Abstract

Tissue engineering can utilize the use of Porous Collagen Scaffolds (PCS) to act as a guide for cells in wound healing applications. Improving healing time and reducing scar formation leading to improved quality of life in treated patients. Interactions between cells and PCS is still not fully understood. Recent research has focused on utilizing computational techniques which can reveal insight not currently achievable experimentally.

The Finite Element method was employed to construct PCS and simulate cell contractile forces. Before modelling cell-matrix interactions the PCS model was validated by unconfined compression simulations. Cell contractile forces were randomly generated throughout scaffolds to observe cell-matrix effects on a microscopic and a macroscopic (spatial) scale. Using defined parameters, the process was automated to allow for design optimization of PCS, which can be used as a guide when fabricating PCS for medical applications.

Unconfined compression simulations revealed scaffolds with a pore size of 110 μm to have an initial linear elastic modulus (E^*) of 179.07 Pa, being within the same order of magnitude as experimental compression tests. The compression simulations were also able to validate the use of 1D Pipe elements to model open-cell elastomeric foams applied to scaffolds constructed from 14-sided tetrakaidekahedrons. Cell-matrix interactions demonstrated an inverse relationship between the average contraction of scaffolds and the average stiffness that is sensed by cells.

Parametrization of variables in the model allowed a design optimization process to be implemented which was able to identify PCS candidates that reduced the contraction based on varying the strut thickness and the elastic modulus of struts (E_s). The work offers techniques that can be applied to a range of PCS such as Collagen-Glycosaminoglycan scaffolds, which is also argued in the prospects for future work in this research direction.

IN SILICO INVESTIGATION OF CELL-MATRIX MECHANOBIOLOGY INSIDE POROUS COLLAGEN SCAFFOLDS

Oliver Santos-Lopes

At Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

At the

University of Cyprus

Recommended for Acceptance

By the Department of Mechanical Engineering

June, 2023

APPROVAL PAGE

Master of Science Thesis

IN SILICO INVESTIGATION OF CELL-MATRIX MECHANOBIOLOGY INSIDE POROUS COLLAGEN SCAFFOLDS

Presented by

Oliver Santos-Lopes

Research Supervisor _____

Research Supervisor's Name

Committee Member _____

Committee Member's Name

Committee Member _____

Committee Member's Name

University of Cyprus

June, 2023

ACKNOWLEDGMENTS

I would like to express my thanks to my supervisor Prof. Vasileios Vavourakis for his insights in the model set up and validation and support throughout the project. I also thank my co-supervisor Prof. Dimitrios Tzeranis for his help on experimental characterization of Porous Collagen-based Scaffolds and feedback on report writing.

TABLE OF CONTENTS

Contents

LIST OF TABLES

LIST OF FIGURES

SYMBOLS

t	Strut Thickness
r	Cell Space Radius
P_s	Unit Cell Size
E^*	Initial Linear Elastic Modulus
E_s	Strut Elastic Modulus
E^*/E_s	Normalised Elastic Modulus
σ_{el}^*	Elastic Buckling Stress
ε_{el}^*	Elastic Buckling Strain
σ_{el}^*/E_s	Normalised Collapse Stress
$\Delta\sigma/\Delta\varepsilon$	Collapse Plateau Elastic Modulus
S_l	Strut Length
ρ^*/ρ_s	Relative Density
k_{ce}	Cell Effective Stiffness

Chapter 1

Introduction

Cell-matrix interactions have been shown to be vital in the wound healing process and a major factor in tissue engineering [8,11,44]. Porous Collagen Scaffolds (PCS) are built to replace damaged or missing tissue acting as a support structure for cell proliferation, differentiation etc. The mechanical stimuli being applied to cells (mechanosensing) is a factor effecting cell proliferation and differentiation [21,24,45]. Contractile forces of cells cause macroscopic contraction of PCS affecting wound healing and scar tissue formation [40,41]. Scar tissue has poor mechanical properties, limiting functionality and resulting in physical disfigurements. An improved understanding of how cells effect the surrounding extra-cellular matrix (ECM) and vice versa can lead to improved design of biomaterial-based scaffolds to reduce scar tissue formation.

Cell contractile forces are applied by attaching to ECM and pulling applying force within the nano newton range during cell migration [16]. Understanding the forces applied per cell and how they attach poses challenges due to the size and quantity of cells within a scaffold. Cell-matrix interactions and dependencies on a micro to nano scale could be used to develop treatments with improved results and faster healing times in a wide range of applications.

One of the primary challenges with investigating mechanics on a small scale is the lack of knowledge on cell properties and cell-matrix interactions. For this reason a technique needed to be developed, by which the mechanics experienced by individual cells could be recorded thus relating microscopic and macroscopic properties.

1.1 Porous Collagen-based Scaffolds (PCS)

The term Porous Collagen-based Scaffolds (PCS) refers to a class of biomaterials that have three major common characteristics: i) they are highly porous biomaterials, with a mean pore

diameter in the order of 100 μm, ii) their chemical backbone is microfibrillar collagen I, and iii) they are fabricated in dry state via lyophilization. PCS of various chemical composition have been reported in the literature, including collagen-glycosaminoglycan (CG) scaffolds and plain collagen scaffolds. Both CG scaffolds and plain collagen scaffolds are of medical interest as they have been utilized in FDA-approved regenerative medicine grafts.

Collagen is the primary constituent in the ECM of several tissue. CG scaffolds are highly porous structures interconnected with randomly orientated collagen struts (Fig. 1.1). They are produced via freeze drying which allows tuneable geometric features, a desirable property for tissue engineering applications.

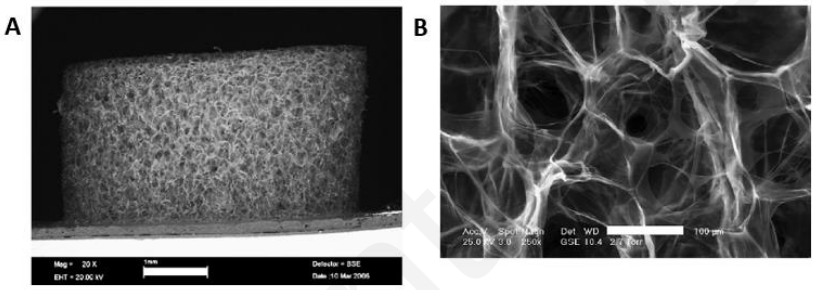


Figure 1.1: Scanning Electron Microscopy (SEM) images of collagen-glycosaminoglycan (CG) scaffolds. A) Low magnification. Scale bar: 1 mm. B) High magnification image. Scale bar: 100 μm [15].

1.2 Mechanical Properties of PCS – Modelling and Experimental Characterization

1.2.1 Macroscopic Characterization

Harley et al. 2007a reported that the macroscopic behaviour of isotropic equiaxed CG scaffolds acts as an open-cell elastomeric foam (Fig. 1.2). Having an initial linear elastic region (E^*), followed by a collapse plateau due to buckling ($\Delta\sigma/\Delta\varepsilon$), and finally a densification regime. The transition from E^* to $\Delta\sigma/\Delta\varepsilon$ can determine the elastic buckling stress (σ_{el}^*) and the elastic buckling strain (ε_{el}^*). The strength of scaffolds is dependent on whether they are dry or hydrated, with hydrated being significantly weaker [15].

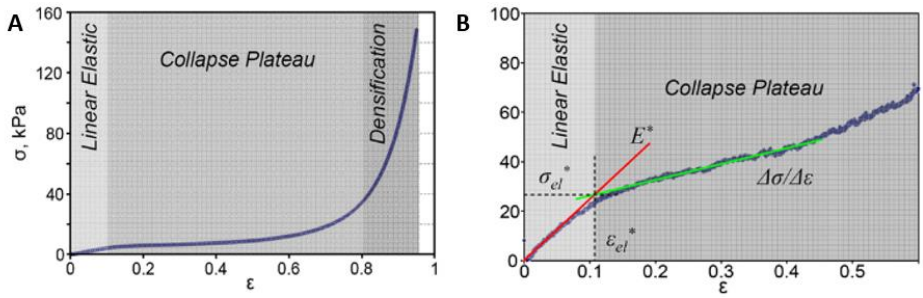


Figure 1.2: A) The stress-strain relationship of open-cell elastomeric foams consists of an initial linear elastic response, a collapse plateau, and a densification regime. B) Linear regression slopes of the linear elastic region (E*) and the collapse plateau ($\Delta\sigma/\Delta\varepsilon$), with intersecting elastic buckling stress (σ_{el}^*) and elastic buckling strain (ε_{el}^*) [15].

Unconfined compression tests on equiaxed CG scaffold samples of 20 mm diameter and 3.4 mm thickness (Tab. 1.1), revealed that the macroscopic young's modulus (Fig. 1.2) of hydrated CG scaffolds in the linear elastic region (E*) was 208 $\pm$41 Pa and 92 $\pm$14 Pa in the collapse plateau region ($\Delta\sigma/\Delta\varepsilon$) for pore sizes ranging from 96-151 μm [15].

Table 1.1: Macroscopic elastic modulus in initial linear elastic region (E*) & collapse plateau region ($\Delta\sigma/\Delta\varepsilon$), elastic buckling strength (σ_{el}^*), and elastic buckling strain (ε_{el}^*) from unconfined compression tests performed on hydrated CG scaffolds by *Harley et al., 2007a*.

Average Pore Size (μm)	**Initial Elastic Modulus (E*)**	**Collapse Plateau Modulus ($\Delta\sigma/\Delta\varepsilon$)**	**Elastic Buckling Stress (σ_{el}^*)**	**Elastic Buckling Strain (ε_{el}^*)**	**Relative Density (ρ^*/ρ_s)**
96 μm	206 $\pm$36 Pa	98 $\pm$15 Pa	18 $\pm$4 Pa	0.079 $\pm$0.017	0.0058 $\pm$0.0003
110 μm	176 $\pm$41 Pa	83 $\pm$11 Pa	14 $\pm$8 Pa	0.076 $\pm$0.031	0.0059 $\pm$0.0003
121 μm	221 $\pm$47 Pa	93 $\pm$11 Pa	31 $\pm$5 Pa	0.151 $\pm$0.055	0.0061 $\pm$0.0003
151 μm	229 $\pm$22 Pa	94 $\pm$18 Pa	22 $\pm$4 Pa	0.107 $\pm$0.022	0.0062 $\pm$0.0005

Mechanical characterization tests have been performed on various PCS. *Herrera et al., 2019* used highly aligned macroporous collagen scaffolds consisting of collagen walls linked by struts. Performing monoaxial compression tests they measured macroscopic elastic modulus to

range from 1 ±0.1 kPa to 29.7 ±2.3 kPa with increasing collagen content, 0.8 wt% to 3.0 wt%. The increase in stiffness when compared to the CG scaffold tested by *Harley et al., 2007a* could be attributed to the alignment of collagen walls in the direction of compression. The compression tests acted perpendicular to the orientation of the struts, so they had no significance mechanically and only their volume and thickness were measured.

1.2.2 Microscopic Characterization

The microscopic properties of a PCS relate to the elastic modulus of struts (E_s) which build the scaffolds. In CG scaffolds Atomic Force Microscopy (AFM) measured the bending stiffness of dry struts to calculate the dry strut modulus ($E_{s,dry}$). The hydrated state modulus ($E_{s,hydrated}$) was calculated using the ratio between macroscopic young's modulus between dry and hydrated scaffolds; $E_{s,hydrated}$ = 5.28 ±0.25 MPa [15].

Herrera et al., 2019 used AFM to measure the elastic modulus of walls (E_{wall}). Increasing collagen content from 1.1 wt% to 1.5 wt% also increased the elastic modulus of walls (E_{wall}) from 96.3 kPa to 554.5 kPa. This increase in stiffness was attributed in part to the increase of the thickness of the scaffold walls.

Experimental results, see *Herrera et al., 2019* and *Harley et al., 2007a*, suggest that the macroscopic elastic modulus (E^*) of PCS is significantly (several orders of magnitude) smaller in comparison to the microscopic elastic modulus of scaffold walls E_{wall} or scaffold struts E_s. Elastic modulus can therefore be defined as macroscopic when referring to a collection of pores, and microscopic when talking about the constituents of pores (e.g., struts, walls etc.).

1.2.3 Defined Relationships Between Microscopic and Macroscopic Properties

Gibson et al., 1997 defined a relationship between the normalised elastic modulus (E^*/E_s) of open-cell elastomeric foams to the relative density (ρ^*/ρ_s) of PCS, defined as the ration of the macroscopic density (ρ^*) to the strut density (ρ_s), measured as 1.3 g cm^{-3}.

$$\frac{E^*}{E_s} = C_1 \left(\frac{\rho^*}{\rho_s}\right)^n \tag{1.1}$$

where C_1 and n are constants (Tab. 1.2), E^*/E_s is the normalised elastic modulus of the linear region, and ρ^*/ρ_s is the relative of the pores in the PCS. ***Gibson et al., 1997*** also defined a relationship between the normalised collapse stress (σ^*_{el}/E_s) and ρ^*/ρ_s as,

$$\frac{\sigma^*_{el}}{E_s} = C_2\left(\frac{\rho^*}{\rho_s}\right)^m \tag{1.2}$$

where C_2 and m are constants (Tab. 1.2), and σ^*_{el} is the elastic buckling stress (Pa). The values of these constants have been defined and redefined through experimental results (Fig. 1.3) [15,22].

Table 1.2: Constants for the relationship between normalised elastic modulus (E^*/E_s) & normalised collapse strength (σ^*_{el}/E_s) with the relative density (ρ^*/ρ_s) of PCS defined from experimental compression tests on CG scaffolds (Eq. 1.1 & 1.2).

Reference	C_1	n	C_2	m
Gibson et al. 1997	1	2	0.2	2
Harley et al., 2007a	0.00416	0.89	0.000873	0.95
Kanungo et al., 2010	0.006	0.98	0.0009	0.94

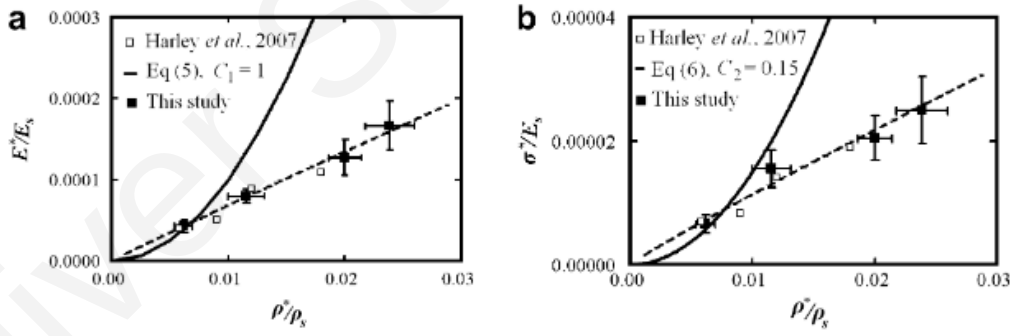


Figure 1.3: Normalized elastic modulus (E^*/E_s) and normalized collapse strength (σ^*_{el}/E_s) plotted against relative density (ρ^*/ρ_s), with experimental data as points, solid line demonstrates the theoretical relationship from cellular solids model, dashed line is from linear regression analysis of the shaded point [14,15,22].

1.3 Geometric Modelling of PCS

In silico investigations of PCS require an accurate characterization of PCS geometry. Various geometric models exist. For example, Rhombicuboctahedrons have been utilised to model porous scaffolds in bone [1,6,35]. Subsequently rhombicuboctahedrons were validated by experimental testing [1].

The design of the PCS can rely on fabrication techniques [13,18]. For example, if 3D printing techniques are used the PCS design should allow for it. *Früh et al., 2022* attempted to create polycaprolactone (PCL) scaffolds via 3D printing for use in bone tissue engineering. The scaffold consisted of square unit cells (pores) allowing for 3D bio-plotting.

PCS geometry can be replicated from *in vitro* studies using imaging techniques, such as scanning electron microscopy (SEM) [5,19]. The geometry must be able to capture the mechanical properties both on a micro- and macro-scopic level making design from experimental images desirable.

14-sided tetrakaidecahedrons, also sometimes referred as the kelvin foam model (Fig. 1.4), can capture the young's modulus, shear modulus, and Poisson's ratio for equi-axed isotropic, open-cell elastomeric foams [31,42]. PCS are considered to act as open-cell elastomeric foam (Fig. 1.2) [17]. Some research has disputed the use of the kelvin foam model due to a lack of randomness and anisotropic properties which is not how real foams are structured [31].

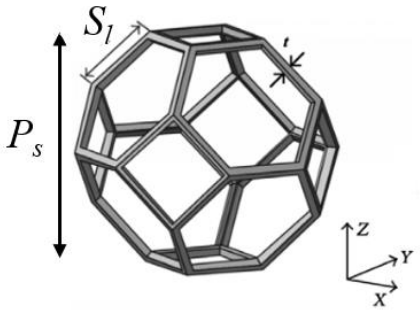


Figure 1.4: Geometry of the Kelvin cell. P_s is the pore diameter, l is the strut length, and t is the strut thickness [31].

Mohammadalipour et al., 2023 demonstrated that the use of tetrakaidekahedrons produced more accurate predictions compared to honeycomb and power-law models for electrospun polyhydroxybutyrate mats.

Strut thickness (Fig. 1.4) and pore diameter is related to the ρ^*/ρ_s of PCS constructed with tetrakaidekahedrons,

$$\frac{\rho^*}{\rho_s} = 8.19\left(\frac{t}{P_s}\right)^2 \quad (1.3)$$

where ρ^*/ρ_s is the relative density of pores, t is the thickness of collagen struts (μm), and P_s is the diameter of the pore (μm) [22]. For PCS of $\rho^*/\rho_s \approx 0.006$ [15,22]. ρ^*/ρ_s is also related to the porosity of a scaffold [30].

$$\%Porosity = \left(1 - \frac{\rho^*}{\rho_s}\right) \times 100 \quad (1.4)$$

The porosity of PCS is highly influential on cell migration and hence a key factor in tissue engineering [9,28]. PCS have extremely high porosity often measured to be 98% or higher depending on the application [26,36].

1.4 Finite Element Modelling of PCS

Experimental data (imaging, mechanical testing) allows one to create a finite element model of PCS. The introduction of improved computational techniques reduces processing times allowing for more detailed analysis with improved efficacy.

PCS act as elastomeric open-cell foams which are hyper-elastic materials modelled by techniques such as the neo-Hookean approach [10]. 3D finite elements are commonly used when modelling hyper-elastic materials. Techniques are required to minimise the computational costs due to model complexity. 2D and 1D elements are more efficient than 3D elements and have been utilized in PCS modelling. *Herrera et al., 2019* created finite element models of scaffolds using 2D shell elements to model the walls and 1D beam elements to model

the struts connecting the walls. The model efficiency improved but did not consider non-linear elastic materials which will produce different results at higher strains.

Früh et al., 2022, used beam elements to model scaffold deformation, stating that results were only valid up to strains of 3%, where the linear elastic region was valid. *Boccaccio et al., 2018*, modelled scaffolds consisting of rhombicuboctahedrons using 3D meshing to allow for non-linear modelling thus allowing for higher strain results. Current PCS FEA models are either very computationally expensive or are limited to linear behaviour meaning they cannot be used to model large strain accurately.

1.5 Cell-Scaffold Interactions

Cell-Scaffold interactions are considered as the contractile forces a cell generates on a PCS. These interactions have been measured in experiments and applied during *In silico* investigations [16,17].

1.5.1 Experimental Studies

Fibroblasts are cells of interest in relation to wound contraction and subsequent scar tissue formation [41]. Myofibroblasts (highly contractile differentiated fibroblasts) appear in a large number around wound sites and are related to wound contraction as well as ECM formation [12]. Control over cell contraction could give control over macroscopic wound contraction which is shown to be related to tissue regeneration [46].

Fibroblasts contract collagen scaffolds by creating at least two attachment sites and pulling towards the cell's center of mass, resulting in bending, and sometimes buckling of struts in the scaffold (Fig. 1.5). The microscopic contraction leads to reduced scaffold dimensions over time. *Freyman et al., 2002* showed free floating scaffolds reduced in diameter over a period of 14 days due to these contractile forces.

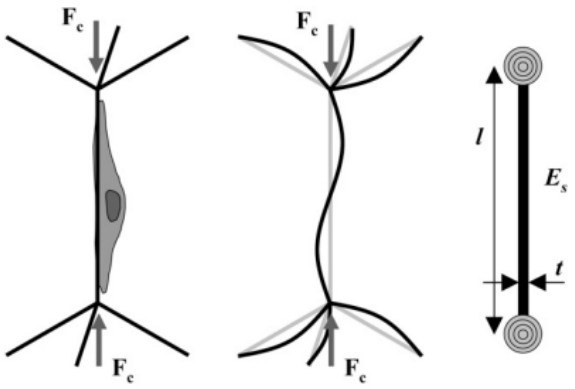


Figure 1.5: Schematic of a contracting fibroblast cell applying force (F_c) to a CG strut with ideal conditions (left) such that the strut buckles (middle). The mechanical and geometrical properties of the strut are characterized by its length (l), thickness (t), and young's modulus (E_s) [16].

The forces cells exert on the struts are difficult to measure due to the size of cells. Being on a scale of micrometers and producing forces in the nN range means that it was almost impossible to accurately assess the mechanical properties produced by a single fibroblast cell.

