DEVELOPMENT OF ABAQUS-MATLAB INTERFACE FOR DESIGN OPTIMIZATION USING HYBRID CELLULAR AUTOMATA AND COMPARISON WITH BIDIRECTIONAL EVOLUTIONARY STRUCTURAL OPTIMIZATION

by

Alen Antony

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Mechanical Engineering

Department of Mechanical and Energy Engineering

Indianapolis, Indiana

December 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Andres Tovar, Chair

Department of Mechanical and Energy Engineering

Dr. Khosrow Nematollahi

Department of Mechanical and Energy Engineering

Dr. Xiaoping Du

Department of Mechanical and Energy Engineering

Approved by:

Dr. Jie Chen

I would like to dedicate this thesis to The Father, The Son and The Holy Spirit.

ACKNOWLEDGMENTS

First, I would like to thank God Almighty for giving me this opportunity and guiding me throughout my life. I believe without him it wouldn't have been possible for a simple person like me to complete this thesis and graduate with a Master's in Mechanical Engineering.

I would like to thank my advisory, Dr. Andres Tovar, for providing me with this opportunity to work on a research topic that helped me achieve so many milestones in my life in the automotive industry. His invaluable guidance, in-depth knowledge in the field of Optimization and have helped me to a very great extent in completing this research. It was a privilege to work with him.

I would like to thank my committee members, Dr.Khosrow Nematollahi and Xiaoping Du for being a part of my defense committee and for the invaluable knowledge they have imparted to me during my master. Dr. Nematollahi's deep understanding of Non-Linear Finite Element Analysis and Dr. Du's deep understanding of Probabilistic Engineering Design and the knowledge I gained from them not only helped me in this thesis but also helps me during my work in the industry.

I would also like to offer my sincere gratitude to Homero Valladares,Dr. Sajjad Raeisi and Joel Najmon for their guidance and the willingness to share their knowledge with me, patiently; which helped me a lot to complete this research.

I would like to thank Indiana University Purdue University Indianapolis, The Purdue Graduate School and the Department of Mechanical and Energy Engineering for providing me with adequate resources and training required for the research and for providing me with the Teaching Assistant Positions which helped me financially during my Masters.

Above all, I am extremely grateful to my parents who believed in me and did everything in their power to help support my dream of attending a graduate school in this land of dreams and helping me achieve the American Dream. At this point I would like to extend my sincere gratitude to them for all their sacrifices to make this happen.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

MPTO	Multiphysics Topology Optimization
MSTO	Multiscale Topology Optimization
TO	Topology Optimization
GUI	Graphical User Interface
NHTSA	National Highway Traffic Safety Administration
IIHS	Insurance Institute for Highway Safety
PDE	Partial Differential Equation
OC	Optimality Criteria
ESL	Equivalent Static Load
FEA	Finite Element Analysis
FEM	Finite Element Method
SIMP	Solid Isotropic Material with Penalization
CAE	Computer-Aided Engineering
HCA	Hybrid Cellular Automata
BESO	Bidirectional Evolutionary Structural Optimization
CA	Cellular Automata
SED	Strain Energy Density
SAND	Simultaneous Analysis and Design
GA	Genetic Algorithm
USEDD	Uniform Strain Energy Density Distribution
ESO	Evolutionary Structural Optimization
AESO	Additive Evolutionary Structural Optimization

ABSTRACT

Topology Optimization is an optimization technique used to synthesize models without any preconceived shape. These structures are synthesized keeping in mind the minimum compliance problems. With the rapid improvement in advanced manufacturing technology and increased need for lightweight high strength designs topology optimization is being used more than ever.

There exist a number of commercially available software's that can be used for optimizing a product. These software have a robust Finite Element Solver and can produce good results. However, these software offers little to no choice to the user when it comes to selecting the type of optimization method used.

It is possible to use a programming language like MATLAB to develop algorithms that use a specific type of optimization method but the user himself will be responsible for writing the FEA algorithms too. This leads to a situation where the flexibility over the optimization method is achieved but the robust FEA of the commercial FEA tool is lost.

There have been works done in the past that links ABAQUS with MATLAB but they are primarily used as a tool for finite element post-processing. Through this thesis, the aim is to develop an interface that can be used for solving optimization problems using different methods like hard-kill as well as the material penalization (SIMP) method. By doing so it's possible to harness the potential of a commercial FEA software and gives the user the requires flexibility to write or modify the codes to have an optimization method of his or her choice. Also, by implementing this interface, it can also be potentially used to unlock the capabilities of other Dassault Systèmes software's as the firm is implementing a tighter integration between all its products using the 3DExperience platform.

This thesis as described uses this interface to implement BESO and HCA based topology optimization. Since hybrid cellular atomata is the only other method other than equivalent static load method that can be used for crashworthiness optimization this work suits well for the role when extended into a non-linear region.

1. INTRODUCTION

1.1 Background

Topology Optimization is a critical tool that has been used to synthesize structures without any preconceived shape. It, in its most basic form can be understood as a tool used to solve minimum compliance problem i.e. minimization of the compliance of a structure while satisfying the volume fraction constraint over it. Topology optimization achieves this by adding and removing material from different elements of a structure in every iteration based on the response surfaces like strain energy.

This tool can sometimes synthesizing structures which in the past if not impossible was prohibitively expensive to manufacture. But with the rapid development in advanced manufacturing technology like metal 3D printing this limitation is virtually non-existent in many cases.

Also, with rapid introduction of battery electric vehicles, it is necessary to achieve maximum weight reduction along with continuous improvement in the battery technology in-order to achieve greater range; a critical factor from a consumers point of view. One should also remember that the reduction in weight cannot come with a compromise to the passenger safety and the vehicle should still adhering to the National Highway Traffic Safety Administration (NHTSA) standards and should have a good Insurance Institute for Highway Safety (IIHS) rating for the North American Market, while other markets have their own standards. All while keeping the cost of manufacturing minimum to boost the sales and mass adoption of the technology.

This in a nutshell is exactly what optimization is, improve a given function in this case it can be weight while respecting other constrains like maximum energy absorption, penetration into the passenger area (displacement) etc. Topology optimizations ability to add and delete materials can also be harnessed for a number of additional uses like the development of flexible rotor for the engine. Such a rotor will theoretically need less moving parts to achieve the flexibility, which in turn leads to less manufacturing cost, less wear and tear which means less maintenance which can be of strategic importance if this kind of flexible rotor engines

are used for defence purposes. All the above needs and a countless other has increased the need for having advanced tools to synthesizing structures under dynamic loading condition.

1.2 Motivation and Problem Statement

MATLAB which is a multi-paradigm programming language has a number of tools that can be used for optimization. The optimization toolbox in MATLAB is a collection of function that boost the capabilities of the software's numeric computing environment. [1]. There are routines for a different kinds of optimization in this tool box like unconstrained nonlinear minimization, contrained linear least squares, nonlinear system of equation solving, quadratic and linear programming, nonlinear least squares and curve-fitting, constrained nonlinear minimization, including semi-infinite minimization problems, minimax problems and goal attainment problems. [1]

Additionally, MATLAB's Partial Differential Equation (PDE) toolbox can be used for finite element analysis which is a critical component of any kind of shape or topology optimization.

Finite Element Analysis is at the core of topology or shape optimization, from the most basic topology optimization method like Optimality Criteria (OC) to one of the most advance methods like Equivalent Static Load (ESL) method FEA is everything. While it might be possible to solve many real world problems with these tools, it cannot match the efficiency and capabilities of a commercial FEA package like ABAQUS or LS-Dyna.

Like for example it is possible to write a code tailored for solving the FEA of an armor plate problem. But doing so would need time, resources and still it's a simple problem to solve compared to the intricate composit armor designs used in the industry, not to mention the extremely large deformation that will be experience by the plate in real world use.

Hence, it is necessary to have a way to harness the capabilities of these commerical FEA software by linking them to MATLAB to solve complex topology optimization.

This thesis aims to develop and use algorithms in MATLAB to exploit the implicit and explicit FEA capabilities of ABAQUS to solve complex non-linear topology optimization problems for crashworthiness.

1.3 Early Works on ABAQUS-MATLAB Interface

There has been work done in the past where ABAQUS was linked with MATLAB for finite element post-processing [2]. The tool that was presented by George Papazafeiropoulos, Miguel Muñiz-Calvente and Emilio Martínez-Pañeda [2] was intended to benefit from the image processing and integrated graph-plotting features of MATLAB as well as open up avenues for statistical analysis, result post-processing and mathematical optimization [2]. The paper [2] describes two examples to demonstrate the capabilities of the method presented.

It is first used to assess cleavage fracture through a 3-parameter Weibull probabilistic framework [2] and the second example demonstrates the potential of the work to create and train neural networks to identify damage parameters through a hybrid numerical-experimental scheme and model the crack propagation in structural materials by using a cohesive zone approach [2].

The paper describes in detail the internal characteristics of the ABAQUS2MATLAB [2] interface and briefly descries the structure of the ABAQUS result file. It uses ASCII format in its approach due its versatility [2].

There has been many use of the ABAQUS-MATLAB link mentioned earlier for different applications but the tools developed during those applications fall short of the capabilities needed for a topology optimization algorithm. A topology optimization algorithm needs to be capable of sensing the neighborhood of an element, the response surfaces of those neighborhood and make the decision to either penalize the material properties for a Solid Isotropic Material with Penalization (SIMP) approach or kill or regenerate the element for a hard-kill method.

1.4 Research Objective (Accomplishments)

The aim of this research was to develop an ABAQUS-MATLAB based interface that can be used for different kind of topology optimization method. The interface was meant to be such that it work well with both a hard-kill and SIMP based approach. The SIMP based capability was achieved by assigning individual material to individual element and penalizing the material property based on the response surfaces. The ability to write data directly into

ABAQUS input file was the key to achieving the capability to penalize a material and there by use SIMP based topology optimization techniques. Similarly the capability to hard-kill and re-add an element back to the model was achieved by using the ABAQUS capability to add or delete the element using the assembly.

The interface developed as a part of this thesis was successfully tested on two types of optimization methods, namely Bidirectional Evolutionary Structural Optimization (BESO) method and Hybrid Cellular Automata (HCA) method. HCA due to its inherent advantages produced appreciable results.

During the interfacing it was necessary to read and write data back into the ABAQUS in order to perform topology optimization. Since ABAQUS files can be read into MATLAB using textscan and various other methods to import data as strings, this capability was extensively used to read the model and it's response surfaces. Similarly MATLAB command fprintf was used to modify ABAQUS model by writing the data into ABAQUS input file.

As a part of work two methods were used to run an ABAQUS input file, by directly launching ABAQUS though system command in MATLAB or by pulling in the ABAQUS license. Since optimization requires data to be written into the model file MATLAB command fprintf was used to fulfil this purpose.

2. TOPOLOGY OPTIMIZATION

2.1 Introduction

As discussed in the introduction chapter, topology optimization can be used to synthesize structures without any preconceived shape. The fact that one can leverage this ability to find innovative, high-performance structural layouts have attracted the interest of engineering designers and applied mathematicians.[3]

Consequently, there are a number of journal publications in this area from Lucien Schmit's work in the 1960s [4] which highlighted the potential of combining finite-element analysis and optimization methods for structural design, to the seminal paper by Bendsøe and Kikuchi in 1988 [5] about the use of homogenization method to generate optimal typologies in structural design [3]. There are also a number of reference books like Hassani and Hinton 2012 [6], Bendsøe and Sigmund 2003 [7], Christensen and Klarbring 2008 [8] and as discussed earlier a number of tools in MATLAB and other platforms to synthesize topology optimized structures [3].

2.2 Theoretical Background

2.2.1 Problem Definition and Ill-Posedness

A topology optimization problem in a nutshell is a binary programming problem with the objective to find the material distribution in a prescribed volume or area known as the design domain [3]. A typical formulation called as the binary compliance problem, is to find the solids and voids (black and white layout) that minimizes the work done by the external forces (compliance) subjected to the volume constraint.[3].

In most cases a binary compliance problem is known to be ill-posed [3], [9]–[11]. It is possible for one to obtain a non-convergent sequence of feasible designs (black and white) that will monotonically reduce a structure's compliance. [3]. For example, if we consider a design with a single hole, it is possible to find an improved design solution with the same mass and lower compliance just by replacing the hole with two smaller holes [3]. Following this logic one can find even better solution by increasing the number of holes and reducing

their size [3]. This will lead to a chattering design with infinite number of holes all having infinitesimal size [3]. This makes the compliance problem unbounded and hence ill-posed [3].

To deal with this problem Haber and Jog C suggested the use of perimeter control of the structure to make the problem well-posed [12], [13]. This method avoids chattering configuration but its implementation is complicated too since it creates fluctuations during the iterative optimization process [3], [7], [14]. This creates the need for internal loops to be incorporated within the optimization process to overcome this fluctuations [7], [14]. Adding perimeter constrain also creates dramatic changes in the final layout for small variations in the parameters of the algorithm [13].

2.2.2 Homogenization Method

An alternative is to use the homogenization method for topology optimization [15], [16]. The method calls for relaxing the binary condition by including intermediate material densities in the problem formulation [3]. This in turn makes the chattering configurations a part of the problem statement itself by assuming a periodically perforated micro-structure [3]. Once can use the homogenization theory to determine the mechanical property of the material [3].

While this method addresses the problem of chattering configuration, the optimal microstructure required in the derivation of the relaxed problem isn't always known and is the main drawback of the same [3]. It is possible to alleviate this problem by restricting the method to a subclass of the microstructure, probably sub-optimal but fully explicit [3]. Bendsøe and Kikuchi [5], Allaire and Khon [17] and Allaire et al. [16] have used this approach referred to as partial relaxation.

Also, the way homogenization method works, it creates infinitesimally small holes in the gray area which are difficult or impossible to manufacture [3]. However, it is possible to use penalization strategies to mitigate this problem [3]. One approach is to force the intermediate densities to take black or white values by post-processing the partially relaxed optimum [18]. This is a purely numerical and mesh dependent procedure called as posteriori procedure and results in a binary design [3].Another approach is to impose a prior restriction that implicitly

leads the micro-structure to black and white designs [15]. While this penalization methods have been effective in avoiding or in mitigating the intermediate densities, they revert back to the original ill-possedness with respect to the mesh refinement [3].

2.2.3 Density Based Approach

The density based approach is an alternate method that is widely used in the industry and avoids the use of microstructures by relaxing the binary problem using a continuous density value. In this method the the material density distribution is used to parametrize the material distribution [3].

The mechanical properties of a material element like stiffness is determined by using a power-law interpolation function between the void and solid [19], [20].The power law can implicitly penalize the intermediate density values and can drive the structure towards a black-and-white configuration [3]. This penalization procedure is referred to as Solid Isotropic Material with Penalization (SIMP) [21]. While the SIMP method does not address a problems ill-possedness, it does make it simpler than other penalization methods [3].

The SIMP method is based on the heuristic relation between the relative element density x_{i} and element Young's Modulus E_{i} [3]. This relation can be denoted as,

$$E_{\mathrm{i}} = E_{\mathrm{i}}(x_{\mathrm{i}}) = x_{\mathrm{i}}^{p} E_0, \qquad x_{\mathrm{i}} \in [0, 1] \tag{2.1}$$

where, E_0 is the Young's modulus of the solid material and p is used to represent the penalization power ($p > 1$) [3]. The modified SIMP approach is given as

$$E_{\mathrm{i}} = E_{\mathrm{i}}(x_{\mathrm{i}}) = E_m in + x_{\mathrm{i}}^{p}(E_0 - E_{min}), \qquad x_{\mathrm{i}} \in [0, 1] \tag{2.2}$$

where, E_{min} is the Young's modulus of the void material [3]. E_{min} is non-zero in order to avoid singularity in the finite element stiffness matrix [3].

The modified SIMP approach as shown in (2.1) offers a number of advantages over the classic SIMP approach, as (2.2), including independence between the minimum value of a material's Young's modulus and the penalization power [22].

However, topology optimization methods are most likely to encounter numerical difficulties like checkerboard patterns, mesh-dependency and local minima [7]. In order to overcome such issues, it is recommended to use regularization techniques [23]. The most common approach that is used is the use of a density filter [24]. A basic density filter function can be defined as,

$$\tilde{x_{\mathrm{i}}} = \frac{\sum_{\mathrm{j}\in N_{\mathrm{i}}} H_{\mathrm{ij}} v_{\mathrm{j}} x_{\mathrm{j}}}{\sum_{\mathrm{j}\in N_{\mathrm{i}}} H_{\mathrm{ij}} v_{\mathrm{j}}}, \tag{2.3}$$

where, N_{i} is the neighborhood of element x_{i}. Element x_{i} possesses a volume v_{i} and H_{ij} is a weight factor [3]. The neighborhood can be defined as

$$N_{\mathrm{i}} = \{\mathrm{j} : dist(\mathrm{i},\mathrm{j}) \leq R\}, \tag{2.4}$$

where, the distance between the center of elements i and j is represented by an operator dist(i,j) and R is the filter size or size of the neighborhood [3]. The weight factor (H_{ij}) can be defined as a function of the distance between the neighboring elements, for example

$$H_{\mathrm{ij}} = R - dist(\mathrm{i},\mathrm{j}), \tag{2.5}$$

where, $\mathrm{j} \in N_{\mathrm{i}}$ [3]. The filtered density $\tilde{x_{\mathrm{i}}}$ defines a modified physical density that is then incorporated into the topology optimization formulation and the SIMP method as

$$E_{\mathrm{i}}(\tilde{x_{\mathrm{i}}}) = E_{min} + \tilde{x}_{\mathrm{i}}^{p}(E_0 - E_{min}), \qquad \tilde{x_{\mathrm{i}}} \in [0,1] \tag{2.6}$$

This regularized SIMP interpolation formula defined in (2.6) is used in the work presented in this thesis.

3. ABAQUS INTERFACE AND MODELING

3.1 Introduction

ABAQUS FEA, is a Finite Element Analysis software suite that was acquired and is owned by Dassault Systèmes since October, 2005. This software has a wide range of capabilities and can be used for used for both implicit and explicit analysis.

The software has a free student version available for download from the SIMULIA Community. A user can simply use a search engine to search for the ABAQUS Student Edition (SE) and this should direct them to Dassault Systèmes 3DExperience Edu page which has the link for downloading ABAQUS. It should be noted that while the student edition has all the capabilities needed to perform an optimization using the MATLAB-ABAQUS interface explained in this thesis, the model is restricted to 1000 nodes and 1 cpu core (no parallel computation). An ABAQUS online manual is available from the makers of this software that can be used for more indepth understanding of the tool, its material cards, element types, output commands etc. [25].

3.2 Creating ABAQUS Model

To create a FEA model ABAQUS offers a user friendly GUI that contains various components used for modeling the problem. The GUI first consist of a "Title Bar" which contains the information about ABAQUS version. Below it is the "Menu Bar" this is the component that holds icons such as File, Model, Viewport etc. Located just below this bar is the "Toolbar" this bar contains a number of icons that can be used to create a new model, open an existing one, save a model, pan, rotate, zoom, views such as mesh view, seeded view etc. Below the "Menu Bar" is the "Context Bar" which contains the drop downs for Module, Model and Part.

Towards the extreme left of the GUI, one can observe a tree like structure similar to a CAD software. This is called the "Model Tree/Results Tree" and can be used as an alternate way of toggling to different steps in creating a CAE model if once chooses to use this over the Module (drop-down) in the context bar. Just adjacent to the "Model Tree" is the "Toolbox

Area" This is the location where one will find all the tools necessary to create a model, seed and mesh the part, apply the forces, observe the results etc. The icons in this toolbox change based on the step in the model creation. Next to this on the right hand side we have the "Viewport" where the model and the results will be visible to the user. Below the "Viewport" is the "Prompt Area" this is where the user will find options like done etc., that one needs to click after selecting a command and the associated geometry. Below this and the very bottom of the GUI is the "Message Area of Command Line Interface" this is where one can find all the messages like the ones related to model creation etc. Fig 3.1 shows the GUI explained below.

