
An Accessible Framework for Optimizing the Structural Performance of Wood-based Building Components

Johannes Belz*, Benjamin Kromoser,

* Institute of Green Civil Engineering,
University of Natural Resources and Life Sciences Vienna
Peter-Jordan-Street 82, 1190 Vienna, Austria
johannes.belz@boku.ac.at

Abstract

The reduction of emissions from the construction industry plays a substantial role in the global efforts to mitigate climate change. While lowering the overall construction output is likely the most effective solution, reducing the environmental impact of building materials, extending the lifetime and cycle of materials as well as the optimization and high utilization of structural elements are additional relevant options. With reference to the findings of a prior literature study, current research on the structural optimization of wooden building components is showing a predominant use of computationally cheap objective functions, a low focus on relevant variables, like the orientation of material, and limitations in structural analysis methods. While finite element methods and the consideration of orientation variables in a minimum compliance problem, for example, are state of research in high-tech industries, a low availability of open-source code limits the practical implementation. To counteract this, the present study introduces an approachable and accessible structural optimization framework for wooden building components that can be used in a wide range of applications. The framework is developed in Python programming language and bridges a genetic algorithm with structural analysis capabilities found in the commercially available finite element software ABAQUS FEA. It presents a unified solution combining established processes of finite element analysis, optimization algorithms and 3D model generation and is implemented for the optimization of a multilayered structural element made from oriented strand board. The specific structural performance of the structurally optimized building component in relation to a conventional component is compared and verified through a calibrated finite element material model based on large and small scale experiments.

Keywords: topology optimization, shape optimization, structural optimization, genetic algorithm, finite element analysis, timber, engineered wood, structural element

1. Introduction

Due to the swift rise in global surface temperature, now standing at 1.1°C above preindustrial times, the window of opportunity to counteract irreversible damages to the planet's ecosystem is rapidly closing. Urgent steps must be taken in the near future to reduce CO₂ emissions, ensuring both human well-being and planetary health [1]. With more than one third of global CO₂ emissions (37%) contributed to the construction sector, especially in relation to used materials (approx. 9%), the importance of climate resilient development in the building industry is not to be questioned [2]. While the reduction of emissions coming from building operations are equally relevant, this work will focus on the material-related reduction of the matter.

To reduce the impact of constructions on our environment, several strategies are applicable. While the overall reduction of output, e.g. new building constructions, would have the greatest impact, this possibility is seen as rather unlikely in the near future due to predominantly social and economic reasons. We can therefore narrow our scope to three realistic options: (1) Conscious material choices and fostering of biobased and natural materials, (2) Extension of material use time within one life cycle and

enablement of circular use for several life cycles, and (3) Increase of material utilization, further referred to as structural optimization (SO). While the shift to natural and bio-based materials is accompanied with a number of advantages, this could result in large environmental loads on forests and surrounding ecosystems. Therefore, a conscious use of planetary resources is a prerequisite for further development and will be addressed in the context of SO of wooden building components.

1.1. State of Research

While structural systems with high material utilization have been an essential practice throughout history, starting from the early 19th century up until now, economically efficient building materials have been highly available for modern construction [3]. Studies [4] show that at the current time, the overdesign of structural systems is common practice, with the ease of construction often valued higher than the time consuming design of efficient structural systems. Much can be learned from historic constructions in terms of material efficiency, yet the implementation of state-of-the-art high-tech approaches could lead to even higher material utilization in building components. Since the beginnings of development of SO techniques in the 1870s, with pioneering works of Maxwell [5] and Michell [6], advances in SO algorithms, starting off in in the 1970s, are attributed to Berke and Khot [7], Bendsøe and Kikuchi [8] and Bendsøe and Sigmund [9], with increasing advancements up until today. While the so called deterministic optimization methods, including the works on gradient based methods like the Method of Moving Asymptotes (MMA) and Optimal Criteria (OC), are effective approaches when it comes to well defined problems [10], probabilistic optimization methods, e.g. genetic algorithm (GA), show a wide range of applications and perform well in black box problems where the function of a problem is not known [11].

