Design Automation & Optimisation for Multi-Material 3D Printing

Benjamin Yi
Supervised by: Dr. John Cater and Emilio Calius

September 14, 2018

1 Abstract

TODO

Contents

List of Figures

List of Tables

2 Introduction

TODO

3D printing advancing, multimaterial possible.
Multimaterial allows for new design dimensions.
Must use heuristics to design new stuff - hence GA, CMA-ES.

3 Model Development

TODO
Overview - with figure.

3.1 Heterogeneous cantilever beam

TODO
Chosen because analytical solutions exist for homogeneous but none for heterogeneous - no closed form solutions.
Natural frequencies are a function of E, rho for fixed geometry.
Some analytical solutions for sandwich, special structures.
Must use eigensolver for true heterogeneous beam.

Cantilever chosen for experimental sake.
Uniformity in natural frequencies for experimental sake.

3.2 Voxelisation

Continuous material allocation within a beam suggests infinitely many potential beams are physically possible, even with finite number of unique materials and finite dimensions. In theory, flexibility of material allocation is limited by the resolution of the 3D printer used. In practice, flexibility is limited by the data storage format of the solution, and the ability to optimise over an infinitely-dimensional solution space. Conventionally, 3D printers input model information through .STL files, which use triangular meshes to define the model. Therefore true continuous optimisation is impossible, as movement to a discrete state (and thus loss of information) is unavoidable. It goes without saying that optimisation over an infinitely-dimensional continuous solution space is simply infeasible in practice.

For these reasons, we discretize the continuous beam into voxels, and assign one material to each voxel instead. Voxels are the discretization of a 3D space into a regular grid, with each voxel representing an identically shaped cube volume of the 3D space. By adjusting the size of the voxels, one can control how closely the voxelisation approximates a continuous space at the cost of computational costs of storing and simulating the model. Figure 1 below demonstrates the voxelisation of a smooth character model - note that while the overall shape is maintained, fine detail is irreversibly lost.

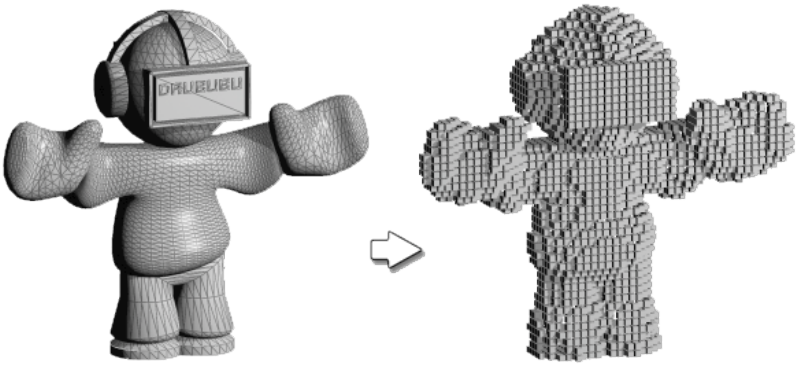

Figure 1: Voxelisation of a character model. Left: Smooth model. Right: Voxelised model.

By assigning one material to each of the n voxels, we reduce the optimisation model down to $n \times m$ dimensions, where m is the amount of information needed to define a unique material. When optimising over a fixed geometry and voxelisation, solutions can then be uniquely defined by a $n \times m$ array.

TODO
For a 40 voxel cantilever beam ...

For a 800 voxel cantilever beam ...

Binary choices

3.3 Heuristics

High level heuristics are used to search the solution space for the allocation of materials that leads to the desired natural frequencies in the beam. A heuristic is defined as a set of low-level rules that allows for automated movement in the search space in a manner that hopefully converges upon the optimal solution. Heuristics do not guarantee obtaining the optimal solution, but can be used to generate solutions of reasonable quality quickly. Because heuristics operate in the solution space, they can be used with "black-box" objective functions, where the objective function is unknown or too complex to analyse.