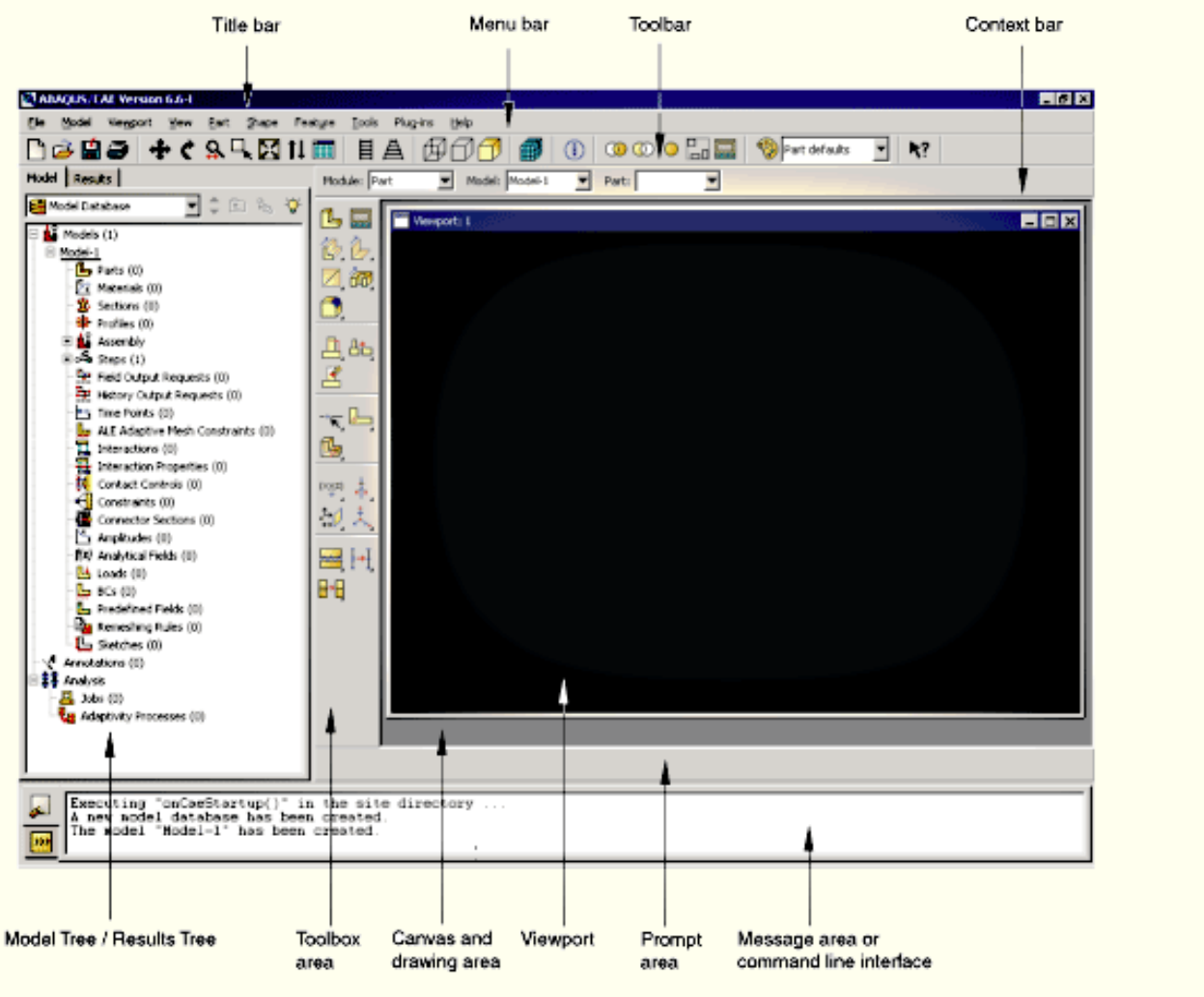

Figure 3.1. ABAQUS GUI
[25]

3.2.1 Linear Model

The example shown below is a simple 100 $mm \times 10\ mm$ cantilever beam made of a steel alloy with Young's Modulus (E) = 2.07×10^5 MPa, Poisson's Ration (γ) = 0.3 and is subjected to a load of 1000 N on the bottom most node at the free edge.

It should be noted that ABAQUS is a unitless software just like LS-Dyna and purely depends on the user to provide input in a consistent unit system. Table 3.1 can be refereed as a base for consistent units.

Table 3.1. Consistent Units
[26]

Mass	Length	Time	Force	Stress	Energy	Density	Young's Modulus	35MPH 56.33Km/Hr	Gravity
kg	m	s	N	Pa	J	$7.83 * 10^3$	$2.07 * 10^{11}$	15.65	9.806
kg	cm	s	$1.0 * 10^{-2}$ N			$7.83 * 10^{-3}$	$2.07 * 10^9$	$1.565 * 10^3$	$9.806 * 10^2$
kg	cm	ms	$1.0 * 10^4$ N			$7.83 * 10^{-3}$	$2.07 * 10^3$	1.56	$9.806 * 10^{-4}$
kg	cm	us	$1.0 * 10^{10}$ N			$7.83 * 10^{-3}$	$2.07 * 10^{-3}$	$1.565 * 10^{-3}$	$9.806 * 10^{-10}$
kg	mm	ms	kN	GPa	kN-mm	$7.83 * 10^{-6}$	$2.07 * 10^2$	15.65	$9.806 * 10^{-3}$
g	cm	s	dyne	$dyne/cm^2$	erg	$7.83 * 10^0$	$2.07 * 10^{12}$	$1.565 * 10^3$	$9.806 * 10^2$
g	cm	us	$1.0 * 10^7$ N	Mbar	$1.0 * 10^7$ Ncm	$7.83 * 10^0$	$2.07 * 10^0$	$1.565 * 10^{-3}$	$9.806 * 10^{-10}$
g	mm	s	$1.0 * 10^{-6}$ N	Pa		$7.83 * 10^{-3}$	$2.07 * 10^{11}$	$1.565 * 10^4$	$9.806 * 10^3$
g	mm	ms	N	MPa	N-mm	$7.83 * 10^{-3}$	$2.07 * 10^5$	15.65	$9.806 * 10^{-3}$
ton	mm	s	N	MPa	N-mm	$7.83 * 10^{-9}$	$2.07 * 10^5$	$1.565 * 10^4$	$9.806 * 10^3$
$lbf - s^2/in$	in	s	lbf	psi	lbf-in	$7.33 * 10^{-4}$	$3.00 * 10^7$	$6.165 * 10^2$	386
slug	ft	s	lbf	psi	lbf-in	$1.52 * 10^1$	$4.32 * 10^9$	51.33	32.17
$kgf - s^2/mm$	mm	s	kgf	kgf/mm^2	kgf-mm	$7.98 * 10^{-10}$	$2.11 * 10^4$	$1.56 * 10^4$	$9.806 * 10^3$
kg	mm	s	mN	$1.0 * 10^3$ Pa		$7.83 * 10^{-6}$	$2.07 * 10^8$		$9.806 * 10^3$
g	cm	ms	$1.0 * 10^1$ N	$1.0 * 10^5$ Pa		$7.83 * 10^0$	$2.07 * 10^6$		$9.806 * 10^{-4}$

The CAD Model used in this example is a 2D cantilever beam (100 mm length, 10 mm wide) which is supported on one end and a concentrated force of 1000 N is applied on the bottom right corner as shown in Fig 3.2. Property of material (steel) is taken from the consistent units Table 3.1. It should be noted that the unit system used here is ton, mm, s, N in order to obtain stresses in MPa.

One should start creating the model by launching ABAQUS CAE from the start menu. Once open it's recommended to change the working directory to a folder that will be used to store the mode. This can be done by selecting files from the menu bar and selecting set work directory. The new folder shouldn't contain any spaces in the name as it can cause issue during interfacing ABAQUS with MATLAB based on the method used.

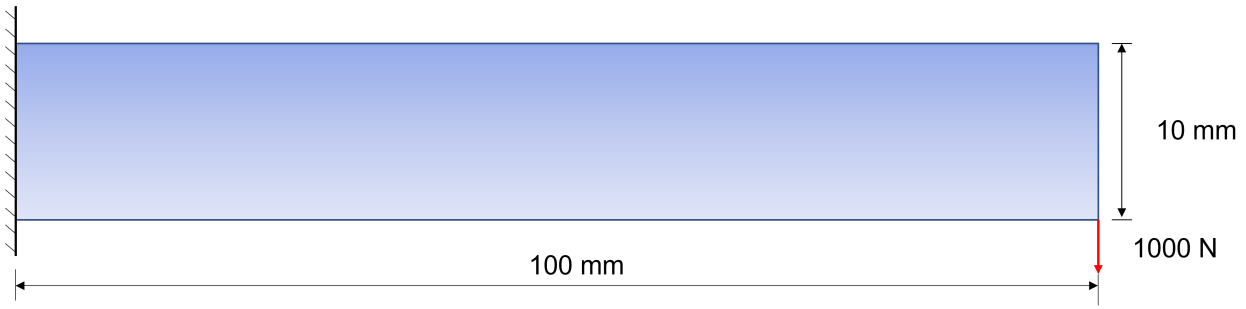

Figure 3.2. 2D Beam Setup

Double click on parts from the model tree. A create part window will pop-up, change the name to a suitable name, here we choose the name Beam2D. Select modeling space as 2D planar, type as deformable, base feature shell, approximate size as 200 and press continue.

Use the "Create Lines:Rectangle (4 Lines)" icon in the toolbox area to draw a rectangle in the viewport. After creating the rectangle use the "Add Dimension" icon in the toolbox area to give dimension to the rectangle, in this case 100×10. While giving dimension after selecting the "Add Dimension" tool click on the side to dimension and follow the instructions in the prompt area. Once done dimensions are given, press the red cross button in the prompt area and then press done for sketch section in the same area.

Then select "Property" from the module drop down list located in the context bar. After this click on "Create Material" icon in the toolbox area. On the edit material pop-up window change the name of the material to "Steel" in this case, select mechanical option located below it and select elasticity - elastic. Populate the Young's Modulus using the value given in Table 3.1 keeping in mind the units system and enter 0.3 for Poisson's Ratio.

For entering the density click on general - density and populate the same. It should be noted that while it's completely possible to have a plastic material property applied to the material, this model is limited to the elastic region for demonstration purposes. However a following subsection will point out the method to introduce material non-linearity by defining the plastic region.

Now to assign the material to the part we need to create a section and assign the section to the part. Select "Create Section" icon in the toolbox area, in the pop-up window we enter the name as "BeamSection", category as solid and type as Homogeneous. Then after pressing

continue a "Edit Section" window pop-up. In this window select the material that was just created in the last step using a drop-down and click the checkbox for plain stress/strain thickness if required. Here it's selected and given 1 mm thickness value then press "OK".

Now, in order to assign individual material property to individual element needed for SIMP method we have two ways. In first method a material can be created for every single element individually and assigned to a section individually which can be later assigned to the individual element using an element set. One can also edit the final output .inp file to make this happen, which is an easier way for big complex model. This will be explained in the future subsection.

Now click on "Assign Section" icon in the toolbox area, left click on the beam model and press Done in the prompt area. On the next window that popup called "Edit Section Assignment" make sure everything is correct this means in our case making sure the section in the drop down is the section that we just created and the thickness assignment radio button is on from section and then press "OK".

From the module drop down in the context bar select "Assembly" and then click on "Create Instances" icon in the toolbox area. In the pop-up window make sure everything is as needed, in this case the create instances from should be parts. Parts highlighted should be Beam2D, instance type we have chosen to go with a dependent mesh and press "OK".

Now, from the module drop down located in the context bar select "Step" and then click on the icon "Create Step" in the toolbox area. In the "Create Step" make sure the everything is as needed, in our case this means inter new step after is initial, procedure type is static general and press "Continue".

For this model we will be using static, general but dynamic, implicit and explicit can be selected for the models that need non-linear characteristics. Also, for extracting mass and stiffness matrix the "Procedure Type" is different and one should refer to the subsection dedicated to mass and stiffness extraction model for creating step.

For this section, in the next popup window called "Edit Step" a description can be given and rest all is left as it is in the default setting and press "OK". If the user chooses to (as per the requirement) the time period can be changed and most importantly the Nlgeom can be

turned on to capture nonlinear effects of large displacements, this affects subsequent steps. Automatic Stabilization also can be turned on if needed along with other settings.

To apply the boundary condition from the module drop down located in the context bar select ”Load” and then click on ”Create Boundary Condition” icon in the toolbox area. Since our model is a cantilever beam, we will be applying a fixed support on the left end of the beam. Hence, after giving a name for example ”FixedSide”, make sure the step is set to step-1 and the category is mechanical. Then select ”Symmetry/Anitsymmetry/Encaster”, this boundary condition will help us lock the degree of freedom for the selected nodes as per the need. More information on this can be found by referring to the documentation ABAQUS Analysis User's Manual [25].

Once selected press continue and select the side of the model that will be fixed using left mouse button. Then press ”Done” in the prompt area. In the ”Edit Boundary Condition” window popup we select ENCASTRE which locks all the degrees of freedom in X, Y and Z and all 3 rotational degrees of freedom and press ”OK”. This model uses solid elements and hence does not have all degrees of freedom mentioned above, but this setting is used as it works throughout the board.

To apply the concentrated load select ”Create Load” icon in the toolbox area. In the create load window give an appropriate name, in this case we give a name ”Concentrated-Force”, make sure the step is set to step-1 and the category is mechanical. Then from types for selected step, select ”Concentrated Force” and press continue. Use the mouse button to select the bottom right corner node in the model and press ”Done” in prompt area. In the ”Edit Load” dialog box populate the CF2 tab with a value -1000 as CF2 represents force in Y direction and a negative symbol will apply the force in negative Y direction relative to the global coordinate system.

For any model this method can be applied to apply loads to only the vertices of the model. A subsection in this chapter will explain the method to apply load to a node anywhere in the model or a set of nodes in the model. Other types of loads and setting can be selected based on the users needs and this step is completely avoided when creating a model to extract the stiffness and mass matrix which can be used for a number of optimization methods. As there the model created for this purpose will only have fixed support and will be subject to

a frequency excitation. This is explained in the next subsection. Also, to if the user needs to apply the load to a node anywhere in the model (specially for 3D elements) the subsequent subsections will discuss about it.

Moving forward to create a mesh in ABAQUS it is important to create seeds, which technically can be referred to as points that directly represents the nodes at the edges. Mesh is a very critical part of the FEA modeling and can have effects on the results and computation time. Universally irrespective of the FEA solver, it's good to have quadrilateral or square elements over tetrahedral or triangular elements as triangle by itself is a stiffer structure and can artificially inflate the stiffness of the elements.

Also, as we are aware there are two types of elements, linear and quadratic elements. The difference being the number of nodes in an element. Linear elements have nodes at the vertex only while quadratic elements have nodes at the mid-point of the edge of the element too. This can have effects on both the computation time and the results.

Also, the number of nodes in an element determines the order of interpolation [25]. Elements such as a 8-node brick element shown in the Fig 3.3, have nodes only at the corners and uses linear interpolation in each direction [25]. These elements are often called linear elements or first-order elements [25].

Elements such as the 20-node brick element also shown in the Fig 3.3, have midside nodes too and uses quadratic interpolation [25]. These elements are often called quadratic elements or second-order elements [25].

Tetrahedral or modified triangular elements with mid-side nodes like the one shown in Fig 3.3 (10-node tetrahedron), uses a modified second-order interpolation [25]. These elements are often called modified elements or modified second-order elements [25].

There are two ways of meshing a part in ABAQUS. One is by using the seed part option which directly seeds the edges of the entire part based on the approximate global sizing. The other is by using seed edges which seeds individual edges by size of the element or by the number of the elements as per the user input. It should be noted that ABAQUS is unitless and hence this value of global sizing can be mm, cm, m based on the unit system followed by the user.

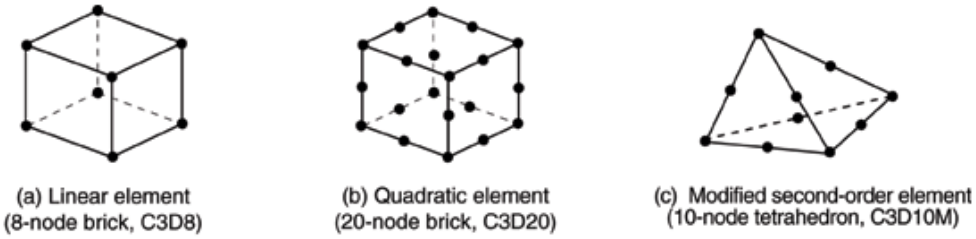

Figure 3.3. Linear Brick, Quadratic Brick, and Modified Tetrahedral Elements [25]

To use global seeding, from the module drop down located in the context bar select "Mesh" and then click on "Seed Part" icon in the toolbox area.In the global seed window pop-up enter the approximate global size; this is global element size. The user can alter other values in this window as per need, but for this model we leave them as default. Once done the user can press "OK" and the seeds should be created. Similarly for edge seeding one can select the "Seed Edge", select the edge to be seeded, press "DONE" in the prompt area and follow the popup on the screen.

Seeds can also be deleted by long pressing the icon used to create seed, this opens a small bar with delete seed option. One can then drag the mouse cursor to this option and delete the seed as needed.

Once the seeds are created click on "Mesh Part" icon in the toolbox area and then press on "Yes" in the prompt area to confirm. This will create a mesh on the part. Since the student version is limited to 1000 nodes, it is necessary to keep this in mind while seeding and check for the same after meshing.

In order to check the number of nodes and elements one can select "view" tab from the menu bar. Then select part display option, click on the mesh tab and select the check box with the name "Show node labels", "Show element labels" and press "OK".

Once the mesh is complete the user needs to assign an element type. For this one needs to select the "Assign Element Type" icon from the toolbox area, left click on the part and click "Done" in the prompt area. On the pop-up window select the element type from the list of elements under title "Family". For this model we select element library as standard, geometric order linear, the family as plane stress. We select Quad with reduced integration

points and keep all the default settings as it is. Our element for this simulation is a 2D 4-node plane stress element with linear integration. One can change the settings based on the problem and type of elements needed for it. Then click "OK" and "Done" once again.

This completes the model building. Now we need to create a job file. In order to build this file navigate to "Job" menu from the module drop down located in the context bar and click on "Create Job" icon in the toolbox area. In the create job pop-up rename the job as needed, we name it 2DBeam. It's necessary to not have spaces since it can interfere with some MATLAB functionality during interfacing. Then click on "Continue". On the pop-up window we leave the setting at default and press "OK". But the user can populate the tabs and select the settings as needed. It's also recommend at this stage to use the save option in files. This will save a ".cae" file in the working directory. This is like a archive and this single file can be transferred to any system to open the entire model in ABAQUS GUI.

Now we need to create the ".inp" file. This is the file that stores all the information about the model and the one we can modify in order to achieve a number of functionality which will be explained in the following subsections and chapters. In order to create this file, click on "Job Manager" icon in the toolbox area and select the option "Write Input". This will write the ".inp" file. The user can use "Submit" option to run this file on ABAQUS GUI to make sure the file runs without an error, "Data Check" can also be used for this purpose. The "Results" icon helps one visualize the results in ABAQUS GUI. It should be noted that we will be running multiple iteration using MATLAB and not though ABAQUS GUI and the above option to "Submit" is just for check.

3.2.2 Material Non-Linearity

For material non-linearity we follow all the process as mentioned earlier for the property tab and define the elastic region and material density.Then in the edit material pop-up window again click on Mechanical, select plasticity, plastic and then define the Yield Stress v/s Plastic Strain. This will define the plastic region for the material.