Following, exemplary presented literature for SO strategies will give an overview on the field, with the authors further referring to a comprehensive summary [27] performed in advance. Even though SO methods are widely adopted within commercial software, practical implementations mostly involve linear isotropic material models [12]. Since the orientation of an anisotropic material is crucial for the high utilization of a component, the research on SO with anisotropic materials, e.g. laminated composite as found in high tech industries, usually focuses on optimization of the orientation with common SO techniques being the optimization of ply layup [13], element orientations [14] or the combination of orientation and density [15]. When it comes to the SO of anisotropic construction materials like wood-based materials, literature can be found on the layout and sizing optimization of flooring systems [16], frames [17] and beams [18]. SO techniques with potentially higher utilizations are found in regard to the optimization of varying strength grade [19], SO of the middle layer [20] or continuous cross section optimization [21] of glue laminated timber beams or in SO approaches for wooden slab elements [22]. While the SO of orientation is researched on trusses [23], [24] or robotically manufactured timber structures [25], [26], accessible approaches on the orientation optimization within a solid building component made from wood are currently not known to the authors. Based on the results of prior literature [27] approx. 80% of the literature on SO of anisotropic materials focuses on the development of algorithms and numerical studies of optimizations tasks. Only 5% of the presented research was conducted on the development of prototypes or other experimental constructions. This current state of limited practical implementations can be explained by the low accessibility of algorithms. Furthermore, the assessment shows that 8% and 18% of the optimization algorithms and frameworks presented within literature were found to be directly or partially accessible through open-source code, respectively. Researchers working on practical implementations of SO building components would benefit greatly from accessible algorithms, giving the presented research within this paper a high value.

1.2. General Idea and Choice of Platform

This study proposes an accessible framework, the Structural Timber Optimizer (STO) to optimize the structural performance of wooden building components. STO unifies strategies found in high-tech industries in a multi-functional and easily adoptable framework for use in the construction industry. The presented SO framework can be implemented to optimize for example the orientation of OSB panels in a multi-layered wall element or in and adapted version also for other anisotropic multi-layered building materials like CLT and for different building components. Taking advantage of meshing algorithms

found in commercial FE software, STO discretizes a 3D model in smaller elements needed for the FE structural analysis, as shown in Figure 1. Through optimization with a genetic algorithm (GA), ideal material orientations in each element can be generated and further interpolated to manufacturable elements. As conventional engineered wood products (EWP) like cross laminated timber (CLT) make use of alternating 0° and 90° orientation in the different layers of a wall element, this reference orientation will be used to benchmark the optimized wall element. With the high accuracy of the FE analysis, the material model can be verified comparing experimental tests to the simulated environment and be later also used for precise dimensioning.

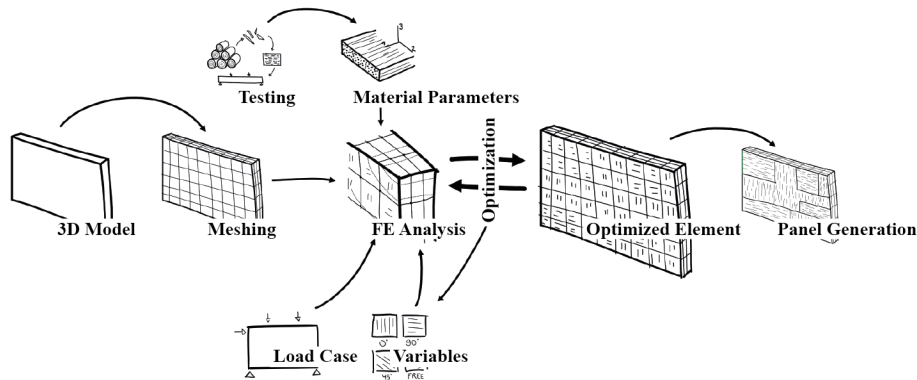


Figure 1: Process sketch of the Structural Timber Optimizer (STO).

Within several case studies the authors compared different platforms and structural analysis tools on their applicability for SO, as shown in Table 1. While Rhinoceros 3D (Rhino) and its visual scripting language Grasshopper are commonly used platforms among researcher in the field of architecture engineering and construction (AEC), plugins for structural analysis, like Karamba3D are lacking key features e.g. the interaction between two solid elements found to be crucial for the presented use case. Other tested software like RFEM and FEM Design in connection with Grasshopper were similarly limited when it came to solid elements and had a low efficiency in combination with Grasshopper. Since Grasshopper and Rhino are widely adopted in AEC and integrate well in existing workflows, efforts were made to develop the framework in Grasshopper, with the limitations in terms of structural analysis leading to further development in the programming language Python. This allowed the integration of ABAQUS as the structural analysis software including its large set of capabilities, native Python environment and a large online documentation and community.

Table 1: Platforms and structural analysis software tested for the STO framework.