Freyman et al., 2002 used a custom Cell Force Monitor (CFM) device to record the total force acting on a scaffold and hence calculate the average force exerted by a single cell. The results estimated that the asymptotic force per cell was close to approximately 3 nN [12]. However, the experiment posed several assumptions meaning there was a margin of error. It has been shown there are times where fibroblast cells present in a scaffold are not active and hence do not exert any forces. PCS can contain thousands of cells per mm^3 [38]. Averaging contractile force becomes inaccurate as it was assumed that every cell seeded into the scaffold was active. The direction of forces was also neglected. Should a contractile force of one cell be opposing another the summation of the two does not fairly represent the actual force that is being implemented.

Attempting to remedy these assumptions *Harley et al., 2007b* developed a technique to investigate the individual cell forces within a three-dimensional CG scaffold. The method worked by taking the mechanical [15] and geometrical properties of an individual CG strut,

measured using live cell imaging, and calculating the force (F_c) required for the cell to buckle the strut. The force to buckle can be calculated using Euler's formula. It was calculated that for an average cell the contractile force was estimated to range from around 11-41 nN with an average of 26 $\pm$13 nN. When considering the variation of strut thickness, the range of forces changed to 11 $\pm$5 nN – 52 $\pm$27 nN [16]. The values are not indicative of the maximum forces applicable by the fibroblast cells rather just the forces seen within these scaffolds. To that end one case was seen in which a contracting fibroblast cell failed to buckle a strut. Using Euler's formula for buckling, it was found that the contractile force required to buckle that strut would have been 450 nN. This suggested fibroblast cells cannot reach forces of 450 nN or higher. Another study suggested that 200 nN is the maximum force a cell can generate [16,20]. Using cell morphology to determine whether a fibroblast cell was active in contraction, *Zahlak et al., 2000* also estimated contractile force of 21 nN.

1.5.2 In Silico

Interactions between cells and CG scaffolds can be studied with in silico investigations [17,33]. To model fibroblast contractile forces in a scaffold two diagonally opposing nodes could be selected with the forces pointing in equal and opposite directions (Fig. 1.5). The distance between the two attaching nodes can reach 200 μm [17]. *Herrera et al., 2019* deduced that with the current studies on contractile cell forces, 30 nN of force was appropriate for in silico models [15,20,25,29,37,39]. After applying the cell force further information about the mechanical environment of cells could be investigated.

1.6 Cell-effective Stiffness of PCS

The concept of "cell-effective stiffness" was introduced by *Herrera et al., 2019* to measure the stiffness that an individual cell senses in its local environment using highly aligned macroporous scaffolds constituted by walls and struts. This study identified the effect of cell-effective stiffness on cell differentiation *in* vitro, validating mechanosensing as important for cell behaviour. The cell-effective stiffness was calculated by modelling the traction forces of a

single cell on a collagen wall. Cell-effective stiffness was shown to increase with the stiffness of collagen walls, ranging from 4.6 ±0.8 kPa, for the scaffold with lowest collagen content (0.8%), to 169.9 ±10.8 kPa, for the scaffold with the highest collagen content (3.0%) [17].

The study *Herrera et al., 2019* was limited to a porous structure composed of walls of varying collagen dispersion interconnected by struts. However, the principles behind the model can be applied to a wide variety of geometries and can be used to find connections between cell-matrix mechanics. With the relationship correctly defined correlations can be drawn between macroscopic scaffold properties and cellular processes such as "migration, proliferation, early differentiation processes, long term stem cell commitment, and ECM formation and maturation" [17].

1.7 Scaffold Design optimization

To check the effects of specific geometrical parameters, such as strut thickness, in vitro and in vivo requires a long and laborious process of changing one feature within the scaffold and repeating the experiment. Considering the amount of scaffold variations possible, scaffold optimization becomes increasingly time consuming. Computational techniques can be used to model various scaffold conditions simultaneously with a large range of conditions. Thus cutting down on cost, material, and time spent compared to experimental parametric studies.

The principle behind using computational methods in tissue engineered scaffolds is to generate optimal conditions for desirable cell/matrix responses. Using variables to define specific features (e.g., young's modulus, strut length etc.) one can adjust parameters to assess optimal conditions for single or multiple objectives (e.g., macroscopic contraction). Some studies have now progressed to optimizing geometrical scaffold design features to give specific results, such as improved cell velocity and reduced wound contraction [33,34,40].

Optimization can also be used to calibrate model parameters to better resemble experimental results. In cell mechanobiology one difficulty lies in analysis the properties on such a small scale hence the calibration technique can be extremely beneficial. AFM was used to obtain the

mechanical response of cells [43]. Through calibration via optimization subcellular components could be assigned young's moduli that is otherwise unobtainable.

A major issue with optimization is computational costs, with many optimization procedures taking more than a day [1]. A lot of work has focused to simulating smaller volumes to avoid excessive computation times. Either reducing the volume or apply symmetry boundary conditions results in improved computation times with a sacrifice in result accuracy. One aspect to the following research is to use a variety of computational techniques to minimize processing time to produce efficient and accurate design optimization with respect to the cell-effective stiffness and the overall macroscopic affects created by cell-matrix interactions.

1.8 Thesis Objectives

Extensive work over the last years focused on bridging gaps in the existing knowledge of cell-matrix interactions. Several papers have focused on optimizing scaffold design based on various objectives such as increasing velocity of cell during migration [1,32,33,40,43]. To build from the current state of the art the following research hopes to identify and fill any assumptions deemed problematic in relation to cell-matrix interactions (Table 1.3).

Despite advancements, still several limitations are apparent. Models either focus on results on the microscopic or the macroscopic scale, except for the work of *Herrera et al., 2019*, and neglect the potential relevance the two set of results may be playing on one another. Despite the inclusion of microscopic results in the form of cell-effective stiffness, the results by *Herrera et al., 2019* consider linear elastic scaffolds and probe stiffness sensed by cells on a single collagen wall. With the effects of contraction between multiple walls or other constituents (i.e., struts) being excluded.

Table 1.3: Checklist of key features for correctly simulating cell-matrix interactions.

Reference	Microscopic Results	Macroscopic Results	Cell-matrix Interactions	Non-Linear mechanics	1D Finite Elements	Year
[1]	✗	✓	✗	✓	✗	2017
[17]	✓	✓	✓	✗	✓	2019
[32]	✗	✓	✗	✗	✗	2021
[40]	✗	✓	✓	✓	✗	2021
[13]	✗	✓	✗	✗	✓	2021
[33]	✓	✗	✗	✗	✗	2022
Presented Research	✓	✓	✓	✓	✓	2023

A relevant *in silico* model in the field of skin tissue was recently presented by *Sohutskay et al., 2021*. This model described wound contraction in rat skin injuries grafted with PCS, and considered cell-matrix interactions, non-linear collagen mechanics, and skin tissue, via a chemo-bio-mechanical finite element model [2-5].

The method in the subsequent section has a more concentrated look on cell-matrix mechanics explicitly outlining how cell contraction effects the surrounding collagen scaffold on the macroscopic level and how the mechanical and geometrical features of the scaffold can be used to influence cells macroscopic effects and what the cell will sense as a result.

Although brute force design optimization comes at significant computational costs, computational costs can be significantly reduced if optimization relies on appropriate *in silico* models that use 1D elements instead of 3D elements. Using 1D elements for non-linear analysis has not yet been validated but could significantly reduce computational costs and provide improved output capabilities. Using previous experimental results the method will provide an additional validation test for CG scaffold using 1D elements not yet previously achieved.

Chapter 2

Methodology

2.1 Materials & Computational Tools

This thesis utilized ANSYS Workbench 2021 R1 consisting of ANSYS SpaceClaim, a 3D CAD software, and ANSYS Mechanical, a Finite Element (FE) solver, both available in the component systems of workbench under the option static structural. ANSYS ran on the Cloudpaging player. Due to unforeseen errors with the CloudPaging player, results were finalized on ANSYS Workbench Student 2023 R1. All work reported was carried out on an MSI GL65 Leopard 10SDR laptop (Table 2.1).

Table 2.1: Hardware specifications of the MSI GL65 Leopard 10SDR computer used for simulations.

Hardware Specifications	
Processor Type	Intel(R)Core(TM)i7-10750H x64-based processor
Number of Cores	6
Frequency	2.60 GHz
RAM	16.0 GB
Operating System	64-bit operating system

2.2 Model Assumptions

In this thesis, a PCS was modeled as a series of interconnected struts organized as a 3D array of a unit cell. The geometry of the model was homogeneous, meaning pore size, strut length, and strut diameter was constant throughout the scaffolds. Strut cross section was assumed circular producing cylindrical beams for the FE solver. Forces applied by each cell were modeled using a pair of equal and opposite force vectors. Every cell was considered to exert the same amount of force. Some assumptions were validated using simulations discussed in section 2.5 The model was time independent. Hence, the static structural system (Fig. 2.1) was used on ANSYS workbench.

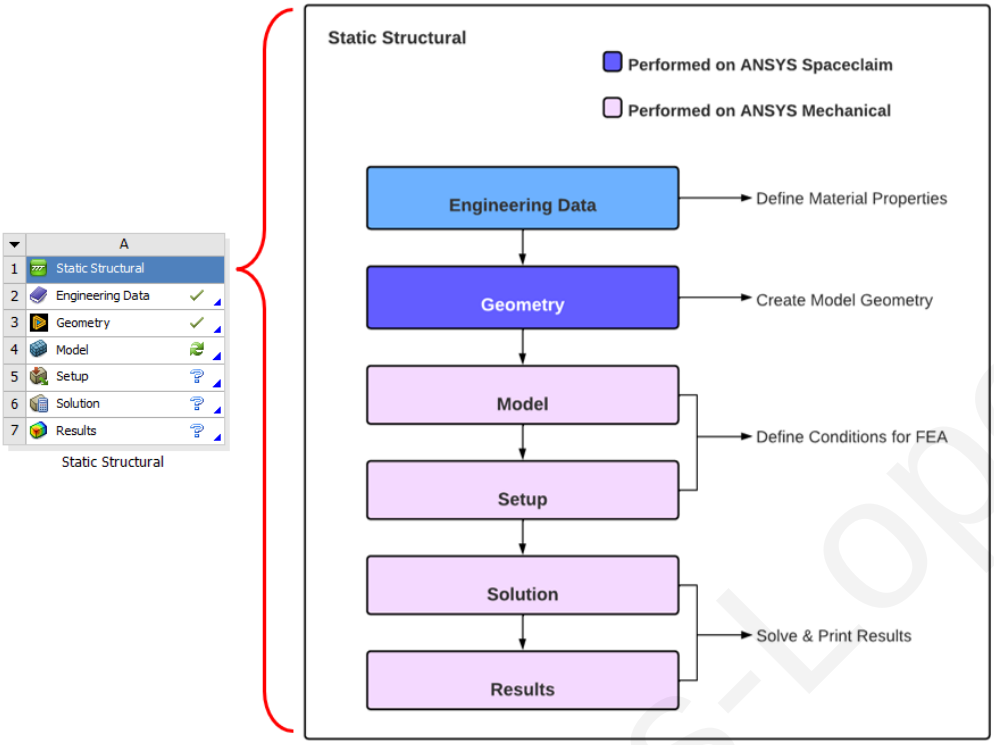


Figure 2.1: Flow chart of the *static structural* pipeline available on ANSYS workbench. Light blue shows the input engineering data (used to define material properties), dark blue is ANSYS SpaceClaim, and light purple is ANSYS Mechanical.

2.3 Material Modelling

2.3.1 Linear Mechanics

Linear models of PCS struts in this thesis assume a homogenous linear elastic material, described by generalized Hooke's equations. This set of constitutive equations utilize two parameters: the young's modulus E, and Poisson's ratio v. According to literature $E = E_s = 5.28\ MPa$ (E_s of hydrated CG struts [15]) and $v = 0.3$ [7]. In ANSYS, linear elastic analysis utilizes Beam188 or Pipe288 1D finite elements. Non-linear geometric changes were not active for linear models.

2.3.2 Non-Linear Mechanics

Non-linear mechanics of PCS struts were modeled using the Neo-Hookean model for hyper-elastic materials, described by the following constitutive law:

$$\psi = \frac{\mu}{2}(I_1 - 3) + \frac{1}{d}(J - 1) \tag{2.1}$$

where ψ is the strain energy in the material, I_1 is the first strain invariant, J is the volumetric change, μ is the initial shear modulus (Pa) defined as

$$\mu = \left(\frac{E}{2(1+v)}\right) \tag{2.2}$$

and d is the incompressibility constant (Pa^{-1}) defined as

$$d = \frac{2}{K} = 2\left(\frac{3(1-2v)}{E}\right) \tag{2.3}$$

where E is the Young's modulus (Pa; here $E \approx E_s$), v is Poison's ratio, and K is the initial bulk modulus (Pa). The estimated neo-Hookean model parameters for the PCS, based on parameters measured in *Harley et al., 2007a*, are shown in Table 2.2.

Table 2.2: Values of neo-Hookean model parameters.

Material Property	Value
Initial young's modulus (E)	5.28 MPa
Poisson's Ratio (v)	0.3
Initial shear modulus (μ)	2.03 MPa
Initial bulk modulus (K)	4.40 MPa
Incompressibility constant (d)	0.45 MPa^{-1}

ANSYS Beam188 elements cannot model hyper elasticity. Instead, non-linear analysis model struts using Pipe288 1D finite elements as they can model hyper-elastic materials. For non-linear modelling non-linear geometrical changes were active.

2.4 Scaffold Geometric Modelling

In this thesis PCS geometry was described using a 3D lattice that is constructed by order repetition of unit cells each representing a pore. Two different unit cells were used. Unit cells were defined in the ANSYS SpaceClaim software by drawing line bodies representing scaffold

struts. A single unit cell was then repeated in x, y, and z directions to construct the PCS lattice. Once the scaffold was constructed a cylindrical cross-sectional area of thickness *t* was applied to all struts.

While beam finite elements have a solid cross section, pipe finite elements have an internal diameter. The internal diameter chosen was 0.1 nm so that it has had no effect on results, which was verified during unconfined compression simulations (Section 2.5). Due to size limitations on ANSYS mechanical modelling the geometry was scaled so that 1mm was equivalent to 1 μm.

2.4.1 Cube Unit Cell

The simple cube unit cell (Fig. 2.2) is not an accurate model of a PCS pore geometry, yet it offers simplicity and low computational cost. The size of each unit cell (P_s) in the scaffold equaled the length of a strut (S_l). Strut thickness *t* was the second unit cell geometry parameter. An ANSYS SpaceClaim script was used to define scaffold structure and define parameters (Appendix A.1). SpaceClaim script uses python which has built in functions for SpaceClaim features. Pores vertices were created, according to their x, y, and z coordinates, and line bodies were drawn to connect these vertices.

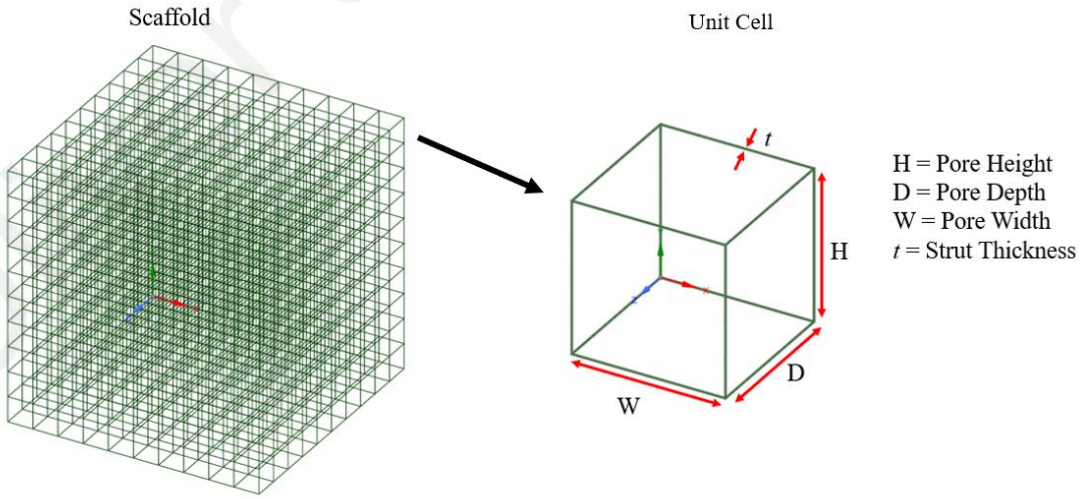


Figure 2.2: Scaffold (left) constructed from cube unit cells (right). *t* is the thickness of collagen struts in the pore, D the depth, H the height, and W the width of a single pore.

Each unit cell has 8 vertices (corners). The distance between neighboring vertices equals S_l. The linear pattern feature of SpaceClaim was utilized to copy and paste unit cells to generate a desired number of pores along each axis. Considering that the minimum producible size of cell-seeded scaffolds producible for in vitro experiments is 1 mm^3, PCS lattices were generated by positioning 7-10 pores along each axis, depending on P_s.

2.4.2 Tetrakaidekahedron Unit Cell

The second unit cell considered was a 14-sided tetrakaidecahedron (Fig. 2.3), a geometry used often to model open-cell elastomeric foams [30].

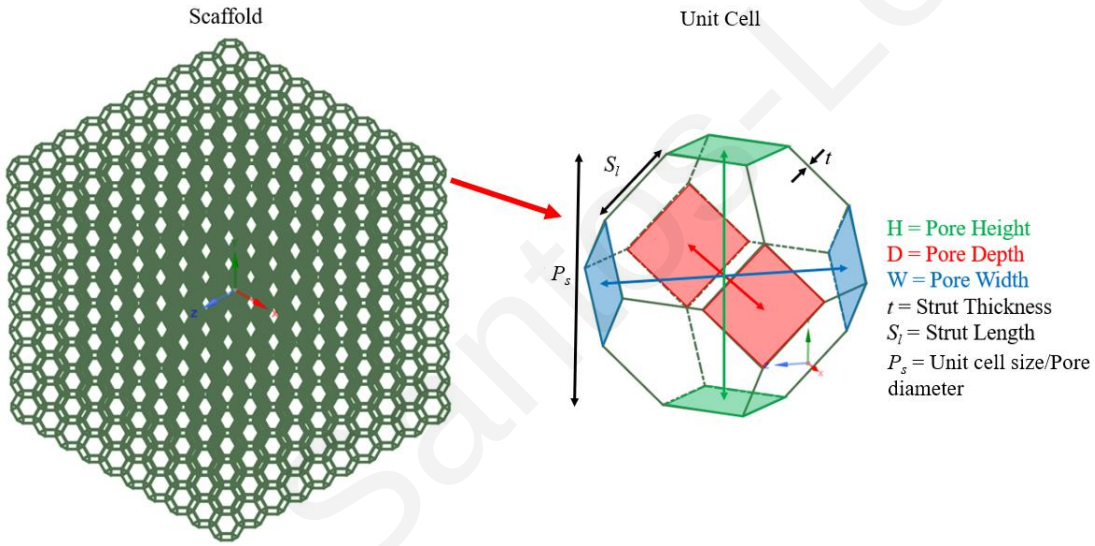


Figure 2.3: Scaffold (left) constructed from pores (right) modelled as tetrakaidekahedral unit cells. t is the thickness of collagen struts, S_l the length of struts, D (red) the depth, H the height (green), and W the width (blue) of a single pore.

Each tetrakaidekahedron has 24 vertices (Fig. 2.4 & Tab. 2.3). Using symmetry and defining several relationships the unit cells were drawn dependent given S_l or P_s (Appendix A.2 & A.3). P_s refers to the diameter of the unit cell whilst S_l refers to the length of struts between vertices.

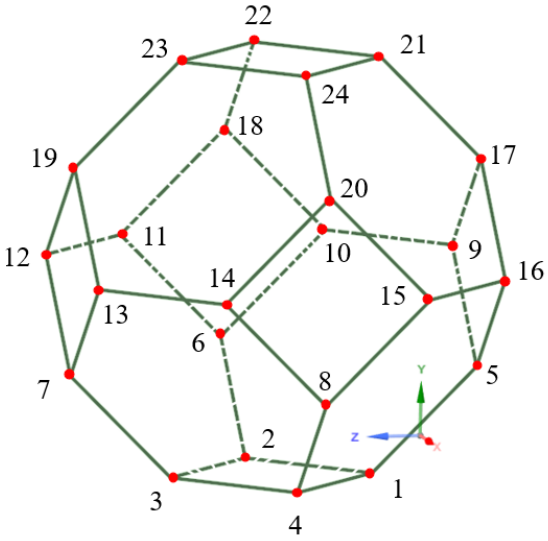


Figure 2.4: Unit cell of P_s=150 μm with the 24 vertices labelled. x,y,z coordinates are defined in Table 2.3.

Table 2.3: List of x,y,z coordinates produced when creating a unit cell of P_s=150 μm, see Figure 2.4 for physical representation. Coordinates derived using Equations A.12-15 (Appendix A.2.2).

Point	x	y	z
1	75	0	37.5
2	37.5	0	75
3	75	0	112.5
4	112.5	0	75
5	75	37.5	0
6	0	37.5	75
7	75	37.5	15
8	15	37.5	75
9	37.5	75	0
10	0	75	37.5
11	0	75	112.5
12	37.5	75	150
13	112.5	75	150
14	150	75	112.5
15	150	75	37.5
16	112.5	75	0
17	75	112.5	0
18	0	112.5	75
19	75	112.5	150
20	150	112.5	75
21	75	150	37.5
22	37.5	150	75
23	75	150	112.5
24	112.5	150	75

Once a unit cell was created it could be copied as a linear pattern in the x, y, and z directions (Fig. 2.5). The unit cells are connected by sharing vertices at a shared face (Fig. 2.5).

