DEVELOPMENT OF A THREE-DIMENSIONAL MESH GENERATOR WITH ANALYTICAL MESH SENSITIVITIES FOR USE IN GRADIENT-BASED SHAPE OPTIMISATION

by

Campbell A. Bam

Submitted in partial fulfillment of the requirements for the degree

Master of Engineering (Mechanical Engineering)

in the

Department of Mechanical and Aeronautical Engineering

Faculty of Engineering, Built Environment and Information Technology

UNIVERSITY OF PRETORIA

2020

SUMMARY

Supervisor:	Prof. Daniel N. Wilke (University of Pretoria, Department of Mechanical and Aeronautical Engineering)
Co-supervisor:	Prof. Schalk Kok (University of Pretoria, Department of Mechanical and Aeronautical Engineering)
Department:	Mechanical and Aeronautical Engineering
Institution:	University of Pretoria
Degree:	MEng (Mechanical Engineering)

Keywords:	Mesh, Meshing, Shape Optimisation, Sensitivity analysis, Analytical Sensitivities, MPCs Multi-point constraints

Structural shape optimisation is a field that has been studied since early on in the development of finite element methods. The sub-fields of shape and topology optimisation are continuously growing in industry and aim to leverage the benefits of technologies such as 3D printing and additive manufacturing. These fields are also being used to optimise designs to improve quality and reduce cost.

Gradient-based optimisation is well understood as an efficient method of obtaining solutions. In order to implement gradient-based optimisation methods in the context of structural shape optimisation, sensitivities describing the change of the domain stiffness are required. To obtain the stiffness sensitivities, mesh deformation sensitivities are required. In this study, a mesh generating method is developed that provides mesh deformation sensitivities.

For shape optimisation it is advantageous to employ an optimisation algorithm that allows for the manipulation of CAD geometry. This means that the CAD geometry is finalised upon completion of the optimisation process. This, however, necessitates the calculation of accurate sensitivities associated with non-linear geometries, such as NURBS (those present in CAD), by the mesher.

The meshing method developed in this study is analogous to a linear truss system. The system is solved for static equilibrium through a geometrically non-linear finite element analysis using Newton's method. Sensitivities are made available by Newton's method for use in generating mesh sensitivities for the system.

It is important for the mesher to be able to accurately describe the geometrical domain which approximates the geometry being modelled. To do so, nodes on the boundary may not depart from the boundary. Instead of prescribing all boundary nodes, this mesher frees the boundary nodes to move

along, but not away from the boundary. This is achieved using multipoint constraints since they allow for an analytical relationship between boundary node movement and the boundary.

Two multipoint constraint (MPC) methods are investigated for boundary discretisation, namely, the Lagrangian and master-slave elimination methods (MSEM). The MSEM presents several difficulties in obtaining convergence on non-linear boundaries in general when compared to the Lagrangian method.

The MSEM has reduced computational requirements for a single Newton step, especially when direct solvers are used. However, when indirect solvers are implemented the time difference between the two MPC methods reduces significantly. For a “medium” curvature geometry the Lagrangian implementation has only a 6% time penalty.

The Lagrangian method is selected as the preferred MPC method for implementation in the mesher to avoid the convergence problems associated with the MSEM. This is justified on the basis of reliability outweighing the 6% time penalty for what is intended to be a tool in the shape optimisation process.

Analytical sensitivities are obtained for the truss system in order to account for the MPC boundaries. The analytical mesh sensitivities are proven to be accurate through comparison with numerical sensitivities. The method is demonstrated to be able to accurately described the mesh deformation throughout the domain for both uniform and non-uniform meshes in the presence of non-linear boundaries.

ACKNOWLEDGEMENTS

My deep-felt gratitude and thanks go to the following people:

- To my supervisors Professor Daniel Wilke and Professor Schalk Kok, University of Pretoria, Department of Mechanical and Aeronautical engineering. Thank you for the support and patience you have offered during this study. Your enthusiasm was infectious, making this a fun and enlightening experience.

- To my general manager Dr. Justin Martens for his encouragement and proofreading of the final draft. I appreciate the time taken out of your busy schedule to assist me.

- To my manager Tyrone Legg for his flexibility with my work schedule in order to assist me with finishing this degree.

- To my wife Tamsin, for your support and encouragement. I thank you for the time spent reading and checking my report, researching, and helping me track down papers. I could not have done this without you.

- To my parents Peter and Faith, for proofreading and grammar checking my initial work.

TABLE OF CONTENTS

LIST OF APPENDICES

LIST OF FIGURES

LIST OF FIGURES IN APPENDICES

LIST OF TABLES

LIST OF TABLES IN APPENDICES

ACRONYMS

CAE	Computer-Aided Engineering
DOF	Degree of Freedom
FEA	Finite Element Analysis
FEM	Finite Element Method
MSEM	Master-Slave Elimination Method
PDEs	Partial Differential Equations

VARIABLES

A	Symmetric invertible matrix
AR	Aspect ratio
B	Full rank matrix
C	Constant
$\mathbb{C}(\mathbf{x})$	Optimisation cost function
e	Natural number
$\boldsymbol{e}_i$	i^{th} unit vector
$\boldsymbol{F}$	Force
F_{scale}	Force scaling factor (used in DistMesh)
F_{truss}	Force exerted by truss
F_{int}	Internal force contribution due to truss
$\mathcal{F}^{int}$	Set of internal forces ($\mathcal{F}_p^{int}, \mathcal{F}_f^{int}, \mathcal{F}_m^{int}, \mathcal{F}_s^{int}, \mathcal{F}_{d\Omega}^{int}$)
$\mathcal{F}^{ext}$	Set of external forces ($\mathcal{F}_p^{ext}, \mathcal{F}_f^{ext}, \mathcal{F}_m^{ext}, \mathcal{F}_s^{ext}, \mathcal{F}_{d\Omega}^{ext}$)
f_{mpc}	MPC function of boundary function in the form of ($f_{mpc} = 0$)
f_{ms}	Master-slave MPC function
$f_{\partial\Omega}(x)$	Boundary function
$\mathbf{f}$	External forces associated with FEA
$\mathbf{f}_f$	External loads on the free DOFs
$\mathbf{f}_p$	External loads plus reactions on the prescribed DOFs (FEA)
$\mathbf{g}(\mathbf{x})$	Optimisation constraint equations
$h(\ \ )$	Desired truss length function
I	Identity matrix
K	Consistent tangent contributions ($K_{ff}, K_{fm}, K_{mf}, K_{mm}$)
$\mathbf{K}$	Stiffness matrix of linear static FEA ($\mathbf{K_{ff}}$, $\mathbf{K_{fp}}$, $\mathbf{K_{pf}}$, $\mathbf{K}_{pp}$)
k	Spring stiffness
k'_{truss}	Truss stiffness factor
$\boldsymbol{L}_c$	Current truss length
$\boldsymbol{L}_{des}$	Desired truss length
$\boldsymbol{L}_0$	Initial seeding length of mesh

L_0	Initial length
ΔL	Difference length ($\Delta L = L_{des} - L_c$)
L_e^*	Denominator length placeholder
$L_{longest}$	Longest edge length
$L_{shortest}$	Shortest edge length
l, m, n	Unit vector components for a given truss
$\mathcal{L}$	Lagrangian functional
m, n	Highest integer in a set
$\boldsymbol{P}$	Linearised coefficient matrix of the master-slave relationship
p	Node/coordinate
p	Node
p_{mid}	Midpoint of truss
p_{cent}	Centroid of simplex
$\boldsymbol{R}$	Reaction forces ($\boldsymbol{R_p}, \boldsymbol{R_f}, \boldsymbol{R_m}, \boldsymbol{R_s}$)
$\mathcal{R}$	Residual equation ($\mathcal{R}_p, \mathcal{R}_f, \mathcal{R}_m, \mathcal{R}_s, \mathcal{R}_{d\Omega}$)
$\mathbb{R}^{\mathrm{n}}$	Set of real numbers
$r_{inscribed}$	Inscribed radius of a simplex
$r_{circumscribed}$	Circumscribed radius of a simplex
S_1	Truss exponential scaling factor
s, t	RHS contributions in saddle point problem
t	Time, or step size
$\boldsymbol{u}$	Mesh nodal coordinate update $\boldsymbol{u} = \Delta\boldsymbol{\mathcal{X}}$
$\mathbf{u}$	Nodal displacements (solution to FEA)
$\mathbf{u}_f$	Nodal displacements associated with the free DOFs (solution to FEA)
$\mathbf{u}_p$	Nodal displacements associated with the prescribed DOFs (FEA)
v, w	Generic problem solution for saddle point problem
W	Preconditioner matrix for a saddle point solution
$\boldsymbol{\mathcal{X}}$	Mesh nodal coordinates
$\boldsymbol{\mathcal{X}}^i$	Mesh nodal coordinates, initial state
$\boldsymbol{\mathcal{X}}^e$	Mesh nodal coordinates, end state $\boldsymbol{\mathcal{X}}^e = \boldsymbol{\mathcal{X}}^i + \boldsymbol{u}$
$\mathcal{X}'$	Place holder for nodal coordinates and Lagrangian multipliers
$\mathbf{x}$	Control parameter
x	Local coordinates ($x_1, x_2, x_3, x_4, x_5 x_6$)
X, Y, Z	Cartesian dimensions
γ	Preconditioner scaling factor for saddle point preconditioner
ϵ	Strain
$\boldsymbol{\lambda}$	Lagrange multiplier
Π	System potential energy
Ω^*	Geometrical domain
$\partial\Omega$	Geometrical domain boundary

Ω	Geometrical domain interior

SUBSCRIPTS

c	Constraint
f	Free DOFs
m	Master DOFs
p	Prescribed DOFs
s	Slave DOFs

SUPERSCRIPTS

f	Free DOFs
m	Master DOFs
p	Prescribed DOFs
s	Slave DOFs
T	Transpose

CHAPTER 1
INVESTIGATION

"Cost, Quality, Deadline: Pick one" – Someone knowledgeable, to a client

1.1. Introduction

Shape optimisation is an important and popular field of design that has been investigated for many years [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. With the ongoing development of technologies such as 3D printing and additive manufacturing, many constraints of traditional manufacturing methods are removed and new approaches to product development are allowed. Shape and topology optimisation are growing fields in industry that are aiming to leverage the benefits of these types of technologies, as well as provide tools to improve existing manufacturing/design methods [11, 12].

There are three general categories in the structural shape optimisation field [6]:

1. Sizing optimisation – optimisation of a nominal dimension of a structure or component
2. Topology optimisation – optimisation of the overall shape and topology of a structure
3. Shape optimisation – optimisation of overall shape without changing topology, sometimes called boundary variation or boundary sensitivity optimisation

Sizing, shape, and topology optimisation may be applied to complex geometries with correspondingly complex displacement, strain, and stress fields. Since the performance of structures is typically defined by criteria relating to the strain and stress, the Finite Element Method (FEM) is used to transfer the continuum into a linear algebraic form. To achieve this the domain is discretised using a mesh and the displacements, strains, and stresses corresponding to each element are then solved using linear algebra. This returns an approximation to the continuum solution.

Sizing optimisation was the focus of the first two decades of structural shape optimisation. It required only the variation in the cross-section or thickness of a component which requires no change in the finite element model (FEM) discretisation when structural member elements are used, only variation in values [13].

Shape and topology optimisation deal with effects associated with the changes of the geometry and are quite closely linked. Topology optimisation aims to determine the optimal layout of material within an allowable space given certain boundary conditions. Methods such as level-set [2] use domain descriptors that allow for comparatively simple manipulation of the shape in the context of the finite element model. The level-set method approximates the geometry through the use of level sets which allows for the geometry to be analysed in discrete layers [2]. Hard-kill/soft-kill methods manipulate the topology of the FEM at the element level. This is performed in a fixed grid layout. Hard-kill methods set

an element's stiffness to either 0 or 1 dictating its absence from or presence in the model, whereas soft-kill methods scale the stiffness of an element between 0 and 1 [8, 14]. Homogenising methods introduce microvoids into a material and the objective is to obtain mean compliance of the structure with a constraint typically applied on the volume fraction [15]. All of these methods are well suited to obtaining optimal topologies.

