Simulating Bone-like Strain Adaptation in Space Trusses

Abstract

In biomechanics, Wolff's law posits that spongy bone is bolstered where internal strain is greatest, and vice versa, producing an anisotropic trabecular structure with superior strength-to-weight performance. In the interest of applying this advantage to building components or entire structures, the present paper pursues a biomimetic application of Wolff's law to space trusses made up of linear beam elements. Two algorithms are compared, the most successful of which reduces mechanical strain in trusses under several different initial conditions.

1 Introduction

This research seeks generative strategies for space trusses mimicking the remodeling process in trabecular bone tissue. Speculative benefits of being able to produce this kind of porous geometry include improved strength-to-weight performance and stress resistance of volumetric building components; greater inter-visibility and permeability of such components; and the potential for them to be used as a scaffolding for other embedded materials.

German anatomist Julius Wolff (1836-1902) posited that human bone adapts to physical loading by strengthening where loads are greater, and vice versa. "Wolff's Law" has since been experimentally verified to a high degree of precision (Maurer et al. 2015; Tsubota et al. 2009). When force is applied to the bone matrix, mechanical loads are translated into biological signals through a process called "mechanotransduction" (Mullender et al. 2004). Where bone is stretched or compressed, it is more likely to be remodeled and to form thicker bone.

The actual cycle of bone remodeling involves at least three kinds of cells (Figure 1).

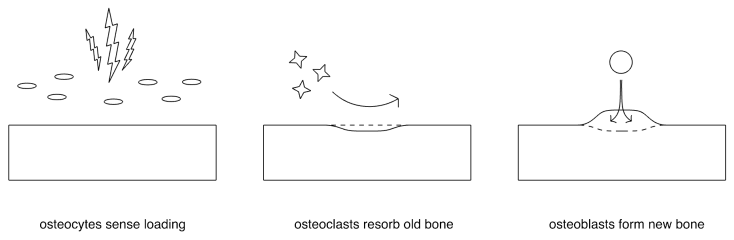


Figure 1. Cycle of Bone Remodelling.

Over time this process rearranges bone tissue such that the matrix of cancellous bone aligns with cyclic loadings (Figure 2).

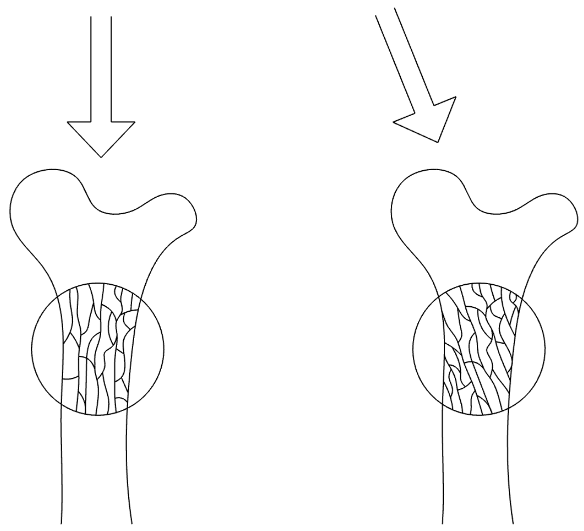


Figure 2. Bone Anisotropy in Response to Loading.

There is precedent for mimicking the bone formation process in simulation, especially in medical fields (Boyle and Kim 2011). Most simulation approaches employ *FEA* (finite element analysis), although there are some exceptions (Velasco, Lancheros, and Garzón-Alvarado 2016). FEA attempts to solve a large system of equations to derive mechanical properties like strain at given points within a geometry subject to certain forces. Since bone remodeling is likely driven by mechanical impulses like strain (Mullender et al. 2004), there must be some framework to predict or calculate strain at key points in the geometry.

In this case we abstract the spongy matrix of bone as a network of essentially linear elements connecting a collection of points, for the sake of computational tractability.

It is worth noting that topology optimization for strain minimization has been implemented in numerous ways in the aerospace and other engineering fields (Zhang et al. 2016). However, bone remodeling is optimized for cyclical strains rather than static loads. Moreover, bone remodeling is intrinsically an agent-based process, with bone cells as the agents, unlike most optimization algorithms which iteratively assess global metrics. These facts favor a different approach to mimicking bone remodeling, and encapsulate the novel potential of this research compared to existing tools. To distinguish the two approaches, we call the present strategy "proxy-optimization."

In earlier iterations, our software pipeline employed Karamba, a structural analysis plugin for Grasshopper, to conduct FEA. The recursive step, shown as a doubled-back arrow in

Figure 3, was accomplished with the plugin Anemone. The latest version of the pipeline uses C# components with direct calls to an FEA library, instead of Karamba and Anemone, in order to reduce dependency on proprietary or poorly-documented plugins. While we use Millipede for the isosurface mesh reconstruction, the technique (the Marching Cubes algorithm) is open source (Lorensen and Cline 1987). The complete latest pipeline is illustrated below (Figure 3).