Heuristics are prone to getting stuck in local optima, where small changes to the solution worsen the objective function value, but better solutions exist elsewhere. Metaheuristics attempt to overcome this by wrapping rules around heuristics to escape from local optima when detected. Well-known metaheuristics include Tabu Search, which maintains a blacklist of forbidden moves, and Harmony Search, which emulates the way musicians compose music.

For our purposes, we required a metaheuristic that could deal with high-dimensionality problems with a complete black-box objective function. The main bottleneck in algorithm speed is the evaluation time for the objective function, as each solution beam must be simulated using a eigensolver

to find the natural frequencies. Thus we seek to minimise the number of objective function calls. Parallelisation of objective function calls was also highly desirable, as they represent increase in algorithm speeds directly proportional to number of computing cores available on the machine running the algorithm.

3.3.1 Genetic Algorithm

To solve the binary voxel case, the genetic algorithm was our metaheuristic of choice. The genetic algorithm (GA) is a metaheuristic that draws inspiration from evolution as it occurs in nature. Solutions are selected by objective function value and the best are merged together to form new solutions. This occurs iteratively, keeping a set of solutions that has an improved average objective function value each iteration. This is analogous to natural selection where a population of individuals breed and grow stronger with each generation, which is highlighted in figure 2. In the 40 voxel beam case and the 800 voxel beam case, solutions are represented by a binary list of length 40 or 800 respectively.

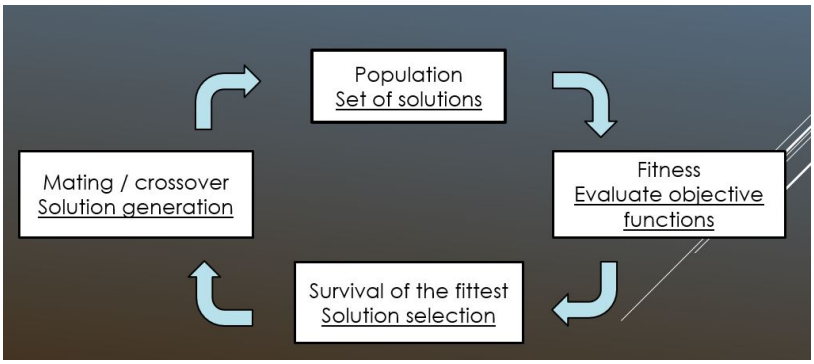

Figure 2: Comparison of GA to natural selection

The merging of good solutions to create better solutions is called crossover. Solutions are split into subsets called features. Crossover mixes the features between two solutions to generate two new solutions, each containing features from both parent solutions. For maximum effectiveness, crossover should be individualised to the problem structure, such that useful features can be extracted from solutions that are representative of a good objective function without information being lost during the extraction. For multimaterial structures, crossover can be implemented by physically separating the structure into features - for example, features of a chair could include the material allocation of the chair legs and chair back. A mutation rate is set that randomly changes some parts of randomly chosen solutions each generation to prevent stagnation of features during the optimisation process.

Because genetic algorithms maintain a set of solutions and generate many new solutions each iteration, they are easily parallelisable. Each iteration, all new solutions need to be evaluated before the next iteration can begin. As the solutions can be evaluated independently of each other, moving from serial evaluation to n-parallel evaluation speeds up the time taken for each generation by a factor of n.

3.3.2 CMA-ES

To solve the continuous voxel case, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) was our metaheuristic of choice. CMA-ES maintains a population of solutions, and updates the solutions each generation by drawing from a multivariate normal distribution. The pairwise variances are updated by estimating the differential of the objective function, in a manner similar to gradient descent. Where genetic algorithms aim to maintain a diverse population of solutions to encourage exploration, CMA-ES tends to have the population converge upon the global optima over time. In the 40 voxel beam case and the 800 voxel beam case, solutions are represented by a list of floats with length 40 or 800 respectively.