Having obtained an optimal topology, the more traditional refinement of the geometry is done through shape optimisation or boundary variation methods. Boundary variation methods directly manipulate the geometric domain through design variables to meet certain criteria [5]. These methods rely on an accurate description of the boundary (often B-splines [5]) that is directly manipulated. Adaptive growth techniques aim to homogenise the stress state on the surface of the structure for their main loads. This is achieved through thermally loading a thin (low elastic modulus) boundary layer based on stress results from the previous analysis. The FEM geometry is then modified in accordance with the growth of this boundary layer [16]. Whilst the adaptive growth technique returns a refined shape, it does not lend itself to geometry manipulation using design variables.

The discrete layers in level-set techniques are each defined by a closed-boundary function. These functions may, in turn, be parameterised to a set of design variables. The level-set functions are then optimised iteratively until a solution is found. The relative simplicity of manipulating a 3D geometry in 2D layers gives the level-set method sufficient flexibility to be extended from the topology optimisation to the shape refinement stage [2].

To perform shape optimisation, the geometry needs to be parameterised with respect to a set of control variables. Ideally, this should be done at the level of the computer-aided design (CAD). However, it is important to note that some shape optimisation methods use simplified geometry descriptions to reduce complexity in parametrisation with respect to a set of design variables and manipulation of the geometrical domain, such as the use of a level-set function. To further complicate matters FEM mesh geometry only approximates the geometry [17]. It therefore follows that methods that manipulate the FEM geometry or simplified geometric descriptions do not feed directly back into the CAD environment. Isogeometric analysis employs a FEM geometry that is identical to the CAD geometry by using elements derived from NURBS [17]. Most FEM packages do not support the isogeometric descriptions. For shape optimisation methods to be practicable in industrial applications, this loop needs to be closed. Therefore, since boundary variation methods can be designed to parametrise the CAD, they are well suited to the shape refinement phase or design optimisation.

Analysis in shape optimisation refers to the minimisation of the cost function given by the solution of a system of partial differential equations (PDEs) constructed with a parametrised geometrical domain [1, 7]. The continuum associated with most problems is complex and not easily solved. Discretisation is a flexible and efficient method of reducing continuum problems to a simpler form requiring standard linear algebra. The main disadvantage meshes pose is that their accuracy is dependent on the quality

of the discretisation strategy employed. For boundary variation methods, the geometrical domain is described using a computational grid, the evolution of which is controlled in a straightforward manner using the control variables.

The strategy employed to manage the continuum discretisation can be handled with several approaches: Fixed grid strategies as used with hard-kill/soft-kill, immersed boundary [10] and evolutionary methods [8, 14]; adaptive meshes that allow for geometrical changes without change in element connectivity; multiple mesh strategies where multiple fixed-grid domains are overlaid and coupled using methods such as Nitche equations [18] and remeshing strategies where the domain is remeshed to retain accuracy in the analysis. Generally, any of these discretisation strategies can be used to conduct shape optimisation.

Discretisation or meshing of the domain contributes significantly to the overall cost of shape optimisation and this can make remeshing strategies prohibitively expensive. This can also introduce numerically induced local minima in the objective function [2]. The large expense associated with remeshing strategies can be offset by conducting local remeshing. Further improvements are realised when the strategy facilitates the provision of exact analytical gradients for use in gradient-based optimisation methods [7]. The introduction of local minima can be overcome using restart and h-refinement strategies [7].

Several approaches are available for capturing boundary deformation in the FEM mesh. The averaging techniques move nodes through the domain by sweeping the boundary deformation effect into the domain in an iterative manner. These are comparatively simple to implement but can result in element distortion around the boundaries where large deformations have occurred [1], and many iterations may be required to reduce this effect.

Function approximation approaches such as radial basis functions (RBFs) [19] or polynomial interpolation to describe the mesh deformation can be prohibitively expensive as they are dependent on the n boundary nodes and exhibit $O(n^3)$ computational cost/time complexity [20]. These methods have been successfully implemented in commercial packages such as Ansys for shape optimisation. Alternatively, methods that depend on an underlying physical principle to accomplish the mesh deformation (such as linear truss systems [7] or torsional springs [21]) allow for deformation throughout the domain with a potentially lower cost of implementation, especially where many design variables are present.

To perform optimisation of the geometry, the geometry needs to be parametrised. This is accomplished with respect to a set of design/control variables in a manner such that meaningful manipulation of the geometry is permitted. These design variables are then the subject of the optimisation process where the cost function is minimised subject to design requirements.

A parametrised geometrical domain that can be meshed and analysed requires optimisation to be applied iteratively to find the best design due to the non-linear nature of the shape optimisation problem. Optimisation methods are well documented [22] and have varying benefits and difficulties associated with them. These come in three general categories: zero-order (nature-inspired/evolutionary and gradient-free), first-order (steepest descent), and second-order (Newton's, quasi-Newton's, and conjugate gradient method) methods.

Nature-based methods such as genetic algorithms and evolutionary strategies can optimise discrete problems and have an advantage in their global optimisation capability [22]. They can, however, be computationally very expensive. Evolutionary methods have been successfully employed by Xie *et al.* [9] in their fixed grid evolutionary strategy.

Gradient-free methods have the advantage of only requiring evaluation of the cost function. This makes them well suited for general problems but with the disadvantage of being computationally expensive and having the tendency to provide inferior solutions for problems with a large number of design variables when compared to solutions obtained using gradient-based methods [22].

First and second-order gradient-based methods have the disadvantage of requiring additional sensitivity information. Strategies available for obtaining sensitivity information vary in complexity, accuracy and computational expense. Their main advantage is the computational efficiency to solve optimisation problems [7]. They are limited to obtaining the solution to the local minima and can converge to saddle points unless sensible restart strategies are employed [2, 7]. However, the objective of shape optimisation/refinement is to obtain the optimum overall shape, or the local minima, associated with a given topology. For this application, gradient-based methods are well suited [1, 7, 22]. The second-order gradient (Hessian) is typically not readily available but is rather approximated using the first-order gradient information, as with the Quasi-Newton methods [22].

Structural design sensitivity analysis has been studied since at least the 1980s [23, 24] and has been effectively implemented in various strategies such as level-set [2], immersed boundary [25], and boundary variation [7, 26]. Obtaining the sensitivities for the design requires mesh sensitivities. This will be discussed in more detail in §1.2, §1.3, and Chapter 5.

As has been demonstrated, structural shape optimisation is a broad field that includes various approaches. The various geometry descriptions and discretisation methods coupled with the optimisation methods ultimately determine the effectiveness of an algorithm in obtaining an optimised solution.

As a whole, shape optimisation is a computationally demanding process and it is therefore important to make use of efficient techniques to improve solution times. This study is focussed on obtaining accurate mesh deformation sensitivities for use in gradient-based shape optimisation techniques. This is specifically an extension of the mesher, and analytical mesh sensitivities work of Wilke *et al.* [7]

which used piecewise linear boundaries to define the domain. Two other limiting factors associated with the Wilke *et al.* [7] method, was that the boundary nodes were prescribed along the piecewise linear boundary and that the method was restricted to two dimensions.

This study's focus is on creating a simple 2D and 3D automatic mesh generator for unstructured tetrahedral meshes that permits the use of unprescribed non-linear boundary descriptions such as NURBS (those present in CAD) that produces accurate mesh movement sensitivities for use with gradient-based shape optimisation methods.

1.2. Overview of the shape optimisation process

The shape optimisation process begins with defining the problem that needs to be solved. The boundary conditions associated with the problem as well as the state equations must be identified and the general working domain initialised as seen in Figure 1 (a). Following this, an initial general shape can be determined by making use of topology optimisation techniques as in Figure 1 (b) [1, 6].

The initial design is parameterised per (c) to allow shape change to be controlled programmatically. The geometry is then meshed and analysed per (d), subject to a set of conditions as in (a). Based on the results of the analysis, the parameters and shape are updated. This process is repeated using optimisation concepts until no change in the shape can be made that results in appreciable improvement of the results of the analysis. The resulting shape is then considered optimal for the set of conditions. This process is represented schematically in Figure 1.

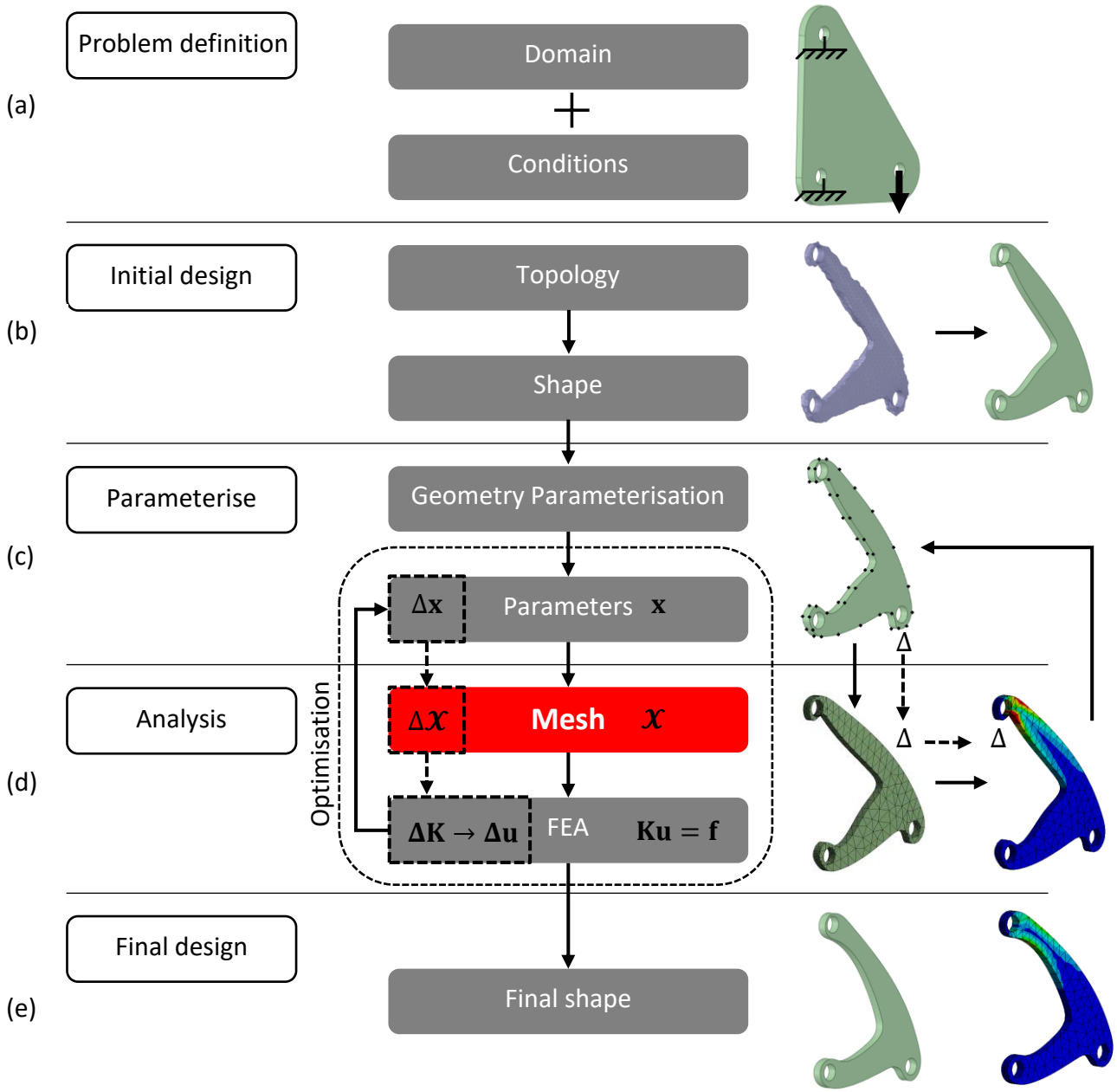


Figure 1 Shape optimisation process

When using gradient-based optimisation methods, the sensitivities need to accurately describe the effect a small change in the parameters has on the state of the system. For the process shown in Figure 1, knowledge of the changes in the FEA results $\mathbf{u}$ is required given a change in the parameters $\mathbf{x}$ defining the geometry. It is seen that the mesh (discretisation) process lies between the parametrised geometry and the finite element analysis (FEA). This implies that prediction of the change in analysis results $\mathbf{u}$ for a given change in geometry requires known sensitivities for the analysis results with respect to the mesh and the mesh with respect to the parameters. This is shown by the use of the Δ symbols in dashed boxes. The specifics of these relationships are detailed in §1.3 in the context of linear static FEM.