	Karamba 3D	FEM Design	RFEM	ABAQUS
Programming Language	Grasshopper Native	API: C#, Grasshopper	API: Python, C#, Grasshopper	Python 2 Native Scripting
Structural Elements	1D / 2D	1D / 2D	1D / 2D / 3D *	1D / 2D / 3D
Efficiency	high	low	low	Medium
Accuracy	Low	Medium	Medium	High
Ease of Use	High	Low	Low	High

*Interaction through contact solids.

2. Structural Timber Optimizer (STO)

The STO framework is build up from four different modules, as shown in Figure 2, consisting of 3D modeling, structural modeling, structural analysis, as well as optimization. It is developed in the programming language Python 3 [28] and integrates the packages CadQuery [29] and PyGAD [30] for

modeling and optimization as well as the ABAQUS [31] environment for structural modeling and analysis.

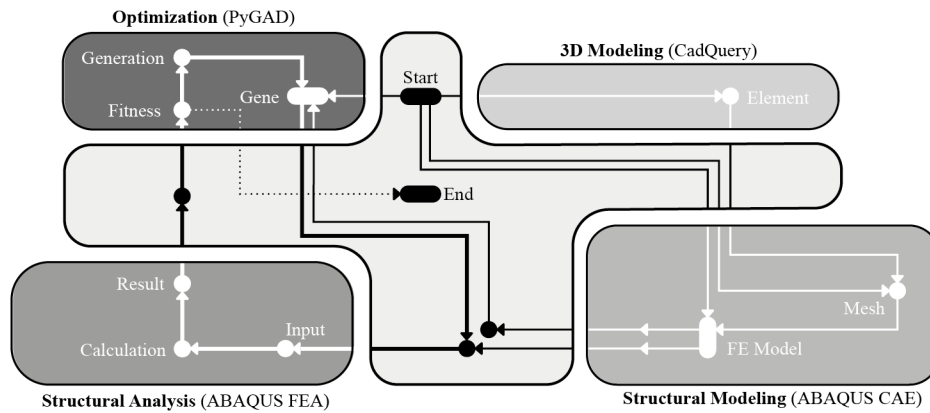


Figure 2: Framework modules of the Structural Timber Optimizer (STO).

After setting the optimization parameters (Start), dimensions of the components are passed to the 3D modeling module that generates a .step file with the 3D model (Element). This element is meshed (Mesh) in the structural modeling module used in combination with information regarding load case and material to generate a FE model (FE Model). The amount of elements in the mesh are passed to the optimization module as the amount of variables (Gene). The initial variables are set by the algorithm and passed down, together with the FE model, to the structural analysis module as input (Input). The model is calculated (Calculation) and the result exported (Result) to be put back in the optimization loop of the optimizer (Fitness) generating a new set of variables (Generation). The algorithm stops if convergence is met, e.g. the best result is found (End).

2.1. Process Walkthrough STO

Following, a detailed description of every step of the process accompanied by a graphical representation through pseudocodes and accessed modules will be given.

The optimization with STO starts with the 3D Model generation, as shown in Figure 3, setting the initial geometric parameters that define the structural element (marked with a red **(a)** Figure 3). The current boundaries are set to rectangular wall elements with rectangular openings but could be modified to support multiple geometric shapes. Implementing the Python package CadQuery, the algorithm iterates through the amount of layers in the wall element, each time offsetting the z-coordinate with the layer thickness and creating box volumes with the provided wall dimensions **(b)**. If data for an opening is given, the previously created box is subtracted with a box with the size and position of the opening **(c)**. The geometry is subsequently exported as a .step file that can be read by ABAQUS. In addition, an info file with the geometry setup information is generated.

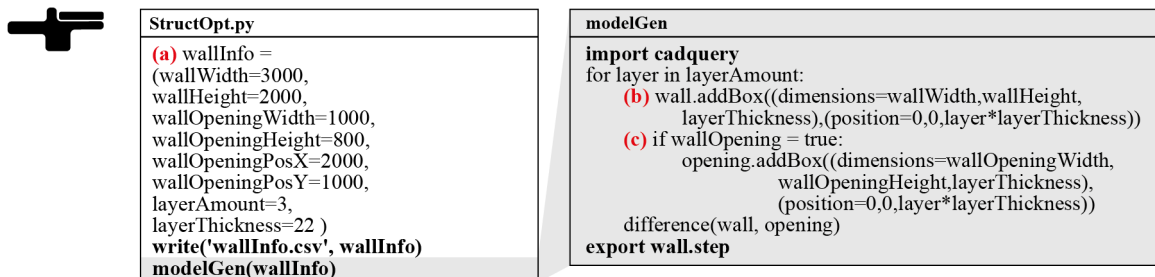



Figure 3: 3D Model generation in the STO framework.