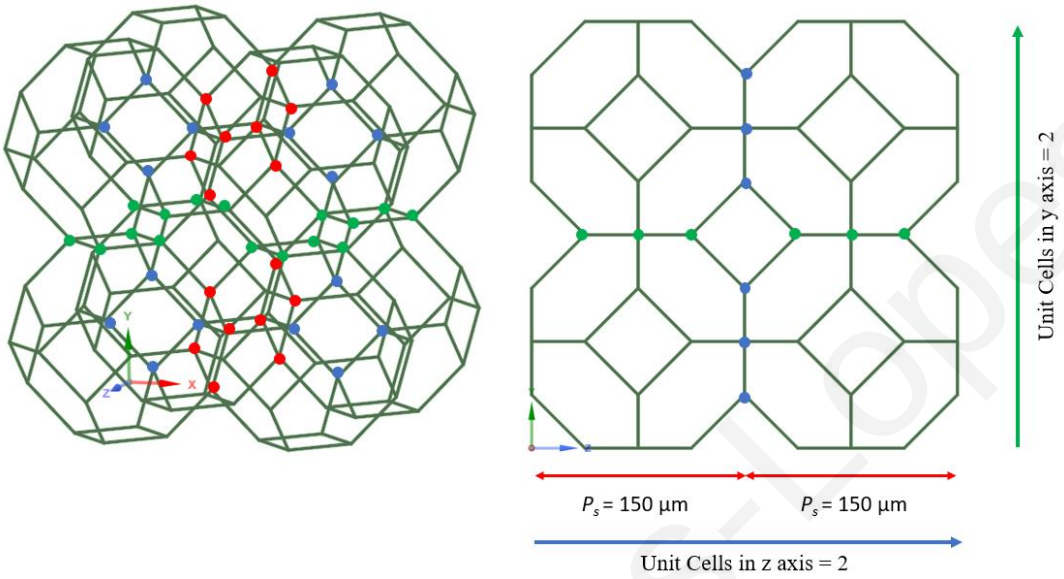


Figure 2.5: PCS model of tetrakaidekahedral unit cells of P_s = 150 µm. Demonstrating how unit cells were copied and pasted along each edge (y and Z). Points are shared vertices between adjacent unit cells.

2.5 Simulations of Scaffold Unconfined Compression

2.5.1 Numerical Analysis

This section describes numerical simulations of the unconfined compression of a PCS. Due to the availability of experimental data in the literature, these simulations were used to validate the FE model of PCS described in sections 2.1 & 2.4 and fine tune its parameters (unit cell, material parameters, finite element type). Based on the literature (section 1.2) simulations consider two mean pore diameters, 96 & 110 μm. Strut elastic modulus was set to E_s = 5.28 MPa. Strut thickness t, initial shear modulus µ and the incompressibility constant d were calculated based on known parameters (Tab. 1.1) using Eq. (1.3), Eq. (2.2) & (2.3).

Table 2.4: Scaffold properties used in simulations of PCS unconfined compression tests: unit cell size P_s, Strut modulus of elasticity E_s, Poisson's ratio v, initial shear modulus μ, initial bulk modulus K, incompressibility constant d, relative density ρ^*/ρ_s, strut thickness t.

P_s (μm)	E_s (MPa)	v	μ (MPa)	K (MPa)	d (MPa^{-1})	ρ^*/ρ_s	t (µm)
110	5.28	0.3	2.03	4.40	0.45	0.0059	2.95
96						0.0058	2.55

Initial simulations utilized lattices P_s and t were used for compression tests on the cubic unit cell scaffolds to allow analysis of the arrangement of struts. Published experiments used scaffolds of 20 mm diameter and 3.4 mm thickness. Due to processing power limitations simulations were limited to scaffolds of volume close to 1 mm^3, or equivalently 10 unit cells per dimensions (Tab. 2.5).

Table 2.5: Scaffold width, thickness, and volume for simulated unconfined compression tests.

Pore Size (µm)	Number of Units Cells Per Edge	Lattice Width (µm)	Lattice Volume (mm^3)
110 µm	10	1100	1.33
96 µm	10	960	0.88

To reduce computational costs only an eighth (1/8) of the scaffold was modelled. Symmetry boundary conditions were applied so that the reaction force of the whole scaffold response could be approximated (Fig. 2.6). The corner was 5-unit cells across along each axis for scaffolds of 96 and 110 μm pore size (Fig. 2.7).

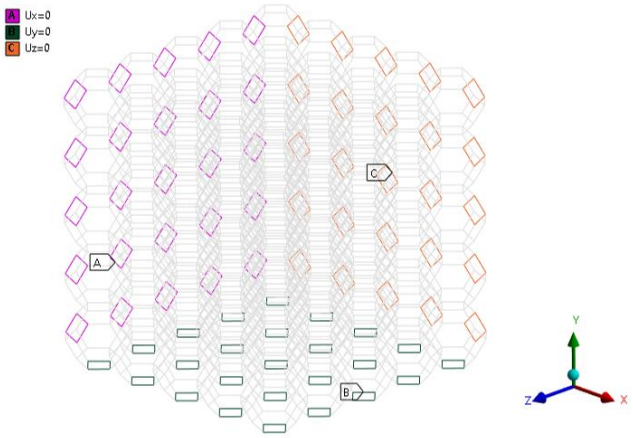


Figure 2.6: Boundary conditions for unconfined compression simulations. $U_{x,y,z}$ represents the displacement in the x, y and z direction. Purple represents nodes constrained in the x direction, black nodes constrained in the y direction and orange nodes constrained in the z direction.

Boundary conditions were applied to the scaffolds at three faces. Faces were constrained in the perpendicular direction, acting as roller constraints, mimicking the actions of the corner of the scaffold.

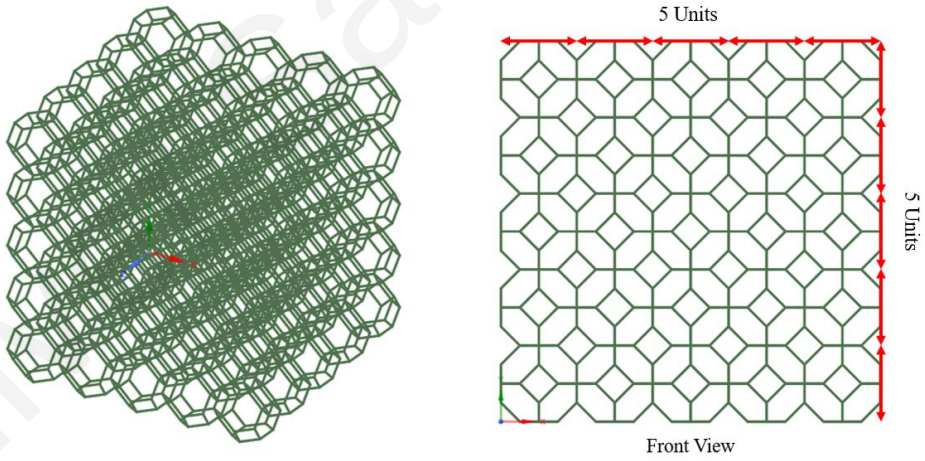


Figure 2.7: Corner of scaffold built from pores of 110 μm used in simulation tests. Number of pores along each edge from a front view is shown on the right.

To simulate the compression test a plate was used acting as a rigid body. The contact between the plate and the scaffold was set to rough so that no translational sliding resulted at the plate-

scaffold interface. The plate was set to move down and press into the scaffolds to a maximum strain of ε = 0.8 or until the FE solver could not converge on a solution. The same procedure was replicated on scaffolds consisting of cube unit cells with the same dimensions (Tab. 2.5) where P_s was thought of as equal S_l.

To produce a set of results the reaction force at the bottom of the scaffold was predicted against the displacement of the rigid plate. The force-displacement curve was then converted to a stress-strain curve by calculating the stress σ (Pa),

$$\sigma = \frac{F}{A} \tag{2.4}$$

where F is the reaction force from the numerical solution (N), and A is the area of the scaffold in contact with the plate (Fig. 2.8; m^2). Strain ε was given as

$$\varepsilon = \frac{\Delta L}{L} \tag{2.5}$$

where ΔL is the displacement of the plate (m), and L is the initial length of the scaffold. From the stress-strain curve E^* was calculated from the initial gradient. $\Delta\sigma/\Delta\varepsilon$, σ_{el}^* and ε_{el}^* were estimated using Microsoft excels linear regression on the collapse plateau region, from 0.2 to ~0.5 strain or maximum strain before the model failed to converge.

2.5.2 Analytical Analysis

In scaffolds consisting of tetrakaidekahedron unit cells, the macroscopic Young Modulus E^* and elastic buckling stress σ_{el}^* can be estimated analytically using equations reported in the literature (Eq. 1.1 & 1.2, Table 1.2).

For scaffolds consisting of cubic unit cells, validation can be performed by estimating the stiffness of each strut. Each vertical strut can be considered as a spring of stiffness,

$$k_s = E_s \frac{A}{S_l} \tag{2.6}$$

where k_s is the stiffness of each individual vertical strut (N/m), E_s is the strut elastic modulus (Pa), A is the strut cross-sectional area (m^2), and S_l is the strut length (m). Once the stiffness of individual struts is calculated the overall stiffness is found as,

$$k_T = N_{par}\left(\frac{k_s}{N_{con}}\right) \tag{2.7}$$

where k_T is the total stiffness (N/m), N_{con} is the number of struts connected in one line of struts (units along axes), and N_{par} is the number of lines of struts in the scaffold (Fig. 2.8). The reaction force (F) can be estimated using,

$$F = k_T \Delta L \tag{2.8}$$

where ΔL is the change in length of the scaffold (m). Using equation 2.4 & 2.5 we can get the stress-strain relationship to calculate macroscopic Young Modulus E^* (Pa),

$$E^* = \frac{\sigma}{\varepsilon} \tag{2.9}$$

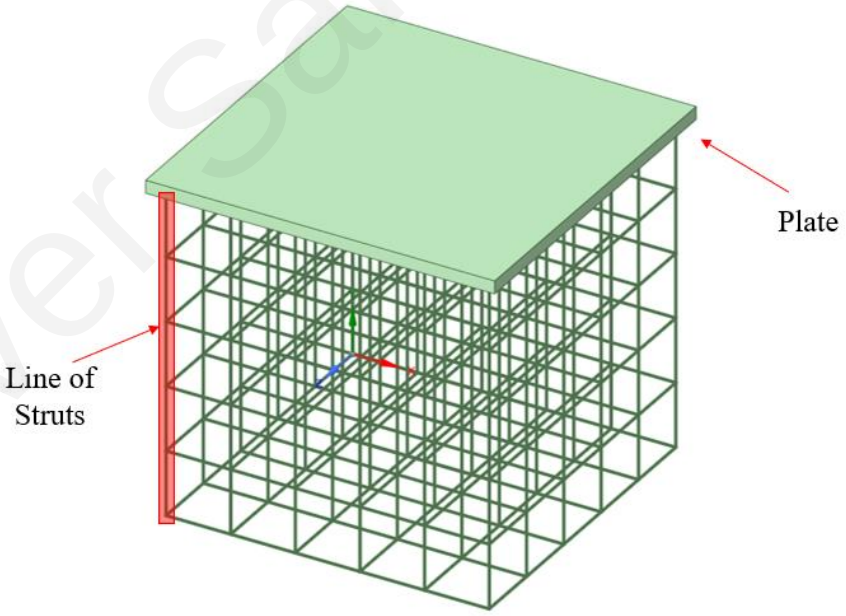


Figure 2.8: Schematic of a scaffold consisting of cube unit cell and the rigid plate used in compression simulations. A line of vertical struts is highlighted red.

2.6 Simulation of Cell-Scaffold Interactions

The scaffolds produced could be used to model the contractile forces (F_c) produced during processes such as cell migration. The contractile forces used in the model were equal to that of fibroblast cells often seen in wound healing.

2.6.1 Cell Attachment Sites

The contractile forces of cells were modelled by creating two attachment sites on struts of the scaffold. At these attachment sites equal and opposite force vectors were generated simulating how cells pull struts towards their centre of mass (Fig. 2.9). Cell attachment sites are chosen at mesh nodes produced by dividing PCS struts into smaller elements of equal length.

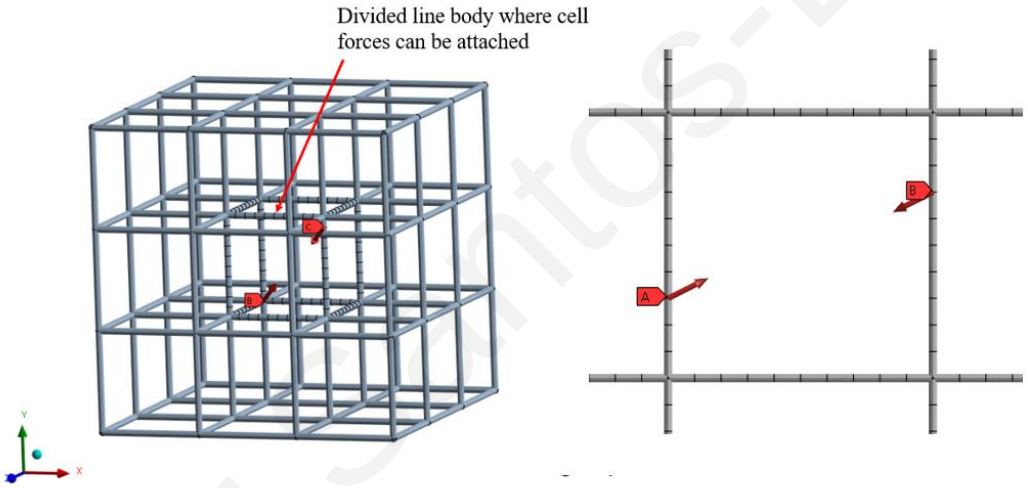


Figure 2.9: Depiction of a PCS lattice cubic unit cells demonstrating a pair of force vectors applied to two mesh nodes. Each strut is divided into 10 finite elements.

To improve the computational efficiency of simulations, the PCS lattice was split into two parts (Fig. 2.10). First, a fine mesh was applied within the part cells are located. As cells attach to mesh nodes, possible orientations of cells increased with a finer mesh. The fine mesh region was limited to 5 divisions or less at the present time due to node number restrictions on ANSYS student. Secondly, a coarse mesh was applied in the lattice part where no cells attached, as deformation is expected to be lower in this region.

A spherical coordinate system was created at the centre of the scaffold lattice, selecting all the nodes within a defined radius (*r*). The sphere known as the cell space defines possible locations for cell attachment sites (Fig. 2.10).

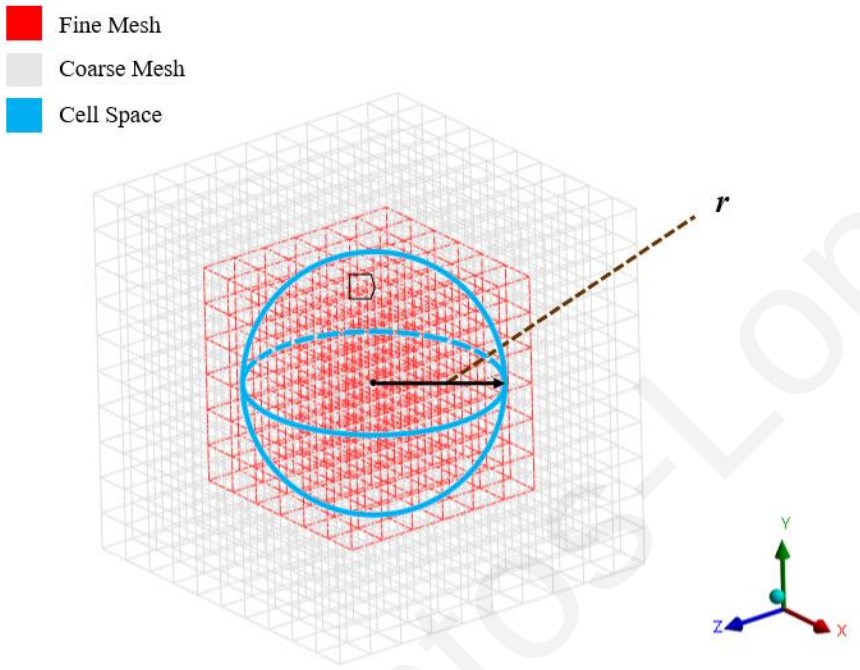


Figure 2.10: Diagram of the sphere in which cell attachment nodes can be located (light blue sphere), the finer meshed lines (red) that enable force vectors to be applied, and the coarser meshed lines (grey).

2.6.2 Cell Contractile Forces

Forces were applied at random in the cell space between a pair of attachment sites. Pair separation was set to be a maximum of 100 μm apart. The magnitude of the force F_c (N) was defined as,

$$F_c^2 = F_x^2 + F_y^2 + F_z^2 \tag{2.10}$$

where F_x, F_y, and F_z are force components along each axis. Using a defined force (F) and the x, y, and z coordinates between the two attachment sites F_x, F_y, and F_z was calculated to ensure force vectors were equal and opposite between pairs of forces.

Like ANSYS SpaceClaim, ANSYS Mechanical offers the ability to automatize procedures via the creation of Mechanical APDL script commands. Mechanical APDL is the FEA solver used by ANSYS mechanical. The use of APDL commands meant that the cell space radius (*r*) (Fig. 2.9), cell number (C_n), and contractile forces (F_c) can be defined by the user using a script (appendix A.4).

2.6.3 Boundary Conditions

Usually PCS *in vitro* are free-floating. However, it is of interest to study PCS mounted via specific means. Along this direction, here it is assumed two lateral sides of the scaffold are fixed (zero displacement) leaving four faces free to translate (Fig. 2.11).

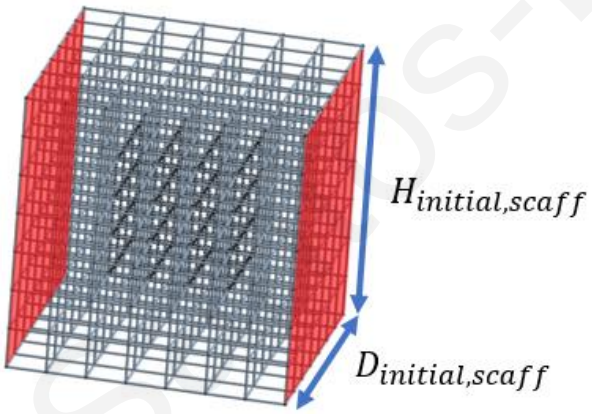


Figure 2.11: Boundary conditions applied to a PCS lattice during simulations of cell-matrix interactions, lateral sides (highlighted in red) are fixed in all directions.

2.6.4 Post-processing: Quantifying Cell-effective stiffness

The stiffness sensed by each cell k_{ce} (N/m) was estimated based on the displacement of nodes where it applied forces (Fig. 2.12).

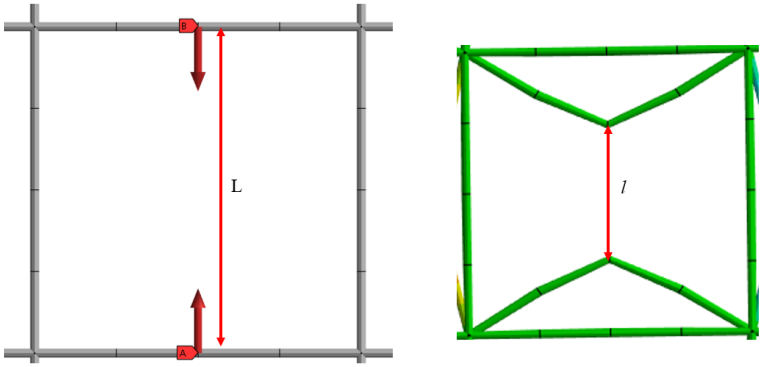


Figure 2.12: Example of scaffold strut deformation between a pair of contractile forces (F_c) in 1D to demonstrate change in distance (ΔL= L - l) between nodes after forces were applied.

Based on Hooke's law, cell effective stiffness k_{ce} is defined as,

$$k_{ce} = \frac{F_c}{\Delta L} = \frac{F_c}{L - l} \tag{2.11}$$

where F_c is the contractile force applied by the cell (N), L is the initial distance between cell attachment sites (m), and l is the distance between attachment sites after contraction (m).

2.6.5 Post-Processing: Macroscopic Scaffold Contraction

In cell-seeded scaffolds, macroscopic effects on the scaffolds can be quantified by calculating, the average node displacement of the 4 unfixed faces of the PCS lattice (Fig. 2.13).

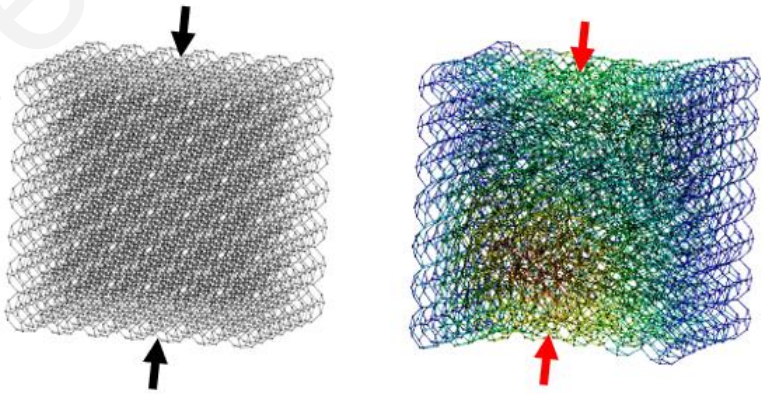


Figure 2.13: Macroscopic contraction of PCS scaffold, showing free faces being pulled towards the centre of the scaffold.