CMA-ES is relatively insensitive to starting parameters. Internal parameters evolve over the algorithm's runtime, so the user only needs to specify an initial solution point and initial step size. Because of this, little parameter tuning is required for the algorithm to perform well, at the cost of loss of flexibility and inability to input specific problem strategies. Regardless, CMA-ES is regarded as the de facto standard for large-scale black-box optimisation problems in industry, and has been proven to work well on many different types of problems.

Similar to genetic algorithms, CMA-ES is bottlenecked by the updating of solutions each generation and thus greatly benefits from parallelisation of the objective function.

4 Model Implementation

TODO

4.1 Python / DEAP

TODO
DEAP as an evolutionary framework for Python.
Able to treat solutions as lists of numbers and use black box as objective function.
Able to be parallelised.

4.2 ElmerSolver modelling

TODO
Choice of eigensolver must be opensource, parallelisable.
ElmerSolver runnable headless.
ElmerSolver opensource, parallelisable, has eigensolver.

Mesh resolution
Mesh shape
Quadratic / linear (errors)

4.3 Heuristic parameters

4.3.1 Material allocations

TODO
State materials used

4.3.2 GA parameter values

Metaheuristics are sensitive to parameters set at the start of algorithm runtime. Users often set these parameters in an ad-hoc manner, where parameter values are randomly changed until performance is deemed satisfactory. Alternatively, "good" parameter values are published in journals and used regardless of individual problem structure.

In the genetic algorithm, the three main parameters are mutation rate, crossover rate, and population size. In our genetic algorithm, mutation rate and crossover rate were set at 0.1 and 0.5 respectively

via ad-hoc optimisation of the parameters, with a population size of 40.

4.3.3 Objective function

The objective function attempts to force the natural frequencies of the solution beam to a set of uniformly-spaced frequencies determined beforehand. The first six natural frequencies of the solution beam were extracted from ElmerSolver and matched to a set of six frequencies defined by an initial frequency f_1 and uniform spacing Δf. The L2-norm was used to determine the distance between the natural frequencies and goal frequencies, with the objective function being the minimisation of the L2-norm. The L2-norm (also known as the Euclidean norm) is defined as:

$$||x||_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} \tag{1}$$

Our objective function is then defined as:

$$\min z = \sqrt{\sum_{i=1}^{6} x_i^2} \tag{2}$$

where x_i is the percentage distance between solution beam frequency s_i and target frequency f_i for mode i.

$$x_i = \frac{s_i - f_i}{f_i} \times 100\% \tag{3}$$

4.3.4 Voxel count

TODO

4.3.5 Custom crossover

TODO
Explain standard
Explain both custom

4.3.6 Continuous voxel clustering

TODO
Explain reasoning / usefulness

5 Model Verification

TODO
ElmerSolver export solution.
3D printer used, materials used.
Acoustic testing.
Sensitivity analysis.

6 Experimental Results

TODO
General results – distance from uniformity after one hour of optimisation vs randomly generated solution.
Accuracy of frequencies when physically tested.

6.1 Parameter tuning results

Experiments were ran on the parameters listed in section 4.3. Each parameter was tested independently of each other, and the results are listed below in this section. The default parameters are shown in table 6.1 and are used in each experiment unless explicitly stated otherwise.

Parameter	Default value
Mutation rate	0.5
Crossover rate	0.1
Population size	40
Voxel count	40
Objective function	$f_1 = 75, \Delta f = 150$
Crossover	Ten-point crossover
Clustering	None

Table 1: Default parameters for parameter testing

6.1.1 Population size

The 40 voxel beam case was ran for one hour each across a range of population sizes with the genetic algorithm. Ten runs were made for each population size case, shown below in figure 3.