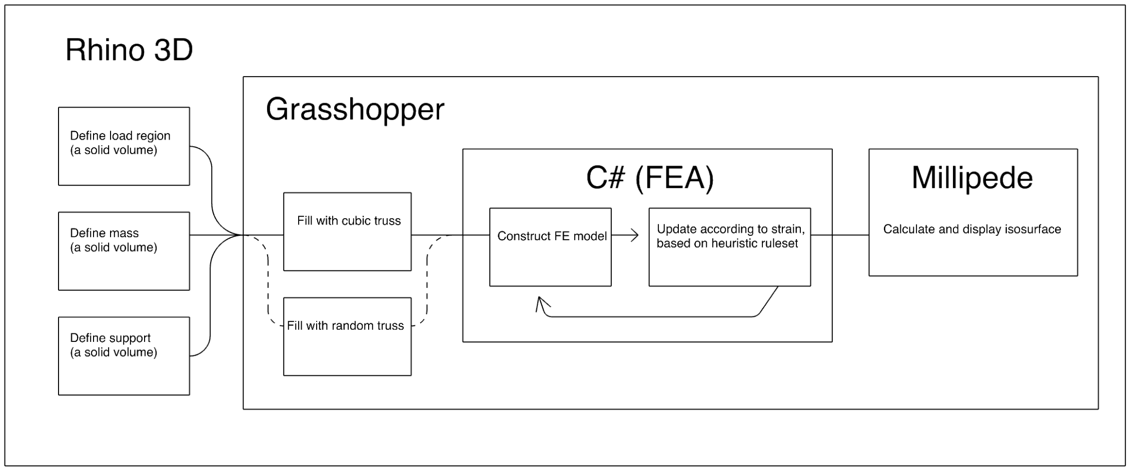

Rhino 3D
Grasshopper
Define load region (a solid volume)
Define mass (a solid volume)
Define support (a solid volume)
Fill with cubic truss
Fill with random truss
C# (FEA)
Construct FE model
Update according to strain, based on heuristic ruleset
Millipede
Calculate and display isosurface

Figure 3. Software Pipeline.

The following section outlines our setup for an FEA-driven topology proxy-optimization algorithm. It is designed to approximate solid geometry as a space-filling truss, though theoretically it can be approximated as any kind of finite element geometry (beam, shell, voxel, etc). The subsequent section demonstrates a use case by comparing the performance of two different applications of this method.

2 Methods

Both algorithms examined had the same basic structure of three stages:

1) The first stage takes as input three solid regions, ‘load’, ‘mass’, and ‘support’ (Figure 4). The mass region is the domain to be filled with a truss of beam-connected nodes. Any distribution and connectivity of nodes can be used, but for the following examples a random distribution is chosen. In this case, their connectivity is determined by generating the 3d Voronoi diagram; two nodes are considered connected if the Voronoi cells containing them share a face. This helps limit connections between relatively distant pairs of nodes, an intuitive constraint on most trusses where beams can only be made so long.

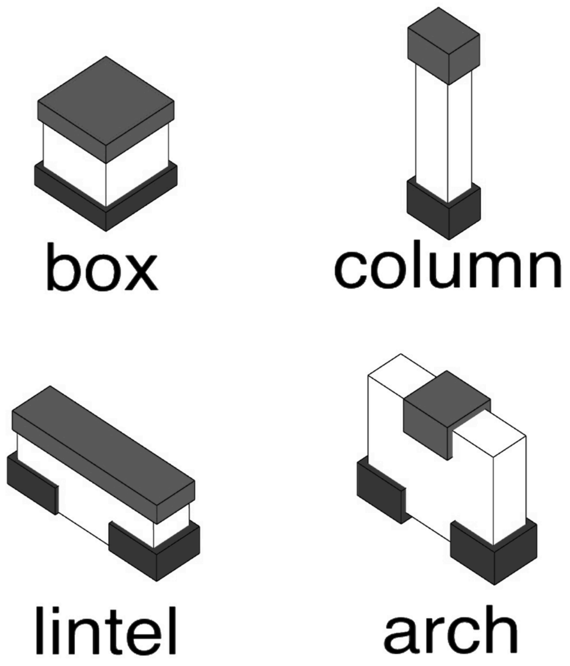


Figure 4. Step 1: Defining Load, Mass, and Support Regions.

2) The next portion of the algorithm begins by solving the finite element model of this truss for physical properties in each node and beam. This information then informs how the model should be changed. This cycle of analyzing and updating is iterated a user-specified number of times. Two different heuristic approaches to the update step were implemented, each described here separately:

   a) *Subtractive*: This heuristic method removes a user-specified number of beams with the highest maximum stress. This is comparable to evolutionary structural optimization (ESO) methods of topology optimization (Zhang et al. 2016). It is illustrated in Figure 5 acting on the arch test geometry, discussed in the following section, over the course of 100 iterations. A sample of 20 of these iterations are arrayed left to right, top to bottom.

   b) *Bidirectional*: This heuristic method adjusts the cross-sections of the beams, thickening those most stressed and thinning those least stressed. At the end of the set number of iterations, a fraction of the thinnest beams are removed. This is comparable to bidirectional evolutionary structural optimization (BESO) methods of topology optimization, which instead iteratively change the density of solid elements as opposed to the cross-section of beam elements.