To normalize the macroscopic contraction of the scaffold a percentage reduction in area was calculated as,

$$\%AC = \frac{\Delta Area}{Initial\ Area} \times 100 \tag{2.12}$$

where the $\%AC$ is the amount of cross-sectional area change in the scaffold along the unconstrained axes, ΔArea is the change of cross-sectional area of the scaffold (m^2), $Initial\ Area$ is the original cross-sectional area of the scaffold (m^2), calculated as,

$$Initial\ Area = D_{initial,scaff} \times H_{initial,scaff} \tag{2.13}$$

where $D_{initial,scaff}$ is the initial scaffold depth (m) and $H_{initial,scaff}$ is the initial scaffold height (m) (Fig. 2.11). Microscopic and macroscopic results were stored and extracted using the scripting features in mechanical APDL (Appendix A.4).

2.7 Variable Effects

To assess the effects of variable in the scaffold each variable was change independently from the other. Some variables during PCS design can be controlled such as the strut thickness t, pore size (or unit cell size) P_s, and the strut elastic modulus E_s. Other variables also exist in the model which cannot be controlled *in vitro* or *in vivo*. These include number of active cells C_n, Contractile force F_c, and the cell distribution (controlled by the cell space radius (r)) and may need to be altered or optimized in future studies. The effects of individual variable were able to be controlled and as such their effect on microscopic and macroscopic results could be determined. To test the effects a standard set of variable values was created, in line with published values (Table 2.6). The number of unit cells along each axis is also tuneable but was not adjusted to save on processing times.

Table 2.6: Nominal simulation case for variable tests, values were derived from previous publications outline in section 1.

Variable	Value	Units
Strut Elastic Modulus (E_s)	5	MPa
Cell Contractile Force (F_c)	25	nN
Cell Number (C_N)	1000	-
Strut Thickness (*t*)	3	μm
Cell Space Radius (r)	400	μm
Unit cell size (P_s)	150	μm
Unit cells in each axis	7	-

With all other variables fixed a model could be produced to run in batch repeating 20 simulations per variable condition (Tab. 2.7). To interpret the result of the parameters box plots were made for the simulations of each value (Tab. 2.7) against the mean cell effective stiffness (k_{ce}) and a summary table with the mean k_{ce} and mean *%AC* was produced.

Table 2.7: Values used to test the effects of individual changes in variables on the macroscopic and microscopic results.

	Variable Test No.				
Variable	**1**	**2**	**3**	**4**	**5**
E_s	100 kPa	500 kPa	1 MPa	5 MPa	10 MPa
F_c	10 nN	20 nN	25 nN	40 nN	50 nN
C_N	500	1000	2000	3000	4000
t	1 μm	2 μm	3 μm	4 μm	5 μm
r	50	100	200	300	400
P_s	100	150	200	250	-

2.8 Design Optimization

Using a parametric study the model can be adapted to be used for design optimization of PCS by controlling tuneable variables such as t, P_s, and E_s. SpaceClaim scripting (geometric modelling) and APDL scripting (cell forces) allows the model to be parametrized (Fig. 2.14). To fully allow automatization the boundary conditions, and meshing which were done manually now were performed on a python script on ANSYS mechanical (Appendix A.5).

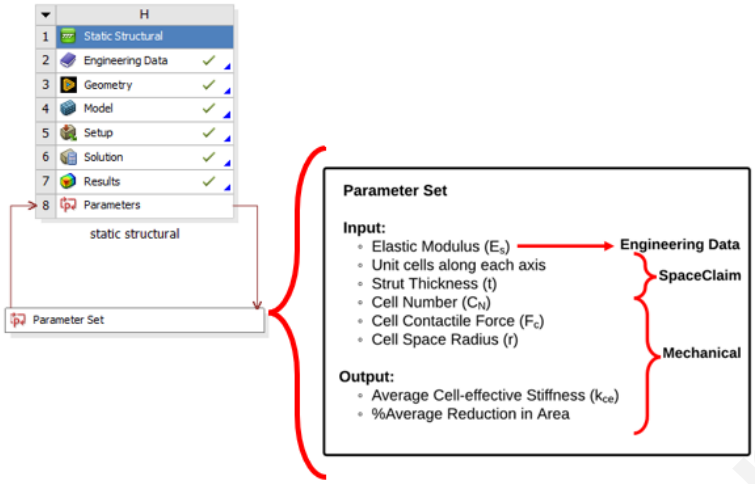


Figure 2.14: Diagram of the set of parameters applied across different phases of the static structural method (i.e., engineering data, SpaceClaim, & mechanical). Parameters include both input parameters from the users and outputs from the solution.

For the optimization process direct optimization was used available on ANSYS workbench (Fig. 2.15). Direct optimization uses an initial model and changes parameters within that model over range of different designs to come up with candidate parameter values that meet user defined conditions. The user can define criteria for the results, constraints of input parameters, and the number of designs modelled. The number of designs tested during this optimization was 100. Only the parameters which are controllable *in vitro/in vivo* such as the sturt modulus (E_s), and strut thickness (t) were varied.

The initial model used the variables of the nominal simulation case (Table 2.6). The range of possible E_s was that of 1 MPa to 10 MPa, and the thickness of struts was constrained so that porosity was at least 98% (Section 1.3; Equation 1.3 & 1.4). The optimizer was set to minimize the macroscopic contraction of the scaffold (Equation 2.12; *%AC*).

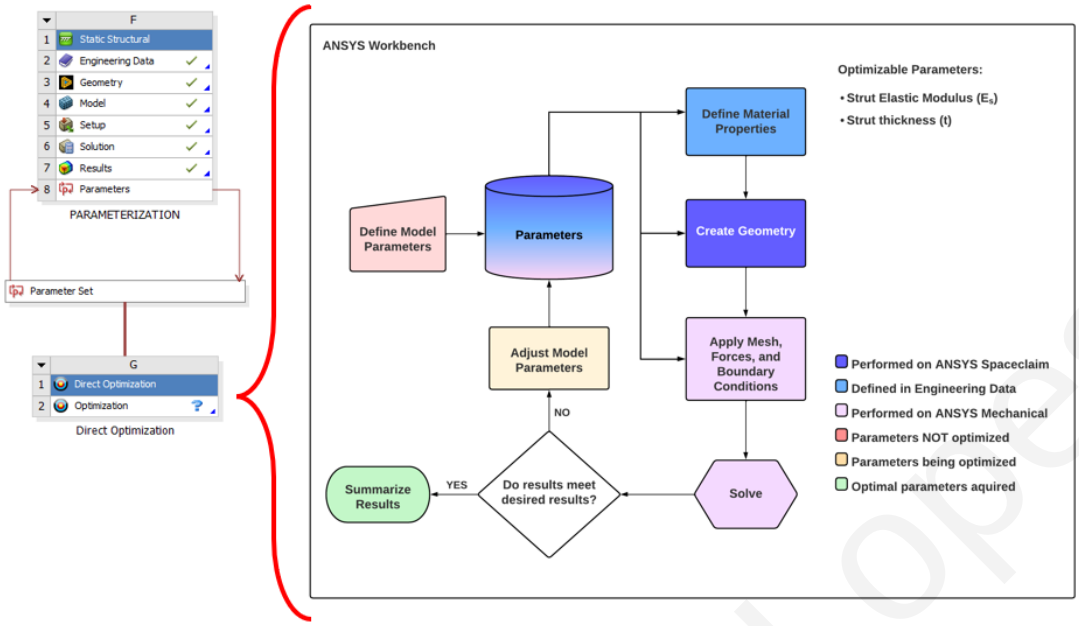


Figure 2.15: Flow chart describing the general procedure direct optimization. Parameters are defined prior to starting optimization including the range of possible values for optimizable parameters. The parameters being optimized are listed in the top right. Optimization stage is indicated as red for initial inputs, amber for optimizing in progress, and green for optimal parameters found. Software where parameters are assigned are defined as baby blue for engineering data, dark blue for ANSYS SpaceClaim, and light purple for ANSYS Mechanical. The optimized parameters are allocated in engineering data and SpaceClaim.

Chapter 3

Results

3.1 Simulation of Unconfined Compression

3.1.1 Lattices Derived from Tetrakaidekahedral Unit Cells

Compression simulations were performed on PCS lattices of P_s = 96 μm & P_s = 110 μm (Fig. 3.1). Stress-strain curves for PCS consisting of 10×10×10 tetrakaidekahedral unit cells (Section 2.5) under unconfined compression simulations were calculated from force-displacement graphs derived from ANSYS mechanical (Fig. 3.2). ANSYS solver provided results for linear elastic scaffolds up to maximum strain of ε=0.8. Unit cell models that consider non-linear material mechanics (Pipe288 elements with neo-Hookean constitutive law) only managed to converge on a solution up until ε=0.46 for unit cells of 110 μm and ε=0.275 for unit cells of 96 μm. Models fail due to mesh element distortion resulting from large deformations of strut.

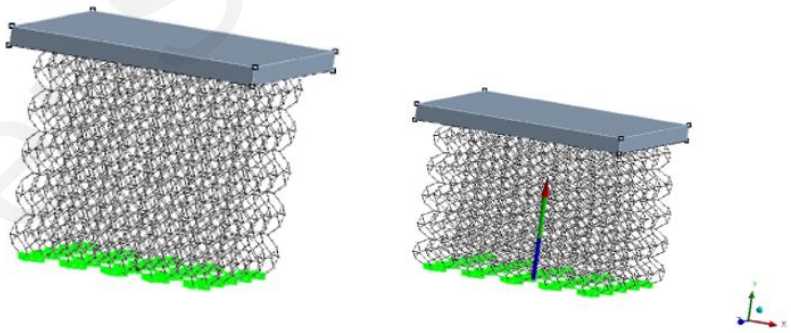


Figure 3.1: Unconfined compression simulation on PCS lattice with tetrakaidekahedron unit cells of P_s = 110 μm. Linear elastic material properties (E_s = 5.28 MPa, v = 0.3), with non-linear geometric changes inactive.

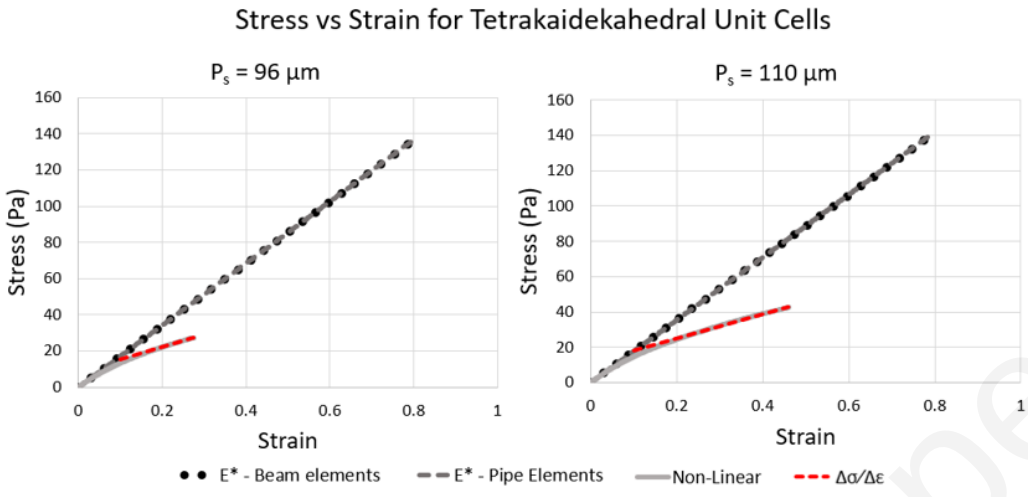


Figure 3.2: Stress-strain curves for unconfined compression simulations on PCS consisting of 10×10×10 tetrakaidekahedral unit cells of pore diameter Ps= 96 µm (left) and 110 µm (right). Black Dots: Results from models that assume linear mechanics and utilize 1D beam elements. Grey Dashes: Results from models that assume linear mechanics and utilize 1D pipe elements. Grey Line: Results from models that assume non-linear mechanics. Red Dashes: Linear regression curve to define the collapse plateau elastic modulus ($\Delta\sigma/\Delta\varepsilon$).

At low strains (up to ε=5%) the non-linear model and the two linear models provide nearly identical results. At strains larger than 10%, results provided by non-linear and linear models diverge due to strut buckling, a phenomenon that only calculated when non-linear geometric changes are active. The use of 1D pipe elements instead of 1D beam elements was validated based on the linear elastic modulus (E^*) of the stress-strain curves (Tab. 3.1). Results show that use of pipe finite elements instead of tube finite elements had minimal effect on results and did not increase computational costs.

Table 3.1: Apparent macroscopic elastic modulus E^* of PCS in unconfined compression derived using linear models that utilize 1D beam or 1D pipe finite elements.

Unit Cell Size (μm)	Beam Elements E* (Pa)	Pipe Elements E* (Pa)	Error (%)
96	170.53	170.89	0.21
110	177.05	177.43	0.21

Models that considered non-linear elastic mechanics converged for strains ε < 50%. Yet, this strain range was enough to simulate *in silico* the collapse plateau ($\Delta\sigma/\Delta\varepsilon$) region, the elastic buckling strain (ε_{el}^*), and the elastic buckling strength (σ_{el}^*) of PCS, shown in (Tab. 3.2). Linear regression was used to fit stress-strain data for ε > 20% to a straight line. As $R^2 \approx 1$, results show that the simulated stress-strain relationship in the collapse plateau region is well described by a linear function, in agreement with experimental data [15].

Table 3.2: Estimations of the macroscopic Young modulus (E^*), the elastic modulus of the collapse plateau ($\Delta\sigma/\Delta\varepsilon$), the elastic buckling strain (ε_{el}^*), and the elastic buckling strength (σ_{el}^*) derived by PCS finite element models that utilized tetrakaidekahedral unit cells and nonlinear mechanics (neo-Hookean model). R^2 refers to the linear regression of $\sigma(\varepsilon)$ for $\varepsilon > 20\%$.

Unit Cell Size (μm)	E^* (Pa)	$\Delta\sigma/\Delta\varepsilon$ (Pa)	σ_{el}^* (Pa)	ε_{el}^*	R^2
96	170.74	70.44	13	0.08	0.9997
110	179.07	72.73	17.5	0.1	0.9985

Simulation results presented in Table 3.2 were compared against estimations of E^*, σ_{el}^* calculated from published experimental data using Eq. (1.1) & (1.2) and (Table 1.2). Results show that estimates of E^* derived from numerical simulations (nonlinear FE model) agree reasonably with estimates based on experimental data. On the other hand, estimates of σ_{el}^* derived from numerical simulations (around 9%) are somewhat smaller compared to estimated

derived from experimental results (Table 3.3). The collapse plateau ($\Delta\boldsymbol{\sigma}/\Delta\boldsymbol{\varepsilon}$) was greater than experimental which may be related to the size of scaffolds in the simulated compression tests [15].

Table 3.3: Estimations of E^*, σ^*_{el} obtained from published data using Eq. (1.1), (1.2) and Table 1.2.

Pore Size (μm)	*Gibson et al. 1997*		*Harley et al., 2007a*		*Kanungo et al., 2010*	
	E^* (Pa)	**σ^*_{el} (Pa)**	**E^* (Pa)**	**σ^*_{el} (Pa)**	**E^* (Pa)**	**σ^*_{el} (Pa)**
96	177.62	35.52	224.4	34.59	203.7	37.54
110	183.80	36.76	227.9	35.15	207.12	38.14

3.1.2 Cubic Unit Cells

Compression simulations were repeated on PCS lattice of cubic unit cells with the same dimensions (Fig. 3.3).

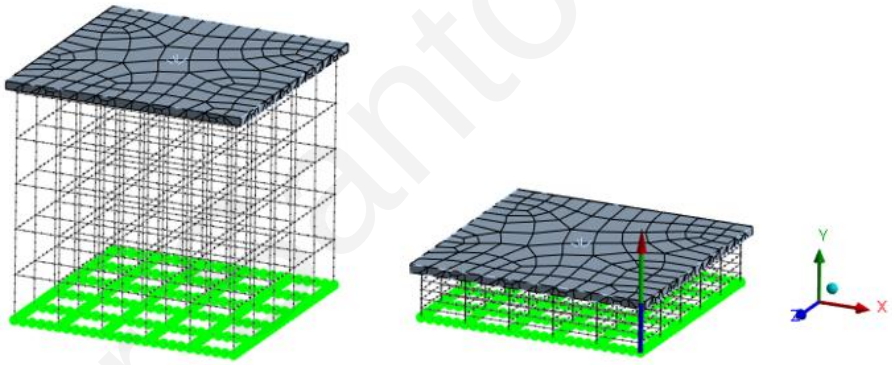


Figure 3.3: Unconfined compression simulation on PCS lattice with cubic unit cells of P_s = 96 μm. Linear elastic material properties (E_s = 5.28 MPa, v = 0.3), with non-linear geometric changes inactive.

Simulation results on PCS unconfined provided from lattices derived from cubic unit cells provide much higher estimates of the scaffold bulk Young Modulus E^* (Fig. 3.4). Numerical estimates of the bulk scaffold modulus E^* based on the slope of the $\sigma(\varepsilon)$ curve derived from simulation results agree very well with analytic predictions derived using Eq. 2.6-2.9. For the linear elastic material 1D Pipe elements results were close to that of analytical values and 1D

beam results (Table 3.4). The increase elastic modulus was due to the alignment of struts in the direction of compression.

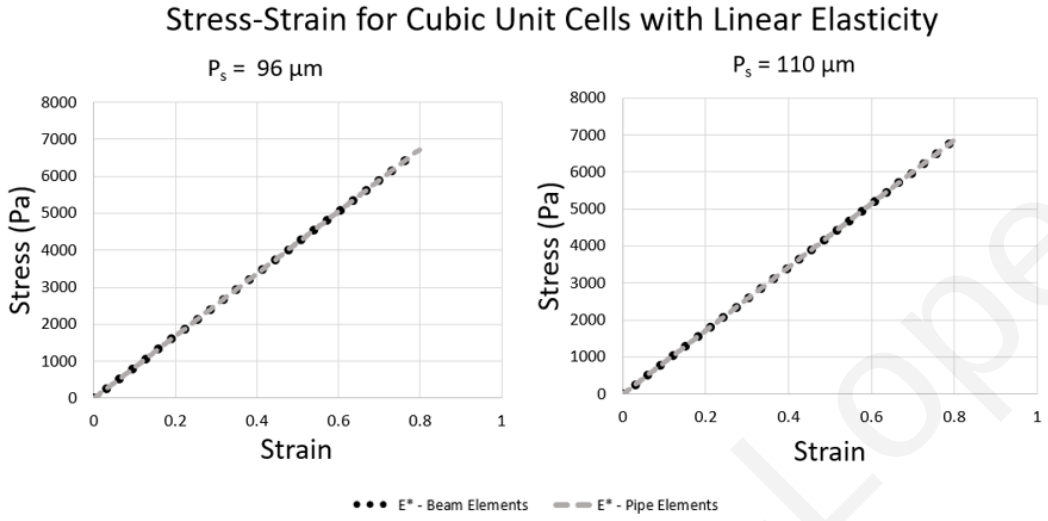


Figure 3.4: The stress-strain of macroscopic elastic modulus considering only linear elasticity with 1D beam (black dots) and pipe (grey dashes) elements.

Table 3.4: Numerical and Analytic elastic modulus for linear elastic unconfined compression simulations.

Unit Cell Size	**Beam Elements** ***E* (Pa)***	**Pipe Elements** ***E* (Pa)***	**Analytical E* (Pa)**
96	8412.534	8426.296	8424.97
110	8572.38	8579.83	8591.60

Simulations of unconfined compression that considered non-linear mechanics required a more refined mesh which led to increased processing time (Fig. 3.5). Despite the refined mesh the maximum strain before the FEA solver could no longer converge on a solution was just above 20% (Fig. 3.6). The solver could not converge due to significant element distortion seen during the buckling of struts (Fig. 3.5). Unlike lattices that utilized tetrakaidekahedral unit cells, in lattices that utilized cubic unit cells the transition from the initial elastic region (E^*) to the plateau $\Delta\sigma/\Delta\varepsilon$ appeared at a low strain $\varepsilon = 0.0016$ and had a sharper transition (Fig. 3.6).

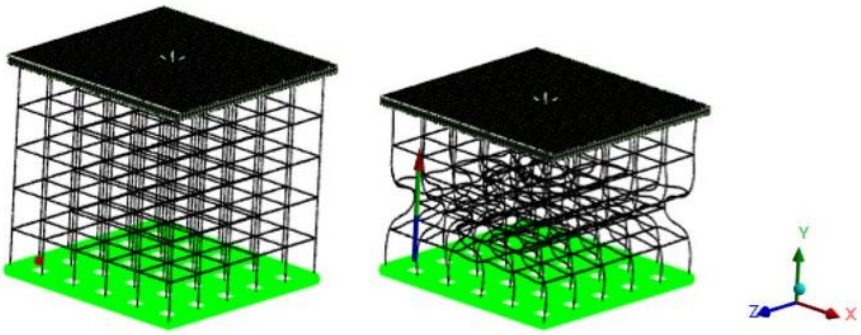


Figure 3.5: Unconfined compression simulation on PCS lattice with cubic unit cells of P_s = 96 μm. Linear elastic material properties (μ = 2.03 MPa, d = 0.45 MPa^{-1}), with non-linear geometric changes inactive. Line bodies were divided into 50 1D pipe elements.