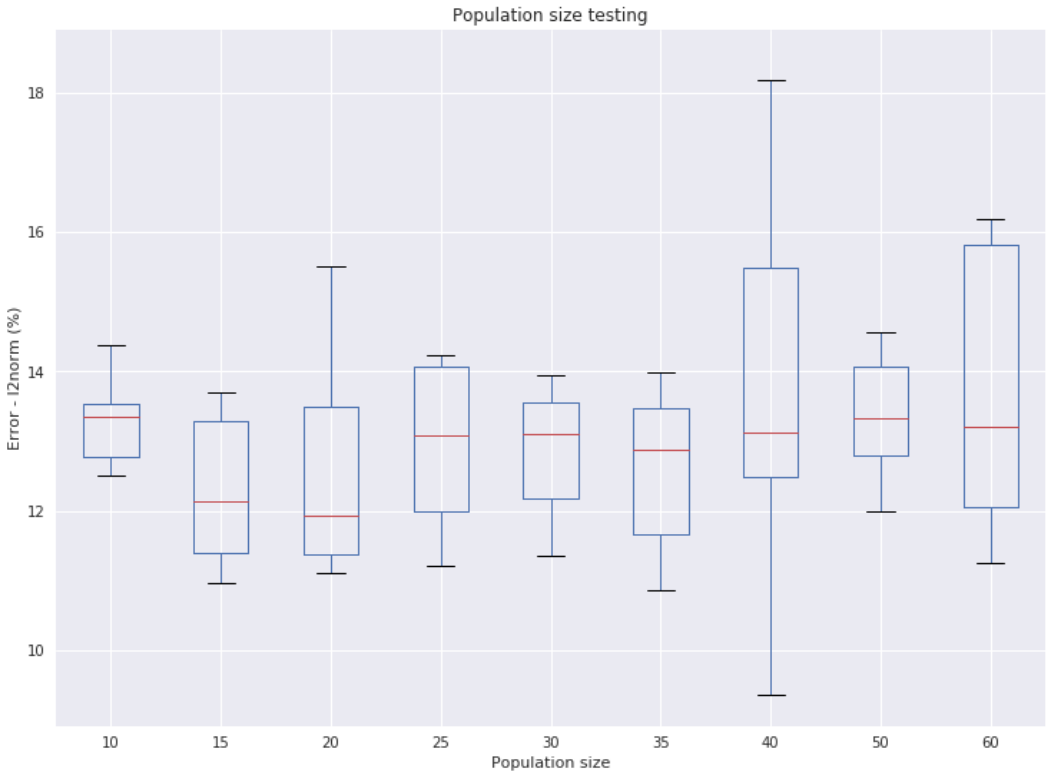

Figure 3: Effect of population size parameter on solution quality

The population size parameter can be grouped into two distinct groups in terms of solution quality. Population sizes of 15 and 20 appear to perform the best, with mean objective function value of 12.5. Population sizes of 10, 25, 30, 35, 40, 50, 60 perform worse, with mean objective function value of 13.6. No clear pattern exists for the spread in sample objective function values within each size group, with the exception of population size 40, whose abnormally large variance is likely due to sampling error.

6.1.2 Voxel count

The 40 voxel beam case and the 800 voxel beam case were ran for three hours each with the genetic algorithm. Thirty runs were made for each voxel count case, shown below in figure 4.