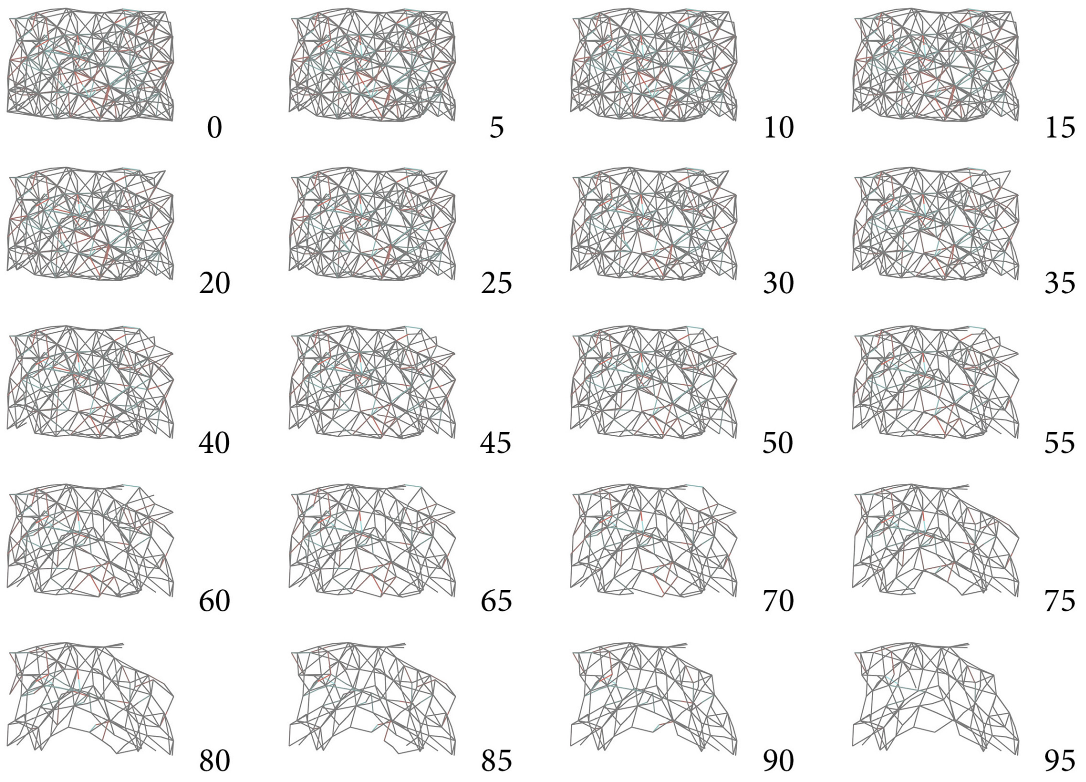


Figure 5. Step 2 Proxy-optimization (subtractive heuristic).

3) After the iterative process is completed, the abstract points and lines defining the nodes and beams can be reinterpreted and processed as solid geometry to suit the needs of visualization and manufacture. The algorithm discussed here uses the Millipede plugin to generate mesh representations of the beams as prisms. Optionally, the output can also be viewed as an isosurface of the wire network, as in Figure 6. Several inputs (Isovalue, Spread, Power, Strength) allow this isosurface mesh to be tailored, affecting the vector field defining the isosurface and hence how much of a buffer is formed between the mesh and the actual geometry. For example, a smaller isovalue amounts to a mesh that is further displaced from the geometry, smoother, and more globular. It should be acknowledged that approximation via isosurface may have unforeseen effects on mechanical performance compared with that of the idealized truss model.

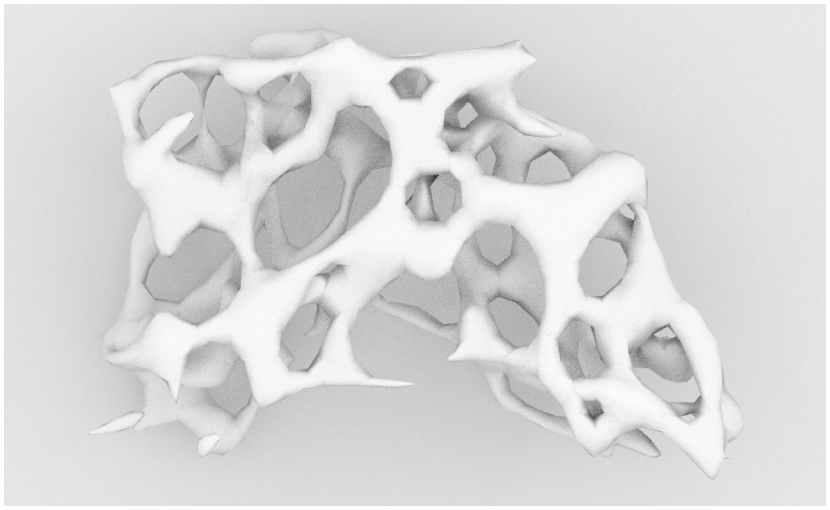


Figure 6. Step 3: Solid Visualization.

3 Results

Early tests showed signs that the algorithm obeyed intuitions, removing material from the least-stressed regions of input geometry, leaving behind only the regions subjected to the greatest strain. Figure 7 shows a simple rectangular prism mass region that was subjected to different angular loads incident on its top, with the vertical axis showing different fractions of the initial material remaining.

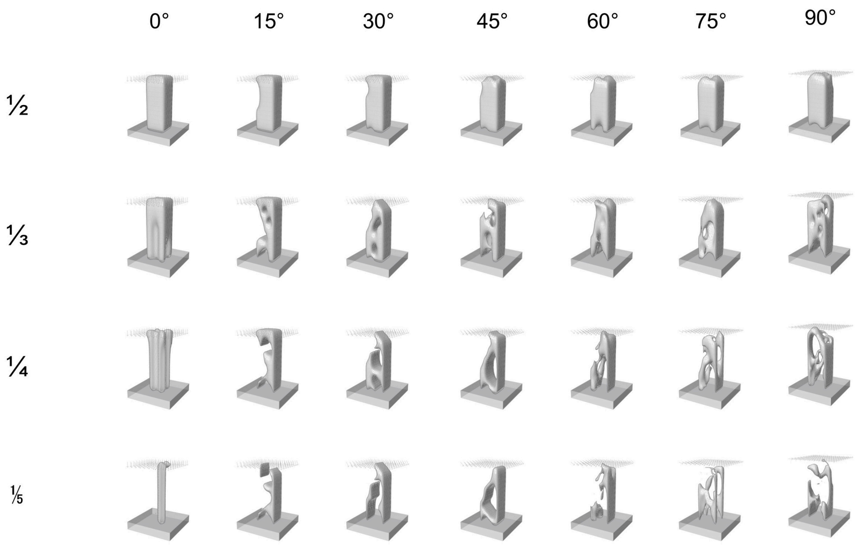


Figure 7. Early Tests.