1.2.1. Geometry domain parametrisation and discretisation

The computational domain Ω^* is subdivided into the interior Ω and boundary $\partial\Omega$. This is shown in Figure 2 (a) and given as:

$$\Omega^* = (\Omega \cup \partial\Omega) \tag{1}$$

Domain boundary is defined by parameters $\mathbf{x}$ (shown as green dots in (b)) varying in complexity from a parameter defining a thickness to the control point locations and weights of a set of Non-Uniform Rational Basis Splines (NURBS[1]). Exactly which aspects the parameters control is immaterial; what is required is an understanding of their effect on the domain boundary and correspondingly the interior and the mesh.

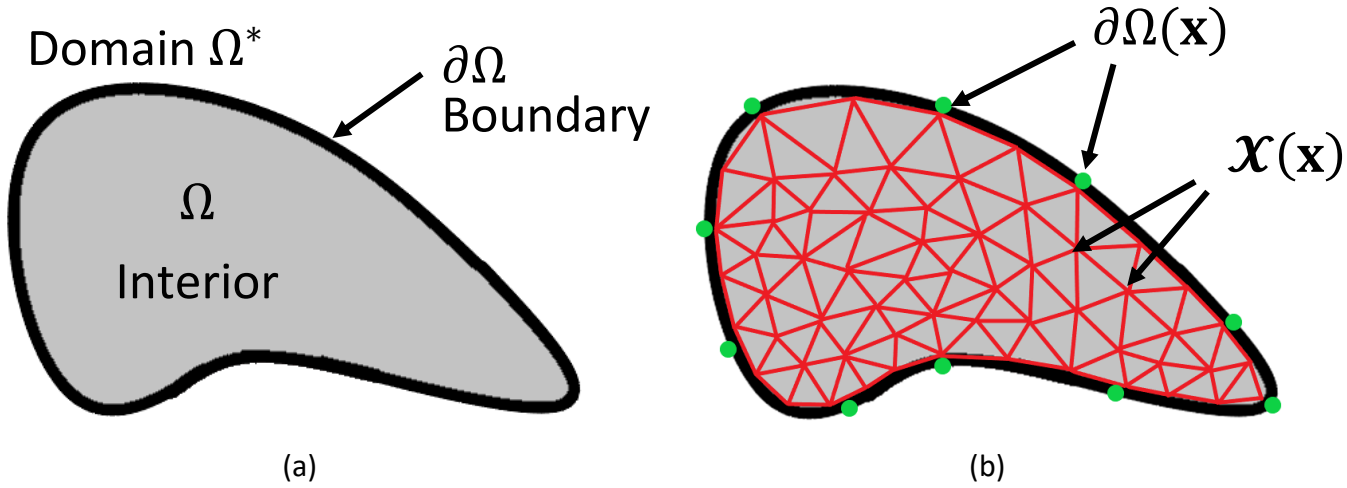


Figure 2 Domain definition
(a) Original domain (b) Discretised domain

Figure 2 (b) shows the domain discretised using a mesh. It is important to note that the mesh is unable to exactly represent the boundary, however reducing the size of the discretisation will improve how well the domain is represented. The mesh is a function of the domain Ω^* and therefore a function of the parameters $\mathbf{x}$ as follows. The mesh representation of the geometry is therefore:

$$\boldsymbol{\mathcal{X}}\left(\boldsymbol{\Omega}^*(\mathbf{x})\right) = \boldsymbol{\mathcal{X}}(\mathbf{x}) \tag{2}$$

[1] NURBS are used to define geometry form in most CAE packages

1.3. Gradient-based shape optimisation in the context of linear static FEA

1.3.1. Cost function and linear static FEA

Given the set of control variables, $\mathbf{x}$, the cost function describing the shape optimisation problem is stated as follows for the general inequality constrained problem: given the cost function $\mathbb{C}\big(\boldsymbol{\mathcal{X}}(\mathbf{x})\big)$, find the minimum $\mathcal{F}^*$ such that a set of constraints $\mathbf{g}(\boldsymbol{\mathcal{X}}(\mathbf{x}), \mathbf{x}) \leq \mathbf{0}, \mathbf{g} \in \mathbb{R}^n$ is satisfied:

$$\mathbb{C}^* = \mathbb{C}\big(\boldsymbol{\mathcal{X}}(\mathbf{x})\big) = \min_{\mathbf{x}\epsilon\mathbb{R}^n} \left\{\mathbb{C}\big(\boldsymbol{\mathcal{X}}(\mathbf{x})\big) : \mathbf{g}(\boldsymbol{\mathcal{X}}(\mathbf{x}), \mathbf{x}) \leq \mathbf{0}\right\} \tag{3}$$

where $\mathbb{R}^n$ represents the n-dimensional space of real numbers.

The cost function $\mathbb{C}\big(\boldsymbol{\mathcal{X}}(\mathbf{x})\big)$ and the constraints $\mathrm{g}_j(\boldsymbol{\mathcal{X}}(\mathbf{x}), \mathbf{x})$, for $j = 1,2,\dots,m$ are scalar functions of the control variables $\mathbf{x}$. For the sake of brevity, the cost and constraint functions will be denoted by $\mathbb{C}(\mathbf{x})$ and $\mathbf{g}(\mathbf{x})$ respectively. The mesh $\boldsymbol{\mathcal{X}}(\mathbf{x})$ is a function of the geometrical domain $\Omega^*(\mathbf{x})$ as shown in §1.2.1. The geometrical domain, $\Omega^*(\mathbf{x})$, is selected as smooth and continuous between fixed points and is also a function of the control variables $\mathbf{x}$.

For simplicity's sake, the shape optimisation problem will be considered subject to the solution of a linear static FEA on the domain. The cost function $\mathbb{C}(\mathbf{x}) = \mathbb{C}\left(\mathbf{u}\big(\boldsymbol{\mathcal{X}}(\mathbf{x})\big)\right)$ is, therefore, an explicit function of the nodal displacements, $\mathbf{u} = \mathbf{u}\big(\boldsymbol{\mathcal{X}}(\mathbf{x})\big)$, which are solved directly from the equilibrium equations for linear static finite element methods:

$$\mathbf{K}\mathbf{u} = \mathbf{f} \tag{4}$$

where $\mathbf{K}$ is the stiffness matrix, $\mathbf{u}$ are the nodal displacements and $\mathbf{f}$ are the external forces. For analysis, the system is partitioned into the free $\mathbf{u}_f$ and prescribed $\mathbf{u}_p$ degrees of freedom (DOFs), such that (4) can be rewritten as:

$$\begin{bmatrix}\mathbf{K}_{ff} & \mathbf{K}_{fp} \\ \mathbf{K}_{pf} & \mathbf{K}_{pp}\end{bmatrix}\begin{Bmatrix}\mathbf{u}_f \\ \mathbf{u}_p\end{Bmatrix} = \begin{Bmatrix}\mathbf{f}_f \\ \mathbf{f}_p\end{Bmatrix} \tag{5}$$

Since the prescribed DOFs $\mathbf{u}_p$ are known, (5) can be rewritten to solve for the unknown DOFs $\mathbf{u}_f$:

$$\mathbf{K}_{ff}\mathbf{u}_f = \mathbf{f}_f - \mathbf{K}_{fp}\mathbf{u}_\mathrm{p} \tag{6}$$

1.3.2. Obtaining the sensitivities for the shape optimisation problem

Since the cost function $\mathbb{C}(\mathbf{x})$ is an explicit function of the nodal displacements $\mathbf{u}$, the structural response of $\mathbf{u}$ with respect to the control variables $\mathbf{x}$ is required to implement a gradient-based shape

optimisation method as in [7, 27]. Since the stiffness, force and the resulting displacement are all functions of the design variables $\mathbf{x}$, (6) can be written as $\mathbf{K}_{ff}(\mathbf{x})\mathbf{u}_f(\mathbf{x}) = \mathbf{f}_f(\mathbf{x}) - \mathbf{K}_{fp}(\mathbf{x})\mathbf{u}_\mathrm{p}(\mathbf{x})$

The analytical gradient $d\mathbf{u}/d\mathbf{x}$ is obtained by taking the derivative of (6) with respect to $\mathbf{x}$:

$$\mathbf{K}_{ff}\frac{d\mathbf{u}_f}{d\mathbf{x}} = \frac{d\mathbf{f}_f}{d\mathbf{x}} - \frac{d\mathbf{K}_{fp}}{d\mathbf{x}}\mathbf{u}_\mathrm{p} - \mathbf{K}_{fp}\frac{d\mathbf{u}_\mathrm{p}}{d\mathbf{x}} - \frac{d\mathbf{K}_{ff}}{d\mathbf{x}}\mathbf{u}_f \tag{7}$$

For a Dirichlet boundary condition, the external forces $\mathbf{f}$ and the prescribed displacements $\mathbf{u}_p$ are known constants (typically zero) and not part of the control variable set $\mathbf{x}$, hence $d\mathbf{u}_\mathrm{p}/d\mathbf{x} = d\mathbf{f}/d\mathbf{x} = \mathbf{0}$. Selecting $\mathbf{u}_p = \mathbf{0}$, (7) then reduces to:

$$\mathbf{K}_{ff}\frac{d\mathbf{u}_f}{d\mathbf{x}} = -\frac{d\mathbf{K}_{ff}}{d\mathbf{x}}\mathbf{u}_f \tag{8}$$

To obtain the gradient $d\mathbf{u}_f/d\mathbf{x}$ the derivative term $\frac{d\mathbf{K}_{ff}}{d\mathbf{x}}$ is required. Since the stiffness matrix $\mathbf{K}$ is a function of the nodal coordinate set $\boldsymbol{\mathcal{X}}$, which in turn is a function of the control variables $\mathbf{x}$, $\mathbf{K}_{ff} = \mathbf{K}_{ff}\big(\boldsymbol{\mathcal{X}}(\mathbf{x})\big)$. Therefore, the derivative can be expanded using the chain rule and computed from:

$$\frac{d\mathbf{K}_{ff}}{d\mathbf{x}} = \frac{d\mathbf{K}_{ff}}{d\boldsymbol{\mathcal{X}}}\frac{d\boldsymbol{\mathcal{X}}}{d\mathbf{x}} \tag{9}$$

The component $\frac{d\mathbf{K}_{ff}}{d\boldsymbol{\mathcal{X}}}$ is obtained by analytical differentiation of the stiffness matrix with respect to the mesh nodal coordinates $\boldsymbol{\mathcal{X}}$. This action can be completed as detailed by Wilke *et al.* [7] or Olaf *et al.* [27]. The sensitivities of the domain nodal coordinates with respect to the control variables $d\boldsymbol{\mathcal{X}}/d\mathbf{x}$ - also referred to as the mesh sensitivities - may be obtained analytically, numerically, or semi-analytically.