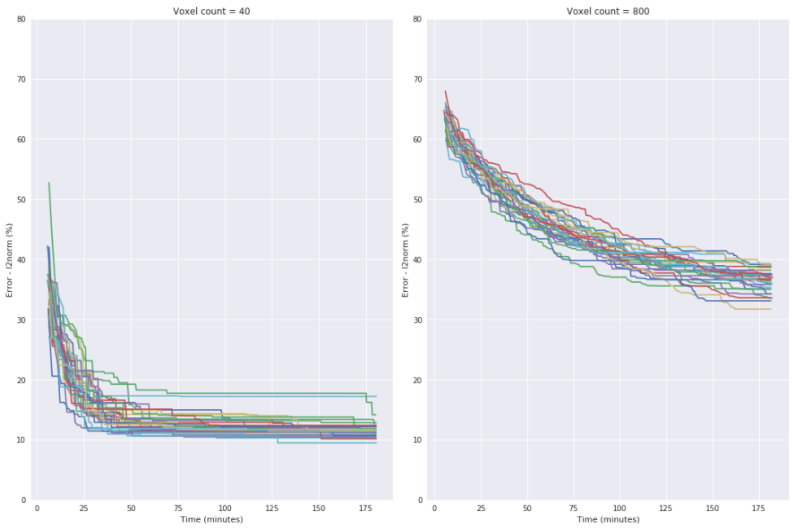

Figure 4: Effect of voxel count in beam on solution quality over time

The 40 voxel beam case performed significantly better than the 800 voxel beam case, with the 40 voxel beam averaging an objective function value of 3.87 in contrast to 800 voxel beam's 13.4. The 40 voxel beam case also reaches local optimality around the one-hour mark, whereas the 800 voxel beam is still continuing at three hours.

The errors were separated by natural mode to compare difficulties in optimising specific modal frequencies, as seen below in figure 5.

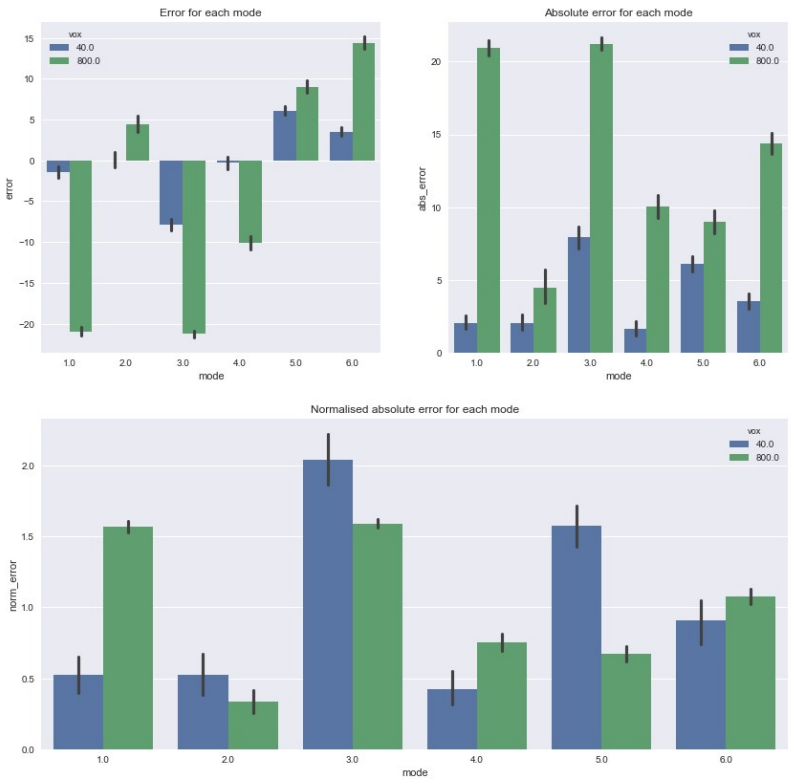

Figure 5: Effect of voxel count in beam on solution quality by mode

Both voxel beam cases have difficulties optimising for mode 3. Modes 2, 4, and 6 are successfully optimised by both beams, with mode 1 being difficult for the 800 voxel beam case and mode 5 being difficult for the 40 voxel beam case. The direction for each mode is consistent across both cases, with modes 1, 3, and 4 being lower than expected and modes 2, 5, and 6 higher than expected.

6.1.3 Uniformity in objective function

The 40 voxel beam case was ran for one hour each across a range of target frequencies as the objective function with the genetic algorithm. The objective function was set to the L2-norm of the relative distance between a set of uniformly-spaced frequencies and the solution beam's natural frequencies. The uniformly-spaced frequencies varied with starting frequency values of [50, 75, 100, 125, 150] Hz and frequency spacing of [50, 100, 150, 200, 250] Hz. Five runs were made for each objective function, with average solution quality shown below in figure 6.