As ABAQUS runs with an older version of Python and does not support packages needed for the complete optimization framework workarounds for communication were developed. While ABAQUS


natively reads Python commands, a direct communication between the IDE and ABAQUS is not available, yet the capability of ABAQUS to execute script over the command line is used. Within the STO framework ABAQUS is called, as shown in Figure 4 (d), passing the input script in a subprocess without opening the graphical user interface (GUI), with the additional possibility of running the full GUI while developing the model. Since the script is not run in the IDE directly but in a separate process, debugging can only take place each time the script is run, therefore line by line debugging is not possible. After reading the element info file, ABAQUS iterates through the layers of the element meshing the given geometry into elements (e). The latter is performed through different algorithms using a variety of mesh types, e.g. triangles or quads, as well as different mesh sizes. For every element of the mesh a set is generated that can receive further information on material or orientation. The number of elements is then written in an info file with geometrical information on position of the boundary nodes of each element also added if necessary. In a next step the sets of nodes and surfaces are generated (f). For the given wall element, lateral and vertical loads are subjected to the nodes on the top surface while supports are added to the nodes at the bottom surface (g). The set generation iterates through the mesh nodes of the element and appends them to a list if the y-coordinate equal to the height of the wall or zero. Sets of contact surfaces between the layers are defined by iterating through the surfaces positioned on a multiple of the layer thickness before the interactions between the surfaces are created (h). Subsequently, the to-be used materials are defined, either by choosing one material for the complete model or assigning different materials or material grades for each element. At the end of this initializing module the .cae file is saved to further receive additional parameters from the optimizer in the following steps.



StructOpt.py command = abaqus cae -noGUI AbqInit.py (d) subprocess.Popen(command)	AbqInit.py import abaqus wallInfo = read('wallInfo.csv') (e) for layer in layerAmount: part = model.PartFromGeometryFile(wall.step) part.generateMesh for element in part.elements: part.Set(element) write(elementInfo.csv, len(part.elements)) (f) for part in parts: for node in part.nodes: if node.coordinate[1] == wallHeight: loadEdge.append(node) elif node.coordinate[1] == 0: boundEdge.append(node) for surface in part-surfaces: if surface.coordinate[2] in contactCoordinates: contactSurface.append(surface) (g) model.ConcentratedForce(loadEdge, cf1=hLoad / len(loadEdge), cf2=vLoad / len(loadEdge)) model.DisplacementBC(boundEdge, (u1, u2, u3, ur1, ur2,ur3)=0) (h) for iteration in range (0, layerAmount -1) model.SurfaceToSurfaceContactStd(main=surfaceA, secondary=surfaceB) model.materials['Wood'].Elastic(type=constants) model.HomogeneousSolidSection(Wood) part.SectionAssignment(Wood) mdb.saveAs('AbqInit.cae')
---	---

Figure 4: Structural model generation in the STO framework.

In the next step the genetic algorithm (GA) provided in the PyGAD package is initialized as shown in Figure 5. With the GA having its origins in evolution as found in nature [32], evolutionary terms will be further used within the description. After reading the information regarding the amount of finite elements generated within the structural modeling module, the number of elements is passed to the number of genes parameter (i). The genome consists of a list of variables that define different variations of the structural element. In the presented case the variables describe the material orientation in each mesh or finite element. Herefore the algorithm takes a list defining the range of possible variables, also called the gene range. For the first optimization runs, discrete integer variables are allowed with values of 0, 45 or 90 degrees (j).



```
StructOpt.py
import pygad
read elementInfo.csv
#Set Optimization Parameters
(i) for gene in range (0, elementSum)
    geneRange.append([0,45,90])
gaInstance = pygad.GA(
    num_generations=40,
    num_parents_mating=30,
    fitness_func=fitnessFunction,
    fitness_batch_size=50,
    sol_per_pop=100,
    (i) num_genes=elementSum,
    gene_type='int',
    gene_space=geneRange,
    parent_selection_type='sss',
    keep_parents=1,
    crossover_type='single_point',
    mutation_type='random',
    mutation_percent_genes=10,
    on_generation=onGeneration)
gaInstance.run()
```

Figure 5: Initialization of the genetic algorithm in the STO framework.