Algorithm performance was compared for four different test geometries, referred to as box, column, lintel, and arch for brevity and clarity (see Figure 4). Each test geometry specified load, mass, and support regions. In each case, a 100-node randomized truss was used, a load of 1000 Newtons was applied to each node in the load region, and each node in the support region was treated as having no degrees of freedom, i.e. being fixed in space. Note that the load region covers the top of the geometry in the box, column, and lintel examples, but only applies to the center in the arch example. Likewise, the support covers the base in the box and column examples, but only applies to the sides in the lintel and arch examples. This is evident in Figure 4, and is shown in detail in Figure 8. Arrows represent applied and reaction force vectors.

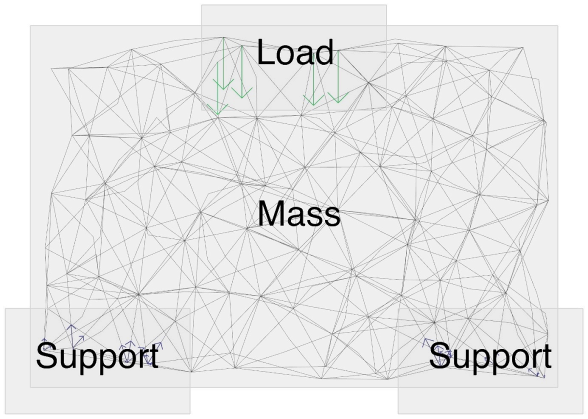


Figure 8. Arch Test Case Load, Mass, and Supports.

These test geometries were run through each algorithm for 400 iterations each. See the Appendix B for graphs of the average peak stress and moment vs. number of iterations for the subtractive algorithm, for each test geometry. Likewise, see Appendix C for quantitative results of the bidirectional algorithm. Visually, in these graphs we observe consistent behaviors across most test cases:

1) Subtractive
    a) Maximum element-wise max stress: decreasing at accelerating rate
    b) Median element-wise max stress: increasing at constant rate
    c) Average element-wise max stress: erratic
2) Bidirectional
    a) Maximum element-wise max stress: decreasing at decelerating rate
    b) Median element-wise max stress: decreasing, then erratic

c) Average element-wise max stress: decreasing at decelerating rate

Qualitatively, the better-performing algorithm is the bidirectional; however, this does not relate stress to the weight of the system, or any number of other metrics that could be taken into account. The actual shape of the resultant trusses is also illuminating. Shown in Figure 9 are isosurfaces generated from the final geometry output by the algorithm.

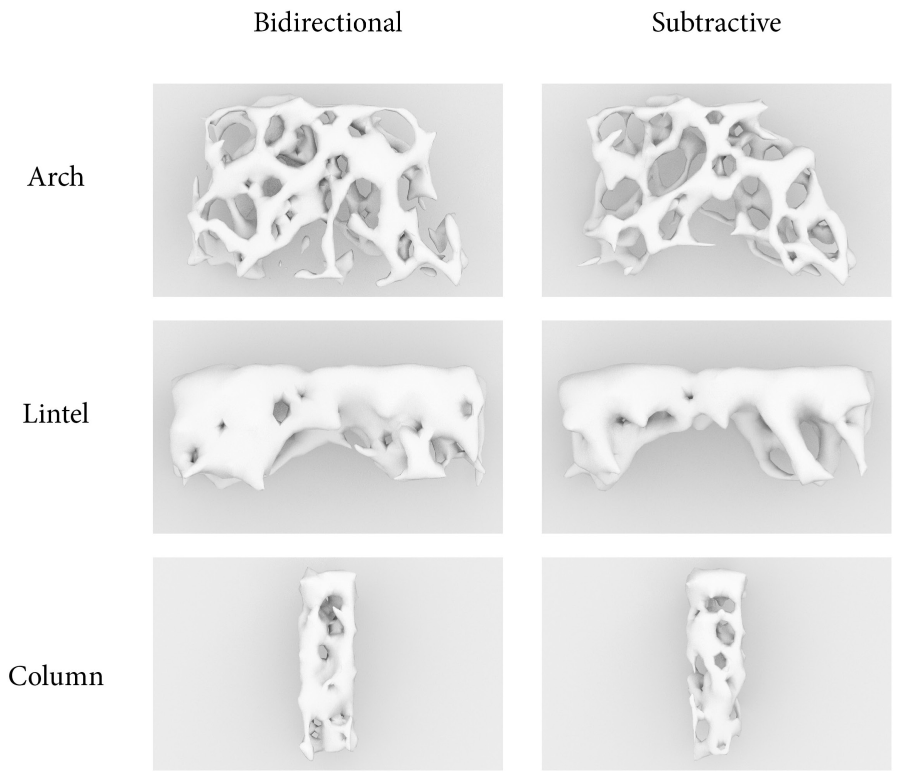


Figure 9. Isosurface Results (column, lintel, and arch).

As we can see in the arch and lintel test cases, from a truss initially randomly filling a rectangular-prism-shaped mass region like those in Figure 4, an arch-like form emerges, with most of the material removed from the center of the base where there are no supports. This evolutionary emergence of the arch form, well known to be optimal in many load-bearing situations, is of interest as a qualitative result, and bears some testament to the intuitive functioning of these heuristics.

A 3d ceramic powder-printed example of the Subtractive Arch geometry is shown in Figure 10, as a proof of concept for future physical prototyping, which might take advantage of the already fertile field of custom 3d-printed ceramics research ([redacted]).