3.2.3 Stiffness and Mass Matrix Specific Modification

For extracting stiffness and mass matrix as explained previously in the ”Step” stage, one should excite the model using a frequency. This can also be called as a model analysis. To do so after completing the ”Assembly” step, from the module drop down list located in the context bar, select ”Step” and then click on ”Create Step” in the toolbox area. In the create instance window from the ”Procedure type” drop down list select ”Linear perturbation” and select ”Frequency”, the rest can be left as default and press continue. In the ”Edit Step” popup window make sure to change the ”Number of eigenvalues requested” radio button to value, set the value to 10 and press ”OK”.

There after all the steps remain the same except for the ”Load” module where only fixed support will be applied i.e. no concentrated force.

3.2.4 Applying Load to any Node as Needed

To apply load to any node(s) as per the user need, we should create the model as specified earlier in the linear model subsection up till the ”Step” step. After step has been created, we can first apply the fixed support as specified earlier without any change in the procedure.Now, to apply load to a specific node or a set of nodes we first need to mesh the geometry to generate these nodes and elements. So we move to the mesh part and create a mesh as specified in the linear model subsection.

Now, if a user need to apply load to a set of nodes then we need to select ”Tools” from the menu bar and in tools select set - create. A ”Create Set” window will pop-up in the GUI, give an appropriate name, in our case we call it ”LoadNode”, set the type to node and click continue. Now using ”Shift+Left Click” select all the nodes that a user needs to apply the load to. The Shift acts like Ctrl when thought of other software’s and needs to be pressed all time while selecting the nodes.Then click ”Done” in the prompt area and this will create a node set containing all the nodes needed

A user can create multiple node set if he/she wishes to apply different loads to different sets of nodes. Now, with the set created use the module drop down located in the context bar to select ”Load” and then click on the icon ”Create Load” in the toolbox area. In the create

load window give an appropriate name, in this case we give the name ”ConcentratedForce”, make sure the step is set to step-1 and the category is mechanical. Then from types for selected step, select ”Concentrated Force” and press continue.Then select ”Sets...” icon from the prompt area and in the ”Region Selection” pop-up window select the set that we just created and click continue. Populate the ”Edit Load” window as needed and by following the instructions given in the linear model subsections and click ”OK”. This will apply the concentrated load to the node/nodes that have been selected by the user.

3.2.5 Assigning Individual Material Properties for each Element using ABAQUS GUI

In order to apply individual material properties for each element using the ABAQUS GUI a user must start by creating a material and a section for each element. This can be done by repeating the steps mentioned earlier to ”Create Material” and ”Create Section” under the module property tab.Once we have n number of material and sections created for n number of elements. We need to create an element set, we start by meshing the geometry to create the elements. It is important to ensure nodal connectivity to get accurate results. Another alternative is to create a partition at the location and use assembly option of mesh independence to ensure connectivity.

Now, we need to create an element set for each element for that select ”Tools” from the menu bar and in tools select set - create. A ”Create Set” window will pop-up in the GUI. Give an appropriate name, in our case we call it ”Element1” for element 1 and so on for each element. Set the type to element and click on continue. Now, using the left mouse button select the appropriate element for each element set and once selected click on done in the prompt area.

Now we go back to property by selecting property in the module dropdown located in the context bar. Click on ”Assign Section” icon in the toolbox area and in the ”Region Selection” pop-up window select the Element1 set and press continue. In the ”Edit Section Assignment” pop-up under the section drop-down select the section that was created for element 1 and press ”OK”. Repeat this step for every single element and now each element will have it’s own material property.

This is a simple but tedious and repetitive way to assign individual material property, a better method using the input (.inp) file edit will be discussed in the subsequent subsections.

3.3 ABAQUS Input File Modification

An ABAQUS .inp file is a simple text file that can be opened and edited from a text editor like Notepad or Notepad++. With this in mind we can do certain modifications to the .inp file to achieve various objectives like generating stiffness and mass matrix, forcing ABAQUS to write output data such as strain energy, nodal displacements and also to add or delete elements which is critical to synthesize new structures using optimization.

There are two points that needs to be noted while doing this, first is that ABAQUS is sensitive to keyword location, and there is a designated location for each keyword. If a keyword is located somewhere else the program won't run. Similarly after this modification is done, the .inp file needs to be run using MATLAB, if one tries to go back to GUI and run the file there is a chance that ABAQUS will simply overwrite the modifications with the original file.

If a user needs to input some comment or comment out a line in the .inp then he should insert two star marks (**) in front of the line, this will let the solver know that it's a comment and hence the line should be skipped.

3.3.1 Generating Stiffness and Mass Matrix

To generate stiffness matrix we need to write the *MATRIX GENERATE and *MATRIX OUTPUT command into the .inp file as shown in Fig 3.4. This command writes global matrix to the file in assembled or element-by-element format [25]. It should be noted that it can only be used in a matrix generation analysis [25]. The FORMAT=COORDINATES used here gives us the output in a matrix format that can be easily read and interpreted using a simple code in MATLAB.

```
*Elastic
207000., 0.3
** ----------------------------------------------------------------
**
** STEP: Step-1
**
*step,name=export matrix
*MATRIX GENERATE , STIFFNESS,MASS
*MATRIX OUTPUT,STIFFNESS,MASS,FORMAT=COORDINATE
*end step
**
*Step, name=Step-1, nlgeom=NO, perturbation
*Frequency, eigensolver=Lanczos, sim, acoustic coupling=on, normalization=mass
10, , , , ,
**
** BOUNDARY CONDITIONS
```

Figure 3.4. Stiffness and Mass Matrix Generate

3.3.2 Extracting Elemental Strain Energy, Integration Points and Element Volume

Strain energy density, element volume data etc., is crucial for synthesizing new structures using topology optimization. Some of this data can be extracted directly others can be calculated by extracting the raw data from ABAQUS. For example strain energy density can be calculated by extracting strain energy from the model and in similar way the neighbourhood data needed to perform Hybrid Cellular Automaton can be calculated using the nodal coordinates, and other element data such as the centroid of the element.

We can write the *EL PRINT command into the .inp file to extract these values and the raw data needed to calculate other crucial values during the run. This keyword is used to provide tabular printed output of the element variables such as stresses, strains, etc. As we can see from Fig 3.5 this keyword is associated with some optional parameters, the once's that are necessary for our work is POSITION and SUMMARY.Position is the point at at which the requested values will be printed, by default the position is set at integration points (POSITION=INTEGRATION POINTS). This can be changed to POSITION=AVERAGED AT NODES, POSITION=CENTROIDAL and POSITION=NODES if the values being written is extrapolated to the nodes of each element in the set but is not averaged at the nodes [25]. While SUMMARY=YES (default) provides the summary and the location of the maximum and minimum values in each column of the table [25].

Other whole element variables output requested here is the total elastic strain energy in the element which is requested using ELSE, current element volume using EVOL and COORD the value of which is later used as the element centroid for neighbourhood function.

```
1126  *Restart, write, frequency=0
1127  **
1128  ** FIELD OUTPUT: F-Output-1
1129  **
1130  *Output, field, variable=PRESELECT
1131  **
1132  ** HISTORY OUTPUT: H-Output-1
1133  **
1134  *Output, history, variable=PRESELECT
1135  *EL PRINT, POSITION=CENTROIDAL, SUMMARY=YES
1136  ELSE,
1137  COORD,
1138  EVOL,
1139  *End Step
1140
```

Figure 3.5. ABAQUS EL PRINT Keyword

3.3.3 Element Delete or Re-Add for Hard-Kill

This set of command is generally written into a secondary .inp file which is similar to the main .inp file and is used as a template to update the main .inp file. To create a secondary template file, once all the modifications has been done to the main .inp file just use the save as option in a text editor like Notepad++ and save with an appropriated name and .inp extension. For this example we save it as "Beamtemp.inp".

To add or delete the element directly we can insert an element set in the ABAQUS assembly line as shown in Fig 3.6. During optimization tool we will use the MATLAB command "fprintf" to write the element that we wish to delete in the next iteration. The optimization algorithm can simply add the element back in any iteration by removing the element number from this Elset as shown in Fig 3.7.

In addition to this keyword we also add a keyword *Model Change, which is used to remove or reactivate the elements or the contact pairs during an analysis as showen in Fig 3.8[25]. It has some required mutually exclusive parameters which defines the model change.

They are ACTIVATE, ADD and REMOVE.Remove is the parameter that is used for our optimization to indicate that the elements or the contact pairs involved are being removed during the step [25]. Add parameter is used to indicate that the elements or the contact pairs involved are being reactivated during the step [25]. Activate parameter is used in any step during an analysis to indicate that elements or the contact pairs may need to be added or removed during a subsequent restart analysis [25]

```
1082  ** ASSEMBLY
1083  **
1084  *Assembly, name=Assembly
1085  **
1086  *Instance, name=BEAM-1, part=BEAM
1087  *End Instance
1088  **
1089  *Nset, nset=Set-1, instance=BEAM-1, generate
1090    51,  561,   51
1091  *Elset, elset=Set-1, instance=BEAM-1, generate
1092    50,  500,   50
1093  *Nset, nset=Set-2, instance=BEAM-1
1094   511,
1095  *Elset, elset=delete_element, instance=Part-1-1
1096  *End Assembly
```

Figure 3.6. Assembly Modification

```
1082  ** ASSEMBLY
1083  **
1084  *Assembly, name=Assembly
1085  **
1086  *Instance, name=BEAM-1, part=BEAM
1087  *End Instance
1088  **
1089  *Nset, nset=Set-1, instance=BEAM-1, generate
1090    51,  561,   51
1091  *Elset, elset=Set-1, instance=BEAM-1, generate
1092    50,  500,   50
1093  *Nset, nset=Set-2, instance=BEAM-1
1094   511,
1095  *Elset, elset=delete_element, instance=Part-1-1
1096  1, 2, 51,
1097  3, 52, 101,
1098  4, 53, 102,
1099  151, 54, 5,
1100  *End Assembly
1101  **
1102  ** MATERIALS
```

Figure 3.7. Element Delete Example

```
1121  ** Name: Load-1   Type: Concentrated force
1122  *Cload
1123  Set-2, 2, -1000.
1124  **
1125  ** Interaction: Remove_Elements
1126  *Model Change, remove
1127  delete_element,
1128  ** OUTPUT REQUESTS
1129  **
1130  *Restart, write, frequency=0
```

Figure 3.8. Delete Elements

3.3.4 Assigning Individual Material Properties for each Element by Editing Input File

In order to assign individual material properties to each element we can write the keyword for the element set, section and section assignment along with the material properties either manually into the inp file or using fprintf command in MATLAB.

In order to create an element set for each element we need to write the *Elset keyword in the .inp file just below any other element or node set that's already present and if not just after the last element number and it's coordinates under the *Element keyword in the file. Fig 3.9 shows how this keyword needs to be written, here "elset" is the name of the set and the line following it is the element number. A user can write n number of sets like this one after another for each element.

To create the section we use the the *Solid Section keyword. This keywords needs to be inserted just below the *Elset that we just created and is written as shown in Fig 3.10. Here we use the elset and material parameters to connect a material with its respective element via element set.

For having n number of material properties for n number of elements we keep on writing *Material Keyword along with its associated parameters for as shown in Fig 3.11 just below the *End Assembly keyword. Here the name parameter serves as the tag that is used in the *Solid Section keyword to identify the correct material for an element set.

```
1009  375, 619, 620, 624, 623, 515, 516, 520, 519
1010  *Nset, nset=Beam, generate
1011     1,  624,    1
1012  *Elset, elset=Beam, generate
1013     1,  375,    1
1014  *Elset, elset=Element_1
1015   1,
1016  *Elset, elset=Element_2
1017   2,
1018  *Elset, elset=Element_3
1019   3,
1020  *Elset, elset=Element_4
1021   4,
1022  *Elset, elset=Element_5
1023   5,
1024  *Elset, elset=Element_6
1025   6,
1026  *Elset, elset=Element_7
1027   7,
```

Figure 3.9. Individual Element Set

3.3.5 Splitting ABAQUS Input File

Splitting an .inp file can be useful when a part of the input needs to be edited or written from scratch using a program such as MATLAB. This can be achieved by placing *INCLUDE keyword in the exact position where the original content (now split into a new file) would have been. For example in Fig 3.12 we use *Include keyword just after the *End Assembly keyword to direct ABAQUS to the material.inp file which contains all the material properties. For this to work seamlessly it is recommended that all the input file should be kept in the same location.

```
1761    374,
1762   *Elset, elset=Element_375
1763    375,
1764   ** Section: Element_1_Mat
1765   *Solid Section, elset=Element_1, material=Steel_1
1766   ,
1767   ** Section: Element_2_Mat
1768   *Solid Section, elset=Element_2, material=Steel_2
1769   ,
1770   ** Section: Element_3_Mat
1771   *Solid Section, elset=Element_3, material=Steel_3
1772   ,
1773   ** Section: Element_4_Mat
1774   *Solid Section, elset=Element_4, material=Steel_4
1775   ,
1776   ** Section: Element_5_Mat
1777   *Solid Section, elset=Element_5, material=Steel_5
1778   ,
1779   ** Section: Element_6_Mat
```

Figure 3.10. Individual Element Section

```
 5    101, 102, 103, 104, 205, 206, 207, 208, 309, 310, 311, 312, 413, 414, 415, 416
 6    517, 518, 519, 520, 621, 622, 623, 624
 7   *Elset, elset=Set-1, instance=BeamSolid-1
 8     73,  74,  75, 148, 149, 150, 223, 224, 225, 298, 299, 300, 373, 374, 375
 9   *End Assembly
10   **
11   ** MATERIALS
12   **
13   *Material, name=Steel_1
14   *Density
15    7.83e-09,
16   *Elastic
17   207000., 0.3
18   *Material, name=Steel_2
19   *Density
20    7.83e-09,
21   *Elastic
22   207000., 0.3
```

Figure 3.11. Individual Element Material

```
2908   *Elset, elset=Set-1, instance=BeamSolid-1
2909     73,  74,  75, 148, 149, 150, 223, 224, 225, 298, 299, 300, 373, 374, 375
2910   *End Assembly
2911   **
2912   *INCLUDE,INPUT=material.inp
2913   ** ----------------------------------------------------------------
2914   **
2915   ** STEP: Step-1
2916   **
2917   *Step, name=Step-1, nlgeom=NO
2918   LinearBeam
2919   *Static
2920   1., 1., 1e-05, 1.
2921   **
2922   ** BOUNDARY CONDITIONS
```

Figure 3.12. ABAQUS File Split

4. MATLAB OVERVIEW AND ABAQUS INTERFACING

4.1 Introduction

MATLAB is multi-paradigm programming language and a numeric computing environment that is used by millions of engineers and scientists from around the globe to analyze and design different systems and products [27].It uses a matrix-based language to express computational mathematics [27] which has been heavily used in this thesis for achieving topology/structural optimization. MATLAB's built-in graphics makes it easier to visualize a data set and by analysing such data set one can improve the quality and safety of their product.

4.2 Running ABAQUS File Though MATLAB

4.2.1 Using System Command and ABAQUS License

ABAQUS input file can be run though MATLAB by pulling in the ABAQUS license which is generally located in the C drive, under the folder SIMULIA as shown in Fig 4.1. The command system let's us run command prompt from MATLAB. Other inputs needed include the hyperlink to .inp file, job name and number of processors which is set to 1 since the student version does not support parallel computation. It is extremely important to note that if there is any space in the folder name or file name, the code won't run and will result an error.

```
% Submitting job to Abaqus %
% This code is ment for windows system only %
% Ncpu = 1 - Student Version Limitation %
function run_abaqus(outputFilePath)
    abq_run = system('C:\SIMULIA\Commands\abq2020se.bat input=E:\IUPUI_Stuffs\Projects\Thesis_Project\ESL\Thesis_writing\Documentation_Model\2DBeam.inp job=2DBeam cpus=1 interactive');
end
```

Figure 4.1. ABAQUS Run Using License

4.2.2 Using Command Prompt Though System Command

One can also use the system command directly from MATLAB without pulling in the ABAQUS license on a system which has ABAQUS installed. This method is illustrated in Fig 4.2, here we first provide the path to the working directory and the ABAQUS .inp file

as an input. Later we set the ABAQUS working directory to the directory we just provided. This will ensure that all the outputs are directed to this location and not the default location.

We then run the code the simulation using the system command to launch ABAQUS from command prompt. Depending on the way an optimization code is written the user might want to add a template file too as an input which can be used to update the main input file. In the above case the entire simulation will be run on command prompt outside MATLAB

```
%% Inputs Needed for a Program
clc
clear all
WorkDir='E:\IUPUI_Stuffs\Projects\Thesis_Project\ESL\Thesis_All_Runs_Combined\HCA'; % Working Directory (Should Have inp Files Too)
Inp_File='3DCant';                                                                  % ABAQUS Input File

%% Setting ABAQUS Working Directory for Results
AbaqusDir=['cd ' WorkDir];                                                          % Assiging Change Directory to AbaqusDir

system(AbaqusDir);                                                                  % Setting Dir Though Command Prompt

%% Running ABAQUS Though MATLAB
Itr0Cmd=['abaqus job=Iteration_0 input=' Inp_File];
system(Itr0Cmd);

PauseMatlab(0)
```

Figure 4.2. ABAQUS Run Though System

and the user needs an extra program loop to hold MATLAB in so it doesn't move ahead before the system completes the simulation. ABAQUS generated a lock file ($job_name.lck$) whenever an output database file is opened by with the write access[25]. This includes the instance when the analysis is running and when the output is being written into an output database file [25]. This file is generated by ABAQUS to prevent having simultaneous write permission to the output database by multiple sources [25]. This file is deleted once the output database file is closed i.e. the run is complete [25].

By writing a loop that keeps checking for this file, MATLAB program can be effectively paused by holding it within the loop till the run is complete. The program given in Fig 4.3 uses this method to hold MATLAB in the check loop i.e. the second loop while ABAQUS is still running the model and prevents the program from going any further. But since ABAQUS does not generate this file immediately as soon as it starts, the first loop holds MATLAB in it while ABAQUS is still generating the .lck file

```
function PauseMatlab(Iter)                %checks for .lck file generated by Abaqus during run (del after run)
%Buffer time till Abaqus can generate .lck file
n=0;

while n==0
    n=exist(['Iteration_' num2str(Iter) '.lck'],'file');  %exisit returs 0 since the file does not exisit
end
%Keeps checking for presense of .lck file and there by pauses MATLAB till simulation completes
n=2;

while n==2
    n=exist(['Iteration_' num2str(Iter) '.lck'],'file');  %exisit returs 2 since file exisit
end
```

Figure 4.3. Pause MATLAB

4.3 Read and Write Data Using MATLAB

It is possible to read and write data into ABAQUS using MATLAB since ABAQUS converts the entire model into a set of nodes and elements along with all the attributes defined to it such as the boundary conditions, materials etc. and stores it in a .inp file that can be read into MATLAB just like any word file. Similarly it stores the requested results of a run into .odb file which can be read by MATLAB too. For this one can use MATLAB commands such as textscan, importdata, fprintf, strncmp etc.

4.3.1 Reading ABAQUS Data

The user would want to have the ability to read ABAQUS data to MATLAB in order to perform optimization. Data such as the element nodes, element numbers, element/model material property can be read from the .inp file generated during the modeling using ABAQUS. Data such as the coordinate of the integration points (which can be used as centroid with right settings), element strain energy, nodal displacement and many other data sets that are generated during a run can be read into MATLAB from the .obd file.