It is therefore clear that the accuracy of FEM displacement sensitivity $\frac{d\mathbf{u}_f}{d\mathbf{x}}$ is dependent on the accuracy of the mesh sensitivities $\frac{d\boldsymbol{\mathcal{X}}}{d\mathbf{x}}$. It is thus imperative for an efficient shape optimisation method to have accurate mesh sensitivities available. Obtaining $\frac{d\boldsymbol{\mathcal{X}}}{d\mathbf{x}}$ will be detailed in Chapter 5.

1.4. Mesh generation

Discretisation of the domain, meshing, is required in FEM to transform the continuum problem into a set of PDEs that can then be solved using a linear algebra problem. This is an approximation whose accuracy is directly dependent on the quality of the mesh.

1.4.1. What is a mesh

A mesh is the representation of a surface, volume, or n-dimensional domain using such finite elements. It defines the connectivity between adjacent elements and completely describes the domain.

1.4.2. Mesh generation

The meshing process is a time-consuming and error-prone process. This is particularly true for the practical sciences and engineering where arbitrary three-dimensional geometries of varying levels of complexity need to be meshed. Attempts have been made to create fully automatic mesh generators for complex bodies since the early 1970s [28]. These have since grown in complexity and efficiency. An example of a mesh, generated using Hypermesh™, is shown in Figure 3.

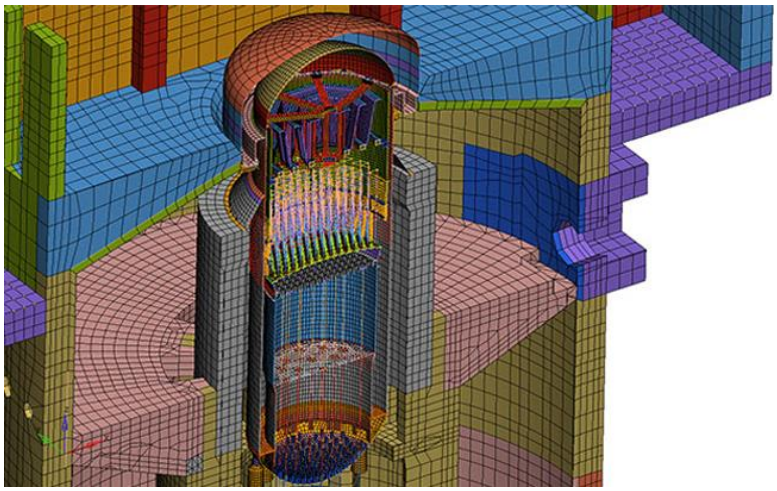


Figure 3 3D mesh cross-section of nuclear reactor created using HyperMesh™ [29]

Mesh generation for analysis must be differentiated from that required for shape optimisation. In analysis, the boundary is fixed, the initial mesh is generated and either retained as is or optimised by the user to meet specific requirements. A change in geometry requires remeshing.

The boundary changes required by shape optimisation are accommodated using either mesh movement through a mapped mesh with initial connectivity essentially remaining as-is; or remeshing with the opportunity of optimising the mesh.

Remeshing introduces non-smooth behaviour to the cost function while mesh movement is usually smooth. Optimisation benefits from smooth functions.

1.4.3. Desirable mesh attributes

Discretised analyses such as FEA and CFD benefit from the mesh having certain attributes. Meshes that are highly distorted cause numerical inaccuracies; meshes that are too coarse might cause information within the domain to be missed thereby giving poor or incorrect results; too fine a mesh and the system can become too cumbersome and numerically expensive to solve reasonably.

The mathematics and programming developed for FEA uses a reference element or set thereof to perform the repeated calculations required to represent the domain. These reference elements are idealised versions of their given shape and are typically created as equal-sided and equiangular. These

idealisations are typically of unit or twice unit size [30, 31]and are located neatly around a local axis system for easy reference.

The transformation of the actual element from/to the reference domain requires a mapping of information from the analysis domain to the reference domain and vice versa. This results in a distortion of information. Calculations are therefore performed in the reference domain for simplicity and the results are mapped back. This process requires numerical integration of irrational functions, which can only be approximately resolved using Gauss quadrature [31]. This introduces an integration error in addition to the already existing discretisation error resulting from utilizing a mesh. When the element in the analysis domain is geometrically similar[2] to the element in the reference domain, the denominator reduces to a constant and the polynomial numerator can be exactly integrated using Gauss quadrature.

1.4.3.1. Element quality

Since it is highly unlikely that a geometry or domain can be meshed so that no element is distorted, these numerical inaccuracies are a reality. The effect of these mapping inaccuracies is not severe until large deviations from the reference shape or skewness occur. This property is often identified as *element quality* for which various metrics have been devised to quantify the mesh quality.

The simplest measure is the ratio of two times the inscribed circle's radius $r_{inscribed}$ to the circumscribed circle's radius $r_{circumscribed}$. For ideal elements, this gives a value of 1 and for a completely degenerated element a value of 0. the element quality is therefore given by:

$$Element\ quality = \frac{2 \times r_{inscribed}}{r_{circumscribed}} \tag{10}$$

1.4.3.2. Element aspect ratio

The aspect ratio (AR) of elements can also affect mesh performance. This ratio is defined as the ratio of the longest edge to the shortest edge. The ideal aspect ratio is 1 and occurs when all edges are the same length. High aspect ratio elements do not always result in poor mapping but may instead fail to adequately capture the underlying field along the coarsely discretised directions.

Poor aspect ratio elements may often be used in CFD boundary layers. This is because the change in state in the direction along the boundary is minimal compared to the variations in the perpendicular direction. The use of poor aspect ratio elements under such conditions allows for efficient use of elements in the domain space. Aspect ratio is given by:

[2] Geometry has the same angles and is only scaled in its dimensions

$$AR = \frac{L_{longest}}{L_{shortest}} \quad (11)$$

1.4.3.3. Mesh refinement

Mesh refinement becomes a serious requirement in order to adequately capture the state of a given domain without making the problem so resource-intensive as to make solution times and/or hardware requirements completely impractical. If the underlying field over part of the spatial domain is constant, a single element could, in theory, be used to accurately capture and represent this state. However, if there are significant variations in the field over the spatial domain, elements that are large compared to these changes in the field or geometry will result in inaccurate representation and solution. See §1.5.2 for adaptive mesh refinement.

1.4.3.4. Mesh smoothness

The smoothness of a mesh is a measure of how rapidly element edge sizes vary relative to their neighbouring elements. The smoothness ratio is also referred to as element growth rate which permits a more intuitive understanding of the fact that a neighbouring element grows by at most the specified value. A ratio of 1.2 indicates that adjacent elements have sizes no more than 20% larger or smaller than their neighbouring elements. A balance is however required between mesh smoothness and refinement in order to optimise the mesh for performance and solution accuracy. Whilst having a very fine smoothness has positive benefits in terms of element quality and reduced distortion, it can create a situation where there is an unnecessarily high element density in regions where it is not required. This results in unnecessarily increased computational costs.

1.5. Mesh generator requirements

There are numerous means of obtaining a sufficient mesh and these are well documented in literature [28, 31, 32]. However, access to accurate mesh deformation sensitivity information is not readily available from most meshing methods.

Obtaining accurate sensitivities requires that the entire mesh construction be dictated by an underlying smooth and continuous process of describing the mesh change when a fixed boundary with an initial mesh is proposed. The mesh boundary movement essentially updates the initial positions on the boundary by remeshing (non-smooth and discontinuous) or mesh mapping (smooth and continuous) that is then optimised or improved under a small change of the boundary. The process that improves the initial mesh is important from the sensitivity point of view. An example of a non-smooth process would be Laplacian smoothing [31] where nodes are moved in isolation as part of a mesh optimisation process. An example of a smooth process is treating the mesh as a truss structure where all nodes are optimised simultaneously. This was done in principle in DistMesh [33] (tension components were

ignored, causing some non-smooth behaviour) and extended to be completely smooth by Wilke *et al.* [7].

1.5.1. Meshing method: Structured vs unstructured meshing

Two overarching approaches to meshing a domain exist; structured and unstructured. The difference between structured and unstructured meshes, or grids, is concisely described as follows by Vladimir Liseikin in his book *Grid Generation Methods, Second Edition* [32]:

"There are two fundamental classes of grid popular in the numerical solution of boundary value problems in multidimensional regions: structured and un-structured. These classes differ in the way in which the mesh points are locally organized. In the most general sense, this means that if the local organization of the grid points and the form of the grid cells do not depend on their position but are defined by a general rule, the mesh is considered as structured. When the connection of the neighbouring grid nodes varies from point to point, the mesh is called unstructured. As a result, in the structured case the connectivity of the grid is implicitly taken into account, while the connectivity of unstructured grids must be explicitly described by an appropriate data structure procedure."

Structured meshes are created through the mapping of a reference (usually uniform) grid to the geometry. This is easily done for simple shapes with higher complexity domains causing problems during mesh generation. An example of mapping a uniform cubic grid to a section of a cylinder is shown in Figure 4.

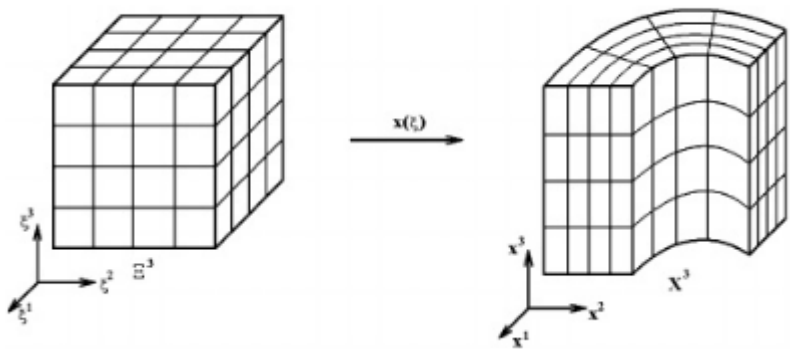


Figure 4 Cylindrical structured grid mapping [32]

Due to the nature of structured meshing, it is not suitable for making local mesh refinement changes programmatically with minimal overall mesh disturbance. Rather, it requires remeshing with a set of new predefined conditions.

Unstructured meshes are considered suitable for discretising domains with complicated shapes. They allow for a more natural approach to local mesh refinement and adaptation through the insertion and removal of local nodes. Furthermore, local element refinement of the mesh can be done by subdividing the elements without requiring reconsideration of any other part of the mesh. Overly refined sections

of the mesh can be altered through the deletion of local nodes and elements. Unstructured methods are therefore the most flexible approach for discretising complicated domains and geometries and in general, will mesh a complicated geometry faster than a structured approach.

Unstructured meshing methods typically use simplexes (triangles[3] for 2D, tetrahedra for 3D) due to the simplicity of discretising a domain with such element geometries. Irrespective of the nodal movement, these shapes can never become concave in their associated domains, which reduces the complexity of meshing. Figure 5 shows the difference between a structured and unstructured mesh on the nose cone cross-section of an aircraft.

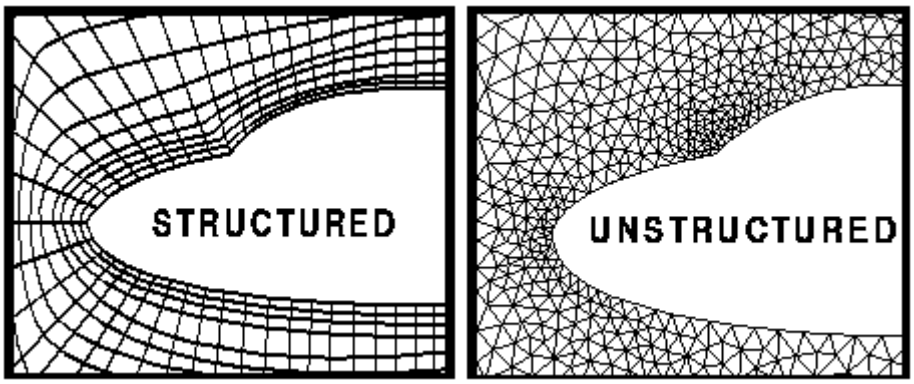


Figure 5 Structured vs unstructured mesh for the same geometry [34]

For mesh movement strategies, the potential for element inversion occurs. This requires special treatment during mesh updates.