The GA allows for a flexible definition of variables with multiple gene ranges of continuous and discrete values. To limit the search space and reduce variables, only the orientation is optimized in the first trials, with an addition of material grade variables possible for increased optimization potential. Following the definition of the `gene_space` and `solution_per_pop` parameters, additional parameters are needed for the setup of the GA, e.g. population size, describing the amount of individuals or genomes as well as the number of generations. It was found that a genome with a length of 1830 genes, a population size of 100, and analysed for over 40 generations, showed good results. Additional parameters were left to default as found in the PyGAD documentation. As the definition of the fitness function is the last critical part of the initialization of the algorithm, it will be described in the next section as part of an iteration through the lifecycle of one generation. Concluding the GA instance would now be ready and could be initiated with the `.run()` command.

The start of every new generation is defined through the creation of a new genome, containing the variables that are defining, e.g. material orientation. In the PyGAD package the genome is called “solution”. The first generation starts by generating the solution either as a random or predefined set of variables called initial population. While the standard fitness function is called for every solution of the population, the PyGAD batch fitness function can be called for a multiple of solutions, increasing efficiency as all solutions in one generation can be accessed at once. Shown in Figure 6 (k), every solution in the population is exported as a `.csv` file and ABAQUS is called, passing the `AbqInp.py` file. For every solution the following process is executed in a loop. After a solution is generated, the orientation variables are passed to the corresponding elements (l) and an input file is produced. Thereafter a new ABAQUS calculation job is generated and the calculation is run again (m). After the calculation is completed, the `.odb` output file is read and the corresponding field data of each node is exported (n). This process allows for the generation of various outputs like displacement or stress of each node element over multiple or only the last increment. After iterating through the solutions, the main script appends the calculated value to a list and returns the list as the fitness values to the batch fitness function (o). The algorithm stops when the convergence criterion is met. Convergence criteria can be freely defined, e.g. as a specified number of generations or if a desired fitness value is reached. The fitness of every solution is then appended to a list containing the generation and solution number and exported.

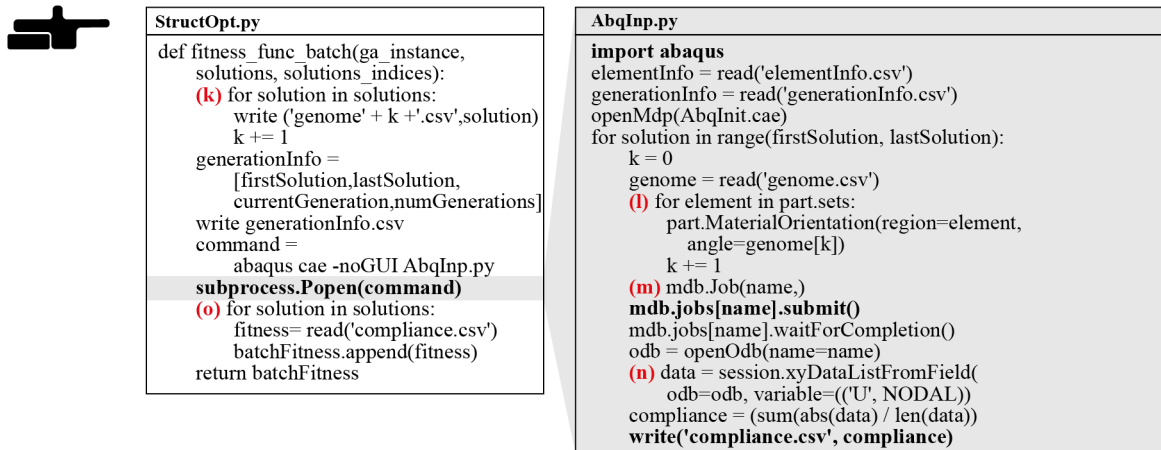


Figure 6: Iterating in the STO framework.

3. Optimized Element and Material Model

The results of the first optimization with the STO framework are presented and compared to the reference element with a corresponding material orientation in the layers of 0° , 90° , 0° . The model consists of a three-layered wall element with dimensions of $3000 \times 2000 \times 66$ mm and a rectangular opening of 1000×800 mm. The element, fixed in all directions ($U_1 = U_2 = U_3 = UR_1 = UR_2 = UR_3 = 0$) at the nodes on the bottom surface, was submitted to a lateral and a vertical load of 200kN to the nodes of the top surfaces. The optimization parameters included the allowed orientations in every FE element to take values of 0° , 45° , 90° , with solutions per generations set to 40 and generations until convergence set to 100. In total the runtime of the algorithm was 35.2 hours equaling to about 32s per calculation. The comparison between the reference wall element and the optimized solution, as presented in Figure 7 showed an approx. 50% lower mean displacement.