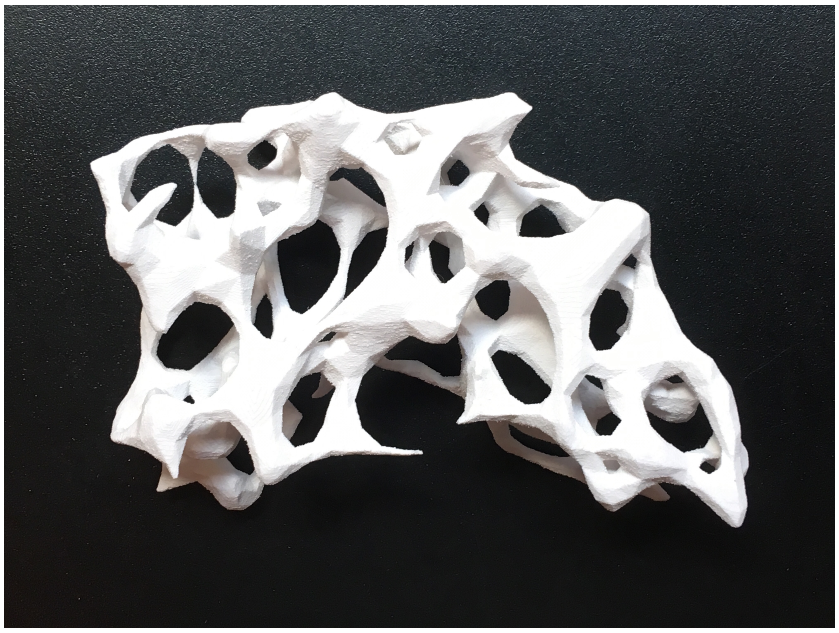


Figure 10. 3d Printed Subtractive Arch Result.

Conclusion

Topology optimization is a readily available tool in architectural and other design fields, but it is often obscured as a backend process in proprietary or poorly-documented plugins (Aage et al. 2014). While this gives confidence of performance since optimization is essentially a solved problem, it precludes experimentation with the algorithm itself. One particular value of this research is its potential to make topology optimization directly editable by designers.

A continuation of this research might proceed with writing from-scratch code to reduce reliance on plugins. While FEA can be quite complex, beam theory and space trusses are among its simplest and best understood applications, suggesting this effort may be viable. Another potential avenue of research is to parametrize the algorithm space itself and optimize those parameters, which could drastically speed the search for a good biomimetic abstraction of bone remodelling processes.

Based on this research so far, a proxy-optimization approach to topology optimization inspired by the mechanobiology of bone remodelling shows promise of combining measurable performance benefits with greater flexibility for the algorithm designer.

Acknowledgements

The author would like to acknowledge the invaluable guidance of [redacted]. Research on [redacted] during the summer of 2018 was generously funded by [redacted]

References

Aage, Niels, Oded Amir, Anders Clausen, Lior Hadar, Dana Maier, and Asbjørn Søndergaard. 2014. "Advanced Topology Optimization Methods For Conceptual Architectural Design". In *Advances In Architectural Geometry 2014*, Springer. 159-179.

Boyle, Christopher, and Il Yong Kim. 2011. "Three-Dimensional Micro-Level Computational Study Of Wolff's Law Via Trabecular Bone Remodeling In The Human Proximal Femur Using Design Space Topology Optimization". *Journal Of Biomechanics* 44 (5): 935-942.

Lorensen, William E., and Harvey E. Cline. 1987. "Marching Cubes: A High Resolution 3D Surface Construction Algorithm". *ACM SIGGRAPH Computer Graphics* 21 (4): 163-169.

Maurer, M.M., Weinkamer, R., Müller, R. et al. "Does Mechanical Stimulation Really Protect the Architecture of Human Bone? A Simulation Study". *Biomechanics and Modeling in Mechanobiology* (2015) 14: 795.

Mullender, M., A. J. El Haj, Y. Yang, M. A. van Duin, E. H. Burger, and J. Klein-Nulend. 2004. "Mechanotransduction Of Bone Cells in Vitro: Mechanobiology Of Bone Tissue". *Medical And Biological Engineering And Computing* 42 (1): 14-21.

[redacted]

[redacted]

Tsubota, Ken-ichi, Yusuke Suzuki, Tomonori Yamada, Masaki Hojo, Akitake Makinouchi, and Taiji Adachi. 2009. "Computer Simulation Of Trabecular Remodeling In Human Proximal Femur Using Large-Scale Voxel FE Models: Approach To Understanding Wolff's Law". *Journal Of Biomechanics* 42 (8): 1088-1094.

Velasco, Marco A., Yadira Lancheros, and Diego A. Garzón-Alvarado. 2016. "Geometric And Mechanical Properties Evaluation Of Scaffolds For Bone Tissue Applications Designing By A Reaction-Diffusion Models And Manufactured With A Material Jetting System". Journal Of Computational Design And Engineering 3 (4): 385-397.

Zhang, Weihong, Piotr Breitkopf, Jihong Zhu, and Tong Gao. 2016. *Topology Optimization In Engineering Structure Design*. London: ISTE Press.