Numerical analyses that require strict adherence of the mesh to the boundary (such as FEA or CFD) typically have the most complex numerical requirements and non-linearities. This, coupled with the mesh distortion at the boundary, reduces the numerical efficiencies and simulation accuracy. These can largely be overcome through mesh refinement, see. §1.4.3.3.

1.5.2. Adaptive meshing based on results, geometry and error estimates

Adaptive meshing is a meshing process that facilitates the description of a region with local mesh refinement. In iterative FEA the program can be designed to utilise local information from a previous solution to dictate mesh refinement and coarsening. This can be done based on either geometrical changes and/or numerical results such as local stress gradients or convergence requirements. Alternatively, user information specifying a region or face can be accommodated with the possibility of including a growth rate option to allow for better overall performance of the model.

In FEA the changes in stress results between adjacent elements can be used to estimate the errors of the solution *a posteriori* and thereby identify areas in which mesh refinement or coarsening is required.

[3] For 2D domains, neighbouring triangles can be easily combined into quadrilaterals, for more efficient solving

Results indicating little to no change between adjacent elements can be used to identify regions where mesh coarsening will not affect the accuracy and large changes can indicate areas requiring refinement.

Shape optimisation requires changes in the boundary. This can result in reduced element quality or in elements no longer being able to adequately capture the new geometrical state. Additionally, changes in shape have a direct impact on domain stress and stress gradients. An example of adaptive meshing around a "crack" is shown in Figure 6. In Figure 6 it is evident, that for this problem, a fine mesh is required only at the boundary and in the region around the crack. The remainder of the geometry may be meshed with courser elements, reducing the solution time for each iteration with no appreciable loss in accuracy.

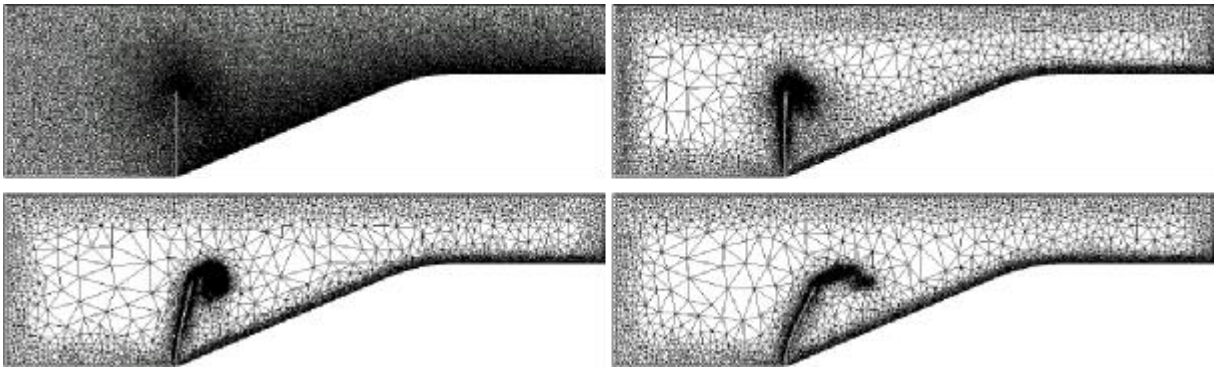


Figure 6 Adaptive mesh refinement around a crack tip [35]

As mentioned in §1.5.1, unstructured meshing strategies are well suited to adaptive meshing through the insertion or removal of nodes locally without the need for remeshing the domain. Structured meshes are not suited to local mesh adaptation, but rather require portions of the domain to be completely remeshed to accommodate the new refinement requirements.

1.5.3. Boundary deformation with boundary node and mesh adaptation

In structured meshing strategies, the boundary analytically dictates the nodal positions. It follows that nodes are updated explicitly with boundary deformation.

In the case of the unstructured meshing strategies, nodes on the boundary are placed in a regular fashion but do not move explicitly with a change in the boundary. The boundary nodes, therefore, need to either be explicitly defined by the user or they must be projected to the boundary after the change, which is a discrete process.

It is preferable that the nodes should be allowed to move along, but not away from, the domain boundary. This allows the mesh to retain better uniformity during updates. This is illustrated in Figure 7 showing a quadrilateral mesh on a 2D surface. The upper edge of the surface was defined using two distinct splines split by a control point, $\boldsymbol{x}$. Surface (a) represents the undistorted domain with the control point $\boldsymbol{x}$ located in the middle of the upper edge, effectively dividing the edge into two segments.

Each segment is seeded with 20 points at equal intervals. The change to the domain boundary of the domain was seen to be negligible for a fairly significant change in location, $\delta \boldsymbol{x}$, of the control point $\boldsymbol{x}$.

The effect of this change on the mesh is illustrated by (b) where nodes are still explicitly placed along the segments. This results in a highly distorted mesh, with reduced element quality. In contrast, if the nodes can move freely along the boundary as in (c), little to no mesh distortion occurs and the mesh remains of high quality.

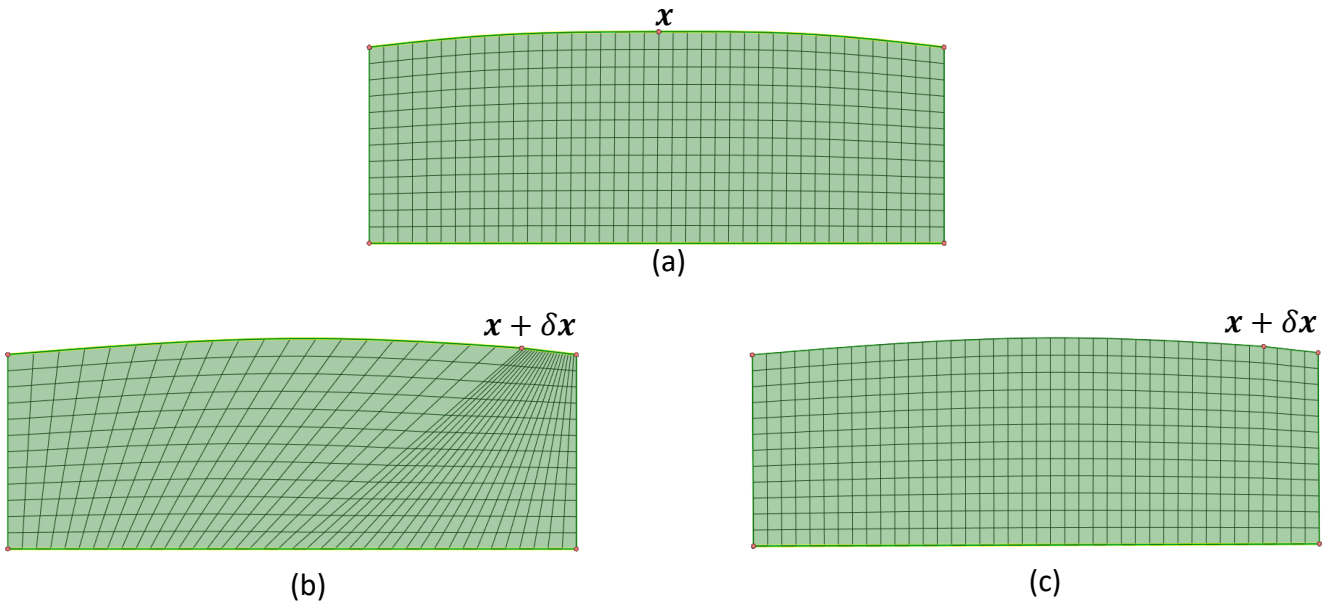


Figure 7 Effect of boundary modification on discretised boundary nodes
(a) Original domain and mesh (b) Boundary update with seeding using explicit nodes relation to control points
(c) Boundary update with free boundary node relation

The further discretisation of an edge is easily achieved by subdividing the edge. The discretisation of a surface is a 2D problem in and of itself and is not as easy. It is therefore important that the mesh boundary be free to be optimised in conjunction with the interior mesh.

1.5.4. Requirements of mesh sensitivities for shape optimisation

Since the accuracy of FEM is directly related to the quality, aspect ratios, and mesh refinement of the elements (see §1.4.3), maintaining these properties during a mesh update is very important. It therefore follows that for quality mesh sensitivities $\frac{d\boldsymbol{x}}{d\mathbf{x}}$ the entire mesh needs to react to any changes on the boundary, and not in an ad-hoc or local manner. The mesh sensitivities must therefore describe this effect at every node in the domain [7].

As discussed in §1.5.1 structured meshes are defined through a mapping process between the reference domain and the geometric domain. This means that the internal nodal locations are defined by the boundary position. In unstructured meshing, the nodes are distributed randomly throughout the

domain. The nodes are therefore not defined relative to the boundary. Hence it follows that movement of the boundary in structured meshes will update the internal nodal positions accordingly, whereas in unstructured meshes boundary movement does not affect the internal nodes and their movement needs to be addressed programmatically. It is therefore seen that the internal nodes of structured meshes are functions of the boundary and naturally possess sensitivities.

Structured meshing strategies are typically challenging to implement on complex geometries and the introduction of local refinement requires at least partial remeshing of the domain. Unstructured meshes are well suited to the discretisation of complex geometries, to the introduction of local refinement and to the accommodation of adaptive meshing as discussed in §1.5.2.

For remeshing optimisation strategies, the domain is remeshed between iterations. This strategy is easier to implement as all local refinement may be readdressed as required while maintaining mesh quality. These strategies are however numerically expensive and the remeshing may introduce discontinuities in the objective/cost function [7]. Both structured and unstructured meshing are suitable for implementation with remeshing optimisation strategies. Remeshing optimisation easily accommodates significant changes to geometry and refinement.

Mesh-movement optimisation strategies require updates to the mesh between iterations. As the name suggests, the mesh with its current connectivity is re-fitted to the updated geometry. Both structured and unstructured meshes can be used for mesh-movement strategies for large changes to boundary dimensions provided the overall shape is maintained. However, where local mesh refinements and changes in geometry occur, structured meshes are not well suited while unstructured meshes can accommodate some local refinement without alteration in element connectivity.

Remeshing optimisation strategies are simpler to implement when compared to mesh-movement strategies. They are however numerically more expensive and tend to introduce discontinuities in the objective/cost function. Unstructured meshes are more flexible when compared to structured meshes in terms of their use in optimisation strategies. Complex geometries are easily discretised and such meshes improve the performance of mesh-movement strategies.

Mesh-movement strategies with unstructured meshes may introduce problems with local mesh quality near altered geometry since the internal nodal coordinates are not a function of the boundary. One way of maintaining mesh quality in unstructured meshes is to use an averaging technique [1]. In these approaches, the movement of the boundary induces a movement in the mesh that is propagated into the mesh iteratively. Many iterations are required to ensure good uniformity is maintained. These methods come in varying degrees of sophistication, but at best the technique can only be used to obtain numerical sensitivities, which are generally computationally expensive [1].

Another option is to use an underlying principle that analytically describes the mesh movement throughout the domain. For unstructured tetrahedral meshes, Wilke *et al.* [7] demonstrated that

analogizing the unstructured mesh to a truss structure allowed for mesh deformation to be taken up evenly throughout the domain. They successfully obtained analytical mesh sensitivities for their 2D piecewise linear domains using this method.

For local mesh changes in the domain, such as local mesh refinement based on error estimates, the local desired truss lengths can be altered. Since the truss mesh is driven by an underlying analytical relationship, the method allows the mesh to adapt, without remeshing, to the new domain requirements.