The non-linear elastic model had a lower initial macroscopic elastic modulus E^* = 7700 Pa (Table 3.5) compared to the linear elastic model with 1D pipe elements E^* = 8426 Pa (Table 3.4). The reduced E^* was due to large deflections in the collagen struts from initial deformation. Buckling is not considered in linear elastic models and so a higher E^* was predicted. A large decrease in the elastic modulus was seen after the initiation of buckling in the non-linear elastic model.

For the cubic unit cell model the elastic buckling strength and strain were at the point connect E^* and $\Delta\sigma/\Delta\varepsilon$ (Fig. 3.6). ε_{el}^* and σ_{el}^* was than the scaffolds with tetrakaidekahedral unit cells despite having higher E^* (Tab. 3.5). The results from these compression simulations demonstrate that the cubic unit cell cannot simulate the response of open-cell elastomeric foams. The initial elastic modulus (E^*) and the collapse plateau elastic modulus ($\Delta\sigma/\Delta\varepsilon$) are too high, and the elastic buckling stress (σ_{el}^*) and strain (ε_{el}^*) are too low and results in a sudden collapse in the scaffold not seen in open-cell elastomeric foams [15].

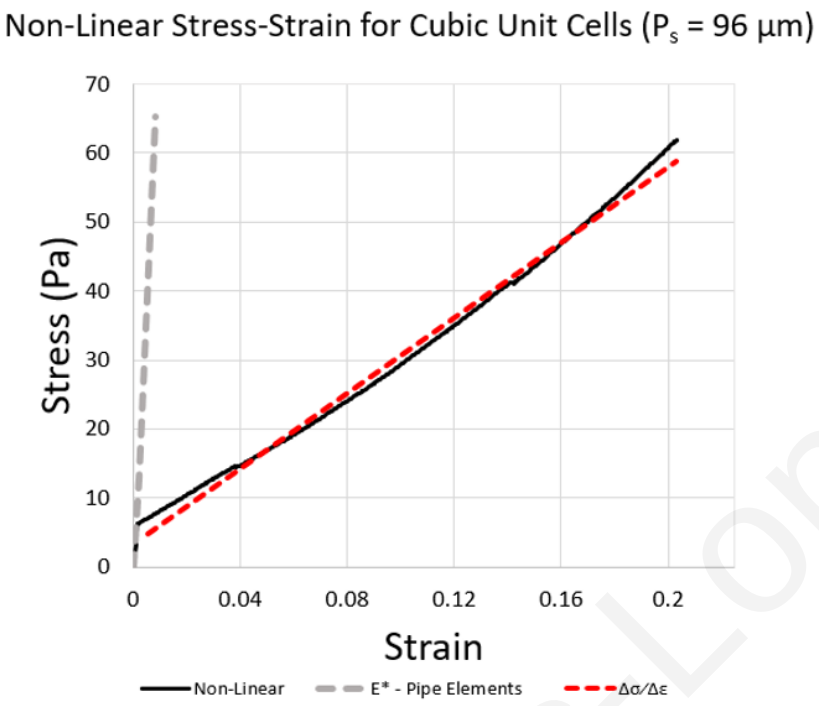


Figure 3.6: The stress-strain curve for scaffold of cubic unit cells of 96 µm pore size. Non-linear relationship represented by black curve, linear relationship by grey, and linear regression of collapse plateau ($\Delta\sigma/\Delta\varepsilon$) by red dashed line.

Table 3.5: Summary of mechanical response of PCS lattice of P_s = 96 µm, with cubic unit cells and non-linear material properties. E^* = initial macroscopic elastic modulus, $\Delta\sigma/\Delta\varepsilon$ = elastic modulus of the collapse plateau, ε_{el}^* = elastic buckling strain, σ_{el}^* = elastic buckling strength.

Unit Cell Size (µm)	E^* (Pa)	$\Delta\sigma/\Delta\varepsilon$ (Pa)	σ_{el}^* (Pa)	ε_{el}^*
96	7700.46	275.95	6.26	0.0016

3.2 Simulation of Cell-Matrix Interactions

Despite the ability of pipe elements to model non-linear material properties the limit of strain meant that cell-matrix interactions were not achievable unless using low cell forces and a low cell number. This was due to the local strain exerted on struts resulting in element distortion. Hence, cell-matrix interactions were modelled using a finite element model that utilizes linear

elastic material (Fig. 3.7). Modelling PCS using lattices and 1D elements enabled running over 100 simulations within an hour in a standard laptop.

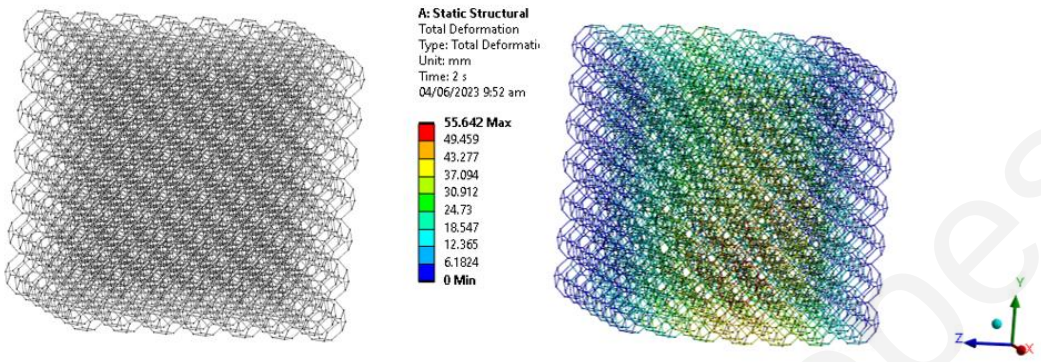


Figure 3.7: Simulation of a cell-matrix interactions with the nominal case. Legend shows the magnitude of deformation of struts within the PCS lattice.

3.2.1 Parameter Effects in Cell-Matrix Interactions

The effects of certain variables on the mean cell effective stiffness and macroscopic contraction (*%AC*) were estimated across various values (Tab. 3.6 & Fig. 3.8).

An increase in the strut Young modulus (E_s) and strut thickness (*t*) led to an increase in the cell effective stiffness and a decrease in the average contraction of the scaffold (Tab. 3.6). An increase in the cell force (F_c), and cell space radius (*r*), decreased the mean cell effective stiffness and increased the average contraction of the scaffold (Tab. 3.6). The mean cell effective stiffness was independent of the active cell number (C_n) . The unit cell size (P_s) appeared to have a decrease in mean k_{ce} with an increase from 100 μm to 200 μm and 250 μm but the nominal case (P_s = 150 μm) did not fit this trend. The results show that there is an inverse relationship between the cell effective stiffness and the contraction of a PCS.

During the cell-matric models the strut thickness and unit cell size were varied independently. The relative density (ρ^*/ρ_s) of the PCS lattice as a result also varied (Eq. 1.3). The macroscopic contraction (*%AC*) was consistently small across the various parameters, with a couple of outliers (t = 1μm & E_s = 100 kPa). The macroscopic contraction is taken for a single load step

which simulates the early stages of cell-matrix interactions. The low macroscopic contraction is hence in agreement with experimental studies of scaffold contraction during early stages of cell-matrix interactions [12].

Table 3.6: Summary of average (mean) microscopic and macroscopic results from varying individual variables within the cell-matrix PCS models. * Denotes models that failed to provide realistic deformations.⋆ Denotes results with poor results range.

Variable	Value	Mean Cell effective Stiffness (N/m)	Mean %AC
Nominal Case	-	2.69	0.027
t	*1 μm	0.01	46.29
	2 μm	0.38	0.230
	4 μm	9.02	0.013
	5 μm	28.67	0.008
E_s	*100 kPa	0.02	15.66
	500 kPa	0.12	0.800
	1 MPa	0.35	0.250
	10 MPa	4.61	0.012
F_c	10 nN	5.48	0.010
	40 nN	2.93	0.051
	50 nN	2.89	0.069
	100 nN	1.53	0.194
C_n	500	6.99	0.016
	2000	2.43	0.052
	3000	2.99	0.072
	4000	2.22	0.079
r	⋆100 μm	25	0.020
	150 μm	10.43	0.027
	200 μm	2.42	0.029
	300 μm	4.76	0.027
P_s	100 μm	185.74	0.040
	200 μm	35.98	0.032
	250 μm	18.07	0.024

A drastic increase in PCS contraction was observed when the strut thickness was reduced to 1 μm or the strut Young modulus was reduced to E_s 100 kPa. The model on closer inspection had collapsed on itself meaning the values from these models should not be considered as reliable. These results were omitted from further analysis and were only included as a reference to the model limits. The simulations on a cell space radius (r) of 100 μm failed to provide a sample

range and hence were not included in the box plots. The range of values differed in orders of magnitude hence to clearly see the range of results a logarithmic scale was required.

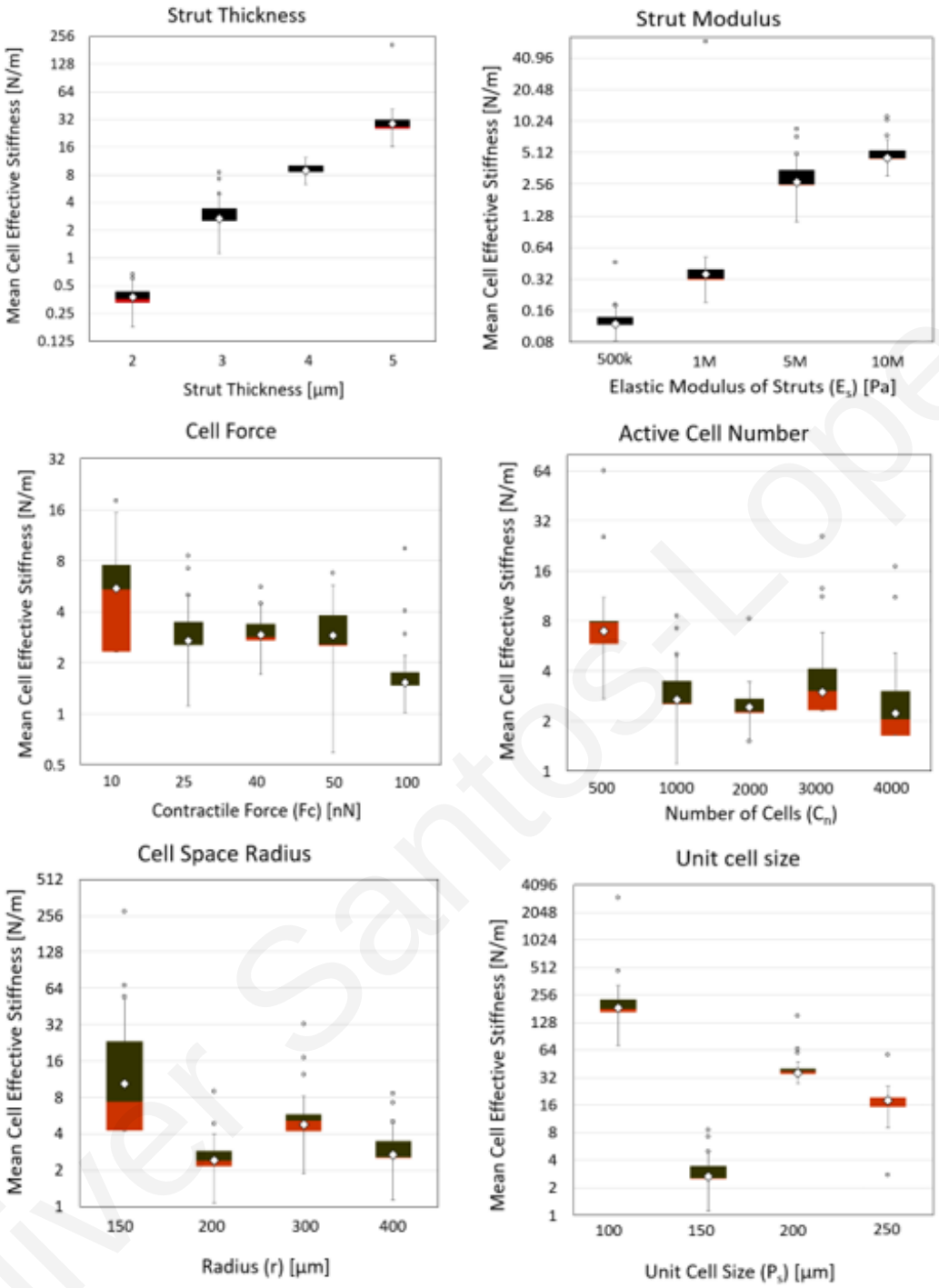


Figure 3.8: Box plots of the simulations carried out on variables; results all plotted against the cell-effective stiffness using a logarithmic scale. Diamonds represent the mean values across 20 simulations provided for each variable value. Points represent outliers in the data.

3.3. Design Optimization

During the production of PCS certain parameters can be controlled such as the strut modulus, strut thickness, and the pore size. Using these parameters in the cell-matrix model meant that optimal scaffold geometries could be produced to reduce phenomena such as wound contraction. Which can improve tissue regeneration. Due to limitations in the parameterization of unit cell size (P_s) the parameter was not included in the optimizer.

The design optimization adjusted the strut modulus E_s and the strut thickness t to minimize the macroscopic contraction of the PCS lattice (*%AC*) whilst maintaining a high porosity (Eq. 1.4; porosity > 98 %). An initial model was created with the parameters of the nominal case after which the optimizer altered only E_s and t. The initial scaffold had a porosity of 99.7% and a *%AC* of 0.04% calculated from Eq. (1.3), (1.4) and (2.12).

The scaffold was then optimized by brute force to reduce the contraction of the scaffold whilst simultaneously maintaining a high porosity. The optimizer ran initially over 100 different samples and then created additional models based on the scaffold parameters that produced the best results. The total procedure took between 1 to 2 hours and 4 candidates were selected which reduced the contraction of the scaffold whilst still maintain a high level of porosity. Other optimization processes performed on similar applications have taken multiple days to produce the same number of results highlighting the model's low computational cost [33].

Table 3.7: Candidates produced by the direct optimization on ANSYS, with %AC, porosity, and cell effective stiffness.

Candidates	**t (µm)**	**E_s (MPa)**	**%AC**	**Porosity (%)**	**Mean Cell effective Stiffness (N/m)**
Nominal Case	3	5	0.04	99.7	5.02
No. 1	3.766	9.736	0.00893	99.48 %	970.4
No. 2	3.702	9.761	0.0092	99.50 %	2592
No. 3	3.094	9.73	0.0153	99.65 %	381
No. 4	3.074	9.73	0.0156	99.66 %	395

All candidates showed a significant decrease in the contraction (Fig. 3.6) of the scaffold whilst still maintain high porosity (Tab. 3.7). In all cases the strut modulus was increased to roughly 9.7 MPa. This is because the optimizer is still missing a condition that would require a reduced strut modulus. Inclusion of a range of values for the mean cell effective stiffness would allow for the correct optimization of E_s. The cell effective stiffness was most effected by the elastic modulus of the struts in comparison to the thickness of struts (Tab. 3.7).

Chapter 4

Discussion

This thesis focuses on creating a computationally efficient method to model the mechanical behaviour of Porous Collagen-based Scaffolds (PCS) in the micrometre and millimetre. The sponge-like structure of PCS was modelled using lattices generated by repetition of a unit cell, either cube or tetrakaidekahedrons. Finite element models of PCS were generated using 1D finite elements. Both linear mechanics and nonlinear (neo-Hookean) mechanics were considered.

The implication is that models can be produced to study interactions of cells with PCS on a microscopic scale and relate this back to macroscopic changes in the scaffold.

Creating cell-matrix interactions allowed the investigation of cell variables such as the distribution of cells throughout PCS, the force (F_c) etc. and scaffold variables e.g., strut thickness (t), unit cell size (P_s) etc. The model was able to utilize design optimization features meaning that controllable aspects of the design such as the strut modulus (E_s) and the thickness of struts (t) could be tuned to achieve desirable contraction whilst maintaining a sufficient porosity for events such as cell migration and proliferation.

Unconfined compression simulations for linear elastic materials could simulate high-strain deformations. The use of pipe elements instead of beam elements was shown to be interchangeable when modelling linear elastic properties. However pipe elements could include hyper elastic material models, in contrast to beam elements. The use of pipe elements demonstrated a good capability to capture strain of up until ε=46% in the case of P_s =110 μm and ε=27.5% for P_s = 96 μm with tetrakaidekahedral unit cells.

Simulations of unconfined compression of PCS of mean pore diameter P_s = 110 μm, the full version of ANSYS was used. The compression simulations for P_s = 96 μm was performed on ANSYS student version which may have been in part a reason the results were only given to ε = 27.5% versus ε = 46%. This is because ANSYS student has a limited processing capacity. For example, despite the hardware used having 6 cores available (Table 2.1), ANSYS student limits the number of cores to a maximum of 4.

Worth noting is that for the compression simulation with P_s = 110 μm the full version of ANSYS was used. The compression simulations for P_s = 96 μm was performed on ANSYS student version which may have been in part a reason the strain was lower.

The initial macroscopic elastic modulus (E^*), the collapse plateau elastic modulus ($\Delta\sigma/\Delta\varepsilon$), the elastic buckling stress (σ_{el}^*) and strain (ε_{el}^*) were all able to be estimated (Tab. 3.2). E^*, σ_{el}^*, and ε_{el}^* by a particular finite element model (tetrakaidekahedron unit cell, pipe finite elements, neo-Hookean mechanics) had a good agreement with the experimental measurements (Tab. 1.1). E^* had a good agreement with the analytical calculations using constants defined by *Gibson et al., 1997* (Table 1.2, Tab. 3.3), the other analytical results had some margin of error but were all in the correct order of magnitude.

The reduction in the elastic modulus of E^* and $\Delta\sigma/\Delta\varepsilon$ may be due to the overall reduced dimensions of the scaffold when compared to the experimental scaffolds [15]. To observe if this is the case the lattice could be expanded, and the compression simulations repeated. If the results do change it suggests that macroscopic elastic modulus is not only related to the relative density (Eq. (1.1)) but also the scaffold dimensions.

Cubic unit cells demonstrated that the geometry of individual unit cells is important to correctly characterize the macroscopic effects of PCS. Alignment of struts in the direction of compression led to increased macroscopic elastic modulus. The effect was like that of the works of *Herrera et al., 2019,* where porous scaffold (composed of collagen walls aligned in parallel)

had a higher macroscopic elastic modulus compared to PCS with randomly orientated struts such as CG scaffolds [15].

Despite the increase in E^* for cubic unit cells, the scaffold began to collapse at lower stress. The reduction in elastic buckling stress (σ_{el}^*) may also be attributed to the number of struts within the scaffold. A single unit cell for the cube scaffold is composed of 12 struts compared to a tetrakaidekahedral unit cell having 36 struts. This leads to a scaffold composed of tetrakaidekahedrons having 3 times the number of struts compared to the cubic unit cell scaffolds. Normally cubic unit cells have a higher strut thickness. However in the simulations performed the unit cell size and strut thickness was equal to that of the tetrakaidekahedral unit cell tests (P_s = 96 μm, t = 2.55 μm). Another way to compare between the unit cells (cubic and tetrakaidekahedrons) is to keep the relative density of the two scaffolds constant. This would give the cubic unit cells a larger strut thickness which is more representative of PCS with cubic unit cells.

The results of the compression simulations show lattices derived from cubic unit cells cannot capture the properties of open-cell elastomeric foams. The use of the cubic unit cells was best suited for testing aspects of the model set up rather than simulating cell-matrix interactions. Despite being unrealistic for modelling elastomeric foams the cube unit cells offer the advantage of faster processing times. The tetrakaidekahedron has 24 vertices connected by a total of 36 struts, whereas the cube has only 8 vertices and 12 struts meaning the geometry was a lot easier to produce. Due to the simpler geometry models with cubic unit cells could make larger scaffolds than the tetrakaidekahedron unit cells. Something which was a limitation throughout simulations.

Models that study cell-scaffold interactions consider extra variables including the number of active cells, the magnitude of forces applied by cells, and cell localization. Due to the inclusion of such variables in the model, specific conditions not controllable experimentally were able to be assessed and trends could be seen. It was seen that when mean cell effective stiffness increased the macroscopic contraction decreased (Table 3.6).

One question that remains is the desirable cell effective stiffness. *Herrera et al., 2019* showed that k_{ce} is important for cell responses such as migration, proliferation, and differentiation. But it has not been shown what is the optimal k_{ce} for applications such as wound healing. The model offers ways to investigate scaffold design parameters and cell variables on the average k_{ce} whilst also being able to predict the macroscopic effects on the scaffold. With identification of ideal k_{ce} the model optimization process could be further improved.

Most of the simulations had *%AC* in the same order of magnitude (Tab. 3.6). As the model was static, essentially applying only initial contractile forces to the scaffold. Scaffold are shown to contract over a period having initially very small levels of contraction [12]. The contraction is due to cell proliferation resulting in increased cell numbers and hence lager contractions, larger contraction was seen in the model with increasing active cell number (Table 3.6). The only models that produced a significantly large contraction either had exceptionally low strut thickness (1 μm), or a low strut modulus (100 kPa). This implies that there are critical values that PCS design should adhere to, to avoid significant contraction which may result in failure.