Fig 4.4 and Fig 4.5 shows two ways one can import data into MATLAB from the .inp or .obd file. Both of these commands store the data in an array which can be accessed later for optimization. A user can use loops to read the data when needed.

```
%% Scanning and Storing Individual Element's Young's Modulus
d = fopen('material.inp');
d = textscan(d, '%s'); d = d{1};
E_idx = find(strncmp(d,'*Elastic',8))+1;
E= str2double(d(E_idx));
```

Figure 4.4. Reading Data Using Textscan

```
%% Reading Nodal Coordinates from .inp
fidinput=fopen([Inp_File '.inp'],'r');
line_counter=1;
read_line = fgetl(fidinput);
while ischar(read_line)        %ischar checks for string - stopping (true/false)
    read_line = fgetl(fidinput);
    line_counter=line_counter+1;
    header=strfind(read_line,'*Node');   %keeps scanning for *Node phrase in inp file

    if ~isempty(header)      %If exited here then the file is missing *Node
        fclose('all');          % This exit generally happens for output data
        break;                      % and is an indication that the run failed
    end
end

Node_Data=importdata([Inp_File '.inp'],',',line_counter);

Node_Coord=Node_Data.data(:,2:4);
```

Figure 4.5. Reading Data Using Importdata

4.3.2 Writing Data into ABAQUS Input File

A user can write data back into the ABAQUS input file using fprintf command in MATLAB. This step is crucial for any kind of optimization and can be used to write penalized densities or write the element number of the deleted elements back into the input file. A "for" loop can be used when needed as shown in Fig 4.6, this can be helpful when a user needs to write data such as modified Young's Modulus in this case for every single element from an array.

```
for n=1:NmEl
    % Material table
    e_steel = E_Final(n,:);
    mattable            = [e_steel,pr];

    %Using Material ID from the ElemLess Matrix
    mid = n;
    % Material Card
    fprintf(fidoutput,'*Material, name=Steel_%d\n',mid);
    fprintf(fidoutput,'*Density\r\n');
    fprintf(fidoutput,'%g,\r\n',ro);
    fprintf(fidoutput,'*Elastic\r\n');
    fprintf(fidoutput, '%, %10g\r\n',mattable);
end
```

Figure 4.6. Writing Data

5. BIDIRECTIONAL EVOLUTIONARY STRUCTURAL OPTIMIZATION

5.1 Introduction

Evolutionary structural optimization (ESO), which is an important part of topology optimization was initially proposed by Xie and Steve [28], [29]. It's based on the simple concept process that, a structure will evolves towards an optimum design by gradually removing the low stress materials [30]. This method is recognized as a hard-kill method and the associated discrete isn't relaxed in contrast to the density-based method [30]. ESO has been extended for a number of design objectives using either empirical or heuristic criteria, which may or may not be based on the sensitivity information [31]. It has been reported that, the ESO method is equivalent to sequential linear programming approximate method when the update criteria that is used is the strain energy [32]. A summary of the early developments of this method can be found in the first book on this subject by Xie and Steven [31].

Querin et al [33]–[35] developed the early version of bidirectional evolutionary structural optimization (BESO) method which enabled the recovery of the deleted elements neighboring highly stressed elements [30]. One of the last major development of the ESO method was the proposition of the convergent and mesh-independent BESO by Huang and Xie [36]. This development incorporated a sensitivity filter scheme and a stabilization scheme using history information [30]. The latest version of BESO method shows promising performance when applied to a wide range of structural design problems including stiffness and frequency optimization [37], energy absorption [38], nonlinear material and large deformation [39], [40], multiple constraints [37], multiple materials [41], periodic structures [42], etc [43], [44]. The development of bi-directional ESO (BESO) and various applications up to year 2010 has been summarized in the second book of this subject by Huang and Xie [45].

The BESO method shows efficient and robust performance and is a widely adopted design methodology for engineering applications as well as academic research [30].

5.2 Early Developments of BESO

Early developments in ESO were limited to material removal from the structure and the material that is once removed in any iteration couldn't be re-added to the structure [30]. This necessitated the need for an over-sized initial design setting to ensure that the final design contains adequate material [30]. In certain cases the optimization can be misled due to incorrect initial setting [46]. An exactly reverse method to ESO known as Additive ESO (AESO) was proposed by Querin et al [33]. In AESO the structure evolves from a base structure with little material by gradually adding material to the highly stressed regions [30]. Both of these methods ESO and AESO allowed for one directional variation of the structure by removing the material or adding the material [30].

Querin et al [34] proposed the early version of BESO by combing ESO and AESO concept together. While doing so elements with the lowest von Mises stresses were removed which satisfied the first criterion in Eq. 5.1 and the void elements near the regions with high von Mises stress was switched to solid elements satisfying the second criterion in Eq. 5.1 [30]. $\tilde{\sigma}_{\mathrm{e}}$ in Eq. 5.1 is approximate von Mises stresses for the void elements from neighboring side elements [30]. The number of elements that are to be added and removed are treated separately with a inclusion ratio C_{ir} and rejection ratio C_{rr} respectively [30]. Since BESO allows for bidirectional evolution a final optimum can be reached irrespective of the initial design settings. Querin et al validated the optimality of BESO method [35].

$$\begin{cases} \sigma_{\mathrm{e}}^{vm} < C_{rr}.\sigma_{max}^{vm} \rightarrow element\ \ removal \\ \tilde{\sigma}_{\mathrm{e}}^{vm} < C_{\mathrm{ir}}.\sigma_{max}^{vm} \rightarrow element\ \ addition \end{cases} \tag{5.1}$$

BESO method's efficiency is known to be highly dependent on the initial setting and hence by choosing a proper starting point for structural topology the BESO method can generally ensure a quicker process than ESO method [30].

Yang et al [47] conducted a research on use of BESO for stiffness optimization. In his study, the sensitivity of the void element was estimated through a linear extrapolation of the

displacement field after the Finite Element Analysis [30]. Element sensitivity α_e was defined as the variation of the element compliance due to an addition or removal of the element [48].

$$\alpha_e = u_e^T k_e u_e \tag{5.2}$$

where u_e and k_e is the element displacement vector and the stiffness matrix respectively. By ranking the elements using their corresponding sensitivity, the solid elements with the lowest sensitivity is removed from the structure, while, the void elements with the highest sensitivity number is changed to a solid element [30]. This method was later extended into three-dimensional structure too [49].

The early works on ESO and BESO methods were largely based on the heuristic concept and lacked theoretical rigor [30]. These works neglected critical numerical problems in topology optimization such as the existence of a solution, mesh-dependency, checker-board, local optimum, etc. Continuous efforts were put in to address these deficiencies, for instance: Li et al [50] addressed the checkerboard problem by averaging the sensitivity of an element with the neighboring elements, Yang et al. [51] introduced a perimeter constraint to the BESO method, Kim et al. [52], [53] introduced a cavity control technique into the BESO method. Also, with the aim of reducing the computation cost, fixed grids have also been introduced into the BESO method [54], [55].

5.3 Standard Hard-Kill BESO Procedure

Consider a structural optimization problem for minimum compliance ($f_c = u^T K u$, i.e., maximizing the stiffness) subject to material volume fraction constrains [30],

$$\begin{aligned} \min_{\xi} &: f_c(\xi, u) \\ subject\ \ to &: Ku = f \\ &: V(\xi) = \sum \xi_e \vartheta_e = V_{req} \\ &: \xi_e = 0\ \ or\ \ 1, \ \ e = 1, ..., N_e \end{aligned} \tag{5.3}$$

where K is the global structural stiffness, u is the displacement vectorm ϑ_{e} is the element volume, V_{req} and $V(\xi)$ are the required and total material volumes respectively [30]. N_{e} is used to denote the total number of finite elements [30]. The binary design variable ξ_{e} is used to declare an element absent (0) or present (1). [30].

Before the elements are removed or added to the current design, a target volume for the current design iteration $V_{(l)}$ needs to be defined [30]. Since the required material volume V_{req} can be smaller or greater than the initial guess, the target volume in each iteration may increase or decrease step by step until the constrain volume is achieved [30]. This evolution of volume can be expressed as [30],

$$V^{(l)} = V^{(l-1)}(1 \pm c_{er}) \tag{5.4}$$

where the evolutionary ratio c_{er} determines the percentage of material to be removed or added from the design of the previous iteration [30]. Once the target volume V_{req} is reached, the optimization algorithm alters only the topology and keeps the volume constant within a tolerance limit [30].

During each design iteration, the sensitivity numbers that denote the relative ranking of element sensitivities is used to determine material addition or removal [30]. When a uniform mesh is used, the sensitivity for an objective function is defined as [30]

$$\alpha_{\mathrm{e}} = \delta f_c / \delta \xi_{\mathrm{e}} = u_{\mathrm{e}}^T k_0 u_{\mathrm{e}} \tag{5.5}$$

where the element sensitivity is represented by Eq. 5.2. In order to avoid the mesh-dependency and the checkerboard patterns the sensitivity is first smoothed by the means of a filter scheme as shown in Eq. 5.6

$$\alpha_{\mathrm{e}} = \frac{\sum_{\mathrm{j}=1}^{N_{\mathrm{e}}} w_{\mathrm{ej}} \alpha_{\mathrm{j}}}{\sum_{\mathrm{j}=1}^{N_{\mathrm{e}}} w_{\mathrm{ej}}} \tag{5.6}$$

where w_{ej} the linear weight factor [30]

$$w_{\mathrm{ej}} = max(0, r_{min} - \Delta(\mathrm{e}, \mathrm{j})) \tag{5.7}$$

which is determined according to the prescribed radius filter r_{min} and the element center-to-center distance $\Delta(\mathrm{e}, \mathrm{j})$ between the elements Ω_{e} and Ω_{j} [30]. In order to improve the convergence, the filtered sensitivity is further averaged with the sensitivity of the previous iteration

$$\alpha_{\mathrm{e}}^{(l)} \leftarrow (\alpha_{\mathrm{e}}^{(l)} + \alpha_{\mathrm{e}}^{(l-1)})/2 \tag{5.8}$$

The updating of the design variables by BESO method is realized by the means of two threshold parameters α_{del}^{th} and α_{add}^{th} for removing and adding material to the model [40], [56].

$$\xi_{\mathrm{e}}^{(l+1)} = \begin{cases} 0 \quad if \quad \alpha_{\mathrm{e}} \leq \alpha_{del}^{th} \quad and \quad \xi_{\mathrm{e}}^{(l)} = 1, \\ 1 \quad if \quad if \alpha_{add}^{th} \leq \alpha_{\mathrm{e}} \quad and \quad \xi_{\mathrm{e}}^{(l)} = 0, \\ \xi_{\mathrm{e}}^{(l)} \quad otherwise \end{cases} \tag{5.9}$$

This scheme indicates that the solid elements are removed when their sensitivity is less than α_{del}^{th} and the void elements are recovered when their sensitivity is greater than α_{add}^{th} [30]. The parameters α_{add}^{th} and α_{del}^{th} is obtained from the following iterative algorithm [36]

1. Let $\alpha_{add}^{th} = \alpha_{del}^{th} = \alpha_{th}$, where the value of α_{th} is determined iterative such that the required material volume usage is met at the current iteration. [30]

2. Compute the admission ration c_{ar}, which can be defined as the volume of the recovered elements divided by the total volume of the current design iteration [30]. If $c_{ar} \leq c_{ar}^{max}$, which is maximum admission ratio, then we skip the next steps, otherwise, α_{add}^{th} and α_{del}^{th} is redetermined in the next steps [30].

3. Determine α_{add}^{th} iterative using just the sensitivity of the void elements until the maximum admission ration is met, i.e., $c_{ar} \approx c_{ar}^{max}$ [30].

4. Determine α_{del}^{th} iterative using just the sensitivity of the solid elements until the required volume usage is met for the current iteration [30].

By introducing c_{ar}^{max} the topology optimization process is stabilized by controlling the number of recovered elements [30]. Generally, c_{ar}^{max} is set to a value that is greater than 1% so as to not suppress the merit of the element recovery scheme [30]. It should be noted that in stiffness related design, this procedure be reduced to a simpler scheme by assuming that $c_{ar} \leq c_{ar}^{max}$ is always satisfied in practice only α_{th} needs to be determined [30].

The iterative process of FEA and element removal and it's addition continues until the target volume i.e., V_{req} is reached and the convergence criterion defined in the variation of the objective function is satisfied:

$$c_{\mathrm{err}} = \frac{|\sum_{\mathrm{i}=1}^{N}(f_c^{(l-\mathrm{i}+1)} - f_c^{(l-N-\mathrm{i}+1)})|}{\sum_{\mathrm{i}=1}^{N} f_c^{(l-\mathrm{i}+1)}} \leq \delta_{\mathrm{err}} \tag{5.10}$$

where δ_{err} is an allowable convergence error, l is the number of current design iteration and N is integral number which is usually selected as 5 in most of the design cases [30]. This does mean that a stable compliance at least in 10 successive iterations [30].

5.4 Application using ABAQUS-MATLAB Interface

The BESO method explained in this chapter was implemented as a proof of concept for the ABAQUS-MATLAB interface using 3 different cantilever beam model. The beam models were modeled using 2D 4-node plane stress element, 3D 8-node brick element and 4 node shell element. Fig 5.1 illustrates the flow of optimization and was the base for the algorithm.

5.4.1 2D Beam Model

The 2D beam was modeled using the CPS4R element in ABAQUS. This element is a 4-node bilinear, reduced integration element with hourglass control and has 2 degree of freedom at it's nodes [25].

As shown in Fig 5.2 the beam was a $100mm \times 20mm$ beam with 1000 N of force applied to its free end via a concentrated load at it's node. The material properties that were used

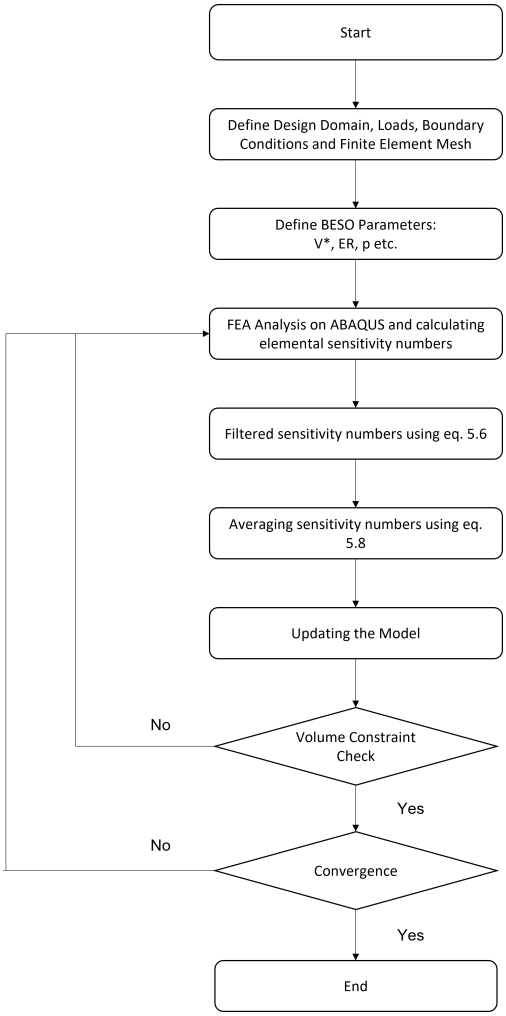

Figure 5.1. BESO Algorithm

was taken from Table 3.1 and the unit system used here is ton, mm, s, N in order to obtain stresses in MPa. The initial run produced a Max Principal Strain of 6.798×10^{-3}, Von Mises Stress of 1.384×10^{3} MPa and a maximum displacement of 2.564 mm in the Y-axis direction and is shown in Fig . 5.3, 5.4, 5.5

This model was optimized using hard-kill BESO with a target volume fraction of 0.5, an evolution rate of 0.01. The model converged at the 58^{th} iteration producing a structure as shown in Fig 5.6

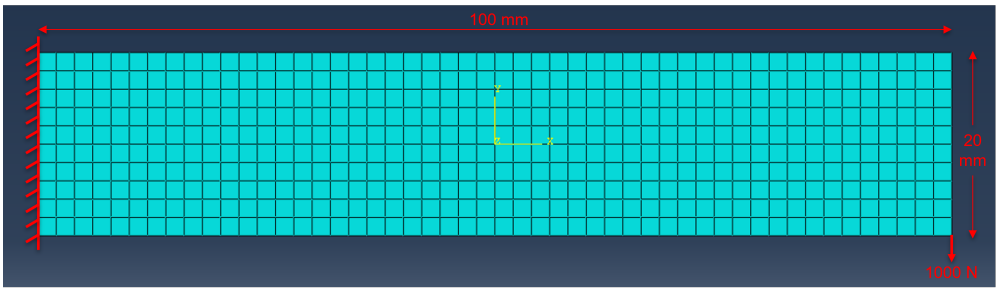

Figure 5.2. 2D Beam - Boundary Condition

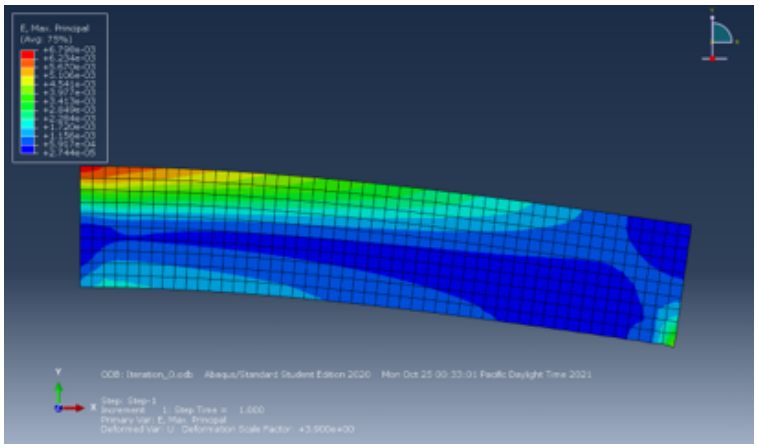

Figure 5.3. 2D Element - Maximum Principal Strain (Iteration 0)

This final run produced a Max Principal Strain of 1.167×10^{-2}, Von Mises Stress of 2.421×10^{3} MPa and a maximum displacement of 4.612 mm in the Y-axis direction and is shown in Fig . 5.7, 5.8, 5.9

The plot between the number of elements deleted v/s iteration shown in Fig 5.10 gives an idea of how the optimization loop worked. The algorithm deleted 250 elements out of the total 500 elements present in the model and resulted in a black and white structure with 50% less elements.