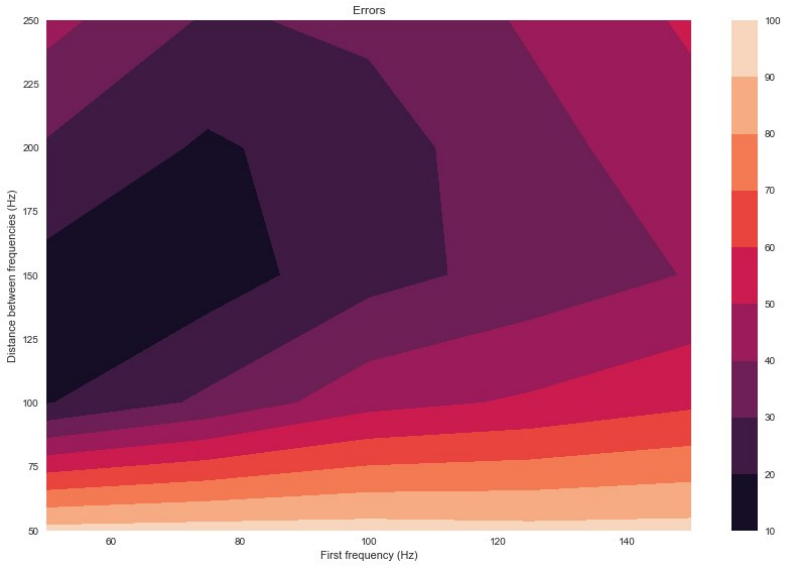

Figure 6: Effect of uniform objective function on solution quality

The set of uniformly-spaced frequencies gives the best objective function value at starting frequency = 75 Hz and uniform spacing of 150 Hz. This has an objective function value of 14.7, with other objective function values shown below in table 2.

grad f1	50	100	150	200	250
50	92.954306	19.358398	16.501543	28.835767	43.330936
75	93.990960	32.000206	14.695473	18.438447	28.659161
100	94.157175	46.362215	26.376269	25.345895	32.033805
125	92.828644	51.415814	33.717527	36.525270	41.194363
150	93.300052	57.947290	40.549948	45.775254	51.483905

Table 2: Effect of uniform objective function on solution quality

6.1.4 Crossover testing

Cross-over testing was performed on both the 40 voxel beam case and the 800 voxel beam case as described in section 4.3. The custom cross-overs were tested against standard two-point crossover for 30 runs each, with no significant difference found, as shown below in figures 7 and 8 below. The

40 voxel beam cases were optimised over one hour, whereas the 800 voxel beam cases were optimised over three hours with the genetic algorithm.

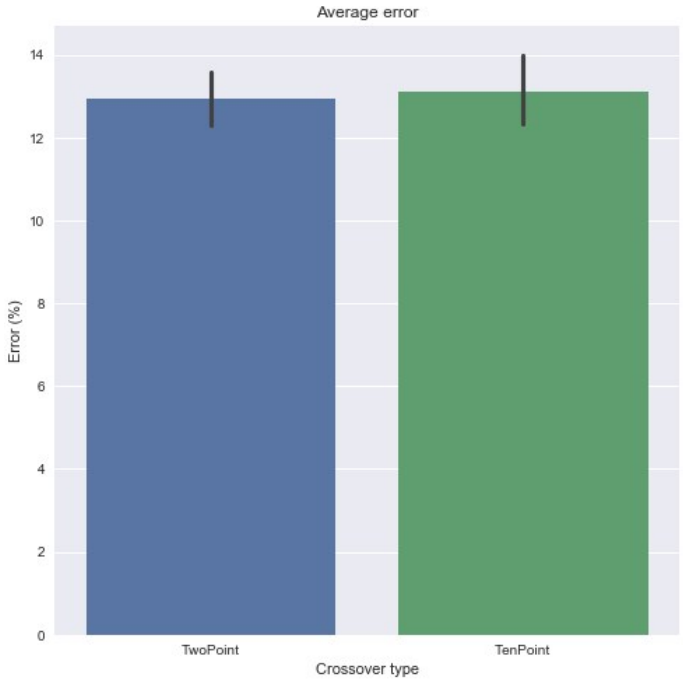

Figure 7: Effect of custom 10-point crossover on solution quality

The 10-point crossover on the 40 voxel beam averaged objective function value of 13.1 $\pm$ 2.4, with the standard 2-point crossover averaging objective function value of 12.9 $\pm$ 1.8.