The following attributes are therefore desirable for a mesher used in shape optimisation:

- Unstructured mesher – this allows for the easy discretisation of complex geometries and execution of adaptive meshing
- Mesh-movement optimisation strategies should be accommodated
- An underlying principle should be used to facilitate access to accurate mesh sensitivities

1.6. Examination and selection of base mesher platform

Several free and open-source meshing programs are available. These have varying degrees of complexity and capability, but all allow for easy mesh creation and adaptation. Robert Schneiders maintains a useful list of both public-domain and commercially available mesh generators. At the date of writing, ninety-six public-domain applications were listed on his website [36]. A summary is provided by Steven Owen who surveyed eighty-one unstructured mesh generators for both the public-domain and commercial use [37].

More than 150 public domain mesh generators were reviewed from the above-mentioned references and none were found that made mesh sensitivities explicitly available for use in gradient-based shape optimisation. Only one mesher, STELLAR [38], indicated the presence of mesh sensitivities as part of the meshing process workflow. These sensitivities were associated with the dihedral angles of the tetrahedral elements and utilised the sensitivities for a mesh optimisation process. STELLAR does not make any mesh sensitivities available to the user. It is conceivable that these sensitivities could be combined to obtain a set of full-mesh sensitivities.

Bischof *et al*. [39] showed a method for modifying the mesher CSCMDO [40] to output numerical mesh sensitivities for gradient-based optimisation applications. The mesher was modified to make numerical sensitivities available using the ADIC automatic differentiation tool. No mesher investigated indicated access to analytical sensitivities, outside of the mesher proposed by Wilke *et al*. [7].

Whilst other mesh generators may exist that are potentially faster, more robust, and create higher-quality meshes, the simplicity offered by DistMesh [33] makes it an easy-to-build-upon platform for use in this study. Wilke *et al.* used DistMesh as the foundation in the development of their 2D mesher [7]. DistMesh offers a convenient analogy between an unstructured tetrahedral mesh and a

compressed set of springs or trusses. This gives DistMesh an underlying physical principle that describes the mesh movement throughout the domain. DistMesh is an unstructured mesher that allows for adaptive meshing through altering the desired or free truss lengths. The underlying principles of DistMesh allow for mesh-movement strategies that can accommodate local changes such as mesh refinement.

The DistMesh implementation was discrete through the use of a forward-Euler implementation and continuous but not smooth through truss stiffness formulation employed to solve equilibrium of the system. Wilke *et al.* utilised the truss concept to implement a Newton-method based solver for the mesh, which already contains the sensitivities required for obtaining the nodal coordinate sensitivities $\frac{d\boldsymbol{x}}{d\mathbf{x}}$.

1.7. Summary

The shape optimisation problem requires the optimisation of an objective or cost function for a particular geometry subject to a set of constraints. The first step in the solution of a shape optimisation problem is the parametrisation of the geometry with respect to a set of control variables. The continuum is then reduced to a standard constrained finite-dimensional model. This reduction is normally accomplished through a discretisation strategy. In discretisation, the continuum PDEs for the domain are described by means of a discretised computational grid.

Discretisation has the advantage of all resulting functions being solvable with standard linear algebra methods. It has the disadvantage of the accuracy of the final solution being dependant on the quality of the meshing strategy employed. Mesh adaptability is therefore a very important part of mesh generation in the shape optimisation environment.

Discretisation methods must consider the construction of the mesh and the control of mesh deformation when the boundary of the geometry is displaced. Two methods for discretising a domain exist, namely structured and unstructured meshing. Structured meshing utilises a mapping method where a reference mesh is mapped to the geometry through a transformation function. Structured meshing methods are however not well suited to discretising complex geometries and require remeshing of the domain to facilitate mesh refinement. Unstructured meshes are defined by the location of the nodes distributed through the domain and easily discretise complex geometries. Furthermore, unstructured meshes are easily adjusted to facilitate mesh refinement through the insertion of nodes in the area of interest.

Adaptive meshing can improve the accuracy with which boundary information is captured and reduce computational resources associated with unnecessary domain refinements. Adaptive meshing is most easily implemented using simplex unstructured meshing methods where nodes can simply be inserted and removed in the area of interest to vary local mesh refinement. For this study the benefits of using

an unstructured meshing strategy outweighed the increased numerical overhead costs and mesh management incurred compared to structured meshing approaches.

The shape optimisation problem requires the geometrical domain discretisation to be updated with each iteration. Two approaches exist to accomplish this; remeshing and mesh-movement strategies. Remeshing strategies, as the name suggests, remesh the domain for every iteration. They are able to ensure good quality meshes regardless of the magnitude of the change in shape. They are however numerically expensive to implement. Since the process of remeshing is not smooth it can result in discontinuities in the sensitivities [7].

Mesh-movement strategies update the mesh from one iteration to the next by refitting the mesh to the new domain. No changes in element connectivity occur, only the movement of the nodes to the new boundary. The process is smooth and does not introduce discontinuities into the sensitivities. This enables the effective use of gradient-based shape optimisation methods. Mesh-movement optimisation strategies are generally restricted to relatively small changes in geometry.

The need for remeshing is reduced when mesh updates do not significantly distort elements. However, this is in general not the case, particularly when large shape changes are at play. The benefit lies in a combination of the two strategies. The mesh-movement approach can provide continuous sensitivities which in turn allows for the use of more efficient gradient-based optimisation methods. Remeshing is then performed when elements become too distorted or when significant changes in mesh refinement are required.

The potentially large expense associated with remeshing strategies can be offset when the strategy facilitates the provision of exact analytical sensitivities. These sensitivities can be used in gradient-based optimisation methods which are known to provide superior solutions [1, 22].

It therefore follows that the requirements of a gradient-based shape optimisation focused mesher are:

1. Simplex[4] unstructured mesher: this is selected for the simplicity of discretising complex domains and the ease of implementing adaptive meshing strategies (§1.5.2).
2. Able to discretise user-defined non-linear domains.
3. Provide (semi)-analytical mesh sensitivities. These are sensitivities that analytically describe the mesh deformation in response to changes in design variables. Partial aspects of the sensitivities may be calculated numerically.
4. Mesh optimisation methods must have a smooth underlying principle that allows access to smooth mesh sensitivities. Remeshing may occur during optimisation, but the mesh improvement strategy must be smooth and continuous towards the end to facilitate access to accurate mesh sensitivities.

[4] Simplexes are triangles in 2D and tetrahedra in 3D

5. Smooth mesh sensitivities: boundary changes, or local refinement, should subject the entire mesh to deformation in a smooth manner, not only locally (§1.5.4).
6. Boundary mesh adaptability: the boundary nodes should be free to move along, but not away from the domain boundary (§1.5.3), in such a manner that this relationship forms part of the mesh analytical sensitivities.
7. User friendly: the mesh generator should be simple to use and modify.

DistMesh is an easy to modify and use unstructured simplex mesher that allows for mesh adaptivity through node insertion and an ideal truss length control function to determine the edge lengths over the geometrical domain. It can discretise non-linear geometrical domains defined by the user. It analogises a simplex mesh to a system of trusses or springs, basing the method in physical mathematics. This, in turn, permits the recovery of smooth sensitivities for the system. DistMesh does not require the prescription of the boundary nodes but allows the nodes to move freely along the boundary while also not being constrained to the boundary. DistMesh was used successfully by Wilke *et al.* [7] to implement their sensitivity based mesher for the 2D piecewise linear domains used to perform shape optimisation.

This study will do three things:

1. Become a workable extension of the Wilke *et al.* [7] mesher from 2D to 3D.
2. Permit the discretisation of non-linear domains rather than piece-wise linear domains.
3. Release the previously prescribed boundary nodes to move along but not away from the boundary.

In order to produce accurate sensitivities for a non-linear domain, the boundary node movement must be incorporated into the sensitivities. Obtaining sensitivities that include the sliding boundary nodes will be the focus of this study.

CHAPTER 2
TRUSS MESHER WITH FREE BOUNDARY NODES

"Engineers ... are not superhuman. They make mistakes in their assumptions, in their calculations, in their conclusions. That they make mistakes is forgivable; that they catch them is imperative. Thus, it is the essence of modern engineering not only to be able to check one's own work but also to have one's work checked and to be able to check the work of others." — Henry Petroski

2.1. Introduction

This chapter discusses the selected base meshing program DistMesh. It describes the fundamentals of program operation with the basic functions used in §2.2. The conceptual changes to DistMesh and their implementations will follow in §2.3 to §2.6.

The first alteration is to change the continuous but not smooth force function in DistMesh. This is discussed in §2.3 where the description of the continuous general truss-equilibrium system is presented. These will consider only a set of prescribed boundary nodes and hence represents the solution to the interior of the domain. This is done in the context of a geometrically non-linear FEA problem.

As stated in the previous chapter, the focus of this study is to allow nodes to move freely along, but not away from the boundary. Potential approaches include smooth classical boundary contact and multi-point constraint (MPC) methods discussed in §2.5 and §2.6 respectively. It will be shown that the analytical requirements for obtaining accurate mesh sensitivities including boundary nodes will be best met by the master-slave elimination (MSEM) and Lagrangian multi-point constraint (MPC) methods.

The MSEM and Lagrangian methods will be presented in §2.6.2 and §2.6.3 respectively. For both methods, the system of truss equations will be extended to include the MPC boundary equations in §2.6.2.1 and §2.6.3.1. In §2.7 the numerical solution strategies for solving these systems of residual equations will be discussed. These equations will be developed directly in the context of the boundary constraint problem.

2.2. Overview of DistMesh

The simple meshing program DistMesh [33], developed by Per-Olof Persson and Gilbert Strang works on a very basic concept: the simplex mesh structure is analogous to that of a simple truss structure where the mesh nodes are the truss joints and the element edges are the trusses. The nodal coordinates of the mesh are determined by solving for the equilibrium of the system of trusses. The trusses are treated as being (on average) in compression. This ensures the propagation of the nodes to the boundary and discretisation of the entire domain. This method results in mesh improvement in a

smooth and continuous manner for smooth and continuous force functions. A brief overview of the main attributes of the mesher follows; for a more thorough explanation see [33].

DistMesh follows a few basic steps. These steps are shown in Figure 8, with the final resulting mesh depicted in Figure 9. Steps 1 through 6 are iterated through until the norm of the mesh updates is below a specific tolerance.

1. Domain seeded with points based on some minimum expected length h_{min} (see Figure 8 (a))
2. Points outside the domain are rejected (see Figure 8 (b))
3. Points throughout the domain are kept/rejected based on a probability function, retaining node density in the areas corresponding to the length function $h\big(p(\boldsymbol{X})\big)$ (see Figure 8 (c)) (see §2.2.4)
4. Delaunay triangulation is used to create the truss connectivity (see Figure 8 (d))
5. Nodes are propagated to the boundary using a forward-Euler step (see §2.2.2) in the directions calculated by the force function (see §2.2.3). The force function acts as if (on average) all truss members are compressed. (see Figure 8 (e))
6. Points outside of the boundary are either rejected (if outside of some tolerance) or returned to the boundary (see §2.2.5) (see Figure 8 (f))
7. The steps 4. through 6. are repeated until equilibrium at all the nodes in the domain is reached and the boundary is satisfied. Retriangulation is completed every n iterations.