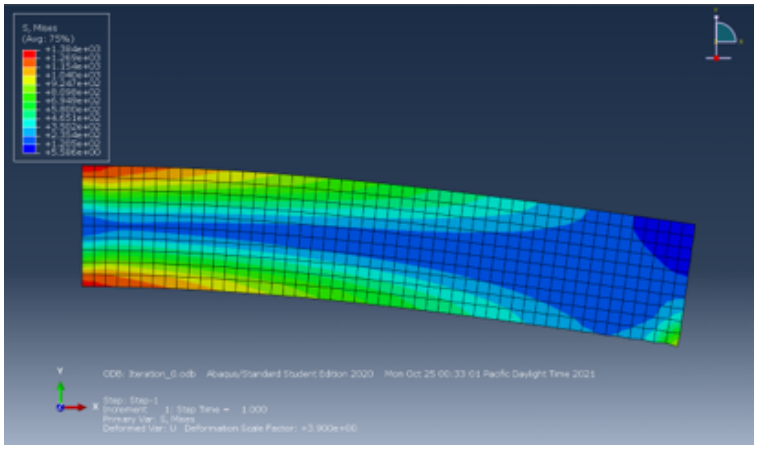

Figure 5.4. 2D Element - Von Mises Stress (Iteration 0)

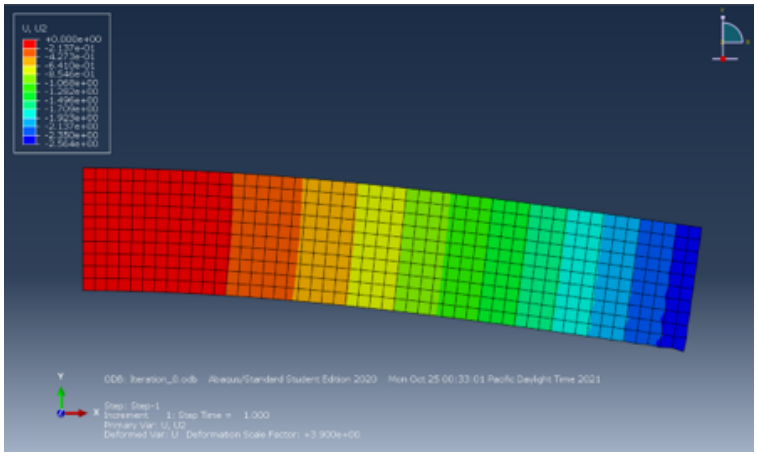

Figure 5.5. 2D Element - Displacement in Y-Axis (Iteration 0)

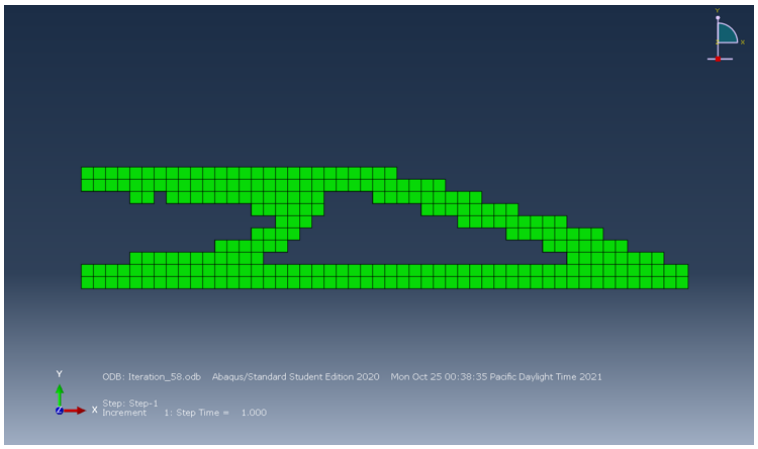

Figure 5.6. 2D Element - Optimized Structure (Iteration 58)

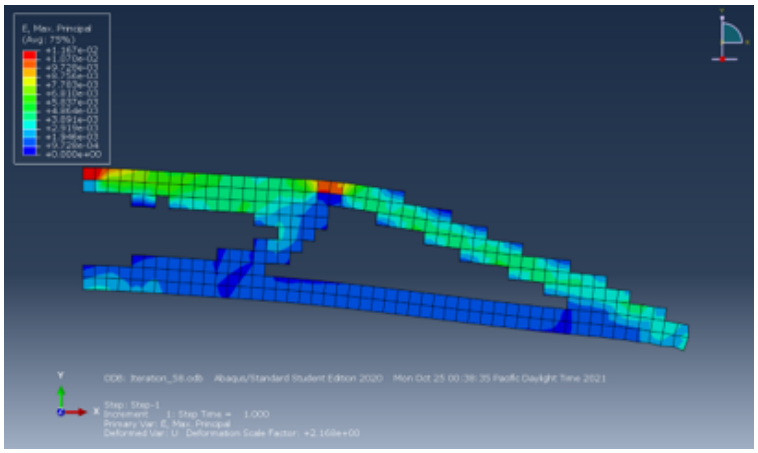

Figure 5.7. 2D Element - Maximum Principal Strain (Iteration 58)

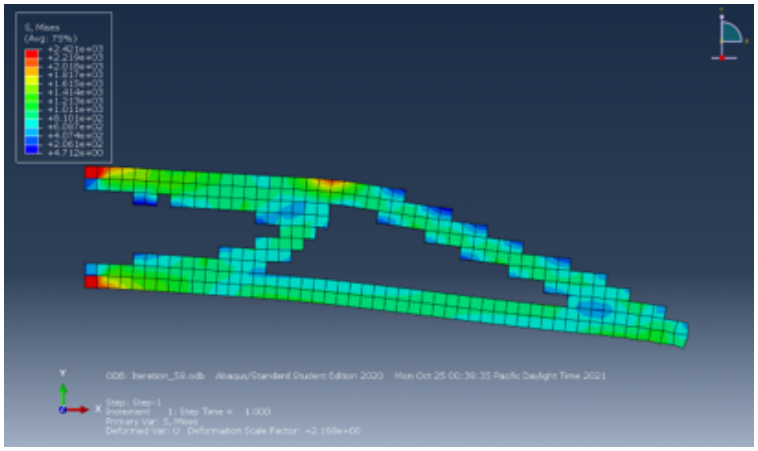

Figure 5.8. 2D Element - Von Mises Stress (Iteration 58)

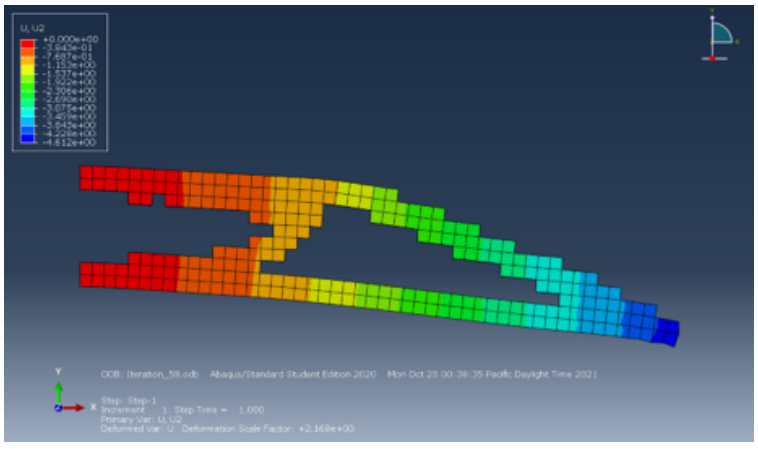

Figure 5.9. 2D Element - Displacement in Y-Axis (Iteration 58)

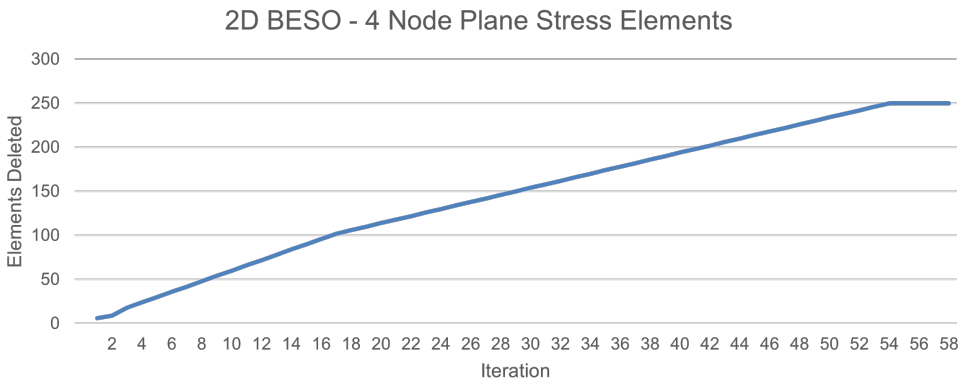

Figure 5.10. 2D Element - Elements Deleted v/s Iteration

5.4.2 3D Beam Model

The 3D beam was created using the C3D8R element in ABAQUS. This element is a 8-node linear brick element, with reduced integration (1 integration point) and has 3 degree of freedom per nodes [25].

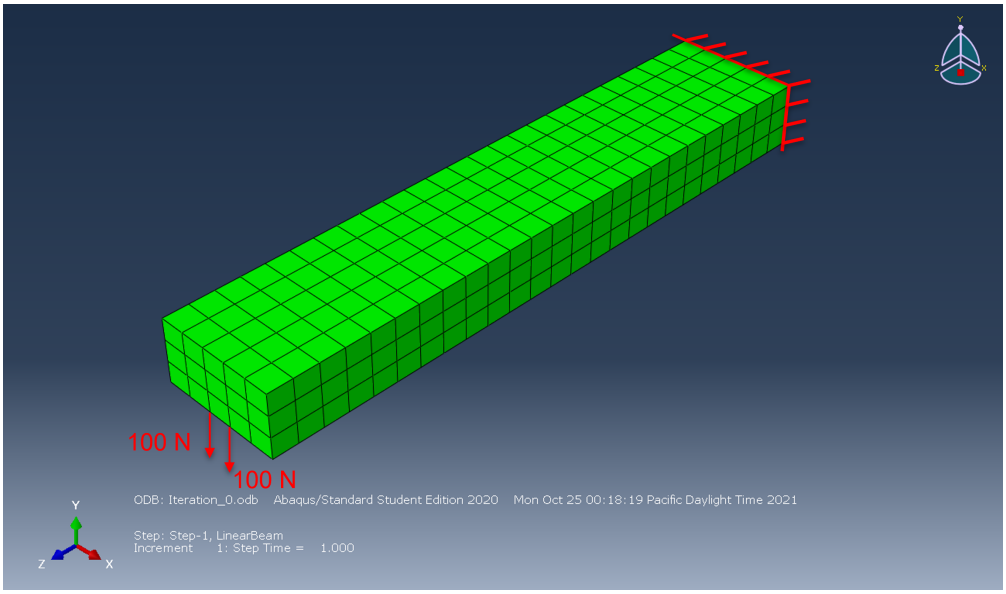

Figure 5.11. 3D Beam - Boundary Condition

Fig 5.11 represents the initial model, the beam was a $100mm \times 20mm \times 10mm$ beam with 100 N of force applied to 2 nodes at the free end. The material properties that were used was taken from Table 3.1 and the unit system used here was ton, mm, s, N in order to obtain stresses in MPa. The initial run produced a Max Principal Strain of 2.127×10^{-4}, Von Mises Stress of 4.423×10^{1} MPa and a maximum displacement of 2.173 mm in the Y-axis direction and is shown in Fig . 5.12, 5.13, 5.14

This model was optimized using hard-kill BESO with a target volume fraction of 0.7, an evolution rate of 0.01. The model converged at the 38^{th} iteration producing a structure as shown in Fig 5.15

This final run produced a Max Principal Strain of 2.123×10^{-4}, Von Mises Stress of 2.421×10^{3} MPa and a maximum displacement of 3.876 mm in the Y-axis direction and is shown in Fig . 5.16, 5.17, 5.18

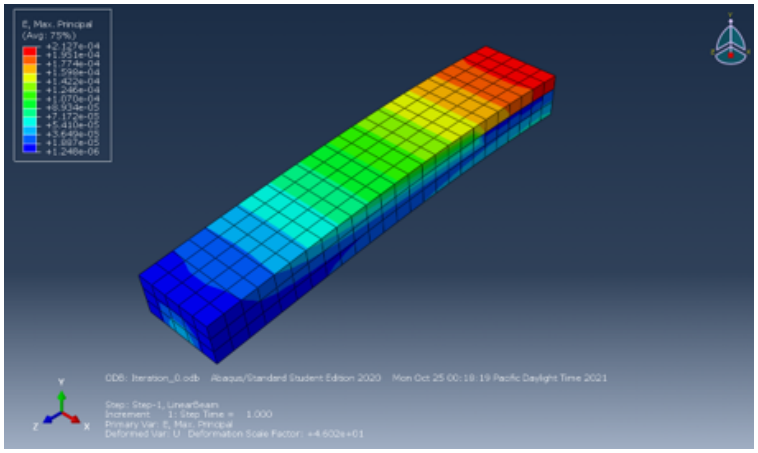

Figure 5.12. 3D Element - Maximum Principal Strain (Iteration 0)

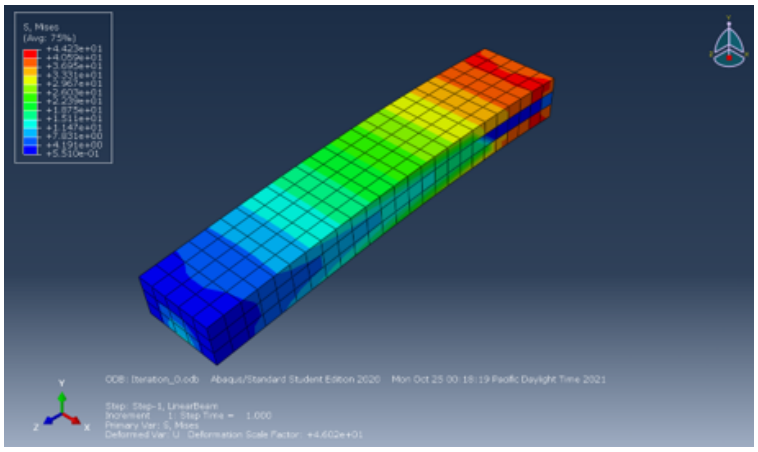

Figure 5.13. 3D Element - Von Mises Stress (Iteration 0)

The plot between the number of elements deleted v/s iteration shown in Fig 5.19 gives an idea of how the optimization loop worked. The algorithm deleted 116 elements out of the total 375 elements present in the model and resulted in a black and white structure with 30.94% less elements.

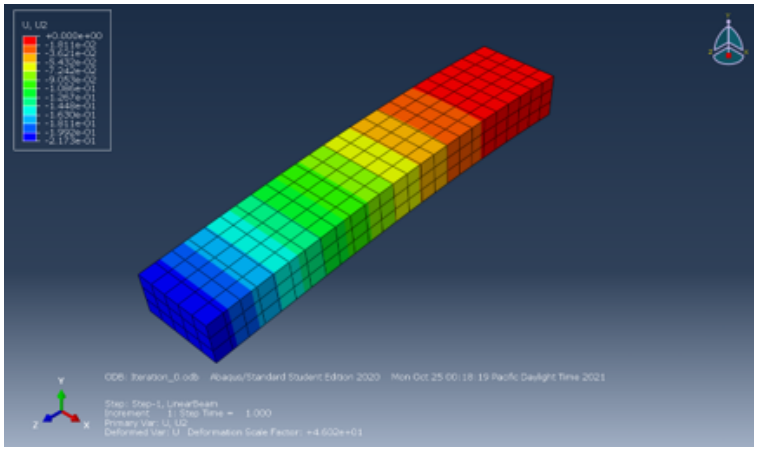

Figure 5.14. 3D Element - Displacement in Y-Axis (Iteration 0)

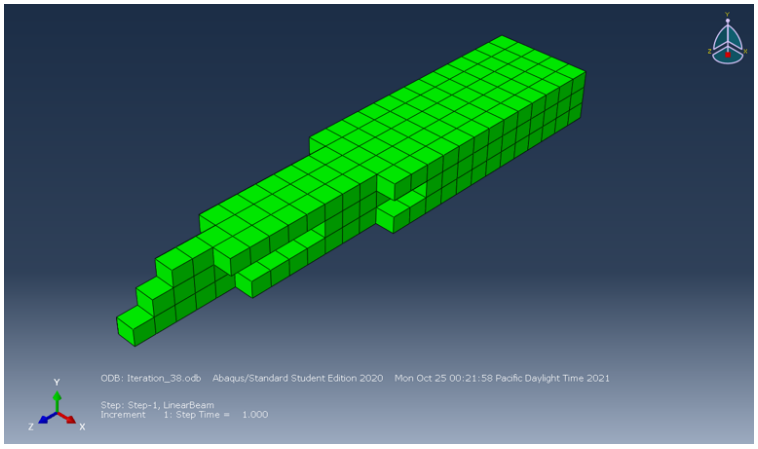

Figure 5.15. 3D Element - Optimized Structure (Iteration 38)

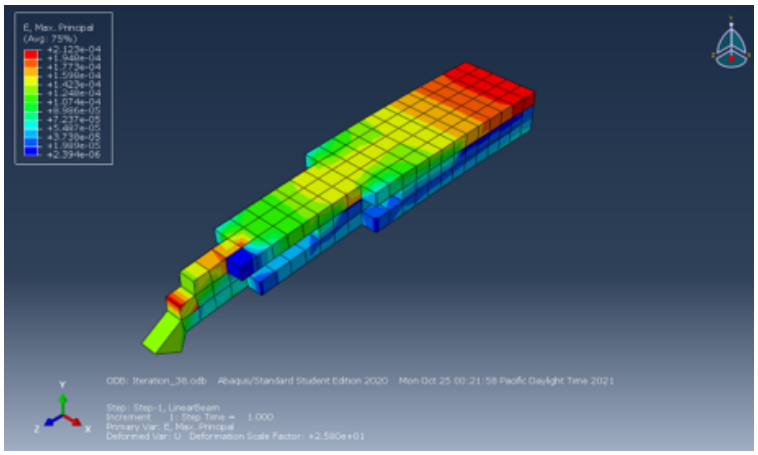

Figure 5.16. 3D Element - Maximum Principal Strain (Iteration 38)

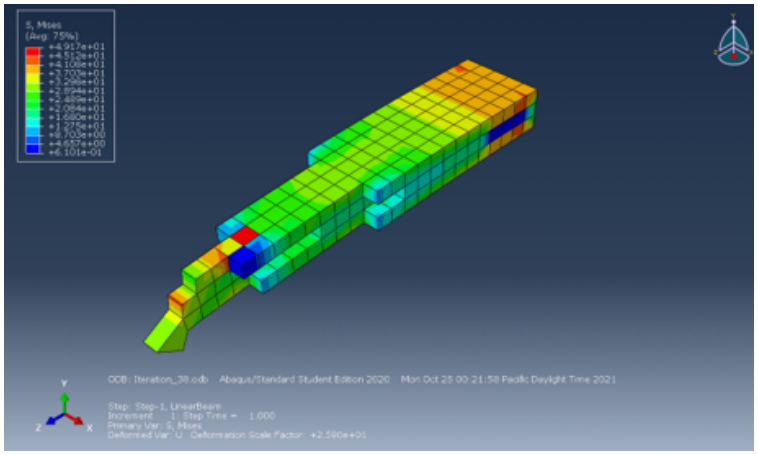

Figure 5.17. 3D Element - Von Mises Stress (Iteration 38)

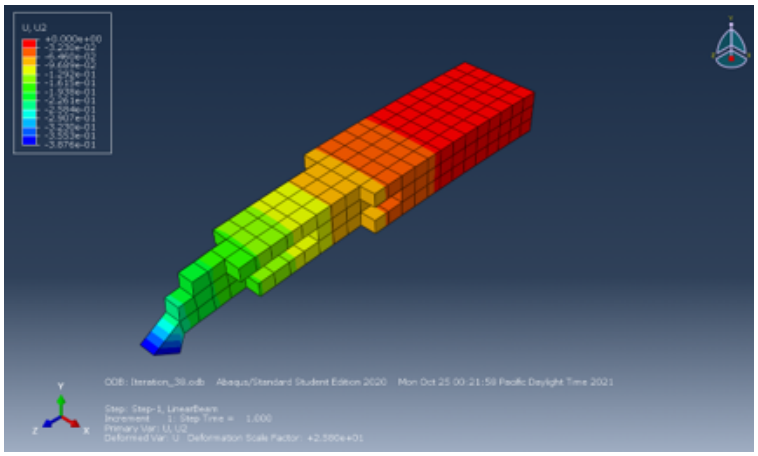

Figure 5.18. 3D Element - Displacement in Y-Axis (Iteration 38)

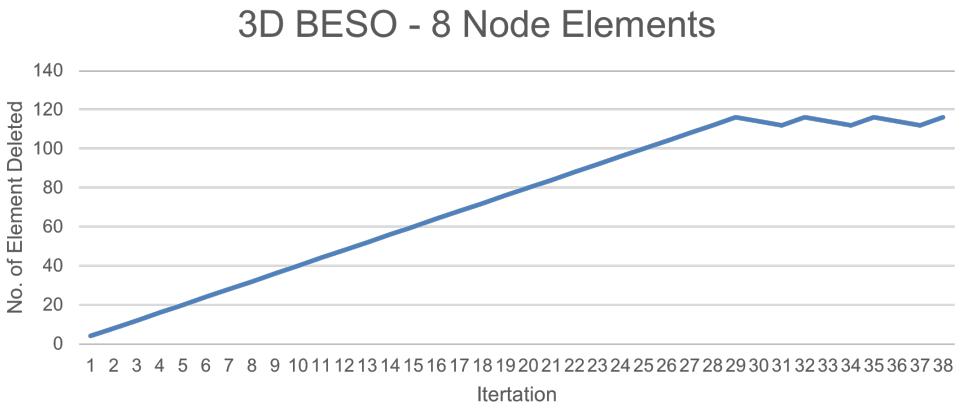

Figure 5.19. 3D Element - Elements Deleted v/s Iteration

6. TOPOLOGY OPTIMIZATION USING HYBRID CELLULAR AUTOMATA

6.1 Introduction

Hybrid Cellular Automata is a topology optimization method that is inspired by a phenomenological model used to simulate bone functional adaptation [57]. The cells named Osteocyte reside within the bone matrix and accounts for 90% to 95% of all the bone cells [58]. These cells are known to be the principal sensors that senses the mechanical loading of the bone and produce soluble factors that regulate the onset of bone formation and resorption [58]. HCA focuses on simulation this exact behaviour of the sensor cells to synthesize optimum structure for given load(s) condition. In HCA, the design domain is discretized into regular lattice of cellular automata (CA) [59]. Each CA in the design domain simulate the behaviour of the sensor cells and senses the strain energy density (SED) within a certain proximity [59]. Based on the SED level, the CA then activates a process of formation or resorption to modify the structure around it [59].