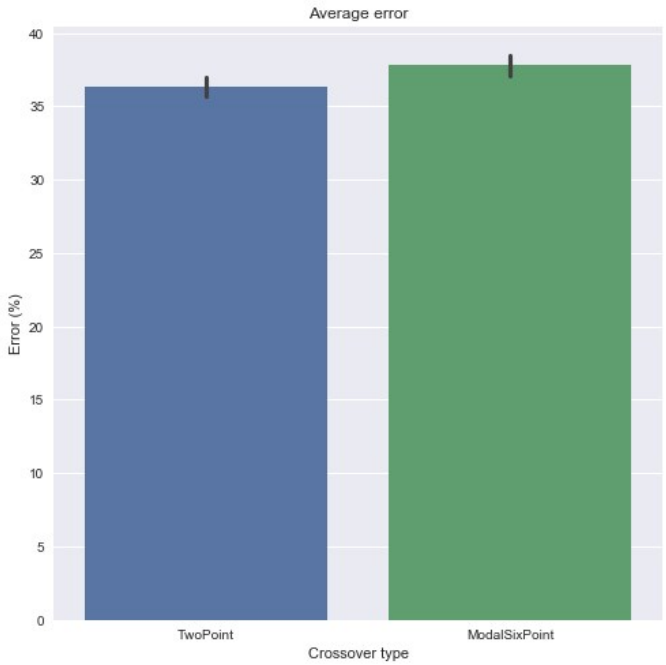

Figure 8: Effect of custom 6-point crossover on solution quality

The 6-point crossover on the 800 voxel beam averaged objective function value of 37.8 ± 2.1, with the standard 2-point crossover averaging objective function value of 36.4 ± 1.9.

6.1.5 Voxel clustering

The effect of voxel clustering for the continuous-case optimisation with CMA-ES was tested by taking known solutions and clustering within each solution. The loss in objective function value for different numbers of clustering centres is shown below in figure 9.

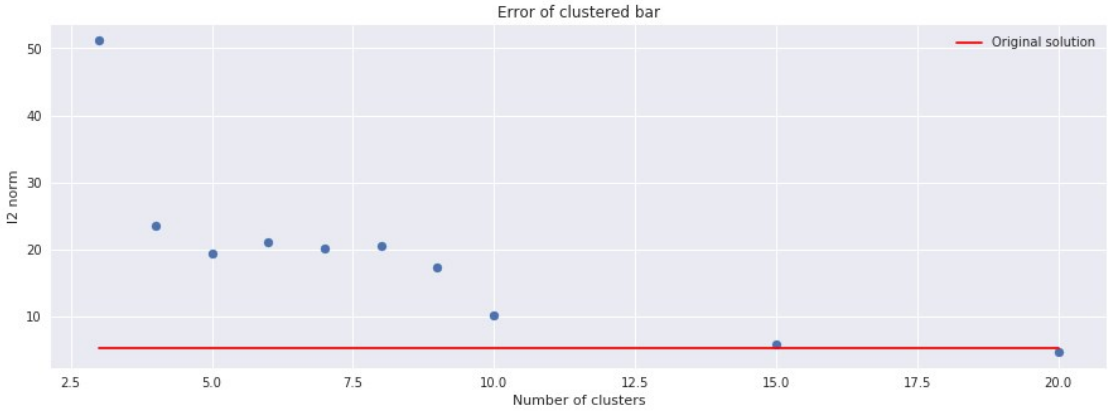

Figure 9: Effect of clustering on solution quality

Solution quality stabilizes from ? clustering centres onwards at 20% above the original objective function value. With number of clustering centres above ?, the solution has recovered its original objective function value, indicating the minimal important information has been lost in the clustering process.