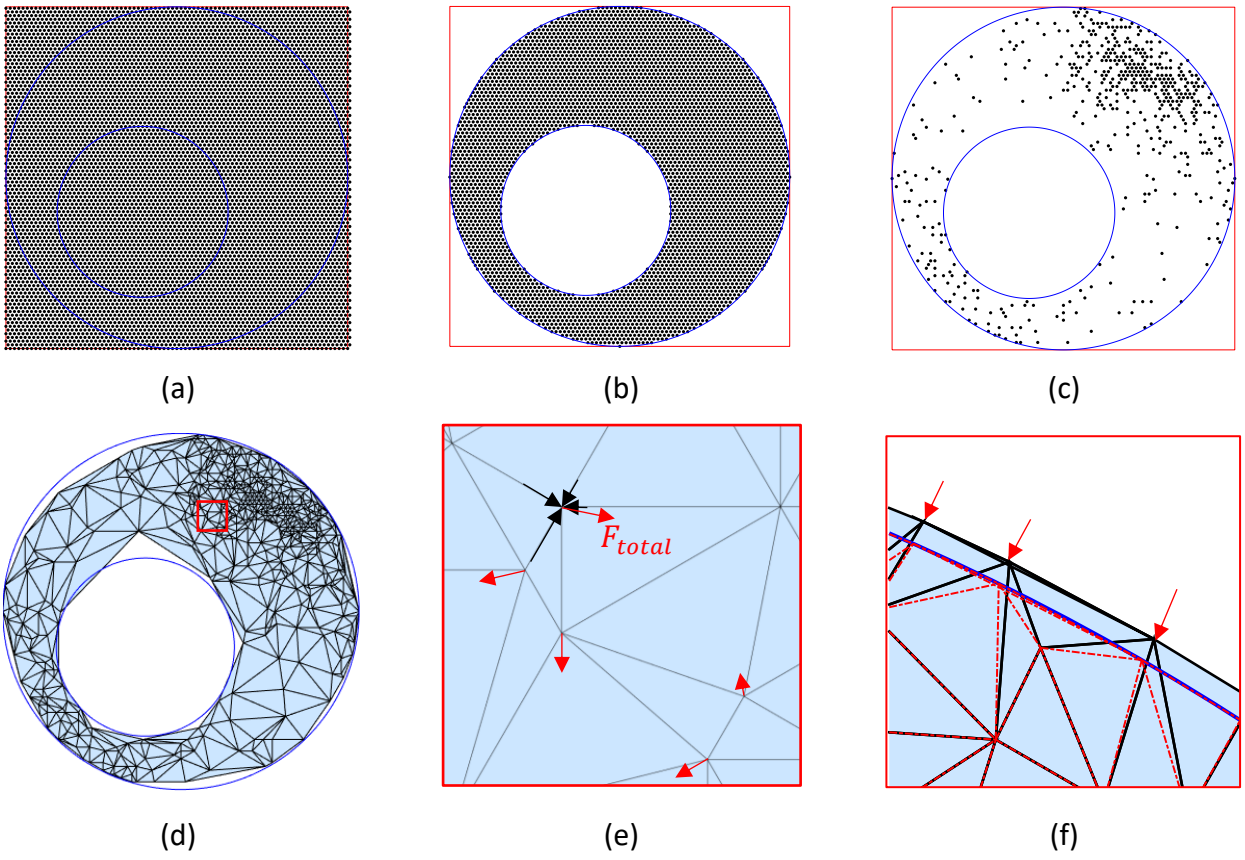


Figure 8 DistMesh process steps

(a) Seeding domain (b) Rejection of points outside the domain (c) Retention of points based on the probability function (d) Delaunay triangulation for truss connectivity (e) Propagation of nodes to the boundary (f) Handling of nodes moved outside the boundary

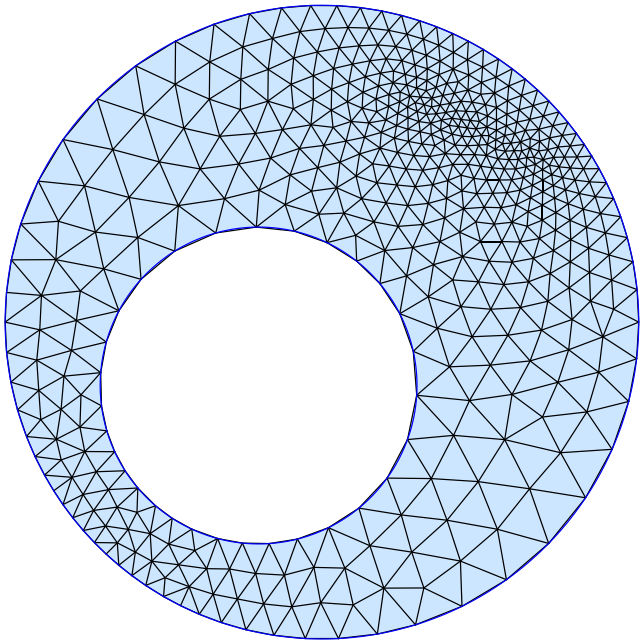


Figure 9 Final mesh from DistMesh

2.2.1. Domain boundary definition

The geometrical domain is described using a signed distance function $f_{\partial\Omega}(\boldsymbol{p})$, where the function evaluates a coordinate p. The function value is a magnitude giving the shortest distance to the boundary and is positive outside of the domain, negative inside of the domain and zero on the boundary $\partial\Omega$. This allows for a combination of simple shapes to describe more complex domains in a simple manner. The geometrical domain is described by:

$$f_{\partial\Omega}(p) = \begin{cases} f_{ellip}(\boldsymbol{p}) \\ f_{rect}(\boldsymbol{p}) \\ f_{poly}(\boldsymbol{p}) \\ \vdots \end{cases} \tag{12}$$

2.2.2. Node propagation and mesh solution

The mesh nodal locations $\boldsymbol{X}$ are solved using a forward Euler approach with an artificial time dependence Δt. This is, in fact, a fixed step size steepest descent method for the minimisation of the larger truss potential energy function. The update step is given by:

$$\boldsymbol{X}_{n+1} = \boldsymbol{X}_n + \boldsymbol{F}(\boldsymbol{L}_c, \boldsymbol{L}_{des}) \cdot \Delta t \tag{13}$$

where $\boldsymbol{X}_{n+1}$ is the updated location, $\boldsymbol{X}_n$ is the current location and $\boldsymbol{F}$ is the truss force function defined in §2.2.3.

This is continued until the magnitude of subsequent nodal coordinate updates is below some tolerance $p_{n+1} - p_n < tol$.

2.2.3. Force function definition

The mesher is designed such that all the trusses of the system are always in compression (on average). Additionally, the force function is designed as continuous but not smooth, meaning that there is a truss reaction force under compression but a zero force under tension. Trusses can only push and cannot pull on nodes – assuming k is positive. The truss force function $\boldsymbol{F}$ is given by:

$$\boldsymbol{F}(\boldsymbol{L}_c, \boldsymbol{L}_{des}) = \begin{cases} k(\boldsymbol{L}_{des} - \boldsymbol{L}_c)\,/\boldsymbol{L}_c & \text{if } \boldsymbol{L}_c < \boldsymbol{L}_{des} \\ 0 & \text{if } \boldsymbol{L}_c \geq \boldsymbol{L}_{des} \end{cases} \tag{14}$$

where $\boldsymbol{L}_{des}$ is the undeformed or desired length assigned to the truss, $\boldsymbol{L}_c$ is the current length and k is the spring stiffness of the truss. This, combined with the length function in (16), ensures the propagation of the nodes to the boundary.

2.2.4. Length function definition

DistMesh is designed as an adaptive mesher where the desired or uncompressed length of the trusses is given by the function $\boldsymbol{h}(\boldsymbol{p})$. This generates a relative size value for the given locations, $\boldsymbol{p}$, when supplied as a point for evaluation. The length function is supplied by the user and can be designed to use local feature sizes, pinch zones, curvature, or any other attribute the user wishes to program in.

Since the objective of their mesher was simplicity, the values returned from the length function $\boldsymbol{h}(\boldsymbol{p})$ are treated as a relative ideal element length. This avoids an implicit connection of the number of nodes to the size of the domain.

The desired lengths utilised for the calculations are therefore given by:

$$\text{Scaling factor} = F_{scale}\left(\frac{\Sigma \boldsymbol{L}_{c_i}^{\,2}}{\Sigma \boldsymbol{h}\left(\boldsymbol{p}_{mid}(\boldsymbol{\mathcal{X}})\right)^2}\right)^{\frac{1}{nD}} \tag{15}$$

where nD is the number of dimensions of the domain and $F_{scale} > 1$ enforces a precompression of the trusses. The desired lengths $\boldsymbol{L}_{des}$ is given by:

$$\boldsymbol{L}_{des} = F_{scale}\left(\frac{\sum_{i=1\ldots n} \boldsymbol{L}_{c_i}^{\,2}}{\sum_{i=1\ldots n} \boldsymbol{h}\left(\boldsymbol{p}_{mid}(\boldsymbol{\mathcal{X}})\right)}\right)^{\frac{1}{nD}} \boldsymbol{h}\left(\boldsymbol{p}_{mid}(\boldsymbol{\mathcal{X}})\right) \tag{16}$$

2.2.5. Boundary node control

When a node reaches the boundary of the domain it is tested to determine whether it should be rejected or retained. If the node has moved outside of some tolerance on the boundary, $f_{d\Omega}(x_n) > tol$, then that point is rejected. If it lies outside of the domain $0 \leq f \leq tol$ but is still within the tolerance value, that node is forced back to the boundary along the local gradient direction to the boundary. In

this manner, the nodes are maintained on the boundary at the end of each iteration[5]. Equilibrium is obtained between the compressive driving forces in the trusses and the boundary acting as a rigid wall pressing the nodes back.

2.3. Truss equilibrium equations

As mentioned in §2.2.3 the force equilibrium equation in DistMesh is non-smooth. In order to create smoothness in the system, the trusses need to be able to act in tension as well as compression. The truss system can, therefore, be viewed as a geometrically non-linear FEM system.

Discretisation of the weak form of the mixed boundary value problem of geometrically non-linear solids results in a system of non-linear nodal residual equations. The residuals are stated as the internal forces $\boldsymbol{\mathcal{F}}^{int}$ minus the external $\boldsymbol{\mathcal{F}}^{ext}$ forces at each node. These residuals can be viewed as the first derivative of the larger energy function of the system and therefore represent the required solution to the first-order necessary condition:

$$\boldsymbol{\mathcal{R}} = \boldsymbol{\mathcal{F}}^{int} - \boldsymbol{\mathcal{F}}^{ext} = \mathbf{0} \tag{17}$$

The internal forces in the system $\boldsymbol{\mathcal{F}}^{int}(\boldsymbol{\mathcal{X}})$ are a function of the nodal coordinates $\boldsymbol{\mathcal{X}}$. $\boldsymbol{\mathcal{F}}^{ext}$ is the set of scalar external forces.

2.4. Freeing the boundary nodes

To obtain sensitivities for the boundary nodes, the nodes on the boundary may not deviate from the boundary due to reaction forces. Two options are available to fulfill this requirement: regular fixing to the boundary, or permission to move along the boundary using constraint(s).

Fixing the nodal co-ordinates along the boundary is quite simple in 2D where points can simply be placed along an edge at regular intervals. In 3D and higher-order dimensions, however, fixed node placement on the boundary presents significant programming challenges, especially where non-linear and/or irregular boundaries are present. The boundary problem then becomes an $(n-1)$-dimensional problem.

When only prescribed constraints are present, the system is typically split into the residuals at the free $\boldsymbol{\mathcal{R}}_f$ and single DOF or prescribed constraints, $\boldsymbol{\mathcal{R}}_p$:

$$\begin{aligned} \boldsymbol{\mathcal{R}}_f &= \boldsymbol{\mathcal{F}}_f^{int} - \boldsymbol{\mathcal{F}}_f^{ext} = \mathbf{0} \\ \boldsymbol{\mathcal{R}}_p &= \boldsymbol{\mathcal{F}}_p^{int} - \boldsymbol{\mathcal{F}}_p^{ext} - \boldsymbol{R}_p = \mathbf{0} \end{aligned} \tag{18}$$

where $\boldsymbol{R}_p$ is the reaction force at the prescribed DOFs. This is the system of non-linear equations that needs to be solved to obtain equilibrium in the truss system.