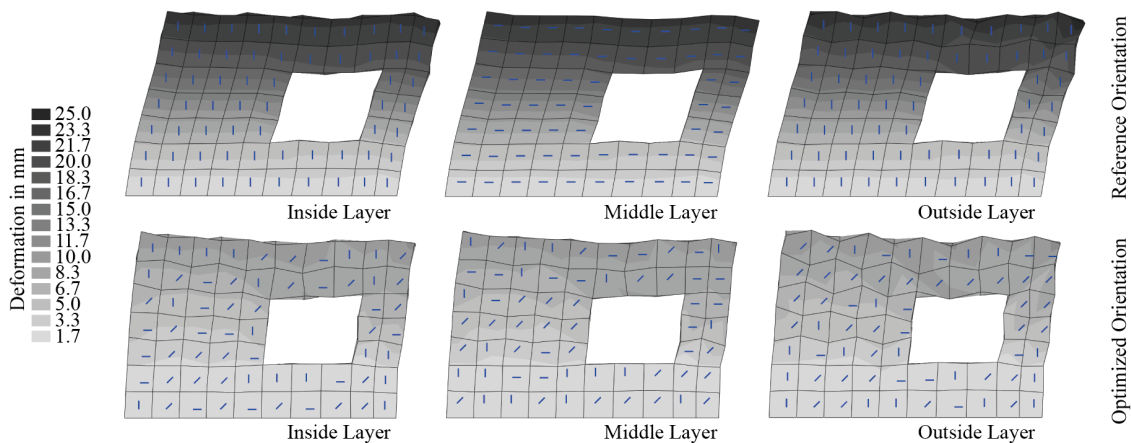


Figure 7: Comparison of a wall element with reference and optimized orientations.

While the structural analysis for the optimization was calculated with a coarse FE mesh to increase the efficiency, the accuracy of the structural behavior can be validated through refining the material model in a separate process. Using the same environment of ABAQUS, a four-point bending test, as presented in Figure 8, was modeled to validate the material model by comparing the reaction forces on the load introduction in the state of maximum displacement. While the first simulation of the four-point bending tests were successful, the reaction forces at 35mm displacement in the simulated environment were found to be approx. 3 times higher than the average force found in experimental testing, showing that the model is still in need of improvement.

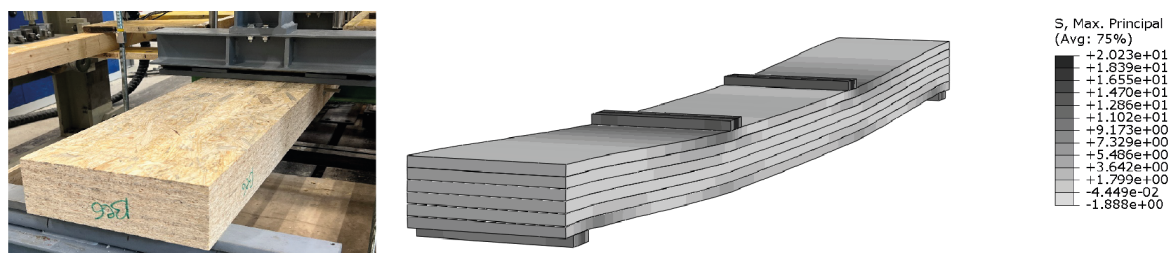


Figure 8: Validation of the material model in ABAQUS

4. Discussion and Conclusion

As presented in the beginning of this paper, an efficient material utilization of bio-based building structures is necessary to mitigate climate change and reduce our impact on the environment. While other scholars have presented various ways to optimize wooden beams or trusses, an accessible framework on optimizing the material orientation of wall and slab elements made from solid timber elements was not found in literature. This research gap has now been filled with the Structural Timber Optimizer (STO). This framework opens new possibilities by integrating state-of-the-art commercial FE software for the calculation as well as open-source libraries and other modules in a Python framework. This not only extends the range of possible applications but also increases accessibility through the large documentations provided for the mentioned modules. The possibility of simulating real world experiments in the ABAQUS environment broadens the range of practical applications since the material model can be refined and validated. While this process can produce reliable results through detailed calculations, the efficiency is rather low. With the efficiency increase of the STO through batch fitness calculation among other parameters, calculation times for one solution amount to around 32s. The large search space, a result from combinations of the different orientations and material grades for every element, challenges the optimizer to find the best solution. Efforts should be made to either decrease calculation times of each calculation or increase the efficiency of the optimization algorithm. The high amount of data generated from detailed calculations could be used for prediction-based models as a way of reducing calculation times. While the first optimization shows a lowering of more than 50% in terms of mean compliance, investigations and statistical analysis of the performance of the algorithm with different parameters needs to be done to find the most influential optimization parameters. Additional steps should be taken towards processing orientation data into manufacturable elements, finding the most relevant load combinations and boundary conditions as well as testing on objectives with minimum compliance in combination with mass reduction. In the scope of this project the authors will further develop and statistically verify the most influential parameters in the STO framework and investigate the practical applications, including experimental testing and comparison of a structurally optimized building component to conventional variants.