The local control strategy that is used to drive the actual SED level to the target value controls the quantity of the material that is added or removed in each iteration [59]. This control strategy also called as the controller influences the robustness and computational efficiency of the iterative process [59]. This work uses the two-position control strategy for the implementation of HCA using ABAQUS-MATLAB interface.

6.2 Cellular Automaton Parading

Cellular Automata (CAs) have been used in the simulation of biological phenomena for over 60 years [59]. Weiner and Rosenblunth [60] used them as early as in 1946 to describe the operation of the heart muscle, but its was John von Neumann who in the late 1940s formalized the CA theory [61]. Later, some researches recognized CAs as a technology that can change the current paradigms in modeling in physics [62]. Woldfram, refers to the CAs as a new kind of science [63].

This technique in simple terms discretizes a complex problem into a set of simple local rules that in turn operates over a large number of CAs which only knows the local conditions [59]. The cellular automata is an idealization of a physical system in which the space and time is discrete [59]. CAs are composed of regular lattice of cells which are defined by a discrete location i and a set of states $\alpha_{\mathrm{i}}(t) = \alpha_{\mathrm{i}}^{\mathrm{j}}(t)$ [59]. The local rule that governs the evolution of each state and is defined as

$$\alpha_{\mathrm{i}}^{\mathrm{j}}(t+1) = R_{\mathrm{i}}^{\mathrm{j}}(\alpha_{\mathrm{i}}(t), \alpha_{\mathrm{i}+\Delta_1}(t), ..., \alpha_{\mathrm{i}+\Delta_{\hat{N}}}(t)), \tag{6.1}$$

where, the states $\alpha_{\mathrm{i}+\Delta_1}(t), ..., \alpha_{\mathrm{i}+\Delta_{\hat{N}}}(t)$ designate the ones that belong to the neighboring cells of the CA [59]. The neighborhood does not have any restriction on location or size, except that it remains the same for all the CAs [59].CAs are also extremely good for parallel computing since the computations are limited to neighbourhoods in CAs and the local rules are identical for the whole lattice [59].

In practice, often the size of the neighborhood is limited to the adjacent cells but it can also be extended. Fig 6.1 depicts some of the common neighborhood layouts. The most commonly used neighborhood is the von Neumann layout that includes four neighboring cells ($\hat{N} = 4$) and Moore layout that includes eight neighboring cells ($\hat{N} = 8$). While these are the common neighborhood layouts the user can reduce it down to an empty layout ($\hat{N} = 0$) or even extend it as much as the model requires.

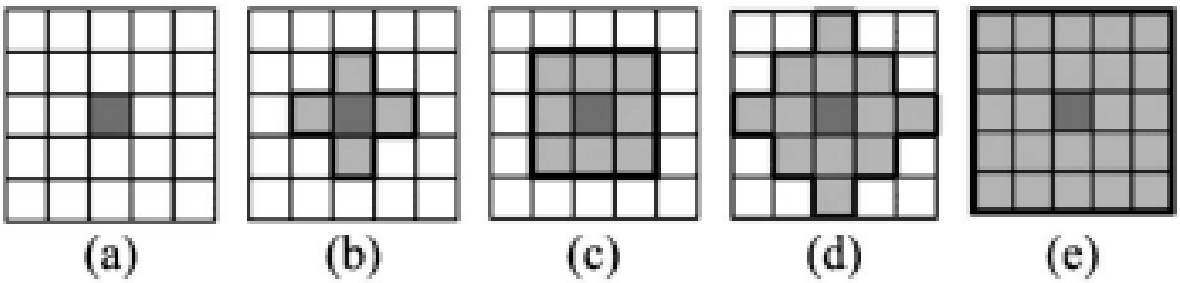

Figure 6.1. Neighborhood Layouts for a Cellular Automaton. (a) Empty, $\hat{N} = 0$; (b) von Neumann,$\hat{N} = 4$; (c) Moore, $\hat{N} = 8$; (d) radial $\hat{N} = 12$; (e) extended,$\hat{N} = 24$.
[59]

In order to define the evolution rule for the cells located on the boundary of the design domain, the domain itself can be extended in different ways [59]. Fig 6.2 depicts some type of boundary conditions that are obtained by extending the design domain [59]. A fixed boundary is defined in order to complete the neighborhood with cells having a pre-assigned fixed state [59]. In an adiabatic boundary, the value of the cell is duplicated into an extra virtual neighbor [59]. For a reflecting boundary, the state of the opposite neighbor is replicated by the virtual cell [59]. A periodic boundary condition is used when the design domain is assumed to be warped in a torus-like shape [59].

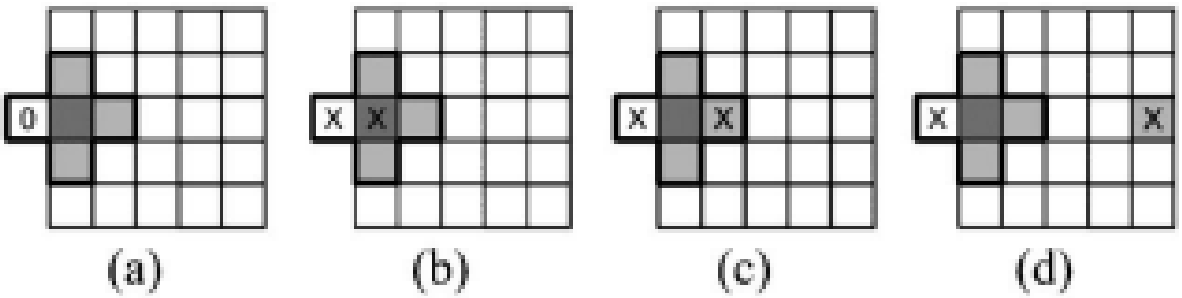

Figure 6.2. Boundary Condition for a CA. (a) Fixed; (b) Adiabatic; (c) Reflecting; (d) Periodic.
[59]

By referring to Eq. 6.1, we can observe that the set of rules $R_{\mathrm{i}} = R_{\mathrm{i}}^{1}, ...R_{\mathrm{i}}^{J^T}$ is identical for all sites and is also applied simultaneously to each of them which leads to a synchronous dynamic [59]. In other words we can say that the rule is homogeneous i.e. it does not depend on the location of a cell [59]. However, it is completely possible to introduce a spatial or temporal inhomogeneities to the model a particular characteristic like a boundary condition or a random phenomena [59]. Asynchronous updating scheme can be useful to model events that does not necessarily occur in parallel [59]. Sometimes it can be desirable to use a specific rule with a certain probability [59]. A CA whose updating rule is driven by an external probability/probabilities is called probabilistic, as opposed to deterministic CAs [59].

In Eq. 6.1, the new state at time t+1 depends on the states at the time t [59]. It can sometimes be necessary to have a lingering memory and introduce a dependence on the states at time t-1,t-2,...,t-T [59]. Depending on definition, the rule can or cannot be reversible [59].

6.3 Cellular Automata for Structural Synthesis

CA models have inspired a number of techniques for structural synthesis and optimization [59]. A basic Cellular Automata like approach was developed by Inou et al [64] in which the elastic modulus of the cell was used as the design variable. In this method a local rule is used to update the modulus of each cell in each iteration based on the difference between the current stress value and the target value [59]. This process is fine-tuned using the evolutionary rules based on the growing/reforming procedure [59].The cells with lower elastic modulus is removed while the cells with a higher elastic modulus create a new cell in an empty space surrounding it [59]. This approach then leads to a structure that is similar to the ones observed in a birds bone [65]. Even though this isn't necessarily a topology optimization algorithm, it does illustrate the application of an evolutionary CA model [59].

Kita and Toyoda later presented the concept of a CA model for use in topology optimization [66]. In this approach, they used the thickness as the design variable and the local design rule is derived from the optimally condition of a multi-objective function [59]. In this function both the the weight of a structure and the deviation between the equivalent stress and the yield stress in a Moore neighborhood is to be minimized [59]. FEM is used to evaluate the stresses in each iteration [59]. This algorithm requires hundreds if iterations in order to achieve a convergence, even for a simple test problem [59].

The CA model that was presented by Tatting and Gürdal [67] implements a simultaneous analysis and design (SAND) approach [59]. In this SAND-CA method, the need for finite element analysis is eliminated by the use of local equilibrium [59]. The optimizer will simultaneously drive the local field states to a target value that converge the residuals to zero, while performing the structural synthesis [59]. This approach does require hundreds of thousands of iterations to achieve convergence; however, the overall computation time is or can be reduced when compared to the techniques based on FEM [59]. In this method, the con-

vergence can deteriorate with increase in the number of elements [59]. This happens because the field variable information is propogated slowly [59]. This problem can be mitigated with multigrid and full multigrid acceleration strategies [68]. In one of the publications, Abdalla and Gürdal [69] demonstrates the application of SAND-CA approach to column design for buckling.

In an another application, Hajela and Kim [70] combined genetic algorithms (GAs) and cellular automata (CAs) for the structural analysis of two-dimensional elastic problems [59]. The local rules in this approach was derived using the principle of minimum energy and GA optimization process [59]. The strain fields exhibited in the results were very close to the ones obtained from analytical solutions [59]. Even though the CA method wasn't used for structural synthesis, their work showed a strategy to develop local rules for use in structural analysis and avoid use of global analysis i.e. FEM [59].

In conventional CA method a global analysis of the field state is not performed [59]. But, HCA method makes use of finite element analysis (FEA) to evaluate the field states, that is strain energy density (SED) [59]. In this context, the work done by Kita and Toyoda [66] can be considered as hybrid method since they used FEA to update the stress states during each iteration of the algorithm [59]. The hybrid cellular automata method is a finite element based approach and hence it sets the residual between the external work and the internal energy to zero at every iteration [59]. Contrarily to this in the SAND-CA implementation by Tatting and Gürdal [67], the residuals are iterative reduced to zero by the optimizer [59].

6.4 Hybrid Cellular Automata Algorithm

In nature all the biological structures continuously adapt to the changes in their physical environment. This can be a slow change like evolution or a fast change like bone formation changes in response to an external loading. If we look at the case of bones in particular we can see that mineral tissue is resorbed in regions that are exposed to a low mechanical stimulus and at the same time new bone is deposited in regions where the stimulus is high [59]. This process of functional adaptation is thought to be the reason why bones can perform its mechanical functions with a minimum amount of mass [59].

A number of theoretical models for bone remodeling use a concept of an error signal as a part of the strategy to simulate bone structural adaptation [71], [72]. These models imply that there exists a state of equilibrium (or zero error condition) where the bone structure is perfectly adapted to the environment and no remodeling is needed [59]. Martin et al [73] and Hart [74] provides good insight into the bone remodeling computational models.

Bone tissue mainly consist of collagen, mineral and water [59]. From the physiological standpoint, the bone contains bone cells besides the mineralized matrix [59]. These bone cells can be divided based on their function, osteoblast are the cells that form new bones, osteoclasts are the cells responsible for to resorb the bone, and osteocytes are the cells that senses mechanical stimuli [59]. When osteocytes detect a deviation in the level of mechanical stimulus with respect to the equilibrium state, it sends a signal to the the osteoclasts and osteoblast to remodel the local bone structure [59]. This local control rule forms the bases of the hybrid cellular automata (HCA) algorithm [57].

In the hybrid cellular automata algorithm, the state of each cell $\boldsymbol{\alpha}_{\mathrm{i}}$, is defined by the design variable $\boldsymbol{x}_{\mathrm{i}}$ (e.g., density, elastic modulus, geometry) and the field variable $\boldsymbol{S}_{\mathrm{i}}$ (e.g., strain energy density, strain, stress) [59]. Hence, the state of a cell can be expresses as [59],

$$\boldsymbol{\alpha}_{\mathrm{i}} = \begin{pmatrix} \boldsymbol{x}_{\mathrm{i}}(t) \\ \boldsymbol{S}_{\mathrm{i}}(t) \end{pmatrix} \tag{6.2}$$

The design variable $\boldsymbol{x}_{\mathrm{i}}$ is defined according to the material model [59]. HCA makes use of the power law for elastic modulus penalization purpose i.e. the elastic modulus of each element E_{i} is modeled as a function of the relative density x_{i}. This can be expressed as [59]

$$E_{\mathrm{i}}(x_{\mathrm{i}}) = x_{\mathrm{i}}^{p} E_0 \qquad (p \geq 1) \tag{6.3}$$

$$\rho_{\mathrm{i}}(x_{\mathrm{i}}) = x_{\mathrm{i}} \rho_0 \qquad (0 \leq x_{\mathrm{i}} \leq 1) \tag{6.4}$$

where E_0 is the modulus of elasticity of the base material, ρ_0 is the density, ρ_{i} is a variable density and p is the penalization power [59]. This power penalizes the intermediate densities and by doing so drives the design towards a black and write structure [59].

The definition of the field variable $\boldsymbol{S}_{\mathrm{i}}$ will depend on the optimization problem to be solved [59]. For design optimization of a rigid structure, strain energy U quantifies the mechanical engineer that is stored while undergoing deformation. Hence, they can be thought to be inversely proportional i.e., the more rigid a structure will be, the lower the strain energy it can store [59]. In a discretized domain, strain energy can be expressed as [59]

$$U = \sum_{\mathrm{i}=1}^{N} U_{\mathrm{i}} v_{\mathrm{i}}, \tag{6.5}$$

where v_{i} is the volume of the CA and U_{i} is the strain energy density (SED) [59]. Since SED is the local indicator of the rigidity of a structure it is used as the field variable in this work. This optimization criteria is called as uniform strain energy density distribution (USEDD) [75]. With this criterion, we can define the state of a CA as [59],

$$\boldsymbol{\alpha}_{\mathrm{i}} = \begin{pmatrix} \boldsymbol{x}_{\mathrm{i}}(t) \\ \boldsymbol{U}_{\mathrm{i}}(t) \end{pmatrix} \tag{6.6}$$

This USEDD criterion follows the principles of a fully stressed design and hence maximum rigidity is reached with minimum amount of material [59]. Locally, this optimization problem can be stated as [59],

$$\begin{aligned} \min_{x_{\mathrm{i}}} \quad & | \, \mathrm{e}_{\mathrm{i}} \, | \\ s.t. \quad & 0 < x_{\mathrm{i}}^{min} \leq x_{\mathrm{i}} \leq 1 \end{aligned} \tag{6.7}$$

The error signal e_{i} can be defined as [59],

$$\mathrm{e}_{\mathrm{i}} = \overline{U}_{\mathrm{i}} - U_{\mathrm{i}}^{*} \tag{6.8}$$

where $\overline{U}_{\mathrm{i}}$ is an average SED value and U_{i}^{*} is a local SED target [59]. This average value can be defined as [59]

$$\overline{U}_{\mathrm{i}} = \frac{U_{\mathrm{i}} + \sum_{\mathrm{j}=1}^{\hat{N}} U_{\mathrm{j}}}{\hat{N} + 1} \tag{6.9}$$

where $\overline{N}$ is the number of neighbours defined in CA neighborhood and U_{j} corresponds to the SED of a neighboring cell [59]. The lower bound x_{i}^{min} in Eq. 6.7, is needed to avoid a singularity in the stiffness matrix for FEA [59].

6.5 Local Control Rule

The design domain comprises of CAs that apply a local design rule using the control theory [59]. The local rule aims to minimize the deviation $\mathrm{e_i}(t)$ between the target SED U_{i}^{*} and the averaged SED $\overline{U}_{\mathrm{i}}(t)$ [59]. Target SED indirectly determines the amount of material in the final structure [59]. The change in mass is given as [59],

$$x_{\mathrm{i}}(t+1) = x_{\mathrm{i}}(t) + \Delta x_{\mathrm{i}}(t) \tag{6.10}$$

The change in relative density is given as [59]

$$\Delta x_{\mathrm{i}}(t) = f(\mathrm{e_i}(t)) \tag{6.11}$$

where the function $f(\mathrm{e_i}(t))$ corresponds to the local control rule and the error signal $(\mathrm{e_i}(t)$ is defined by Eq. 6.8.

6.5.1 Two-Position Control

Two-position control is the simplest strategy that can be used to modify the relative density or the design variable [59]. In this rule, the change in relative density can be determined by a discrete value c_T and the sign of error signal [59]. This is given by [59],

$$\Delta x_{\mathrm{i}}(t) = c_T \times sgn[\mathrm{e_i}(t)] \tag{6.12}$$

where c_T is a positive constant and [59]

$$sgn[\mathrm{e_i}(t)] = \begin{cases} +1.0 \quad if \quad \mathrm{e_i}(t) > 0 \\ 0.0 \quad if \quad \mathrm{e_i}(t) = 0 \\ -1.0 \quad if \quad \mathrm{e_i}(t) < 0 \end{cases} \tag{6.13}$$

If the error signal $\mathrm{e_i}(t)$ in the CA is non-zero, the relative mass changes the amount of c_T [59]. This change in relative density will continue until the CA saturates, a condition that occurs when the CA reaches the minimum or maximum relative density value $x_\mathrm{i} = x_\mathrm{i}^{min}$ or $x_\mathrm{i} = 1$ [59]. This control is simple but it does not work well with all the systems [59].