6.2 ElmerSolver analysis

TODO

6.2.1 Meshing ?

TODO
Time to get solution.
Accuracy of solution.
Sensitivity of solution.

6.3 3D print testing

TODO

6.3.1 Sensitivity, accuracy ?

TODO
Sensitivity of printables.
Accuracy of printables.
Feasibility of printables.

7 Discussion

TODO
Optimal parameters for GA, specific to problem?
Need for proper cross-overs to take advantage of GA strengths.

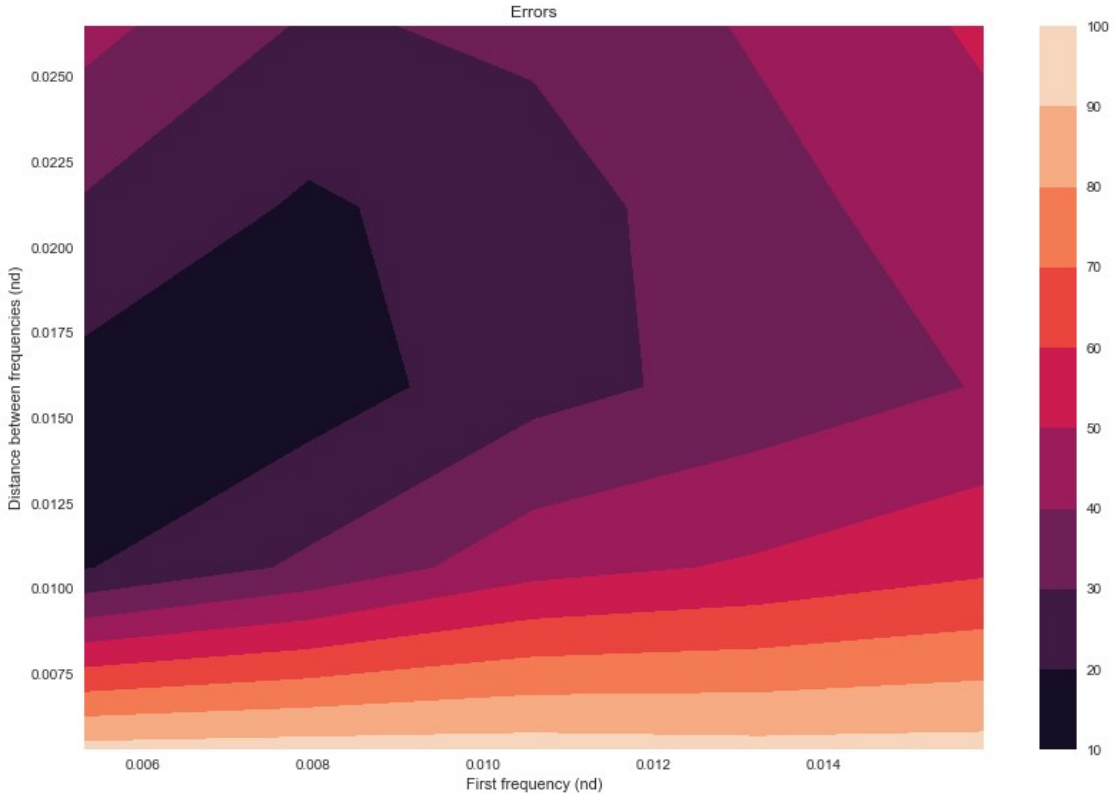

Figure 10: Effect of voxel count in beam on solution quality by mode

Difficulties in 3D printing.
Agreement between printing and modelling.
Accuracy in measurements.

8 Conclusions

TODO
GA successful, can get good results in short time for simple cantilever beam.
ElmerSolver useful, simulation results verified by experimental tests.

Problem scalable, computation limited by number of cores.
Complicated structures can take advantage of crossover for GA.
CMA-ES results suggest another, more accurate, pathway at the cost of feasibility.

We have successfully proven the potential of GA to solve problems and next step is to prove the generalisation.

9 Appendices