Appendix A

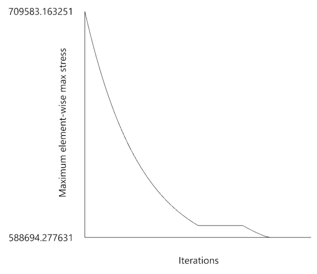
Bidirectional Sample Code (abridged):

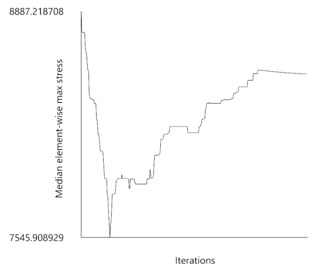
```
    //iteratively update radii of beams

    for (int i = 0; i < iter; i++){
      foreach (RStatBeam b in FEM.Beams)
      {
        if (Math.Abs(b.MaxStress) < lower_threshold)
        {
          double rad = b.CrossSection.rr * (1 - delta);
          StatCrossSection cs = FEM.AddSection(FEM.Materials[0], "");
          cs.CircSolid(rad, b.CrossSection.Segments.Count);
          b.CrossSection = cs;
        }
        else if (Math.Abs(b.MaxStress) > upper_threshold)
        {
          double rad = b.CrossSection.rr * (1 + delta);
          StatCrossSection cs = FEM.AddSection(FEM.Materials[0], "");
          cs.CircSolid(rad, b.CrossSection.Segments.Count);
          b.CrossSection = cs;
        }
      }
      FEM.SolveSystem();
    }

    //remove beams with smallest radii

    beamList = beamList.OrderBy(b => Math.Abs(b.CrossSection.rr)).ToList().GetRange(0,
(int) (cutoff * FEM.Beams.Count));

    foreach (RStatBeam b in beamList)
    {
      FEM.Beams.Remove(b);
    }
```

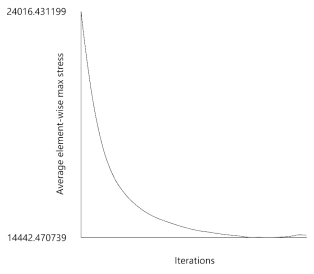
Appendix B

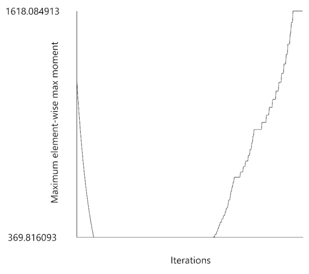
Subtractive Heuristic Results

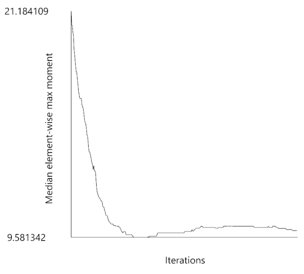
Box:

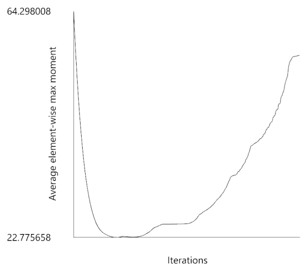




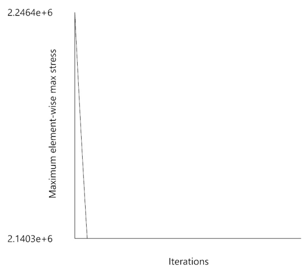


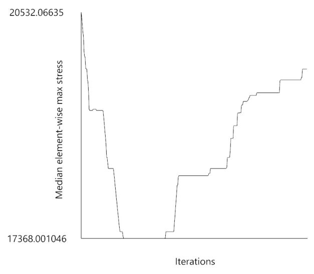


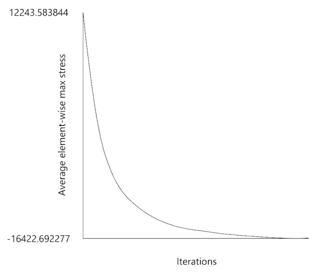


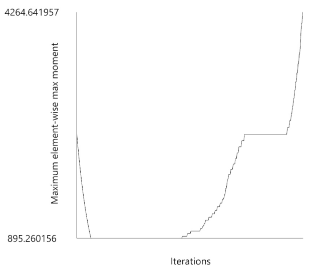


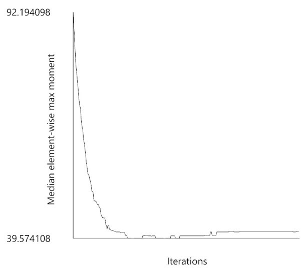
Column:

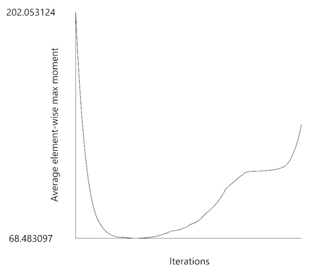




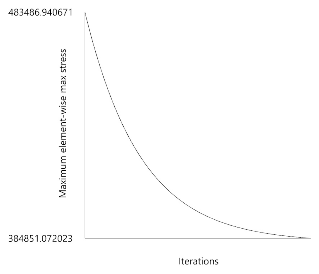
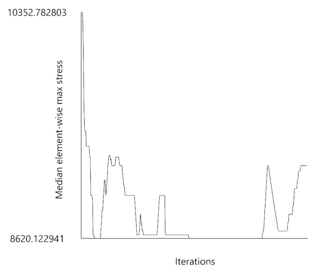
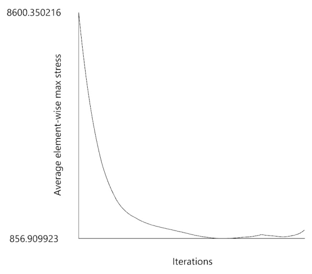
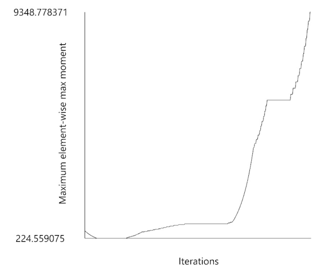
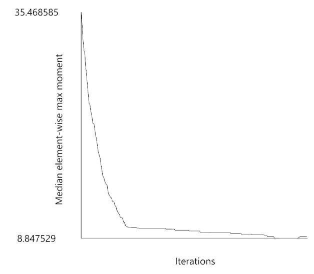
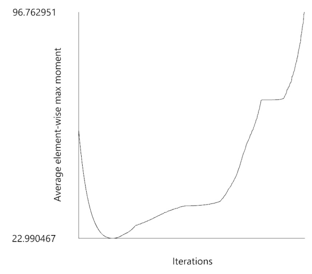