6.6 Application using MATLAB-ABAQUS Interface

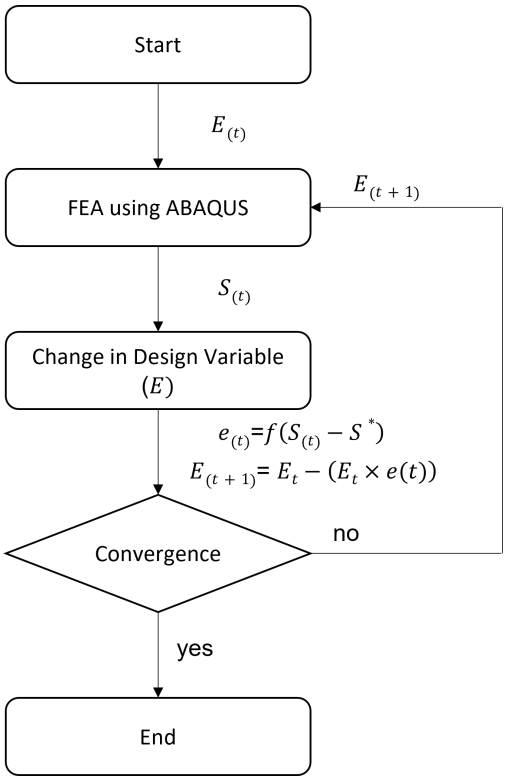

Figure 6.3. HCA Algorithm

Fig 6.3 represent that algorithm that is used for this work. Since ABAQUS was used as the FEA solver Young's Modulus (E) was penalized in order to reach the targeted SED. The algorithm starts by running the model on ABAQUS using the ABAQUS-MATLAB interface

developed in this work. Once the run is complete critical data sets such as the strain energy, nodal coordinates, element number and nodes associated with it and the centroid is extracted from the run. The user can use the above data to develop a neighborhood function for defining neighbourhood of each element. The neighbourhood can be of any limit depending on the model and its needs.

For this work the neighbourhood was considered as the entire model and the strain energy density was calculated for each element and averaged to get the average strain energy density of the entire model (neighbourhood). This is the target value $S*$ for the model. By using the two-position control explained earlier, the error signal is calculated. When the CA is saturated as explained in two-position control the optimization is complete. Till then the design variable i.e. Young's Modulus (E) is penalized as shown in Fig 6.3 in order to increase or decrease the contribution of an element to the entire model.

This change in E is same as penalizing the density of the model in order to stabilize the model during optimization and drive the solution to a black and white structure.

6.6.1 2D Beam Model

The 2D beam was modeled using the CPS4 element in ABAQUS. This element is a 4-node place stress element, meaning it has 2 degree of freedom at it's nodes [25]. It should also be noted that no filters were used for this run which explains the checkerboard pattern observed in the result. By using a filter it is possible to overcome the pattern.

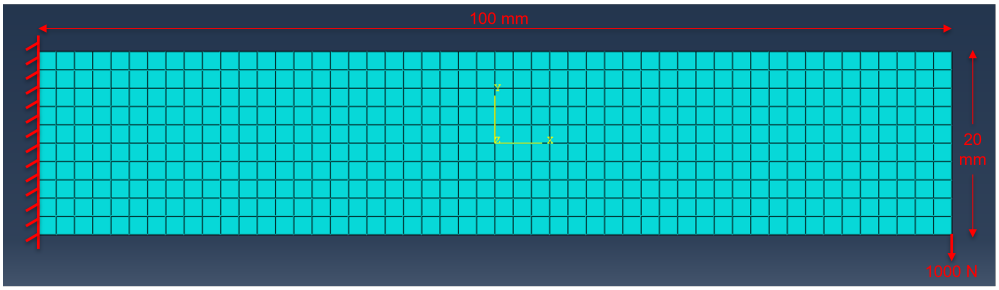

Figure 6.4. HCA 2D Beam - Boundary Condition

As shown in Fig 6.4 the beam was a $100mm \times 20mm$ beam with 1000 N of force applied to its free end via a concentrated load at it's node. The material properties that were used

was taken from Table 3.1 and the unit system used here is ton, mm, s, N in order to obtain stresses in MPa. The initial run produced a Max Principal Strain of 8.218×10^{-3}, Von Mises Stress of 1.658×10^{3} MPa and a maximum displacement of 1.870×10^{-1} mm in the Y-axis direction and is shown in Fig . 6.5, 6.6, 6.7. It should also be noted that this work exist in the linear elastic region and hence all the examples used here are linear elastic. For some higher stress problems it can be desirable to move to the plastic/non-linear region but isn't a part of this work.

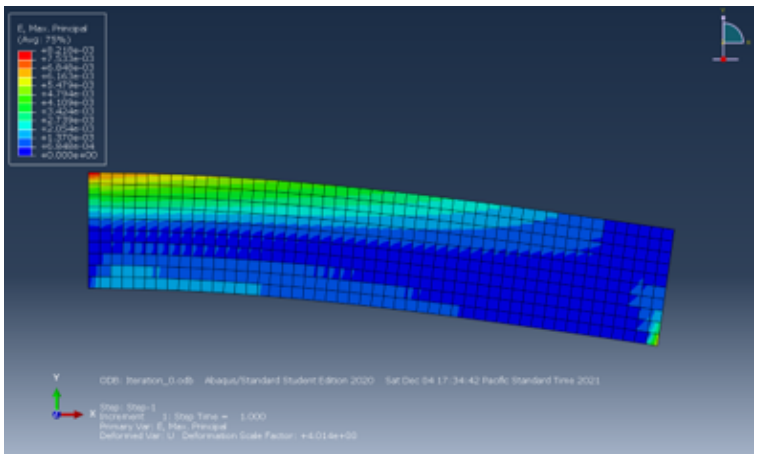

Figure 6.5. HCA 2D Element - Maximum Principal Strain (Iteration 0)

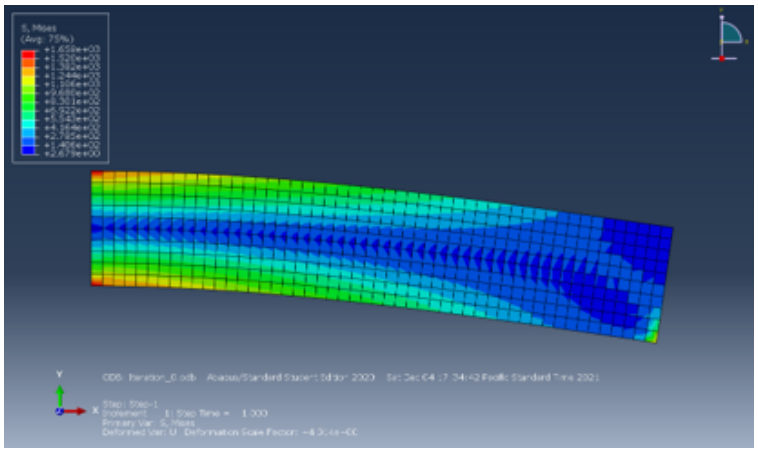

Figure 6.6. HCA 2D Element - Von Mises Stress (Iteration 0)

This model was optimized using HCA with a penilazation power of 0.01. The model converged at the 200^{th} iteration producing a structure as shown in Fig 6.8

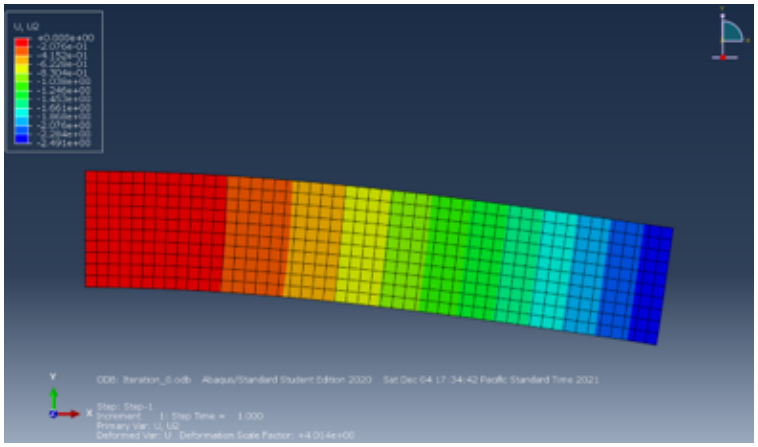

Figure 6.7. HCA 2D Element - Displacement in Y-Axis (Iteration 0)

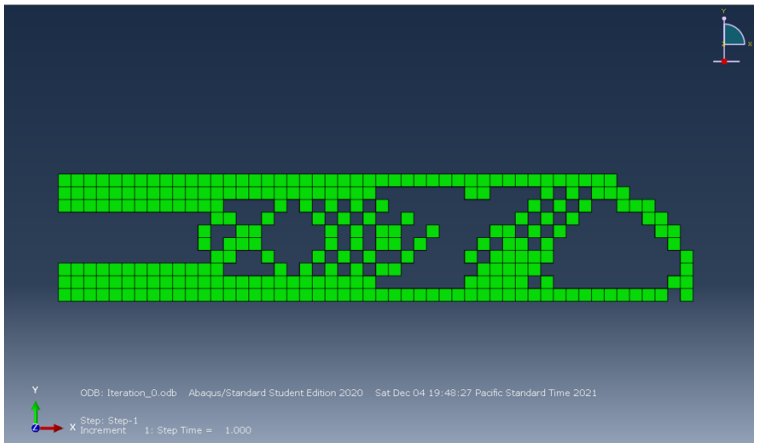

Figure 6.8. HCA 2D Element Optimized Structure (Iteration$_2$00)

This final run produced a Max Principal Strain of 8.682×10^{-3}, Von Mises Stress of 6.193×10^3 MPa and a maximum displacement of 4.9×10^{-1} mm in the Y-axis direction and is shown in Fig . 6.9, 6.10, 6.11.

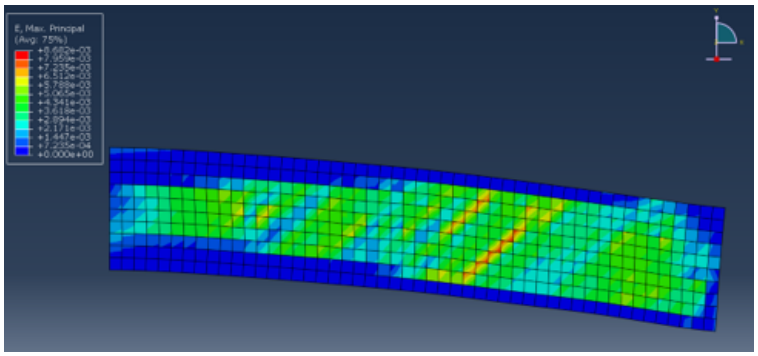

Figure 6.9. HCA 2D Element - Maximum Principal Strain (Iteration 200)

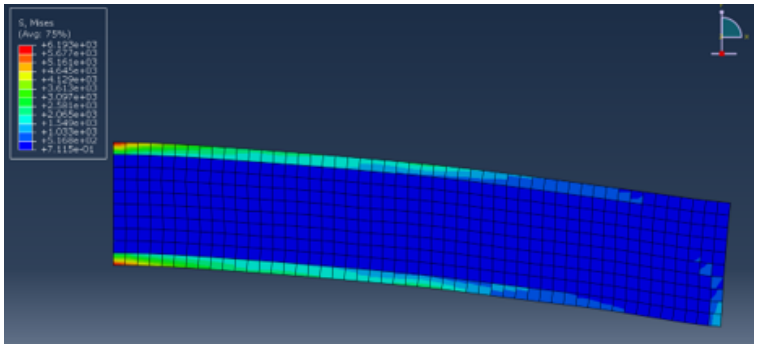

Figure 6.10. HCA 2D Element - Von Mises Stress (Iteration 200)

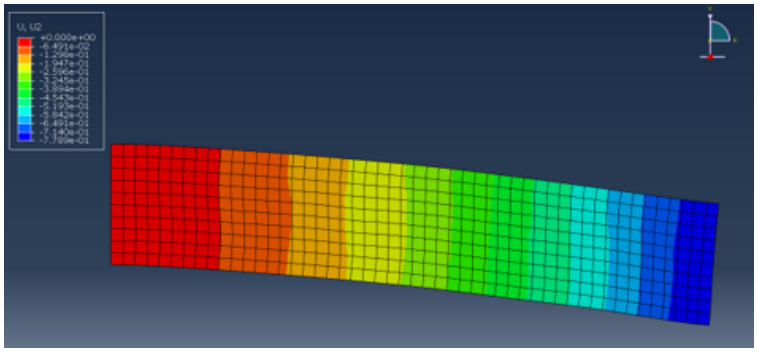

Figure 6.11. HCA 2D Element - Displacement in Y-Axis (Iteration 200)

6.6.2 3D Beam Model

The 3D beam was created using the C3D8R element in ABAQUS. This element is a 8-node linear brick element, with reduced integration (1 integration point) and has 3 degree of freedom per nodes [25].

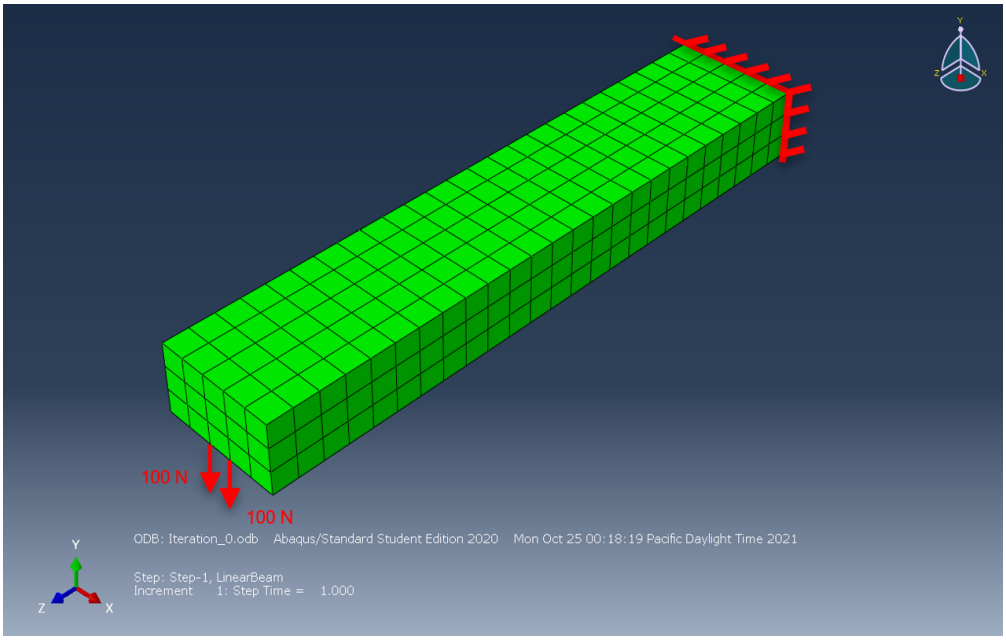

Figure 6.12. HCA 3D Beam - Boundary Condition

Fig 6.12 represents the initial model, the beam was a $100mm \times 20mm \times 10mm$ beam with 100 N of force applied to 2 nodes at the free end. The material properties that were used was taken from Table 3.1 and the unit system used here was ton, mm, s, N in order to obtain stresses in MPa. The initial run produced a Max Principal Strain of 2.127×10^{-4}, Von Mises Stress of 4.423×10^{1} MPa and a maximum displacement of 2.173 mm in the Y-axis direction and is shown in Fig . 6.13, 6.14, 6.15

This model was optimized using HCA with a penalization power of 0.01. The model converged at the 200^{th} iteration producing a structure as shown in Fig 6.16

. The element with the highest amount of material is represented in red ($E = 1.514 \times 10^{6}$ MPa). and the element with the lowest amount of material in dark blue ($E = 2.77 \times 10^{4}$ MPa).

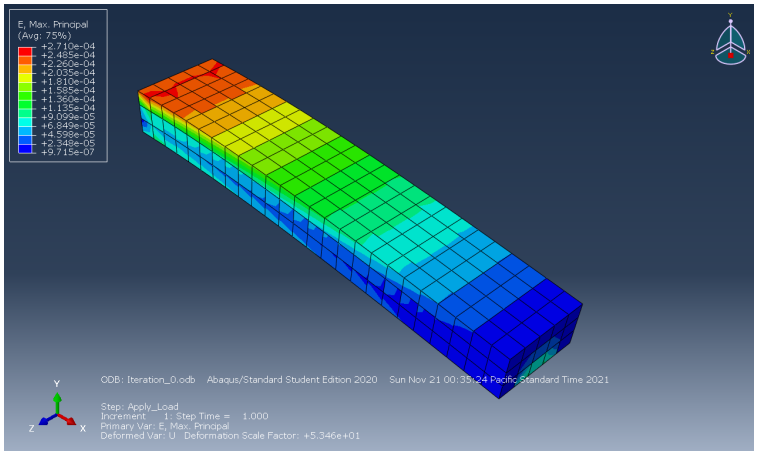

Figure 6.13. HCA 3D Element - Maximum Principal Strain (Iteration 0)

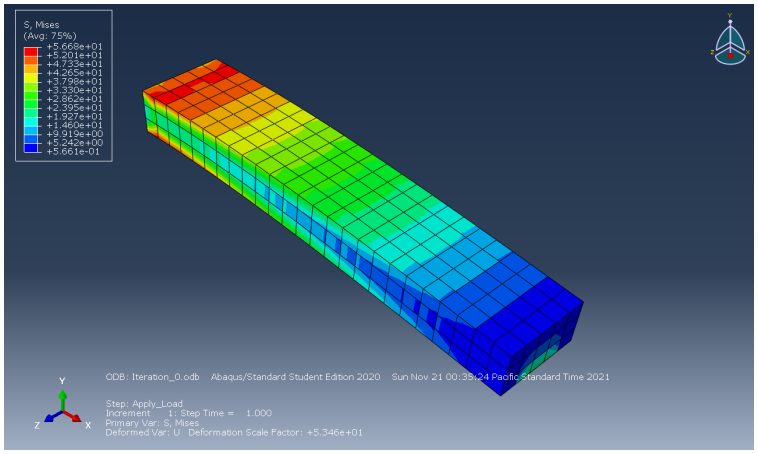

Figure 6.14. HCA 3D Element - Von Mises Stress (Iteration 0)

As we can see by using the Hybrid Cellular Automaton method we can optimize the structure more efficiently compared to Bidirectional Evolutionary Optimization method. The response surfaces are within the permissible limits of commonly used industrial grade steel too.

