of $\boldsymbol{K}$ does not change between iterations. The matrices into which the matrix is decomposed are, similar to the global stiffness matrix, also sparse matrices. The sparsity pattern of the matrices must be calculated before the numerical decomposition of the matrix, since it is required for the generation of the sparse storage structure (e.g. in the CSR format), which is called symbolic factorization. This occupation can be calculated on the basis of an elimination tree [27]. Since the sparsity pattern of the factorized matrices does not change, this can also be done once in the first iteration and taken over in all further iterations.

The factorized matrix usually has more entries than the original matrix, the so-called fill-in. This not only leads to an increased memory requirement, but also has a negative effect on the computation time, since the computation time t required for numerical factorization of a matrix depends quadratically on the number of nonzero elements of the factorized matrices [27].

Therefore, before symbolic factorization, the FE node numbers respectively matrix columns and rows are usually renumbered in order to minimize this fill-in effect. This matrix permutation is mostly done by heuristic methods [27], e.g. the *Minimum Degree Algorithm* [28] or *Nested Dissection Algorithm* [29]. According to the authors' experience, depending on the model, the fill-in reduction and symbolic factorization make up to about one third of the total calculation time needed to solve the equation system when using the PARDISO [23] solver in combination with the *Nested Dissection Algorithm*. Therefore, the step of solving the equation system can also significantly be accelerated by performing the symbolic factorization and fill-in reduction only in the first iteration.

4. CONCLUSION

Topology optimization based on the FEM has become a widely used tool in virtual product development. However, the high calculation effort often prevents a broad application in the industry. In this article, different methods have been presented to reduce this calculation time by taking advantage of different characteristics of FEA in topology optimization.

Not all proposed methods have to be implemented; even the use of individual methods can lead to a significant acceleration of the optimization. It should be noted that the methods presented here can be used not only for mechanical analyses, but also for thermal calculations, for example. To quantify the calculation acceleration in precise terms a combined topology optimization software and finite element solver, which implements all the discussed methods, has to be developed.

ACKNOWLEDGEMENT

The European Union supported this research via the European Regional Development Fund (ERDF):

European Union
European Regional
Development Fund

REFERENCES

[1] Bendsoe, M. P.; Sigmund, O. (2004). *Topology Optimization: Theory, Methods, and Applications*, 2nd edition, Springer, Berlin
[2] Deaton, J. D.; Grandhi, R. V. (2014). A survey of structural and multidisciplinary continuum topology optimization: post 2000, *Structural and Multidisciplinary Optimization*, Vol. 49, No. 1, 1-38, doi:10.1007/s00158-013-0956-z
[3] Deese, K.; Frisch, M.; Hautsch, S. (2018). Strukturoptimierung, Rieg, F.; Steinhilper, R. (Eds.), *Handbuch Konstruktion*, 2nd edition, Hanser, Munich, 511-520
[4] Billenstein, D.; Dinkel, C.; Rieg, F. (2018). Automated topological clustering of design proposals in structural optimisation, *International Journal of Simulation Modelling*, Vol. 17, No. 4, 657-666, doi:10.2507/IJSIMM17(4)454
[5] Deb, K. (2012). *Optimization for Engineering Design: Algorithms and Examples*, 2nd edition, PHI Learning, New Delhi
[6] Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*, Wiley, Hoboken
[7] Coello Coello, C. A.; Lamont, G. B.; Van Veldhuizen, D. A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*, 2nd edition, Springer, New York
[8] Deese, K.; Geilen, M.; Rieg, F. (2018). A two-step smoothing algorithm for an automated product development process, *International Journal of Simulation Modelling*, Vol. 17, No. 2, 308-317, doi:10.2507/IJSIMM17(2)437
[9] Bathe, K.-J. (2014). *Finite Element Procedures*, 2nd edition, Bathe, Watertown
[10] Rieg, F.; Hackenschmidt, R.; Alber-Laukant, B. (2014). *Finite Element Analysis for Engineers*, Hanser Publishers, Munich
[11] Viebahn, F. (2018). *Systematische Untersuchung der Anwendbarkeit der Finite-Elemente-Analyse zur Abbildung des Wälzlagerwanderns mittels der statistischen Versuchsplanung*, PhD Thesis, Shake, Herzogenrath
[12] Liao, W.; Li, L.; Liu, D.; Dai, B.; Wang, C. (2018). Nonlinear FEM analysis on composite beams with web opening under negative bending moment, *Technical Gazette*, Vol. 25, No. 5, 1546-1552, doi:10.17559/TV-20180626222438
[13] Glenk, C.; Hüter, F.; Billenstein, D.; Rieg, F. (2018). Consideration of body forces within finite element analysis, *Strojniski vestnik – Journal of Mechanical Engineering*, Vol. 64, No. 5, 303-309, doi:10.5545/sv-jme.2017.5081