Acknowledgements

This project is funded by the Austrian Forest Fund, an initiative of the Austrian Federal Ministry of Agriculture, Forestry, Regions and Water Management, and is carried out as part of the Think.Wood program of the Austrian Wood Initiative (FFG 893351). The financial contribution by the company partners is gratefully acknowledged.

Data Availability

The code is available from the corresponding author upon reasonable request.

References

- [1] K. Calvin *et al.*, "IPCC, 2023: Climate Change 2023: Synthesis Report. Contribution of Working Groups I, II and III to the Sixth Assessment Report of the Intergovernmental Panel on Climate

- Change [Core Writing Team, H. Lee and J. Romero (eds.)]. IPCC, Geneva, Switzerland.,” Intergovernmental Panel on Climate Change (IPCC), Jul. 2023. doi: 10.59327/IPCC/AR6-9789291691647.
- [2] “2022 Global Status Report for Buildings and Construction: Towards a Zero-emission, Efficient and Resilient Buildings and Construction Sector.,” United Nations Environment Programme (2022), Nairobi, 2022.
- [3] T. C. Jester, *Twentieth-Century Building Materials: History and Conservation*. Getty Publications, 2014.
- [4] J. Orr, M. P. Drewniok, I. Walker, T. Ibell, A. Copping, and S. Emmitt, “Minimising energy in construction: Practitioners’ views on material efficiency,” *Resources, Conservation and Recycling*, vol. 140, pp. 125–136, Jan. 2019, doi: 10.1016/j.resconrec.2018.09.015.
- [5] J. C. Maxwell, “I.— *On Reciprocal Figures, Frames, and Diagrams of Forces*,” *Trans. R. Soc. Edinb.*, vol. 26, no. 1, pp. 1–40, 1870, doi: 10.1017/S0080456800026351.
- [6] Michell, “LVIII. The limits of economy of material in frame-structures,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 8, no. 47, pp. 589–597, Nov. 1904, doi: 10.1080/14786440409463229.
- [7] L. Berke and N. S. Khot, *Use of Optimality Criteria Methods for Large Scale Systems*. Air Force Flight Dynamics Laboratory, 1974.
- [8] M. P. Bendsøe and N. Kikuchi, “Generating optimal topologies in structural design using a homogenization method,” *Comput Methods Appl Mech Eng*, vol. 71, no. 2, pp. 197–224, Nov. 1988, doi: 10.1016/0045-7825(88)90086-2.
- [9] M. P. Bendsøe and O. Sigmund, “Material interpolation schemes in topology optimization,” *Archive of Applied Mechanics*, vol. 69, no. 9, pp. 635–654, Nov. 1999, doi: 10.1007/s004190050248.
- [10] S. Y. Wang, K. Tai, and M. Y. Wang, “An enhanced genetic algorithm for structural topology optimization,” *International Journal for Numerical Methods in Engineering*, vol. 65, no. 1, pp. 18–44, 2006, doi: 10.1002/nme.1435.
- [11] S. Y. Yuen, Y. Lou, and X. Zhang, “Selecting evolutionary algorithms for black box design optimization problems,” *Soft Comput*, vol. 23, no. 15, pp. 6511–6531, Aug. 2019, doi: 10.1007/s00500-018-3302-y.
- [12] N. Stoiber and B. Kromoser, “Topology optimization in concrete construction: A systematic review on numerical and experimental investigations,” *Struct Multidisc Optim*, vol. 64, no. 4, pp. 1725–1749, Oct. 2021, doi: 10.1007/s00158-021-03019-6.
- [13] X. Peng, M. Wang, B. Yi, J. Li, H. Wu, and S. Jiang, “Optimization design of stacking sequence and material distribution for variable thickness hybrid composite structure based on improved stacking sequence table,” *Composite Structures*, vol. 307, p. 116641, Mar. 2023, doi: 10.1016/j.compstruct.2022.116641.
- [14] Y. Muramatsu and M. Shimoda, “Distributed-parametric optimization approach for free-orientation of laminated shell structures with anisotropic materials,” *Struct Multidisc Optim*, vol. 59, no. 6, pp. 1915–1934, Jun. 2019, doi: 10.1007/s00158-018-2163-4.
- [15] Y. Lu and L. Tong, “Concurrent optimization of topologies and fiber orientations for laminated composite structures,” *Composite Structures*, vol. 295, p. 115749, Sep. 2022, doi: 10.1016/j.compstruct.2022.115749.
- [16] P. Jelušič and S. Kravanja, “Optimal Design and Competitive Spans of Timber Floor Joists Based on Multi-Parametric MINLP Optimization,” *Materials*, vol. 15, no. 9, Art. no. 9, Jan. 2022, doi: 10.3390/ma15093217.
- [17] D. N. Kaziolas, G. K. Bekas, I. Zygomas, and G. E. Stavroulakis, “Life Cycle Analysis and Optimization of a Timber Building,” *Energy Procedia*, vol. 83, pp. 41–49, Dec. 2015, doi: 10.1016/j.egypro.2015.12.194.
- [18] M. Simón-Portela, J. R. Villar-García, D. Rodríguez-Robles, and P. Vidal-López, “Optimization of Glulam Regular Double-Tapered Beams for Agroforestry Constructions,” *Applied Sciences*, vol. 13, no. 9, Art. no. 9, Jan. 2023, doi: 10.3390/app13095731.