This final run produced a Max Principal Strain of 1.61×10^{-4}, Von Mises Stress of 4.454×10^{1} MPa and a maximum displacement of 1.013e-1 mm in the Y-axis direction and is shown in Fig . 6.17, 6.18, 6.19

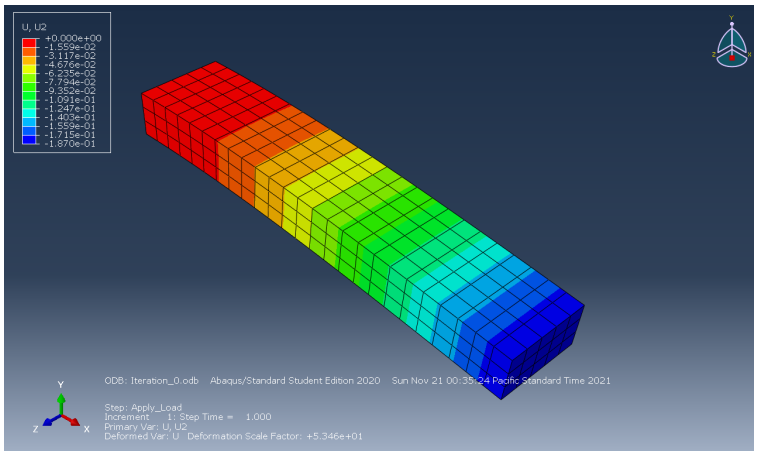

Figure 6.15. HCA 3D Element - Displacement in Y-Axis (Iteration 0)

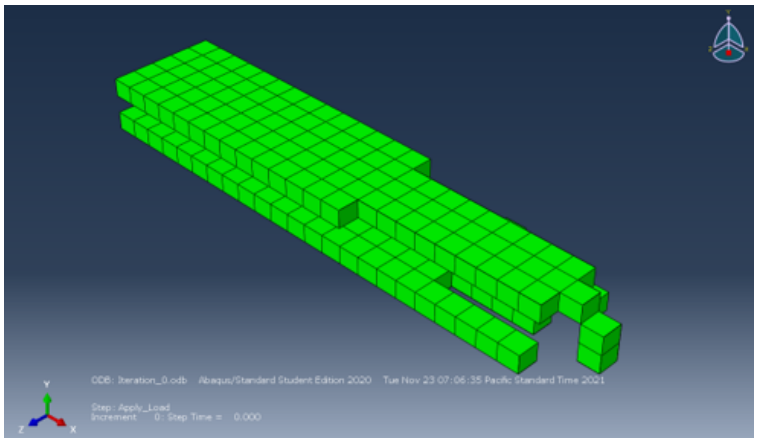

Figure 6.16. HCA 3D Element - Optimized Structure (Element Delete) (Iteration 200)

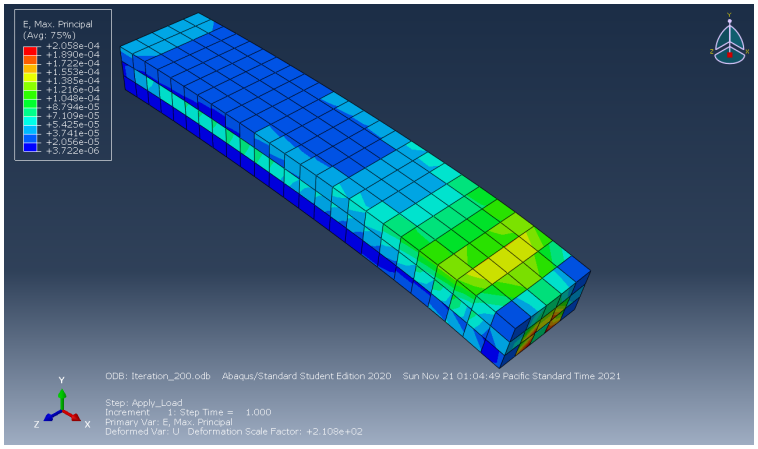

Figure 6.17. HCA 3D Element - Maximum Principal Strain (Iteration 200)

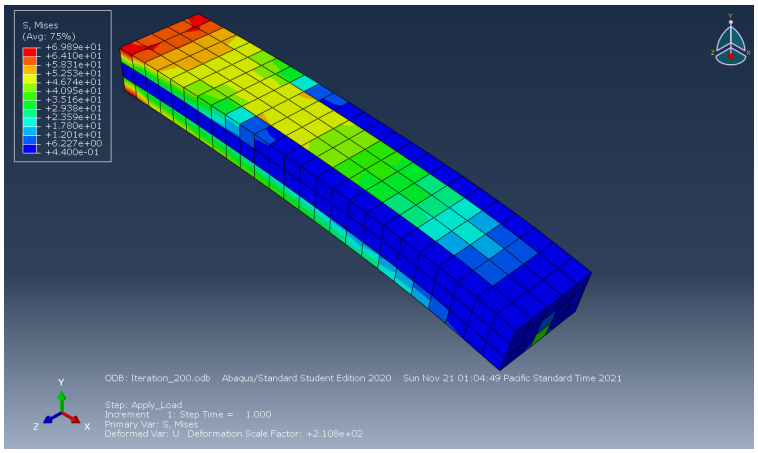

Figure 6.18. HCA 3D Element - Von Mises Stress (Iteration 200)

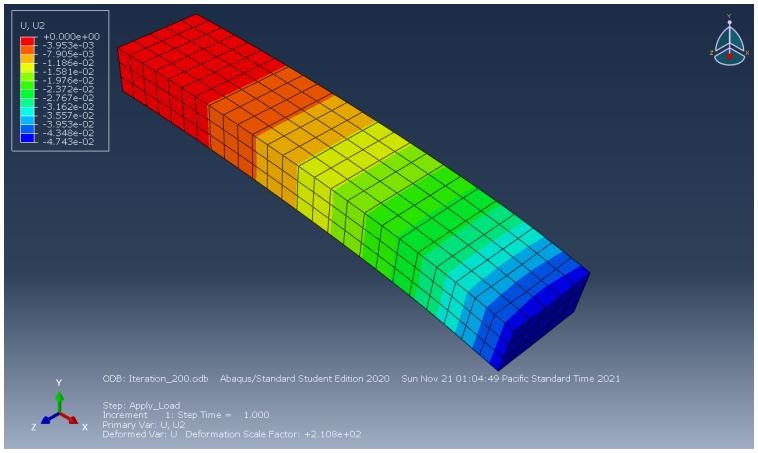

Figure 6.19. HCA 3D Element - Displacement in Y-Axis (Iteration 200)

7. SUMMARY

As a part of this research an ABAQUS-MATLAB based interface was developed and successfully tested on two types of optimization methods, namely Bidirectional Evolutionary Structural Optimization (BESO) method and Hybrid Cellular Automata (HCA) method.

Since ABAQUS files can be read into MATLAB using textscan and various other methods to import data as strings, this capability was extensively used to build this interface. As a part of work two methods were used to run an ABAQUS input file, by directly launching ABAQUS though system command in MATLAB or by pulling in the ABAQUS license. Since optimization requires data to be written into the model file MATLAB command fprintf was used to fulfil this purpose.

During this research the algorithm was tested for a hard-kill method using Bidirectional Evolutionary Structural Optimization (hard-kill) method and for a SIMP based approach using Hybrid Cellular Automita.

This work exist in linear elastic region and can be used as the base to extend the optimization to non-linear region. By extending this work to non-linear region an engineer or researcher can use this method for crashworthiness problems.

Since this algorithm interfaces with ABAQUS which gives an option to extract the stiffness matrix for a model, this method is well suited for Equivalent Static Load based topology optimization method which is the only other method other than HCA that can be used for non-linear/crash analysis.

REFERENCES

[1] T. Coleman, M. A. Branch, and A. Grace, "Optimization toolbox," *For Use with MATLAB. User's Guide for MATLAB 5, Version 2, Relaese II*, 1999.

[2] G. Papazafeiropoulos, M. Muñiz-Calvente, and E. Martınez-Pañeda, "Abaqus2matlab: A suitable tool for finite element post-processing," *Advances in Engineering Software*, vol. 105, pp. 9–16, 2017.

[3] K. Liu and A. Tovar, "An efficient 3d topology optimization code written in matlab," *Structural and Multidisciplinary Optimization*, vol. 50, no. 6, pp. 1175–1196, 2014.

[4] L. A. Schmit, "Structural design by systematic synthesis," in *Proceedings of the Second National Conference on Electronic Computation, ASCE, Sept., 1960*, 1960.

[5] M. P. Bendsøe and N. Kikuchi, "Generating optimal topologies in structural design using a homogenization method," *Computer methods in applied mechanics and engineering*, vol. 71, no. 2, pp. 197–224, 1988.

[6] B. Hassani and E. Hinton, *Homogenization and structural topology optimization: theory, practice and software*. Springer Science & Business Media, 2012.

[7] M. P. Bendsoe and O. Sigmund, *Topology optimization: theory, methods, and applications*. Springer Science & Business Media, 2003.

[8] P. W. Christensen and A. Klarbring, *An introduction to structural optimization*. Springer Science & Business Media, 2008, vol. 153.

[9] R. Kohn, "Optimal design and relaxation of variational problems, part i," *Comm. Pure Appl. Math.*, vol. 39, pp. 1–25, 1986.

[10] R. V. Kohn and G. Strang, "Optimal design and relaxation of variational problems. ii," MASSACHUSETTS INST OF TECH CAMBRIDGE, Tech. Rep., 1986.

[11] R. V. Kohn and G. Strang, "Optimal design and relaxation of variational problems. iii," MASSACHUSETTS INST OF TECH CAMBRIDGE, Tech. Rep., 1986.

[12] R. B. Haber, C. S. Jog, and M. P. Bendsøe, "A new approach to variable-topology shape design using a constraint on perimeter," *Structural optimization*, vol. 11, no. 1, pp. 1–12, 1996.

[13] C. Jog, "Topology design of structures using a dual algorithm and a constraint on the perimeter," *International Journal for Numerical Methods in Engineering*, vol. 54, no. 7, pp. 1007–1019, 2002.

[14] P. Duysinx, "Layout optimization: A mathematical programming approach," Tech. Rep., 1997.

[15] M. P. Bendsøe and O. Sigmund, *Optimization of structural topology, shape, and material*. Springer, 1995, vol. 414.

[16] G. Allaire, F. Jouve, and A.-M. Toader, "Structural optimization using sensitivity analysis and a level-set method," *Journal of computational physics*, vol. 194, no. 1, pp. 363–393, 2004.

[17] G. Allaire and R. V. Kohn, "Optimal design for minimum weight and compliance in plane stress using extremal microstructures," *European journal of mechanics. A. Solids*, vol. 12, no. 6, pp. 839–878, 1993.

[18] G. Allaire, Z. Belhachmi, and F. Jouve, "The homogenization method for topology and shape optimization. single and multiple loads case," *Revue européenne des éléments finis*, vol. 5, no. 5-6, pp. 649–672, 1996.

[19] M. P. Bendsøe, "Optimal shape design as a material distribution problem," *Structural optimization*, vol. 1, no. 4, pp. 193–202, 1989.

[20] H. Mlejnek, "Some aspects of the genesis of structures," *Structural optimization*, vol. 5, no. 1, pp. 64–69, 1992.

[21] M. Zhou and G. Rozvany, "The coc algorithm, part ii: Topological, geometrical and generalized shape optimization," *Computer methods in applied mechanics and engineering*, vol. 89, no. 1-3, pp. 309–336, 1991.

[22] O. Sigmund, "Morphology-based black and white filters for topology optimization," *Structural and Multidisciplinary Optimization*, vol. 33, no. 4-5, pp. 401–424, 2007.

[23] O. Sigmund and J. Petersson, "Numerical instabilities in topology optimization: A survey on procedures dealing with checkerboards, mesh-dependencies and local minima," *Structural optimization*, vol. 16, no. 1, pp. 68–75, 1998.

[24] T. E. Bruns and D. A. Tortorelli, "Topology optimization of non-linear elastic structures and compliant mechanisms," *Computer methods in applied mechanics and engineering*, vol. 190, no. 26-27, pp. 3443–3459, 2001.

[25] M. Smith, *ABAQUS/Standard User's Manual, Version 6.9*, English. United States: Dassault Systèmes Simulia Corp, 2009.

[26] *Consistent units*, https://www.dynasupport.com/howtos/general/consistent-units, Accessed: 2021-11-06.

[27] *Matlab documentation*, https://www.mathworks.com/help/matlab/, Accessed: 2021-11-11.

[28] Y. Xie and G. P. Steven, "Shape and layout optimization via an evolutionary procedure," in *Proceedings of the international conference on computational engineering science*, 1992.

[29] Y. M. Xie and G. P. Steven, "A simple evolutionary procedure for structural optimization," *Computers & structures*, vol. 49, no. 5, pp. 885–896, 1993.

[30] L. Xia, Q. Xia, X. Huang, and Y. M. Xie, "Bi-directional evolutionary structural optimization on advanced structures and materials: A comprehensive review," *Archives of Computational Methods in Engineering*, vol. 25, no. 2, pp. 437–478, 2018.

[31] Y. M. Xie and G. P. Steven, "Basic evolutionary structural optimization," in *Evolutionary structural optimization*, Springer, 1997, pp. 12–29.

[32] P. Tanskanen, "The evolutionary structural optimization method: Theoretical aspects," *Computer methods in applied mechanics and engineering*, vol. 191, no. 47-48, pp. 5485–5498, 2002.

[33] O. Querin, G. Steven, and Y. Xie, "Evolutionary structural optimisation using an additive algorithm," *Finite elements in Analysis and Design*, vol. 34, no. 3-4, pp. 291–308, 2000.

[34] O. M. Querin, G. P. Steven, and Y. M. Xie, "Evolutionary structural optimisation (eso) using a bidirectional algorithm," *Engineering computations*, 1998.

[35] O. Querin, V. Young, G. Steven, and Y. Xie, "Computational efficiency and validation of bi-directional evolutionary structural optimisation," *Computer methods in applied mechanics and engineering*, vol. 189, no. 2, pp. 559–573, 2000.

[36] X. Huang and Y. Xie, "Convergent and mesh-independent solutions for the bi-directional evolutionary structural optimization method," *Finite elements in analysis and design*, vol. 43, no. 14, pp. 1039–1049, 2007.

[37] X. Huang and Y. Xie, "Evolutionary topology optimization of continuum structures with an additional displacement constraint," *Structural and multidisciplinary optimization*, vol. 40, no. 1, pp. 409–416, 2010.

[38] X. Huang, Y. Xie, and G. Lu, "Topology optimization of energy-absorbing structures," *International Journal of Crashworthiness*, vol. 12, no. 6, pp. 663–675, 2007.

[39] X. Huang and Y. Xie, "Bidirectional evolutionary topology optimization for structures with geometrical and material nonlinearities," *AIAA journal*, vol. 45, no. 1, pp. 308–313, 2007.

[40] X. Huang and Y. Xie, "Topology optimization of nonlinear structures under displacement loading," *Engineering structures*, vol. 30, no. 7, pp. 2057–2068, 2008.

[41] X. Huang and Y. M. Xie, "Bi-directional evolutionary topology optimization of continuum structures with one or multiple materials," *Computational Mechanics*, vol. 43, no. 3, pp. 393–401, 2009.

[42] X. Huang and Y. Xie, "Optimal design of periodic structures using evolutionary topology optimization," *Structural and Multidisciplinary Optimization*, vol. 36, no. 6, pp. 597–606, 2008.

[43] X. Huang and Y.-M. Xie, "A further review of eso type methods for topology optimization," *Structural and Multidisciplinary Optimization*, vol. 41, no. 5, pp. 671–683, 2010.

[44] L. Xia, *Multiscale structural topology optimization*. Elsevier, 2016.

[45] X. Huang and M. Xie, *Evolutionary topology optimization of continuum structures: methods and applications*. John Wiley & Sons, 2010.

[46] O. M. Querin, *Evolutionary structural optimisation: stress based formulation and implementation*. University of Sydney Sydney, Australia, 1997.

[47] X. Yang, Y. Xie, G. Steven, and O. Querin, "Bidirectional evolutionary method for stiffness optimization," *AIAA journal*, vol. 37, no. 11, pp. 1483–1488, 1999.

[48] D. N. Chu, Y. Xie, A. Hira, and G. Steven, "Evolutionary structural optimization for problems with stiffness constraints," *Finite elements in analysis and design*, vol. 21, no. 4, pp. 239–251, 1996.

[49] V. Young, O. M. Querin, G. Steven, and Y. Xie, "3d and multiple load case bi-directional evolutionary structural optimization (beso)," *Structural optimization*, vol. 18, no. 2, pp. 183–192, 1999.

[50] Q. Li, G. Steven, and Y. Xie, "A simple checkerboard suppression algorithm for evolutionary structural optimization," *Structural and Multidisciplinary Optimization*, vol. 22, no. 3, pp. 230–239, 2001.

[51] X. Yang, Y. Xie, J. Liu, G. Parks, and P. Clarkson, "Perimeter control in the bidirectional evolutionary optimization method," *Structural and Multidisciplinary Optimization*, vol. 24, no. 6, pp. 430–440, 2002.

[52] H. Kim, O. Querin, G. Steven, and Y. Xie, "A method for varying the number of cavities in an optimized topology using evolutionary structural optimization," *Structural and Multidisciplinary Optimization*, vol. 19, no. 2, pp. 140–147, 2000.

[53] H. Kim, O. M. Querin, G. P. Steven, and Y. M. Xie, "Determination of an optimal topology with a predefined number of cavities," *AIAA journal*, vol. 40, no. 4, pp. 739–744, 2002.

[54] H. Kim, M. Garcia, O. Querin, G. Steven, and Y. Xie, "Introduction of fixed grid in evolutionary structural optimisation," *Engineering computations*, 2000.

[55] H. Kim, O. Querin, G. Steven, and Y. Xie, "Improving efficiency of evolutionary structural optimization by implementing fixed grid mesh," *Structural and Multidisciplinary Optimization*, vol. 24, no. 6, pp. 441–448, 2002.

[56] F. Fritzen, L. Xia, M. Leuschner, and P. Breitkopf, "Topology optimization of multiscale elastoviscoplastic structures," *International Journal for Numerical Methods in Engineering*, vol. 106, no. 6, pp. 430–453, 2016.

[57] A. Tovar, G. Niebur, M. Sen, J. Renaud, and B. Sanders, "Bone structure adaptation as a cellular automaton optimization process," in *45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference*, 2004, p. 1914.

[58] M. B. Schaffler and O. D. Kennedy, "Osteocyte signaling in bone," *Current osteoporosis reports*, vol. 10, no. 2, pp. 118–125, 2012.

[59] A. Tovar, N. M. Patel, G. L. Niebur, M. Sen, and J. E. Renaud, "Topology optimization using a hybrid cellular automaton method with local control rules," 2006.

[60] N. Weiner and A. Rosenblunth, "The mathematical formulation of the problem of conduction of impulses in a network of connected excitable elements specifically in cardiac muscle," 1946.

[61] A. W. Burks, *Essays on cellular automata*. Urbana: University of Illinois Press, 1970.

[62] B. Chopard and M. Droz, *Cellular Automata Modeling of Physical Systems*, ser. Collection Alea-Saclay: Monographs and Texts in Statistical Physics. Cambridge University Press, 1998. DOI: 10.1017/CBO9780511549755.

[63] S. Wolfram, *A new kind of science*. Wolfram media Champaign, IL, 2002, vol. 5.

[64] N. Inoue, N. Shimotai, and T. Uesugi, "Cellular automaton generating topological structures," in *Second European Conference on Smart Structures and Materials*, International Society for Optics and Photonics, vol. 2361, 1994, pp. 47–50.

[65] N. Inou, T. Uesugi, A. Iwasaki, and S. Ujihashi, "Self-organization of mechanical structure by cellular automata," in *Key Engineering Materials*, Trans Tech Publ, vol. 145, 1998, pp. 1115–1120.

[66] E. Kita and T. Toyoda, "Structural design using cellular automata," *Structural and Multidisciplinary Optimization*, vol. 19, no. 1, pp. 64–73, 2000.

[67] B. Tatting and Z. Gurdal, "Cellular automata for design of two-dimensional continuum structures," in *8th Symposium on Multidisciplinary Analysis and Optimization*, 2000, p. 4832.

[68] M. Abdalla, S. Kim, and Z. Gurdal, "Multigrid accelerated cellular automata for structural design optimization: A 1-d implementation," in *45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference*, 2004, p. 1644.

[69] M. Abdalla and Z. Gürdal, "Structural design using cellular automata for eigenvalue problems," *Structural and Multidisciplinary Optimization*, vol. 26, no. 3-4, pp. 200–208, 2004.

[70] P. Hajela and B. Kim, "On the use of energy minimization for ca based analysis in elasticity," *Structural and Multidisciplinary Optimization*, vol. 23, no. 1, pp. 24–33, 2001.

[71] R. Huiskes, H. Weinans, H. Grootenboer, M. Dalstra, B. Fudala, and T. Slooff, "Adaptive bone-remodeling theory applied to prosthetic-design analysis," *Journal of biomechanics*, vol. 20, no. 11-12, pp. 1135–1150, 1987.

[72] D. R. Carter, "Mechanical loading history and skeletal biology," *Journal of biomechanics*, vol. 20, no. 11-12, pp. 1095–1109, 1987.

[73] R. Martin, D. Burr, and N. Sharkey, "Skeletal tissue mechanics springer-verlag: New york," *NY, USA*, 1998.

[74] R. Hart, "Bone modeling and remodeling: Theories and computation," *Bone mechanics handbook*, vol. 1, pp. 31–1, 2001.

[75] L. L. Howell, "Compliant mechanisms," in *21st century kinematics*, Springer, 2013, pp. 189–216.