[14] Ramadani, R.; Belsak, A.; Kegl, M.; Predan, J.; Pehan, S. (2018). Topology optimization based design of lightweight and low vibration gear bodies, *International Journal of Simulation Modelling*, Vol. 17, No. 1, 92-104, doi:10.2507/IJSIMM17(1)419
[15] Harl, B.; Predan, J.; Gubeljak, N.; Kegl, M. (2017). On configuration-based optimal design of load-carrying lightweight parts, *International Journal of Simulation Modelling*, Vol. 16, No. 2, 219-228, doi:10.2507/IJSIMM16(2)3.369
[16] Ooi, J. B.; Wang, X.; Lim, Y. P.; Tan, C. S.; Ho, J. H.; Wong, K. C. (2013). Parametric optimization of the output shaft of a portal axle using finite element analysis, *Strojniski vestnik – Journal of Mechanical Engineering*, Vol. 59, No. 10, 613-619, doi:10.5545/sv-jme.2012.887
[17] Marzbanrad, J.; Hoseinpour, A. (2017). Structural optimization of MacPherson control arm under fatigue loading, *Technical Gazette*, Vol. 24, No. 3, 917-924, doi:10.17559/TV-20150225090554
[18] Mattheck, C. (1998). *Design in Nature: Learning from Trees*, Springer, Berlin
[19] Harzheim, L. (2014). *Strukturoptimierung: Grundlagen und Anwendungen*, Europa Lehrmittel, Haan-Gruiten
[20] Dassault Systèmes Simulia Corp. (2015). *Abaqus Analysis User's Guide*, Version IV, Dassault Systèmes Simulia Corp., Providence
[21] Chair of Engineering Design and CAD, University of Bayreuth. Z88Aurora User Manual (Version 5), from *https://en.z88.de/manuals/*, accessed on 23-06-2019
[22] Chair of Engineering Design and CAD, University of Bayreuth. Z88Arion Theorie-Benutzerhandbuch (Version 2), from *https://en.z88.de/manuals/*, accessed on 23-06-2019
[23] Intel Corporation. Intel® Math Kernel Library: Developer Reference (Revision 024), from *https://software.intel.com/en-us/mkl-developer-reference-c*, accessed on 23-06-2019
[24] Guyan, R. J. (1965). Reduction of stiffness and mass matrices, *AIAA Journal*, Vol. 3, No. 2, 380, doi:10.2514/3.2874
[25] Lutz, N. (2015). *FEM-Formelsammlung Statik und Dynamik*, 3rd edition, Springer Vieweg, Munich
[26] Weinberger, U.; Glenk, C.; Stahl, K.; Rieg, F. (2017). *RIKORplusZ88: Einbindung elastischer Gehäusestrukturen in die Getriebeauslegung mit RIKOR und Visualisierung des Getriebegesamtsystems in der FVA-Workbench (Report No. 1250)*, Forschungsvereinigung Antriebstechnik, Frankfurt
[27] Hüter, F. (2017). *Entwicklung eines direkten Gleichungslösers für das Finite-Elemente-Programm Z88 auf Basis der Cholesky-Zerlegung dünnbesetzter Matrizen*, Master Thesis, University of Bayreuth, Bayreuth
[28] Li, X. S.; Demmel, J. W. (1999). A scalable sparse direct solver using static pivoting, *Proceedings of the 9th SIAM Conference on Parallel Processing for Scientific Computing*, 22-34
[29] Karypis, G.; Kumar, V. (1999). A fast and high quality multilevel scheme for partitioning irregular graphs, *SIAM Journal on Scientific Computing*, Vol. 20, No. 1, 359-392, doi:10.1137/S1064827595287997