- [19] S. Pech, G. Kandler, M. Lukacevic, and J. Füssl, “Metamodel assisted optimization of glued laminated timber beams by using metaheuristic algorithms,” *Engineering Applications of Artificial Intelligence*, vol. 79, pp. 129–141, Mar. 2019, doi: 10.1016/j.engappai.2018.12.010.
- [20] A. F. de Vito, W. M. Vicente, and Y. M. Xie, “Topology optimization applied to the core of structural engineered wood product,” *Structures*, vol. 48, pp. 1567–1575, Feb. 2023, doi: 10.1016/j.istruc.2023.01.036.
- [21] P. Mayencourt and C. Mueller, “Hybrid analytical and computational optimization methodology for structural shaping: Material-efficient mass timber beams,” *Engineering Structures*, vol. 215, p. 110532, Jul. 2020, doi: 10.1016/j.engstruct.2020.110532.
- [22] H. Hofmeyer, M. Schevenels, and S. Boonstra, “The generation of hierarchic structures via robust 3D topology optimisation,” *Advanced Engineering Informatics*, vol. 33, pp. 440–455, Aug. 2017, doi: 10.1016/j.aei.2017.02.002.
- [23] B. Kromoser, M. Braun, and M. Ortner, “Construction of All-Wood Trusses with Plywood Nodes and Wooden Pegs: A Strategy towards Resource-Efficient Timber Construction,” *Applied Sciences*, vol. 11, no. 6, p. 2568, Mar. 2021, doi: 10.3390/app11062568.
- [24] Y. Gao, Y. Shao, and M. Akbarzadeh, “Application of Graphic Statics and Strut-and-Tie Models Optimization Algorithm in Innovative Timber Structure Design,” *Buildings*, vol. 13, no. 12, Art. no. 12, Dec. 2023, doi: 10.3390/buildings13122946.
- [25] R. Naboni and A. Kunic, “A computational framework for the design and robotic manufacturing of complex wood structures,” in *Blucher Design Proceedings*, Porto, Portugal: Editora Blucher, Dec. 2019, pp. 189–196. doi: 10.5151/proceedings-ecaadesigradi2019_488.
- [26] H. Chai, H. J. Wagner, Z. Guo, Y. Qi, A. Menges, and P. F. Yuan, “Computational design and on-site mobile robotic construction of an adaptive reinforcement beam network for cross-laminated timber slab panels,” *Automation in Construction*, vol. 142, p. 104536, Oct. 2022, doi: 10.1016/j.autcon.2022.104536.
- [27] J. Belz and B. Kromoser, “An Analysis of Established Practices in the Structural Optimization of Wooden Building Components and Anisotropic Materials. [Manuscript submitted for publication],” 2024.
- [28] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.
- [29] D. Cowden, “CadQuery Documentation”.
- [30] A. F. Gad, “PyGAD: an intuitive genetic algorithm Python library,” *Multimed Tools Appl*, Dec. 2023, doi: 10.1007/s11042-023-17167-y.
- [31] *AbaqusCAE User’s Guide*. Dassault Systèmes Simulia Corp, 2016.
- [32] S. Katoch, S. S. Chauhan, and V. Kumar, “A review on genetic algorithm: past, present, and future,” *Multimed Tools Appl*, vol. 80, no. 5, pp. 8091–8126, Feb. 2021, doi: 10.1007/s11042-020-10139-6.