The average cell effective stiffness was shown to be lower than the substrate for which they are adhering and lower than the macroscopic stiffness. *Herrera et al., 2019* reported that the stiffness sensed by cells was lower than the substrate but greater than the macroscopic stiffness. The reason for the difference requires further investigation but could be due to the fact many cells were used compared to the use of only a single cell by *Herrera 2019*. Cell effective stiffness could be affected by the actions of neighbouring cells in the PCS lattice.

The design optimization procedure improved upon an initial scaffold design having 3 μm strut thickness (t) and 5 MPa strut elastic modulus (E_s). The aim of the optimizer was to reduce the scaffold contraction by adjusting the strut modulus and strut thickness. The procedure produced 4 possible candidates (Tab. 3.7) which all showed reductions in contraction whilst still maintain a high porosity (~99.5%). The candidates that had the lowest contraction had a higher thickness and strut modulus which was expected. The stiffness sensed by cells was also recorded alongside these scaffolds. The highest k_{ce} was found in the candidate with the greatest strut

modulus suggesting that E_s is more influential on k_{ce} than the strut thickness. The current model was able to simulate cell-matrix interactions and perform mass simulations with lower computational cost than other PCS models solving within hours compared to days it takes for other procedures. For example, *Parsons et al., 2022* optimized fibre dimensions to increase cell velocity. *Parson's* optimization procedure took 24 days to solve 105 simulations. The optimization used on the cell-matrix model in this study solved 100 simulations in between 1 to 2 hours.

Limitations

Design optimization is used to produce possible design for experimental applications. A key feature in scaffold design is the mean size of pores in the scaffold. The model was unable to include the pore size (P_s) in optimization at the present state. The geometry however did consider pore size as a variable and the effects of increasing pore size on cell-matrix interactions was recorded (Fig. 3.4). P_s should be included in design optimization studies in the future as they are a key feature which can be tuned when designing PCS.

The implementation of non-linear material properties was done during compression simulations but not in cell-matrix interaction models. The struts in the PCS lattice underwent significant deformations. At high strain 1D finite elements that model the struts were subject to element distortion. Hence the model failed to converge with normal to high cell force (20 – 100 nN), and normal to high active cell numbers (1000+). The model was in some instances able to solve with a refined mesh, low force, low cell number, and reduced rate of loading. However, the model then represents scaffolds under low strain which is comparable with linear elastic materials. This is not useful as it does not show the non-linear effects and comes with an increase in computational costs.

Using a finer mesh was shown to reduce element distortion and is a recommended and tested solution. However, ANSYS student, for which a large portion of the cell-matrix interactions were modelled, has a limit to the number of mesh nodes and cannot produce the desired mesh.

Despite not being able to include the non-linear material properties in cell-matrix interactions the use of 1D elements was validated. With future investigations as to why the model failed 1D elements could greatly improve computational costs and model non-linear materials which have so far been only done using 3D elements.

An assumption that was made was that the cell-matrix interactions were independent of time. This meant that cell forces were only applied at one instant. It has been shown experimentally that for large scaffold contraction the cells apply forces over a period of days [12]. The model could be adapted to apply forces, randomly move the location of these forces, and apply them again in a repetitive manner thus produced scaffold deformations over a period. Doing this may allow for easier comparison with experimental data and would also provide better insight for scaffold design optimization.

Cell effective stiffness was averaged by taking the displacements of mesh nodes after solving the model and calculating the change in distance between attachment sites. The model cannot tell the user whether the cell is in compression (due to the contraction of the cell) or tension (due to contraction of surrounding cell) which may be important for understanding mechanosensing. The model may be adapted to be able to divide cell effective stiffness into two groups. One under tension and another under compression. This can give more information about cell-cell interactions.

The design optimization was set to achieve low scaffold contraction whilst maintaining a high porosity by adjusting E_s and t. It has been shown that the elastic modulus of scaffold constituents (such as struts) can be controlled [17]. However, the increase of elastic modulus can in turn affect the geometrical features of scaffold pores. For example, *Herrera et al., 2019* demonstrated that increasing the wt% of collagen led to increase in the elastic modulus of collagen walls but also influenced their thickness. The optimizer in this research does not factor that an increase to the thickness of struts may be associated with increase strut modulus. Unconfined compression tests can relate the pore size and strut thickness to E_s (Eq. (1.1) &

(1.2)). The use of these relations could be applied to the model to adjust the thickness as a function of E_s.

Future Work

The model was limited to only being able to model linear cell-matrix interactions. This was due to the limitations with mesh size. In future a mesh study should be carried out to observe if a finer mesh can be implemented to resolve element distortion. The model should also be adapted to be able to optimize the size of pores. The inclusion of pore size will greatly influence the porosity of the scaffold and is influential on cell interactions such as migration.

Variables that were used during the model set up are useful for defining specific conditions seen in experiment. For example, it is shown that the distribution of cells is affected by the scaffold in which they are seeded [27]. The cell space radius can control the distribution of cells but could also be used to create cluster of cells at various volumes in the scaffolds (Fig. 4.1).

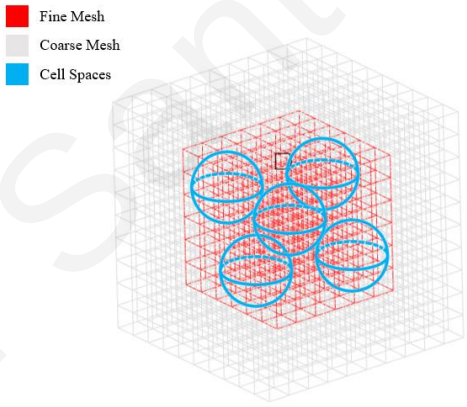


Figure 4.1: Use of multiple cell spaces in a single PCS model, demonstrating how clustering of cells could be modelled.

It was suggested that the orientation of struts in the cubic unit cells were responsible for the change in macroscopic stiffness. This can be validated using the script for the tetrakaidekahedral unit cells (Appendix A.3). The unit cells produced in this script included the aspect ratio between the height of the pore and its width/length. This means that the orientation of struts can be aligned either vertically or horizontally (Fig. 4.2).

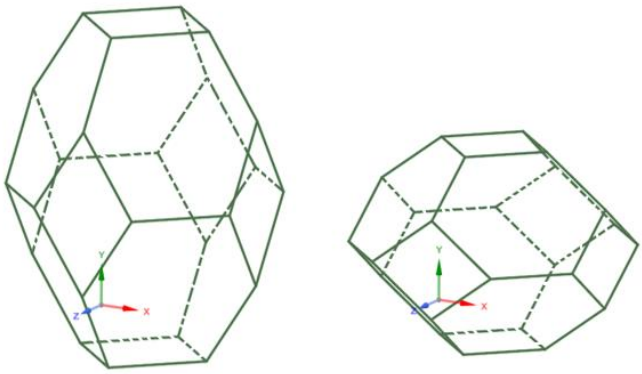


Figure 4.2: Effect of adjusting the aspect ratio on a unit cell using the tetrakaidekahedron script (Appendix A.3)

The model presented was set to only consider scaffolds of minimal dimensions ($\sim 1mm^3$). With improved hardware the scripts that produced the geometries could be applied to produce large scale scaffolds. Better hardware could also improve upon modelling times. The use of script not only allows the adaptability of variables (e.g., strut thickness) but also allows the cell-matrix interactions to be applied to various PCS designs. Provided the model is set up with 1D elements.

Conclusions

A new technique for modelling Porous Collagen-based Scaffolds was presented. Demonstrating capabilities to model open-cell elastomeric foams using 1D Pipe elements in place of 1D beam elements. The mechanical response of 1D pipe elements to model PCS was validated through unconfined compression simulations. The model also validated the use of tetrakaidekahedrons in capturing the macroscopic stress-strain behaviour. Cell-matrix interactions were simulated allowing for control over cellular variables such as the magnitude of cell contractile forces, the number of active cells within a PCS, and how these cells are distributed within the scaffold. The results demonstrated that the average stiffness sensed by cells increases with both strut thickness and elastic modulus of struts. The simulations were computationally efficient due to the use of 1D elements and automatization of the model set up through scripting available on ANSYS. The results demonstrated that cell effective stiffness is influenced by the surrounding cells as well as the macroscopic and microscopic properties of the scaffolds. The method provided can be adapted to fit various types of PCS by changing unit cell geometry and the mechanical properties of the struts.

Bibliography

[1] Boccaccio, A. et al. (2018) "Rhombicuboctahedron unit cell-based scaffolds for bone regeneration: Geometry optimization with a mechanobiology – driven algorithm," Materials Science and Engineering: C, 83, pp. 51–66. Available at: https://doi.org/10.1016/j.msec.2017.09.004.

[2] Buganza Tepole, A. et al. (2011) "Growing skin: A computational model for skin expansion in reconstructive surgery," Journal of the Mechanics and Physics of Solids, 59(10), pp. 2177–2190. Available at: https://doi.org/10.1016/j.jmps.2011.05.004.

[3] Buganza Tepole, A. and Kuhl, E. (2014) "Computational modeling of chemo-bio-mechanical coupling: A systems-biology approach toward wound healing," Computer Methods in Biomechanics and Biomedical Engineering, 19(1), pp. 13–30. Available at: https://doi.org/10.1080/10255842.2014.980821.

[4] Buganza Tepole, A. (2017) "Computational Systems Mechanobiology of Wound Healing," Computer Methods in Applied Mechanics and Engineering, 314, pp. 46–70. Available at: https://doi.org/10.1016/j.cma.2016.04.034.

[5] Buganza Tepole, A. (2019) "Constitutive modelling of wound healing," Studies in Mechanobiology, Tissue Engineering and Biomaterials, pp. 101–133. Available at: https://doi.org/10.1007/978-3-030-13279-8_4.

[6] Chantarapanich, N. *et al.* (2012) 'Scaffold Library for tissue engineering: A geometric evaluation', *Computational and Mathematical Methods in Medicine*, 2012, pp. 1–14. doi:10.1155/2012/407805.

[7] Chen, C.S., Yannas, I.V. and Spector, M. (1995) 'Pore strain behaviour of collagen-glycosaminoglycan analogues of extracellular matrix', *Biomaterials*, 16(10), pp. 777–783. doi:10.1016/0142-9612(95)99640-8.

[8] Eckes, B., Nischt, R. and Krieg, T. (2010) 'Cell-matrix interactions in dermal repair and scarring', *Fibrogenesis & Tissue Repair*, 3(1). doi:10.1186/1755-1536-3-4.

[9] Feng, X. *et al.* (2020) 'Influence of pore architectures of silk fibroin/collagen composite scaffolds on the regeneration of osteochondral defects *in vivo*', *Journal of Materials Chemistry B*, 8(3), pp. 391–405. doi:10.1039/c9tb01558b.

[10] Frankel, A. *et al.* (2022) "Machine learning constitutive models of elastomeric foams," *Computer Methods in Applied Mechanics and Engineering*, 391, p. 114492. Available at: https://doi.org/10.1016/j.cma.2021.114492.

[11] Freyman, T.M. et al. (2001) "Micromechanics of fibroblast contraction of a collagen-gag matrix," Advances in Bioengineering [Preprint]. Available at: https://doi.org/10.1115/imece2001/bed-23150.

[12] Freyman, T.M. et al. (2002) "Fibroblast contractile force is independent of the stiffness which resists the contraction," Experimental Cell Research, 272(2), pp. 153–162. Available at: https://doi.org/10.1006/excr.2001.5408.

[13] Früh, A., Rolauffs, B. and Seidenstuecker, M. (2022) 'Parametric numerical modeling and fabrication of PCL scaffolds for Bone Tissue Engineering Applications', *Applied Sciences*, 12(23), p. 12280. doi:10.3390/app122312280.

[14] Gibson, L.J. and Ashby, M.F. (1997) "Cellular solids." Available at: https://doi.org/10.1017/cbo9781139878326.

[15] Harley, B. et al. (2007a) "Mechanical characterization of collagen–glycosaminoglycan scaffolds," Acta Biomaterialia, 3(4), pp. 463–474. Available at: https://doi.org/10.1016/j.actbio.2006.12.009.

[16] Harley, B.A. et al. (2007b) "A new technique for calculating individual dermal fibroblast contractile forces generated within collagen-gag scaffolds," Biophysical Journal, 93(8), pp. 2911–2922. Available at: https://doi.org/10.1529/biophysj.106.095471.

[17] Herrera, A. et al. (2019) "From macroscopic mechanics to cell-effective stiffness within highly aligned macroporous collagen scaffolds," Materials Science and Engineering: C, 103, p. 109760. Available at: https://doi.org/10.1016/j.msec.2019.109760.

[18] Huebner, P. *et al.* (2019) 'Mechanical properties of tissue formed in vivo are affected by 3D-bioplotted scaffold microarchitecture and correlate with ECM collagen fiber alignment', *Connective Tissue Research*, 61(2), pp. 190–204. doi:10.1080/03008207.2019.1624733.

[19] Hutmacher, D.W., Sittinger, M. and Risbud, M.V. (2004) 'Scaffold-based tissue engineering: Rationale for computer-aided design and solid free-form Fabrication Systems', *Trends in Biotechnology*, 22(7), pp. 354–362. doi:10.1016/j.tibtech.2004.05.005.

[20] Jeon, H., Kim, E. and Grigoropoulos, C.P. (2010) "Measurement of contractile forces generated by individual fibroblasts on self-standing fiber scaffolds," *Biomedical Microdevices*, 13(1), pp. 107–115. Available at: https://doi.org/10.1007/s10544-010-9475-5.

[21] Jiang, S. et al. (2018) "Physical properties of implanted porous Bioscaffolds regulate skin repair: Focusing on mechanical and structural features," Advanced Healthcare Materials, 7(6), p. 1700894. Available at: https://doi.org/10.1002/adhm.201700894.

[22] Kanungo, B.P. and Gibson, L.J. (2010) "Density–property relationships in collagen–glycosaminoglycan scaffolds," Acta Biomaterialia, 6(2), pp. 344–353. Available at: https://doi.org/10.1016/j.actbio.2009.09.012.

[23] Könnig, D. et al. (2017) "Mechanosensation across borders: Fibroblasts inside a macroporous scaffold sense and respond to the mechanical environment beyond the scaffold walls," Journal of Tissue Engineering and Regenerative Medicine, 12(1), pp. 265–275. Available at: https://doi.org/10.1002/term.2410.

[24] Lacroix, D. et al. (2006) "Micro-finite element models of bone tissue-engineering scaffolds," Biomaterials, 27(30), pp. 5326–5334. Available at: https://doi.org/10.1016/j.biomaterials.2006. 06.009.

[25] Legant, W.R. *et al.* (2009) "Microfabricated tissue gauges to measure and manipulate forces from 3D microtissues," *Proceedings of the National Academy of Sciences*, 106(25), pp. 10097–10102. Available at: https://doi.org/10.1073/pnas.0900174106.

[26] Ma, L. *et al.* (2004) 'Biodegradability and cell-mediated contraction of porous collagen scaffolds: The effect of lysine as a novel crosslinking bridge', *Journal of Biomedical Materials Research*, 71A(2), pp. 334–342. doi:10.1002/jbm.a.30170.

[27] Mi, H.-Y. *et al.* (2014) 'Properties and fibroblast cellular response of soft and hard thermoplastic polyurethane electrospun nanofibrous scaffolds', *Journal of Biomedical Materials Research Part B: Applied Biomaterials*, 103(5), pp. 960–970. doi:10.1002/jbm.b.33271.

[28] Minor, A.J. and Coulombe, K.L. (2020) 'Engineering a collagen matrix for cell-instructive regenerative angiogenesis', *Journal of Biomedical Materials Research Part B: Applied Biomaterials*, 108(6), pp. 2407–2416. doi:10.1002/jbm.b.34573.

[29] Mitrossilis, D. *et al.* (2009) "Single-cell response to stiffness exhibits muscle-like behavior," *Proceedings of the National Academy of Sciences*, 106(43), pp. 18243–18248. Available at: https://doi.org/10.1073/pnas.0903994106.

[30] Mohammadalipour, M. *et al.* (2023) 'Theoretical and experimental investigation of solubility and Young's modulus models for polyhydroxybutyrate-based electrospun scaffolds', *Journal of Applied Polymer Science*, 140(13). doi:10.1002/app.53666.

[31] Nie, Z., Lin, Y. and Tong, Q. (2017) "Modeling structures of open cell foams," Computational Materials Science, 131, pp. 160–169. Available at: https://doi.org/10.1016/j.commatsci.2017. 01.029.

[32] Page, M.I., Linde, P.E. and Puttlitz, C.M. (2021) "High Throughput Computational Evaluation of how scaffold architecture, material selection, and loading modality influence the cellular micromechanical environment in Tissue Engineering Strategies," JOR SPINE, 4(3). Available at: https://doi.org/10.1002/jsp2.1152.

[33] Parsons, R., Sestito, J.M. and Luke, B.S. (2022) "Computational analysis and optimization of geometric parameters for fibrous scaffold design," ACS Omega, 7(45), pp. 41449–41460. Available at: https://doi.org/10.1021/acsomega.2c05234.s001.

[34] Reina-Romo, E. et al. (2019) "Computational design of tissue engineering scaffolds," Handbook of Tissue Engineering Scaffolds: Volume One, pp. 73–92. Available at: https://doi.org/10.1016/b978-0-08-102563-5.00004-6.

[35] Rodríguez-Montaño, Ó.L. *et al.* (2018) 'Comparison of the mechanobiological performance of bone tissue scaffolds based on different unit cell geometries', *Journal of the Mechanical Behavior of Biomedical Materials*, 83, pp. 28–45. doi:10.1016/j.jmbbm.2018.04.008.

[36] Ryan, A.J. *et al.* (2014) 'Effect of different hydroxyapatite incorporation methods on the structural and biological properties of porous collagen scaffolds for bone repair', *Journal of Anatomy*, 227(6), pp. 732–745. doi:10.1111/joa.12262.

[37] Saez, A. *et al.* (2005) "Is the mechanical activity of epithelial cells controlled by deformations or forces?," *Biophysical Journal*, 89(6). Available at: https://doi.org/10.1529/biophysj.105.071217.

[38] Samourides, A. *et al.* (2020) 'The effect of porous structure on the cell proliferation, tissue ingrowth and angiogenic properties of poly(glycerol sebacate urethane) scaffolds', *Materials Science and Engineering: C*, 108, p. 110384. doi:10.1016/j.msec.2019.110384.

[39] Shin, J.-W. et al. (2014) "Contractile forces sustain and polarize hematopoiesis from stem and progenitor cells," Cell Stem Cell, 14(1), pp. 81–93. Available at: https://doi.org/10.1016/j.stem.2013.10.009.

[40] Sohutskay, D.O., Buganza Tepole, A. and Voytik-Harbin, S.L. (2021) "Mechanobiological wound model for improved design and evaluation of collagen dermal replacement scaffolds," Acta Biomaterialia, 135, pp. 368–382. Available at: https://doi.org/10.1016/j.actbio.2021.08.007.

[41] Soller, E.C. *et al.* (2012) 'Common features of optimal collagen scaffolds that disrupt wound contraction and enhance regeneration both in peripheral nerves and in skin', *Biomaterials*, 33(19), pp. 4783–4791. doi:10.1016/j.biomaterials.2012.03.068.

[42] Sullivan, R.M., Ghosn, L.J. and Lerch, B.A. (2008) 'A general tetrakaidecahedron model for open-celled foams', *International Journal of Solids and Structures*, 45(6), pp. 1754–1765. doi:10.1016/j.ijsolstr.2007.10.028.

[43] Vaiani, L. et al. (2021) "Coarse-grained elastic network modelling: A fast and stable numerical tool to characterize mesenchymal stem cells subjected to AFM nanoindentation measurements," Materials Science and Engineering: C, 121, p. 111860. Available at: https://doi.org/10.1016/j.msec.2020.111860.

[44] Woodley, D.T., O'Keefe, E.J. and Prunieras, M. (1985) 'Cutaneous wound healing: A model for cell-matrix interactions', *Journal of the American Academy of Dermatology*, 12(2), pp. 420–433. doi:10.1016/s0190-9622(85)80005-0.

[45] Yang, B. *et al.* (2021) 'Enhanced Mechanosensing of cells in synthetic 3D matrix with controlled biophysical dynamics', *Nature Communications*, 12(1). doi:10.1038/s41467-021-23120-0.

[46] Yannas, I.V., Tzeranis, D.S. and So, P.T. (2017) "Regeneration of injured skin and peripheral nerves requires control of wound contraction, not scar formation," Wound Repair and Regeneration, 25(2), pp. 177–191. Available at: https://doi.org/10.1111/wrr.12516.

[47] Zahalak, G.I. et al. (2000) "A cell-based constitutive relation for bio-artificial tissues," Biophysical Journal, 79(5), pp. 2369–2381. Available at: https://doi.org/10.1016/s0006-3495(00)76482-4.

Appendix A

Appendix A.1: SpaceClaim Script – Cubic Unit Cells

The code in the following section is used to create scaffolds consisting of cubic unit cells. The user must input the size of the unit cell, how many units are along each edge, and the thickness of the struts in the scaffold. The script will then automatically construct the scaffolds to the specified values.