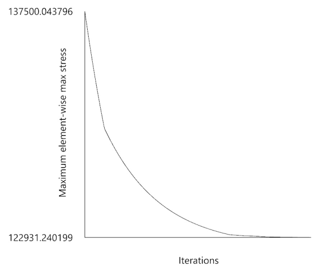
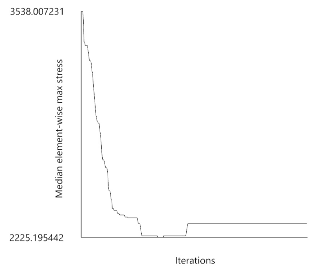
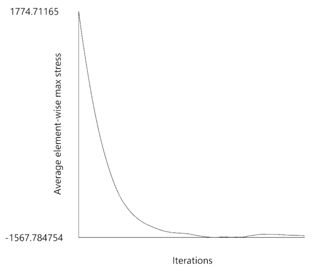
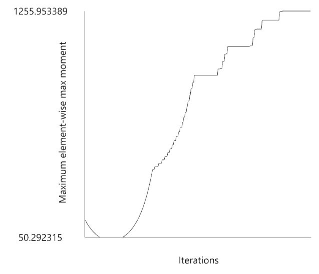
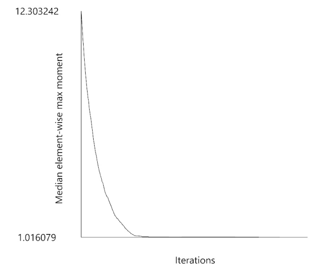
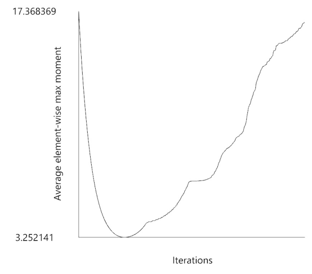




Lintel:


483486.940671
Maximum element-wise max stress
384851.072023
Iterations

10352.782803
Median element-wise max stress
8620.122941
Iterations

8600.350216
Average element-wise max stress
856.909923
Iterations

9348.778371
Maximum element-wise max moment
224.559075
Iterations

35.468585
Median element-wise max moment
8.847529
Iterations

96.762951
Average element-wise max moment
22.990467
Iterations

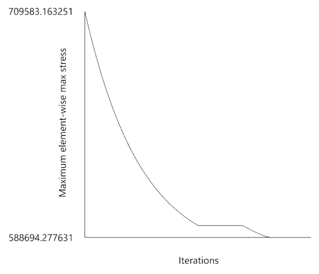
Arch:

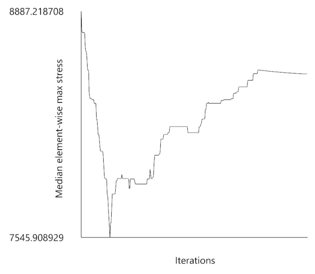

137500.043796
Maximum element-wise max stress
122931.240199
Iterations

3538.007231
Median element-wise max stress
2225.195442
Iterations

1774.71165
Average element-wise max stress
-1567.784754
Iterations

1255.953389
Maximum element-wise max moment
50.292315
Iterations

12.303242
Median element-wise max moment
1.016079
Iterations

17.368369
Average element-wise max moment
3.252141
Iterations

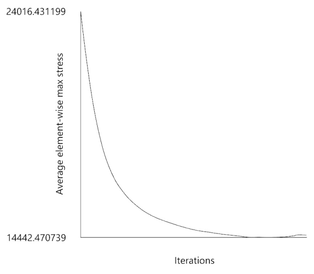
Appendix C

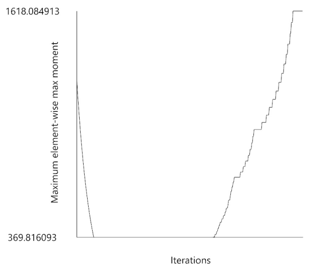
Bidirectional Heuristic Results

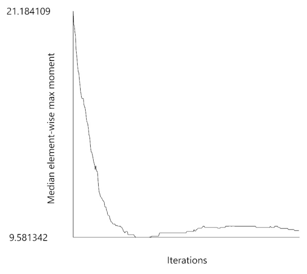
Box:

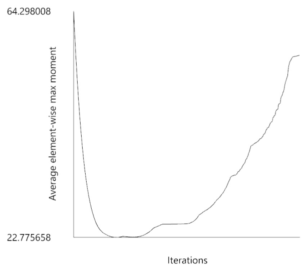




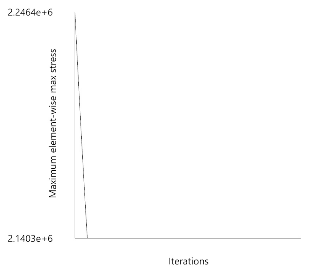


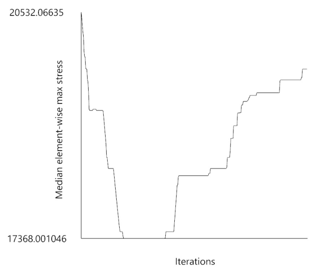


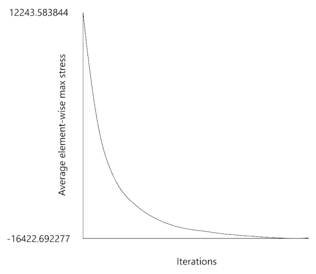


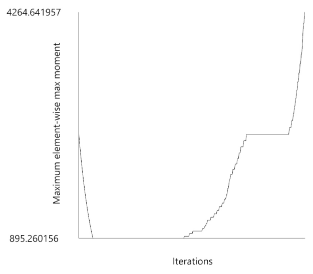


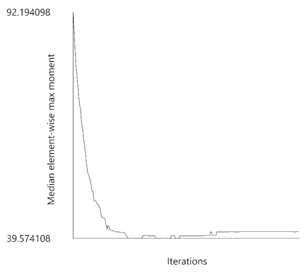
Column:

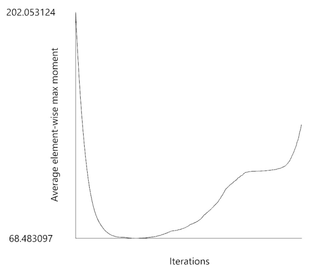




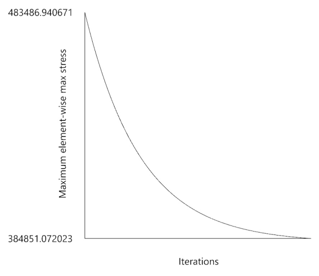
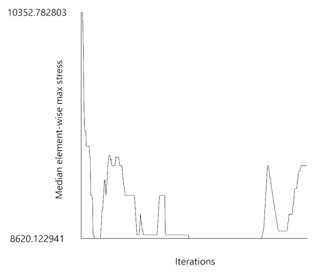
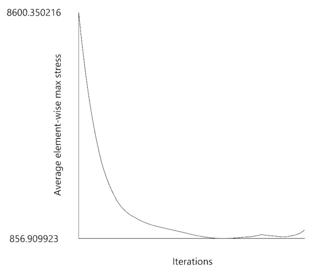
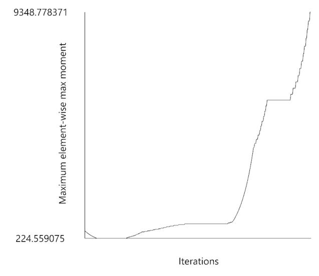
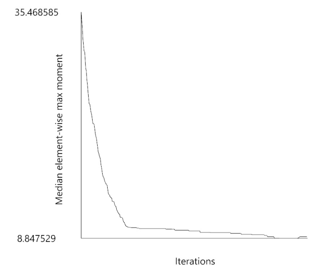
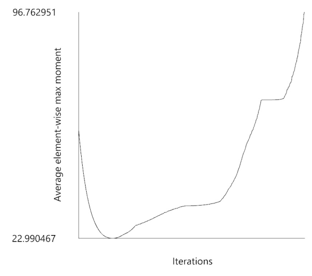




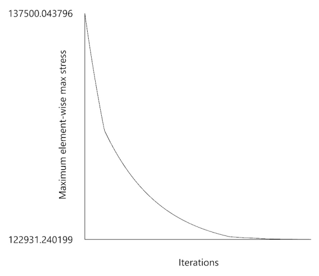
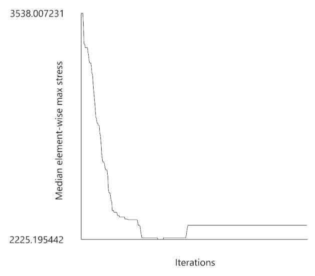
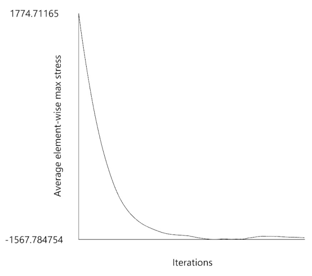
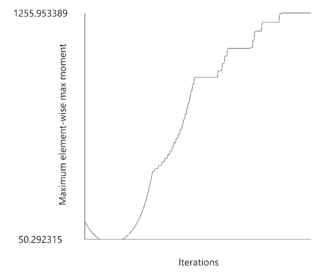
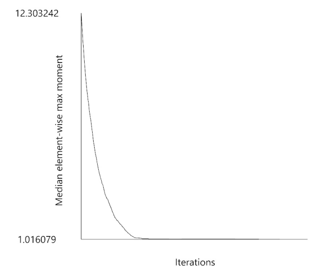
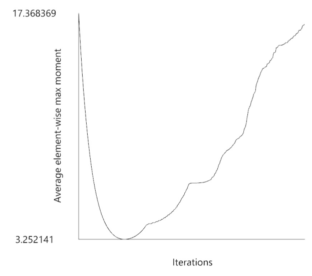




Lintel:


483486.940671
Maximum element-wise max stress
384851.072023
Iterations

10352.782803
Median element-wise max stress
8620.122941
Iterations

8600.350216
Average element-wise max stress
856.909923
Iterations

9348.778371
Maximum element-wise max moment
224.559075
Iterations

35.468585
Median element-wise max moment
8.847529
Iterations

96.762951
Average element-wise max moment
22.990467
Iterations

Arch:


137500.043796
Maximum element-wise max stress
122931.240199
Iterations

3538.007231
Median element-wise max stress
2225.195442
Iterations

1774.71165
Average element-wise max stress
-1567.784754
Iterations

1255.953389
Maximum element-wise max moment
50.292315
Iterations

12.303242
Median element-wise max moment
1.016079
Iterations

17.368369
Average element-wise max moment
3.252141
Iterations