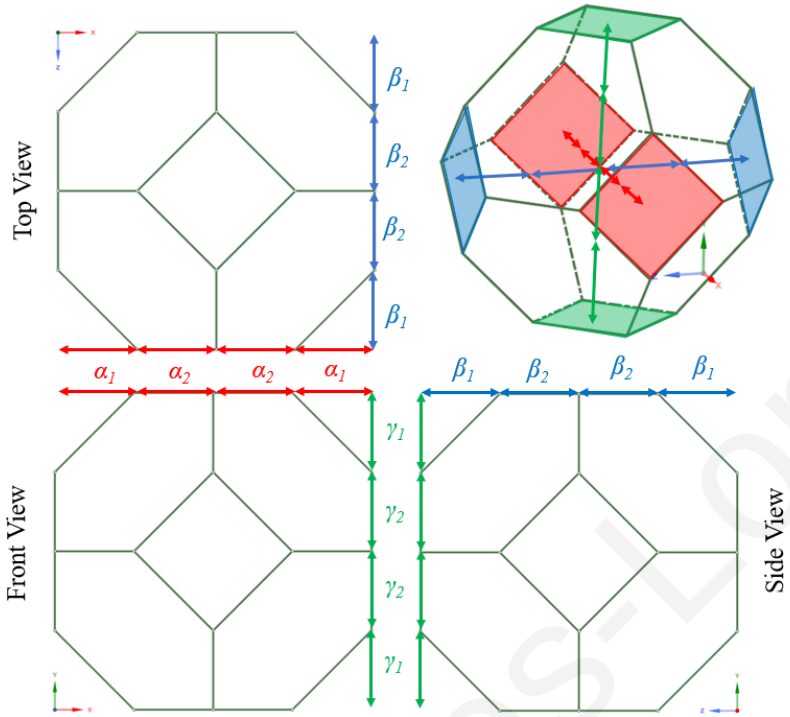
```
# Python Script, API Version = V19
ClearAll() # Clear Current geometry

#Prepare Sketch Area
CreateSketch = Sketch3D.Set3DSketchMode(True)

#Define number of cells, diameter, and strut length

num_cells = 3 # Number of units along each axis
strut_length = 100 # Strut length or unit cell size
Diam = 3 # Strut thickness

#Create Points (vertices)
p1x = 0
p1y = 0
p1z = 0
P1 = Point.Create(MM(p1x),MM(p1y),MM(p1z))

p2x = p1x + strut_length
p2y = p1y
p2z = p1z
P2 = Point.Create(MM(p2x),MM(p2y),MM(p2z))

p3x = p1x
p3y = p1y
p3z = p1z + strut_length
P3 = Point.Create(MM(p3x),MM(p3y),MM(p3z))

p4x = p1x + strut_length
p4y = p1y
p4z = p1z + strut_length
P4 = Point.Create(MM(p4x),MM(p4y),MM(p4z))

p5x = p1x
p5y = p1y + strut_length
p5z = p1z
P5 = Point.Create(MM(p5x),MM(p5y),MM(p5z))

p6x = p1x + strut_length
p6y = p5y
p6z = p1z
P6 = Point.Create(MM(p6x),MM(p6y),MM(p6z))

p7x = p1x
p7y = p5y
p7z = p1z + strut_length
P7 = Point.Create(MM(p7x),MM(p7y),MM(p7z))

p8x = p1x + strut_length
p8y = p5y
p8z = p1z + strut_length
P8 = Point.Create(MM(p8x),MM(p8y),MM(p8z))

#Create Lines (miss lines – not included initially to prevent overlapping)
```

```
L1 = SketchLine.Create(P4,P8)
L2 = SketchLine.Create(P6,P8)
L3 = SketchLine.Create(P7,P8)

if num_cells > 1:
    select = Selection.SelectAll()
    data = LinearPatternData()
    data.PatternDimension = PatternDimensionType.Two
    data.LinearDirection = Selection.Create(GetRootPart().CoordinateSystems[0].Axes[1])
    data.CountX = num_cells
    data.PitchX = MM(strut_length)
    data.CountY = num_cells
    data.PitchY = MM(strut_length)
    result = Pattern.CreateLinear(select, data, None)

    select = Selection.SelectAll()
    data = LinearPatternData()
    data.LinearDirection = Selection.Create(GetRootPart().CoordinateSystems[0].Axes[2])
    data.CountX = num_cells
    data.PitchX = MM(strut_length)
    result = Pattern.CreateLinear(select, data, None)

# Fill in missing lines (Bottom)

L1 = SketchLine.Create(P2,P4)
L2 = SketchLine.Create(P3,P4)

num_lines1 = 3*(num_cells**3)

# Create Named Selection Group
primarySelection = Selection.Create([GetRootPart().Curves[num_lines1],
    GetRootPart().Curves[num_lines1+1]])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)

if num_cells > 1:
    select = Selection.CreateByGroups("Group1")
    data = LinearPatternData()
    data.LinearDirection = Selection.Create(GetRootPart().CoordinateSystems[0].Axes[2])
    data.CountX = num_cells
    data.PitchX = MM(strut_length)
    result = Pattern.CreateLinear(select, data, None)

num_lines2 = num_lines1 + 2*num_cells

if num_cells > 1:
    select = Selection.Create(GetRootPart().Curves[(num_lines1):(num_lines2)])
    data = LinearPatternData()
    data.LinearDirection = Selection.Create(GetRootPart().CoordinateSystems[0].Axes[0])
    data.CountX = num_cells
    data.PitchX = MM(strut_length)
    result = Pattern.CreateLinear(select, data, None)

#Fill in missing lines (left)

L1 = SketchLine.Create(P5,P7)
L2 = SketchLine.Create(P3,P7)

num_lines3 = 3*(num_cells**3)+(2*num_cells**2)

# Create Named Selection Group
primarySelection = Selection.Create([GetRootPart().Curves[num_lines3],
    GetRootPart().Curves[num_lines3+1]])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)
# EndBlock

if num_cells > 1:
    select = Selection.CreateByGroups("Group2")
    data = LinearPatternData()
    data.PatternDimension = PatternDimensionType.Two
    data.LinearDirection = Selection.Create(GetRootPart().CoordinateSystems[0].Axes[2])
    data.CountX = num_cells
    data.PitchX = MM(strut_length)
    data.CountY = num_cells
```

```
    data.PitchY = MM(strut_length)
    result = Pattern.CreateLinear(select, data, None)

#Fill in missing lines (back face)
L1 = SketchLine.Create(P2,P6)
L2 = SketchLine.Create(P5,P6)

num_lines4 = (3)*(num_cells**3)+2*(2*num_cells**2)

# Create Named Selection Group
primarySelection = Selection.Create([GetRootPart().Curves[num_lines4],
    GetRootPart().Curves[num_lines4+1]])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)
# EndBlock

if num_cells > 1:
    select = Selection.CreateByGroups("Group3")
    data = LinearPatternData()
    data.PatternDimension = PatternDimensionType.Two
    data.LinearDirection = Selection.Create(GetRootPart().CoordinateSystems[0].Axes[1])
    data.CountX = num_cells
    data.PitchX = MM(strut_length)
    data.CountY = num_cells
    data.PitchY = MM(strut_length)
    result = Pattern.CreateLinear(select, data, None)

# Fill in lines (final)
L1 = SketchLine.Create(P1,P3)

num_lines5 = (3)*(num_cells**3)+3*(2*num_cells**2)

# Create Named Selection Group
primarySelection = Selection.Create(GetRootPart().Curves[num_lines5])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)

#Linear Pattern
if num_cells > 1:
    select = Selection.CreateByGroups("Group4")
    data = LinearPatternData()
    data.LinearDirection = Selection.Create(GetRootPart().CoordinateSystems[0].Axes[2])
    data.CountX = num_cells
    data.PitchX = MM(strut_length)
    result = Pattern.CreateLinear(select, data, None)

# Fill in missing lines
L1 = SketchLine.Create(P1,P2)
num_lines6 = (3)*(num_cells**3)+3*(2*num_cells**2)+(num_cells)

primarySelection = Selection.Create(GetRootPart().Curves[num_lines6])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)

if num_cells > 1:
    select = Selection.CreateByGroups("Group5")
    data = LinearPatternData()
    data.LinearDirection = Selection.Create(GetRootPart().CoordinateSystems[0].Axes[0])
    data.CountX = num_cells
    data.PitchX = MM(strut_length)
    result = Pattern.CreateLinear(select, data, None)

L1 = SketchLine.Create(P1,P5)

num_lines7 = (3)*(num_cells**3)+3*(2*num_cells**2)+2*(num_cells)

# Create Named Selection Group
primarySelection = Selection.Create(GetRootPart().Curves[num_lines7])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)

if num_cells > 1:
    select = Selection.CreateByGroups("Group6")
    data = LinearPatternData()
    data.LinearDirection = Selection.Create(GetRootPart().CoordinateSystems[0].Axes[1])
```

```
    data.CountX = num_cells
    data.PitchX = MM(strut_length)
    result = Pattern.CreateLinear(select, data, None)

#Create Cross Section Area
CrossArea = BeamProfile.CreateDisk(MM(Diam),'Cross') #Comment out for pipe elements
CrossArea = BeamProfile.CreateCircular(MM(Diam),MM(0.0001),"Cross") #Comment out for beam elements

# Create Beam
BeamAss = Part1 #SpaceClaim creates Part1 to be able to assign the cross sectional area to the line bodies
select = Selection.SelectAll()
result = Beam.Create(select,BeamAss)

# Assign all bodies to act as one component
result = ComponentHelper.MoveBodiesToComponent(select, None)

#Set the topology of the line bodies to be shared thus acting as a single body
options = ShareTopologyOptions()
result = ShareTopology.FindAndFix(options)
```

Appendix A.2: Tetrakaidekahedral Equations Derivation



Appendix Figure A.1: Top, front, and side view of tetrakaidekahedral unit cell. Labels correlate to change in x (α), y (γ), and z (β) coordinates which were used to derive an expression relating the

To draw the vertices for the tetrakaidekahedral unit cell the width (W), depth (D), and height (H), which define the pore size, were broken down into $\alpha_{1,2}$, $\beta_{1,2}$, and $\gamma_{1,2}$ which denote the change in x coordinate, change in z coordinate and change in y coordinate between each vertex (Fig. A.1).

A.2.1: Equations for unit cell dependent on strut length

Using the strut length as the input and considering that the depth (D) is equal to the width (W) gave the following,

$$D = W = 2\alpha_1 + 2\alpha_2 = 2\beta_1 + 2\beta_2 \quad \text{(A.1)}$$

and,

$$H = 2\gamma_1 + 2\gamma_2 \tag{A.2}$$

where α_1 and α_2 is the change in x coordinate between vertices, β_1 and β_2 is the change in z coordinate between vertices, and γ_1 and γ_2 is the change in y coordinates between vertices. Due to symmetry in the y axis from the AR, the change in y coordinate is constant from one end of the pore to the other. Hence eq. 2.6 becomes,

$$H = 4\gamma \tag{A.3}$$

where γ is the change in y coordinate between vertices. Considering that W and D are equal,

$$\alpha_1 = \beta_1 \quad \& \quad \alpha_2 = \beta_2$$

The aspect ratio can now be written as,

$$AR = \frac{W}{H} = \frac{2\alpha_1 + 2\alpha_2}{4\gamma} \tag{A.4}$$

Using a defined strut length as an input and Pythagoras' theorem to determine the change in coordinates gave,

$$S_l^2 = \gamma^2 + \alpha_2^2 \tag{A.5}$$

$$S_l^2 = \alpha_1^2 + \beta_1^2 = 2\alpha_1^2 \tag{A.6}$$

$$\therefore \gamma^2 = S_l^2 - \alpha_2^2 \tag{A.7}$$

$$\therefore \alpha_1 = \sqrt{\frac{S_l^2}{2}} = \frac{S_l}{\sqrt{2}} \tag{A.8}$$

where S_l is the strut length. With the relationships defined eq. (2.8) was squared giving,

$$AR^2 = \frac{(2\alpha_1 + 2\alpha_2)^2}{16\gamma^2} = \frac{4\alpha_1^2 + 8\alpha_1\alpha_2 + 4\alpha_2^2}{16(S_l^2 - \alpha_2^2)} \tag{A.9}$$

$$\therefore AR^2 = \frac{4\left(\frac{S_l}{\sqrt{2}}\right)^2 + 8\left(\frac{S_l}{\sqrt{2}}\right)\alpha_2 + 4\alpha_2^2}{16(S_l^2 - \alpha_2^2)} = \frac{2S_l^2 + \frac{8}{\sqrt{2}}S_l\alpha_2 + 4\alpha_2^2}{16(S_l^2 - \alpha_2^2)} \tag{A.10}$$

$$\therefore 4\alpha_2^2 + 16\alpha_2^2 AR^2 + \frac{8}{\sqrt{2}}S_l\alpha_2 + 2S_l^2 - 16S_l^2 AR^2 = 0 \tag{A.11}$$

With quadratic formula α_2 can be determined, hence γ is found (2.11) and the positions of all vertices can be calculated using a script **(Adjust script in Appendix A3)**.

A.2.2: Equations for unit cell dependent on pore size (unit cell size)

Using AR (A.2.1 - Eq. A.4) and defining either the height or the width as the unit cell size allows the change in y coordinate (γ) to be calculated. The change in the x (α) or z (β) coordinates can be thought of as equal due to symmetry of the depth and width (Eq. A.1) giving,

$$\alpha_1 = \frac{D}{2} + \alpha_2 \tag{A.12}$$

and with Eq. A.5 &A.6,

$$\alpha_1^2 = \gamma^2 + \alpha_2^2 \tag{A.13}$$

Eq. A.9 with the substitution of Eq. A.12 & A.13 becomes,

$$AR^2 = \frac{(2\alpha_1 + 2\alpha_2)^2}{16\gamma^2} = \frac{4\left(\frac{\gamma^2}{2} + \frac{\alpha_2^2}{2}\right) + 8\left(\frac{D}{2} - \alpha_2\right)\alpha_2 + 4\alpha_2^2}{16\gamma^2} \tag{A.14}$$

$$\therefore 2\alpha_2^2 - 4D\alpha_2^2 + 16AR^2\gamma^2 - 2\gamma^2 = 0 \tag{A.15}$$

Using quadratic formula on Eq. A.15 gives α_2 which is substituted back into A.12 to get α_1, the equation was implement into a script to produce the tetrakaidekahedron unit cells **(see Appendix A.3).**

Appendix A.3: SpaceClaim Script – Tetrakaidekahedral Unit Cells (pore size dependent)

The code in this section constructs scaffolds composed of tetrakaidekahedral unit cells. The script works by creating unit cells dependent on the diameter of a tetrakaidekahedral unit (Section A.2.2). The user must input the unit cell size, the number of units along each edge, and the thickness of the struts in the scaffold.

```
# Python Script, API Version = V19
ClearAll()

#Prepare Sketch
CreateSketch = Sketch3D.Set3DSketchMode(True)

#Create Lines

#Define number of cells, diameter, and strut length
num_cells = 3 #number of units along each axis
PW = 100 #Define the pore size
Diam = 2.95 #Define strut thickness
AR = 1 #Cell Ascpect Ratio
PH = PW/AR

#DEFINE dx,dz, and dy (pore size dependent)

import math

#Define values using equations in appendix A.2.2
dy = float(PH)/4

a = 2
b = -4*PW
c = (16*AR**2*dy**2-2*dy**2)

dx2 = (-b - math.sqrt((b**2) - 4*a*c))/(2*a) #Quadratic formula
dz2 = dx2

dx1 = (PW/2)-dx2
dz1 = dx1

strut_length = math.sqrt(dx2**2 + dy**2)
checkSL = math.sqrt(2*(dx1**2))


### CREATE SINGLE UNIT ###
# P1
x1 = dx2 + dx1
y1 = 0
z1 = dz1
P1 = Point.Create(MM(x1),MM(y1),MM(z1))
#P2
x2 = x1 - dx1
y2 = 0
z2 = z1 + dz1
P2 = Point.Create(MM(x2),MM(y2),MM(z2))
#P3
x3 = x2 + dx1
y3 = 0
z3 = z2 + dz1
```

```
P3 = Point.Create(MM(x3),MM(y3),MM(z3))
#P4
x4 = x3 +dx1
y4 = 0
z4 = z3 - dz1
P4 = Point.Create(MM(x4),MM(y4),MM(z4))
#P5
x5 = x1
y5 = y1 + dy
z5 = z1 - dz2
P5 = Point.Create(MM(x5),MM(y5),MM(z5))
#P6
x6 = x2 - dx2
y6 = y2 + dy
z6 = z2
P6 = Point.Create(MM(x6),MM(y6),MM(z6))
#P7
x7 = x3
y7 = y3 +dy
z7 = z3 + dz2
P7 = Point.Create(MM(x7),MM(y7),MM(z7))
#P8
x8 = x4 + dx2
y8 = y4 + dy
z8 = z4
P8 = Point.Create(MM(x8),MM(y8),MM(z8))
#P9
x9 = x5 - dx2
y9 = y5 + dy
z9 = z5
P9 = Point.Create(MM(x9),MM(y9),MM(z9))
#P10
x10 = x6
y10 = y6 + dy
z10 = z6 - dz2
P10 = Point.Create(MM(x10),MM(y10),MM(z10))
#P11
x11 = x6
y11 = y6 + dy
z11 = z6 + dz2
P11 = Point.Create(MM(x11),MM(y11),MM(z11))
#P12
x12 = x7 - dx2
y12 = y7 + dy
z12 = z7
P12 = Point.Create(MM(x12),MM(y12),MM(z12))
#P13
x13 = x7 + dx2
y13 = y7 + dy
z13 = z7
P13 = Point.Create(MM(x13),MM(y13),MM(z13))
#P14
x14 = x8
y14 = y8 + dy
z14 = z8 + dz2
P14 = Point.Create(MM(x14),MM(y14),MM(z14))
#P15
x15 = x8
y15 = y8 + dy
z15 = z8 - dz2
P15 = Point.Create(MM(x15),MM(y15),MM(z15))
#P16
x16 = x5 + dx2
y16 = y5 + dy
z16 = z5
P16 = Point.Create(MM(x16),MM(y16),MM(z16))
#P17
```

```
x17 = x9 + dx2
y17 = y9 + dy
z17 = z9
P17 = Point.Create(MM(x17),MM(y17),MM(z17))
#P18
x18 = x10
y18 = y10 + dy
z18 = z10 + dz2
P18 = Point.Create(MM(x18),MM(y18),MM(z18))
#P19
x19 = x12 + dx2
y19 = y12 + dy
z19 = z12
P19 = Point.Create(MM(x19),MM(y19),MM(z19))
#P20
x20 = x14
y20 = y14 + dy
z20 = z14 - dz2
P20 = Point.Create(MM(x20),MM(y20),MM(z20))
#P21
x21 = x17
y21 = y17 + dy
z21 = z17 + dz2
P21 = Point.Create(MM(x21),MM(y21),MM(z21))
#P22
x22 = x18 + dx2
y22 = y18 + dy
z22 = z18
P22 = Point.Create(MM(x22),MM(y22),MM(z22))
#P23
x23 = x19
y23 = y17 + dy
z23 = z19 - dz2
P23 = Point.Create(MM(x23),MM(y23),MM(z23))
#P24
x24 = x20 - dx2
y24 = y20 + dy
z24 = z20
P24 = Point.Create(MM(x24),MM(y24),MM(z24))
#Draw Lines
#L13-L32 (Group 4: miss lines – not included initially to prevent overlapping
L13 = SketchLine.Create(P1,P5)
L14 = SketchLine.Create(P2,P6)
L15 = SketchLine.Create(P3,P7)
L16 = SketchLine.Create(P4,P8)
L17 = SketchLine.Create(P7,P12)
L18 = SketchLine.Create(P7,P13)
L19 = SketchLine.Create(P8,P14)
L20 = SketchLine.Create(P8,P15)
L21 = SketchLine.Create(P9,P10)
L22 = SketchLine.Create(P11,P12)
L23 = SketchLine.Create(P13,P14)
L24 = SketchLine.Create(P15,P16)
L25 = SketchLine.Create(P12,P19)
L26 = SketchLine.Create(P13,P19)
L27 = SketchLine.Create(P14,P20)
L28 = SketchLine.Create(P15,P20)
L29 = SketchLine.Create(P17,P21)
L30 = SketchLine.Create(P18,P22)
L31 = SketchLine.Create(P19,P23)
L32 = SketchLine.Create(P20,P24)
L33 = SketchLine.Create(P21,P22)
L34 = SketchLine.Create(P22,P23)
L35 = SketchLine.Create(P23,P24)
L36 = SketchLine.Create(P24,P21)

if num_cells > 1:
```

```
    select = Selection.SelectAll()
    data = LinearPatternData()
    data.PatternDimension = PatternDimensionType.Two
    data.LinearDirection = Selection.Create(GetRootPart().CoordinateSystems[0].Axes[1])
    data.CountX = num_cells
    data.PitchX = MM(PW)
    data.CountY = num_cells
    data.PitchY = MM(PH)
    result = Pattern.CreateLinear(select, data, None)

if num_cells > 1:
    select = Selection.SelectAll()
    data = LinearPatternData()
    data.LinearDirection = Selection.Create(GetRootPart().CoordinateSystems[0].Axes[2])
    data.CountX = num_cells
    data.PitchX = MM(PW)
    result = Pattern.CreateLinear(select, data, None)

num_lines1 = (36-12)*(num_cells**3)

#L1-L4 (Group 1)
L1 = SketchLine.Create(P1,P2)
L2 = SketchLine.Create(P2,P3)
L3 = SketchLine.Create(P3,P4)
L4 = SketchLine.Create(P4,P1)

# Create Named Selection Group
primarySelection = Selection.Create([GetRootPart().Curves[num_lines1],
    GetRootPart().Curves[num_lines1+1],
    GetRootPart().Curves[num_lines1+2],
    GetRootPart().Curves[num_lines1+3]])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)

num_lines2 = num_lines1 + 4*num_cells

if num_cells > 1:
    select = Selection.CreateByGroups("Group1")
    data = LinearPatternData()
    data.LinearDirection = Selection.Create(GetRootPart().CoordinateSystems[0].Axes[2])
    data.CountX = num_cells
    data.PitchX = MM(PW)
    result = Pattern.CreateLinear(select, data, None)

if num_cells > 1:
    select = Selection.Create(GetRootPart().Curves[(num_lines1):(num_lines2)])
    data = LinearPatternData()
    data.LinearDirection = Selection.Create(GetRootPart().CoordinateSystems[0].Axes[0])
    data.CountX = num_cells
    data.PitchX = MM(PW)
    result = Pattern.CreateLinear(select, data, None)

num_lines3 = (36-12)*(num_cells**3)+(4*num_cells**2)

#L5-L8 (Group 2)
L5 = SketchLine.Create(P5,P9)
L6 = SketchLine.Create(P5,P16)
L7 = SketchLine.Create(P9,P17)
L8 = SketchLine.Create(P16,P17)

# Create Named Selection Group
primarySelection = Selection.Create([GetRootPart().Curves[num_lines3],
    GetRootPart().Curves[num_lines3+1],
    GetRootPart().Curves[num_lines3+2],
    GetRootPart().Curves[num_lines3+3]])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)
```

```
# EndBlock
if num_cells > 1:
    select = Selection.CreateByGroups("Group2")
    data = LinearPatternData()
    data.PatternDimension = PatternDimensionType.Two
    data.LinearDirection = Selection.Create(GetRootPart().CoordinateSystems[0].Axes[1])
    data.CountX = num_cells
    data.PitchX = MM(PW)
    data.CountY = num_cells
    data.PitchY = MM(PH)
    result = Pattern.CreateLinear(select, data, None)

num_lines4 = (36-12)*(num_cells**3)+2*(4*num_cells**2)

#L9-L12 (Group 3)
L9 = SketchLine.Create(P6,P10)
L10 = SketchLine.Create(P6,P11)
L11 = SketchLine.Create(P10,P18)
L12 = SketchLine.Create(P11,P18)

# Create Named Selection Group
primarySelection = Selection.Create([GetRootPart().Curves[num_lines4],
    GetRootPart().Curves[num_lines4+1],
    GetRootPart().Curves[num_lines4+2],
    GetRootPart().Curves[num_lines4+3]])
secondarySelection = Selection.Empty()
result = NamedSelection.Create(primarySelection, secondarySelection)

if num_cells > 1:
    select = Selection.CreateByGroups("Group3")
    data = LinearPatternData()
    data.PatternDimension = PatternDimensionType.Two
    data.LinearDirection = Selection.Create(GetRootPart().CoordinateSystems[0].Axes[2])
    data.CountX = num_cells
    data.PitchX = MM(PW)
    data.CountY = num_cells
    data.PitchY = MM(PH)
    result = Pattern.CreateLinear(select, data, None)

###APPLY CROSS SECTIONAL AREA###

select = Selection.SelectAll()

#Create Cross Section Area – Comment one out depending on whether you want beam or pipe elements
CrossArea = BeamProfile.CreateDisk(MM(Diam),'Cross')
CrossArea = BeamProfile.CreateCircular(MM(Diam),MM(0.0001),'Cross')
BeamASS = Part1

result = Beam.Create(select, BeamASS)

# Assign all bodies to act as one component
result = ComponentHelper.MoveBodiesToComponent(select, None)


#Set the topology of the line bodies to be shared thus acting as a single body
options = ShareTopologyOptions()
result = ShareTopology.FindAndFix(options)
```

Appendix A.4: Mechanical APDL Script

The script for APDL in mechanical is split into three different sections which correlate to what order they are read during the solving procedure. /Prep7 refers to code executed during the model set up e.g., material properties (section A.4.1). The second part of the script is executed before the solve command is issued by ANSYS mechanical (Section A.4.2) and is responsible for applying loading conditions and updating nodal coordinate positions for post-processing. The third part of the script is executed once the model is solved and is used to extract results specifically the cell effective stiffness (section A.4.3).

A.4.1: /Prep7 APDL Commands

The first section of commands is performed at the end of /PREP7 in APDL correlating to model set up. The script selected mesh nodes within the cell space for cell forces to be applied to and defined features of the scaffold such as the scaffold volume which was used for macroscopic results. The cell number was also input in this part of the automation. The user must define the overall length of the scaffold for the %AC to be measured.

```
#START OF SCRIPT
#Define the dimensions of the scaffold and calculate initial cross-section area
SCAFFOLD_LENGTH = 1050
APPROX_AREA = SCAFFOLD_LENGTH**2

#Define number of cells
CELL_NUMBER = 1000

#Define a radius in which cells can possibly be
RADIUS = 400

#Create local coordinate system at center of scaffold
CLOCAL,50,2,525,525,525
*DIM,CELL_PAIRS_,ARRAY,2,CELL_NUMBER
*DIM,COORDINATES_,ARRAY,6,CELL_NUMBER
*DO,I,1,CELL_NUMBER,1

#Select nodes in the radius
CSYS,50
ESEL,S,CENT,X,0,RADIUS
NSLE,S,ACTIVE
*GET,NumNodes,NODE,0,COUNT
*DIM,ntab,ARRAY,NumNodes
*VGET,ntab(1),NODE,,NLIST
*SET,Randomnum,NINT(RAND(1,NumNodes))
CELL_A = ntab(Randomnum)

CSYS,0

*GET,X1,NODE,CELL_A,LOC,X
*GET,Y1,NODE,CELL_A,LOC,Y
*GET,Z1,NODE,CELL_A,LOC,Z

*SET,CELL_A,NODE(X1,Y1,Z1)
```

```
COORDINATE_SYS = 21 + I

#Select a secondary node

LOCAL,COORDINATE_SYS,2,X1,Y1,Z1
CSYS,COORDINATE_SYS
ESEL,S,CENT,X,0,100
NSLE,S,ACTIVE
*GET,NUMNODES,NODE,0,COUNT
*DIM,DTAB,ARRAY,NUMNODES
*VGET,DTAB(1),NODE,,NLIST
*SET,RANDOMNUM,NINT(RAND(1,NUMNODES))
CELL_B = DTAB(RANDOMNUM)


CSYS,0

*GET,x2,NODE,Cell_B,LOC,X
*GET,y2,NODE,Cell_B,LOC,Y
*GET,z2,NODE,Cell_B,LOC,Z

*SET,CELL_B,NODE(X2,Y2,Z2)

#Assign selected nodes to a list
*SET,CELL_PAIRS_(1,I),CELL_A
*SET,CELL_PAIRS_(2,I),CELL_B


#Redord the initial coordinates of the nodes
*SET,COORDINATES_(1,I),X1
*SET,COORDINATES_(2,I),Y1
*SET,COORDINATES_(3,I),Z1
*SET,COORDINATES_(4,I),X2
*SET,COORDINATES_(5,I),Y2
*SET,COORDINATES_(6,I),Z2


*ENDDO
```

A.4.2: Solve APDL Commands

The script in this section is performed just before the solve command in APDL. It applies forces on all the selected nodes and updates the nodal coordinate positions to extraction information for the cell effective stiffness. The user must define the cell contractile force in this section.

```
#START OF SCRIPT
#Define the force
F_IN = 0.06 #due to scaling mN = nN

*DIM,INITIAL_LENGTH_,ARRAY,1,CELL_NUMBER


*DO,I,1,CELL_NUMBER,1

#Extract the coordinates saved from previous script
X1 = COORDINATES_(1,I)
Y1 = COORDINATES_(2,I)
Z1 = COORDINATES_(3,I)

X2 = COORDINATES_(4,I)
Y2 = COORDINATES_(5,I)
Z2 = COORDINATES_(6,I)


#Calculate the force components
X = X2-X1
Y = Y2-Y1
Z = Z2-Z1
```

```
LENGTH_AB = SQRT(X**2+Y**2+Z**2)

*SET,INITIAL_LENGTH_(1,I),LENGTH_AB

*IF,X,NE,0,THEN

   X_FACTOR = X/X
   Y_FACTOR = Y/X
   Z_FACTOR = Z/X

   F_X = F_IN/SQRT(X_FACTOR**2 + Y_FACTOR**2 + Z_FACTOR**2)


   F_Y = F_X*Y_FACTOR
   F_Z = F_X*Z_FACTOR

   *ELSEIF,Y,NE,0

      X_FACTOR = 0
      Y_FACTOR = Y/Y
      Z_FACTOR = Z/Y


      F_Y = F_IN/SQRT(X_FACTOR**2 + Y_FACTOR**2 + Z_FACTOR**2)

      F_X = F_Y*X_FACTOR
      F_Z = F_Y*Z_FACTOR

      *ELSEIF,Z,NE,0


         X_FACTOR = 0
         Y_FACTOR = 0
         Z_FACTOR = Z/Z

         F_Z = F_IN/SQRT(X_FACTOR**2 + Y_FACTOR**2 + Z_FACTOR**2)

         F_X = F_Z*X_FACTOR
         F_Y = F_Z*Y_FACTOR

*ENDIF


*IF,X2,LT,X1,THEN

   F_X=-F_X

*ENDIF

*IF,Y2,LT,Y1,THEN

   F_Y=-F_Y

*ENDIF

*IF,Z2,LT,Z1,THEN


   F_Z=-F_Z

*ENDIF

CELL_A = CELL_PAIRS_(1,I)
CELL_B = CELL_PAIRS_(2,I)

F,CELL_A,FX,F_X
F,CELL_A,FY,F_Y
F,CELL_A,FZ,F_Z


F,CELL_B,Fx,-F_X
F,CELL_B,Fy,-F_Y
F,CELL_B,Fz,-F_Z

*ENDDO
```

```
SOLVE

/SOLU

#Update coordinates and calculate cell effective stiffness for each cell
UPCOORD,1

*DIM,CELL_EFF_STIFF_,ARRAY,1,CELL_NUMBER


*DO,I,1,CELL_NUMBER,1

CELL_A = CELL_PAIRS_(1,I)
CELL_B = CELL_PAIRS_(2,I)

*GET,FINAL_X1,NODE,CELL_A,LOC,X
*GET,FINAL_Y1,NODE,CELL_A,LOC,Y
*GET,FINAL_Z1,NODE,CELL_A,LOC,Z

*GET,FINAL_X2,NODE,CELL_B,LOC,X
*GET,FINAL_Y2,NODE,CELL_B,LOC,Y
*GET,FINAL_Z2,NODE,CELL_B,LOC,Z


FINAL_LENGTH = SQRT((FINAL_X2-FINAL_X1)**2+(FINAL_Y2-FINAL_Y1)**2+(FINAL_Z2-FINAL_Z1)**2)

DELTA_LENGTH = INITIAL_LENGTH_(1,I)-FINAL_LENGTH

*IF,DELTA_LENGTH,NE,0,THEN

CELL_STIFFNESS = F_IN/DELTA_LENGTH

*ELSEIF,DELTA_LENGTH,EQ,0

CELL_STIFFNESS = 0

*ENDIF


*IF,CELL_STIFFNESS,LT,0,THEN
CELL_STIFFNESS = -CELL_STIFFNESS
*ENDIF

*SET,CELL_EFF_STIFF_(1,I),CELL_STIFFNESS

*ENDDO
```

A.4.3: /Post1 APDL Commands

The commands in this section were performed after /POST1 which is responsible for the post processing of results. The code averages the cell effective stiffness and outputs the results for the design optimization process. The contraction is calculated by taking the average deformation of the nodes on the free faces of the scaffold, which must have a named selection applied.

```
#START OF SCRIPT

#Initate the total cell effective stiffness and calculate the average
TOT_CES = 0.0

*DO,I,1,CELL_NUMBER,1

   TOT_CES = TOT_CES + CELL_EFF_STIFF_(1,I)

*ENDDO
```

```
MY_AVE_CELL_STIFFNESS = TOT_CES/CELL_NUMBER

ALLSEL,ALL

SET,LAST
CMSEL,S,TOP_FACE,NODE
TOT_DIS_Y = 0.0
*GET,NNUM,NODE,0,COUNT
NCOUNTER=0

#Calculate the displacement of the free faces
*DO,I,1,NNUM,1
   NCOUNTER=NDNEXT(NCOUNTER)
   NAREA=ARNODE(NCOUNTER)
   *GET,DIS_Y,NODE,NCOUNTER,U,Y
   TOT_DIS_Y = TOT_DIS_Y +DIS_Y

*ENDDO

MY_AVE_DIS_Y_TOP = TOT_DIS_Y/NNUM

CMSEL,S,BOTTOM_FACE,NODE
TOT_DIS_Y = 0.0
*GET,NNUM,NODE,0,COUNT
NCOUNTER=0

*DO,I,1,NNUM,1
   NCOUNTER=NDNEXT(NCOUNTER)
   NAREA=ARNODE(NCOUNTER)
   *GET,DIS_Y,NODE,NCOUNTER,U,Y
   TOT_DIS_Y = TOT_DIS_Y +DIS_Y

*ENDDO

MY_AVE_DIS_Y_BOTTOM = TOT_DIS_Y/NNUM

CMSEL,S,FRONT_FACE,NODE
TOT_DIS_X = 0.0
*GET,NNUM,NODE,0,COUNT
NCOUNTER=0

*DO,I,1,NNUM,1
   NCOUNTER=NDNEXT(NCOUNTER)
   NAREA=ARNODE(NCOUNTER)
   *GET,DIS_X,NODE,NCOUNTER,U,X
   TOT_DIS_X = TOT_DIS_X +DIS_X

*ENDDO

MY_AVE_DIS_X_FRONT = TOT_DIS_X/NNUM

CMSEL,S,BACK_FACE,NODE
TOT_DIS_X = 0.0
*GET,NNUM,NODE,0,COUNT
NCOUNTER=0

*DO,I,1,NNUM,1
   NCOUNTER=NDNEXT(NCOUNTER)
   NAREA=ARNODE(NCOUNTER)
   *GET,DIS_X,NODE,NCOUNTER,U,X
   TOT_DIS_X = TOT_DIS_X +DIS_X

*ENDDO

MY_AVE_DIS_X_BACK = TOT_DIS_X/NNUM
MY_FINAL_HEIGHT = SCAFFOLD_LENGTH + (- MY_AVE_DIS_Y_BOTTOM) + (MY_AVE_DIS_Y_TOP)
MY_FINAL_WIDTH = SCAFFOLD_LENGTH + (- MY_AVE_DIS_X_BACK) + (MY_AVE_DIS_X_FRONT)

MY_FINAL_AREA_APPROX = MY_FINAL_HEIGHT*MY_FINAL_WIDTH
#Calculate the percentage of area reduced
MY_AVE_CONTRACTION = ((APPROX_AREA - MY_FINAL_AREA_APPROX)/APPROX_AREA)*100
FINISH
```

Appendix A.5: Mechanical Python Script

The final script used created the mesh, boundary conditions, and the named selection which were used in the post processing (A.4.3). The user defines the scaffold length and number of divisions in the finer meshed region (see Fig. 2.10).

```
def after_object_changed(this, object_changed, property_name):# Do not edit this line
    """
    Called after an object is changed.
    Keyword Arguments :
        this -- the datamodel object instance of the python code object you are currently editing in the tree
        object_changed -- The object that was changed
        property_name -- The property that was changed
    """
    pass


model = ExtAPI.DataModel.Project.Model
geom = model.Geometry
mesh = model.Mesh
connections = model.Connections
materials = model.Materials
analysis = model.Analyses[0]
solution = analysis.Solution


#Create Name Selections
sel = model.AddNamedSelection()
sel.Name = "Scaffold"
sel.ScopingMethod = GeometryDefineByType.Worksheet
PointA = sel.GenerationCriteria
PointA.Add(None)
PointA[0].EntityType = SelectionType.GeoEdge
PointA[0].Criterion = SelectionCriterionType.LocationX
PointA[0].Operator = SelectionOperatorType.GreaterThanOrEqual
PointA[0].Value=Quantity(0, "mm")
sel.Generate()

ns = model.AddNamedSelection()
ns.Name = "Mesh_Zone"
ns.ScopingMethod = GeometryDefineByType.Worksheet
PointA = ns.GenerationCriteria
PointA.Add(None)
PointA[0].EntityType = SelectionType.GeoEdge
PointA[0].Criterion = SelectionCriterionType.LocationX
PointA[0].Operator = SelectionOperatorType.GreaterThanOrEqual #CHECKK
PointA[0].Value=Quantity(100, "mm")
PointA.Add(None)
PointA[1].Action=SelectionActionType.Remove
PointA[1].EntityType = SelectionType.GeoEdge
PointA[1].Criterion = SelectionCriterionType.LocationX
PointA[1].Operator = SelectionOperatorType.GreaterThan #CHECKK
PointA[1].Value=Quantity(950, "mm")
PointA.Add(None)
PointA[2].Action=SelectionActionType.Remove
PointA[2].EntityType = SelectionType.GeoEdge
PointA[2].Criterion = SelectionCriterionType.LocationY
PointA[2].Operator = SelectionOperatorType.GreaterThan #CHECKK
PointA[2].Value=Quantity(950, "mm")
PointA.Add(None)
PointA[3].Action=SelectionActionType.Remove
PointA[3].EntityType = SelectionType.GeoEdge
PointA[3].Criterion = SelectionCriterionType.LocationZ
PointA[3].Operator = SelectionOperatorType.GreaterThan #CHECKK
PointA[3].Value=Quantity(950, "mm")
PointA.Add(None)
PointA[4].Action=SelectionActionType.Remove
PointA[4].EntityType = SelectionType.GeoEdge
PointA[4].Criterion = SelectionCriterionType.LocationY
```

```
PointA[4].Operator = SelectionOperatorType.LessThan
PointA[4].Value=Quantity(100, "mm")
PointA.Add(None)
PointA[5].Action=SelectionActionType.Remove
PointA[5].EntityType = SelectionType.GeoEdge
PointA[5].Criterion = SelectionCriterionType.LocationZ
PointA[5].Operator = SelectionOperatorType.LessThan
PointA[5].Value=Quantity(100, "mm")
ns.Generate()
mesh_1 = mesh.AddSizing()
Name = "Scaffold"
mesh_1.Location = ExtAPI.DataModel.GetObjectsByName(Name)[0]
mesh_1.Type = SizingType.NumberOfDivisions
mesh_1.NumberOfDivisions = 1
mesh_2 = mesh.AddSizing()
Name = "Mesh_Zone"
mesh_2.Location = ExtAPI.DataModel.GetObjectsByName(Name)[0]
mesh_2.Type = SizingType.NumberOfDivisions
mesh_2.NumberOfDivisions = 10
 sel = model.AddNamedSelection()
sel.Name = "TOP_FACE"
sel.ScopingMethod = GeometryDefineByType.Worksheet
PointA = sel.GenerationCriteria
PointA.Add(None)
PointA[0].EntityType = SelectionType.MeshNode
PointA[0].Criterion = SelectionCriterionType.LocationY
PointA[0].Operator = SelectionOperatorType.Equal #CHECKK
PointA[0].Value=Quantity(1050, "mm")
sel.Generate()

sel = model.AddNamedSelection()
sel.Name = "BOTTOM_FACE"
sel.ScopingMethod = GeometryDefineByType.Worksheet
PointA = sel.GenerationCriteria
PointA.Add(None)
PointA[0].EntityType = SelectionType.MeshNode
PointA[0].Criterion = SelectionCriterionType.LocationY
PointA[0].Operator = SelectionOperatorType.Equal #CHECKK
PointA[0].Value=Quantity(0, "mm")
sel.Generate()

sel = model.AddNamedSelection()
sel.Name = "FRONT_FACE"
sel.ScopingMethod = GeometryDefineByType.Worksheet
PointA = sel.GenerationCriteria
PointA.Add(None)
PointA[0].EntityType = SelectionType.MeshNode
PointA[0].Criterion = SelectionCriterionType.LocationX
PointA[0].Operator = SelectionOperatorType.Equal #CHECKK
PointA[0].Value=Quantity(1050, "mm")
sel.Generate()

sel = model.AddNamedSelection()
sel.Name = "BACK_FACE"
sel.ScopingMethod = GeometryDefineByType.Worksheet
PointA = sel.GenerationCriteria
PointA.Add(None)
PointA[0].EntityType = SelectionType.MeshNode
PointA[0].Criterion = SelectionCriterionType.LocationX
PointA[0].Operator = SelectionOperatorType.Equal #CHECKK
PointA[0].Value=Quantity(0, "mm")
sel.Generate()

sel = model.AddNamedSelection()
sel.Name = "BOUNDARY"
sel.ScopingMethod = GeometryDefineByType.Worksheet
PointA = sel.GenerationCriteria
PointA.Add(None)
PointA[0].EntityType = SelectionType.MeshNode
PointA[0].Criterion = SelectionCriterionType.LocationZ
PointA[0].Operator = SelectionOperatorType.Equal #CHECKK
PointA[0].Value=Quantity(1050, "mm")
PointA.Add(None)
PointA[1].EntityType = SelectionType.MeshNode
PointA[1].Criterion = SelectionCriterionType.LocationZ
```

```
PointA[1].Operator = SelectionOperatorType.Equal
PointA[1].Value=Quantity(0, "mm")
sel.Generate()

mesh.GenerateMesh()

#Define fixed support
FS = analysis.AddFixedSupport()
FS.Location = DataModel.GetObjectsByName("BOUNDARY")[0]
```