[5] This assumes that the value returned by the boundary distance function is the exact shortest distance from the boundary, otherwise the boundary is approximated

Since the truss system can be treated as a geometrically non-linear FEM problem the tools are already available to meet the requirements for movement along the boundary: Contact [41] and MPCs [42].

To address the boundary nodes the system of residual equations is partitioned into free $\boldsymbol{\mathcal{R}}_f$, prescribed $\boldsymbol{\mathcal{R}}_p$ and boundary $\boldsymbol{\mathcal{R}}_{d\Omega}$ DOFs. The system of residuals is then given by:

$$\boldsymbol{\mathcal{R}}_f = \boldsymbol{\mathcal{F}}_f^{int} - \boldsymbol{\mathcal{F}}_f^{ext} = \mathbf{0} \tag{19}$$

$$\boldsymbol{\mathcal{R}}_{\partial\Omega} = \boldsymbol{\mathcal{F}}_{\partial\Omega}^{int} - \boldsymbol{\mathcal{F}}_{\partial\Omega}^{ext} = \mathbf{0} \tag{20}$$

$$\boldsymbol{\mathcal{R}}_p = \boldsymbol{\mathcal{F}}_p^{int} - \boldsymbol{\mathcal{F}}_p^{ext} - \boldsymbol{R}_P = \mathbf{0} \tag{21}$$

This system needs to be satisfied with the addition of the domain boundary constraint functions:

$$\boldsymbol{\mathcal{R}}_c = \boldsymbol{f}_{\partial\Omega} = \mathbf{0} \tag{22}$$

2.5. Classical contact

Contact is a non-linear boundary condition used in FEA to describe the interaction between two bodies. The non-linear nature is due to a boundary constraint that is either active or inactive and the unknown and interdependent displacement and force values on the boundary [41]. This requires an iterative solution, either explicitly or implicitly, of the problem until equilibrium is found.

The main disadvantage associated with contact in terms of this work is that the nature of contact between bodies is discrete: the constraint is either active or inactive. A small change in displacement results in a sudden increase in force from zero to a small reaction value, which in turn results in a force profile that is not smooth, hence a non-smooth objective function for solvers. Since it is numerically unstable to allow a force on the boundary to change in a piece-wise linear manner, contact boundaries are generally implemented using a form of the penalty method or a Lagrangian multiplier in an unconstrained optimisation approach [41].

Gradient-based root finders require the force profile at the constraint to be smooth and continuous. In classical contact, for some small distance away from the boundary, the force needs to be zero; and for contact to be stable the force needs to increase accordingly to maintain equilibrium on the boundary [41] resulting in a force profile that is non-smooth and continuous.

Contact methods rely on the penetration of the contacting surfaces to enforce equilibrium. Penetration and resulting deviations from the domain boundary are somewhat overcome by using highly exponential penalty contact functions. However, for a mesh to be suitable for FEM the domain boundary needs to be accurately captured. Furthermore, in accordance with its intended design, classical contact only offers compression resistance at the boundary. This implies that the sensitivities

of the mesh in compression and tension may respond differently. This could therefore reduce the performance of the shape optimisation algorithm.

Additionally, identification of contacting and non-contacting nodes is required at every iteration. This is computationally expensive, and convergence may be slowed by nodes making and breaking contact. These are elaborate models that allow for the inclusion of frictional effects and the resolution of tangential forces etc. that makes contact an inappropriate selection for obtaining analytical mesh sensitivities.

2.6. Multi-point constraints (MPCs)

Conventional FEA, with simple boundaries, can have "single-degree-of-freedom constraints" where a single degree of freedom (DOF) is subject to a prescribed displacement under conditions where the prescribed displacement is aligned with the global coordinate system in which the nodal displacements are expressed. Since FEA solves for the displacements, it means that the global stiffness equations, once assembled, are partitioned to remove these DOFs from the system. The exception is when they are subject to a non-zero displacement; in this case, their contribution is combined with the external forces to completely define the conditions for the unprescribed or free DOFs. Reaction forces for all constrained degrees of freedom are calculated *a posteriori*.

However, realistic engineering problems seldom require boundary conditions that fit so neatly into the limitations of the single-degree-of-freedom constraint. Additional constraint formulations are therefore required to allow for the representation of complex boundaries.

Multi-point constraints (MPCs), sometimes called multi-freedom constraints (MFCs) [43] or multi degree-of freedom-constraints are a subset of contact methods that allows for the analytical definition of relationships between the displacements for various DOFs during FEA. This allows for more complex boundary conditions to be modelled, which in turn allows for more accurate analyses to be performed, e.g. movement along an arbitrarily oriented plane as opposed to a plane aligned with the global coordinate axis.

MPC relationships are defined by equations that describe the movement of DOFs relative to one another. These equations can also take the form of displacement DOFs equated to a prescribed value. The MPC relationship is given by:

$$\boldsymbol{f}_{mpc}(\boldsymbol{x}_{mpc}) = \mathbf{0} \tag{23}$$

In all MPC implementations, the stiffness equations are modified or amended to apply these additional constraint relationships together with a reaction force. Depending on the MPC's implementation, the modification may:

- Reduce the number of stiffness equations to be solved, as with the master-slave elimination method (MSEM)
- Keep the number of equations the same, as with the penalty method
- Or increase the number of equations, as with the Lagrangian approach.

For methods where the system of equations is reduced by removing displacement DOFs, the DOFs missing from the solution, as well as the constraint reaction forces, are recovered by evaluating the MPC relationships in (23). For the other types of implementations, the displacements of all DOFs are solved directly.

2.6.1. Penalty method

The penalty approach to MPCs does not add additional equations to the system, but rather treats the constraint relations as additional elements of stiffness. This method works by adding an element that is very stiff compared to the general element stiffnesses.

When the constraint equations are given as $\mathbf{A}\mathbf{u}_{mpc} = \mathbf{d}$, the general stiffness matrix takes the following form [43]:

$$\left(\mathbf{K}_{mpc} + \mathbf{A}^T\mathbf{W}\mathbf{A}\right)\mathbf{u}_{mpc} = \mathbf{f}_{mpc} + \mathbf{A}^T\mathbf{W}\mathbf{d} \qquad (24)$$

where $\mathbf{W}$ is a diagonal matrix of the penalty weights w_i for each constraint equation. The subscript mpc denotes the modified elements of the stiffness matrix $\mathbf{K}$ associated with the mpc DOFs.

The penalty method is an aggregation of pros and cons. Except for selecting weight values for the constraints, the penalty method is easy to implement due to the simple augmentation of the stiffness matrix with additional element stiffnesses. It is unaffected by linear dependence between constraints (i.e. repeated constraints) since this merely results in additional stiffness between constraint DOFs. However, multiple penalty factors occurring in a single equation could, despite careful selection of $\mathbf{W}$, result in poor matrix conditioning. The resulting system remains positive definite and can, therefore, be efficiently solved using a Cholesky $\mathbf{L}\mathbf{L}^T$ decomposition [43].

The main disadvantage of the penalty method is that the solution is always slightly violated and there is difficulty in selecting appropriate penalty weight values $\mathbf{W}$ that minimise errors without the need for considerable experimentation. A value that is too small results in a large violation of the boundary condition, whereas a value that exceeds a particular computer's precision will result in the computer regarding the stiffness matrix as exactly singular. A compromise is therefore required to select a value of w_i such that the absolute value of the constraint violation error is approximately equal to that of the solution error [43].

The calculation of sensitivities may run into numerical problems when using the penalty method if the stiffness and penalty weights are orders apart. Furthermore, since the penalty method is inexact and

has an error associated with its solution to the boundary function, it is deemed inappropriate and will therefore not be considered further for this study. The inability of it to exactly match the domain boundary results in a situation where the truss sensitivity may respond differently in compression compared to tension. This will especially be the case if numerical sensitivities are used.

2.6.2. MSEM MPCs

In the master-slave elimination method (MSEM), the MPC DOFs are separated into two groups: the masters and the slaves. The MPC equations are rearranged such that the slave DOF is a function of the master DOFs and prescribed displacement. Each MPC equation can only have one slave DOF with a resulting unknown reaction force. A slave DOF can, however, be described by multiple constraint equations, provided it does not become over-constrained or linearly dependent [43, 44, 45].

For the MSEM approach, the global system is modified by recognising that the system can be partitioned into free, prescribed, master, and slave DOFs. Since the slave-DOFs are a function of the master-DOFs, the DOFs to be solved are reduced to only the free-DOFs and the master-DOFs, hence the master-slave “elimination” method.

To further complicate matters, it is not always obvious what is master and what is slave, and selection of the masters vs slaves can have consequences in terms of numerical stability if the constraint equations have coefficients whose orders of magnitude differ substantially. This makes the constraint more unstable and can cause deterioration in the condition of the resulting system. It is thus preferable to select master and slave DOFs such that the magnitude of the coefficients of the master DOFs is minimised. More complications arise when geometrically non-linear problems need to be solved since the curvature (second-order contribution of a non-linear constraint) will also affect the selection of the masters- and slaves-DOFs.

These facts make the automation of MSEM’s difficult. The implication being that packages that support the MSEM typically require the user to specify the slave- and master-DOFs.

One of the notable features of the MSEM is that the constraint is exactly satisfied at each iteration through the calculation of the updated slave positions using constraint equations. This, however, has the associated negative of potentially large displacement updates resulting in problems such as element inversion.

The primary benefit of the MSEM lies in the reduction of system size by the number of slave DOFs, potentially allowing for improved solution times. However, manipulation of the stiffness matrix $\mathbf{K}$ may result in increased bandwidth due to added terms that may increase the bandwidth, increasing the cost and difficulty of allocating memory. Furthermore, the system of equations is positive definite which allows for an efficient solution using a Cholesky $\mathbf{L}\mathbf{L}^T$ decomposition in a 2D environment, or a preconditioned conjugate gradient method in higher dimensions [46, 47].

Since the method analytically satisfies the boundary, it can be used to obtain accurate (semi)-analytical mesh sensitivities. This makes it a suitable choice for the control along the boundary surface and success will be dependent on ease and robustness of implementation.

2.6.2.1. MSEM system of equations

Considering a structure in its initial configuration $\boldsymbol{\mathcal{X}}^i$ that ends in the final configuration $\boldsymbol{\mathcal{X}}^e = \boldsymbol{\mathcal{X}}^i + \boldsymbol{u}$, where $\boldsymbol{u} = \Delta\boldsymbol{\mathcal{X}}$, the relationship between the slave and master DOFs is obtained by rearranging (22) and has the general form:

$$\boldsymbol{u}_s = \boldsymbol{f}_{ms}(\boldsymbol{u}_m) \tag{25}$$

where the displacements $\boldsymbol{u}_s$ are the slave (dependant) DOFs and $\boldsymbol{u}_m$ are the master (independent) DOFs.

For non-linear problems, the system of residual equations in (19) to (22) is modified to account for the master-slave relationships expressed in (25). This is achieved by partitioning the system boundary constrained residual equations $\boldsymbol{\mathcal{R}}_{d\Omega}$ into the master $\boldsymbol{\mathcal{R}}_m$ and slave $\boldsymbol{\mathcal{R}}_s$ DOFs. It should be noted that the reaction forces $\boldsymbol{R}_m$ and $\boldsymbol{R}_s$, for the master and slave DOFs respectively, are also present in the residual system of equations:

$$\boldsymbol{\mathcal{R}}_f = \boldsymbol{\mathcal{F}}_f^{int} - \boldsymbol{\mathcal{F}}_f^{ext} = \boldsymbol{0} \tag{26}$$

$$\boldsymbol{\mathcal{R}}_m = \boldsymbol{\mathcal{F}}_m^{int} - \boldsymbol{\mathcal{F}}_m^{ext} - \boldsymbol{R}_m = \boldsymbol{0} \tag{27}$$

$$\boldsymbol{\mathcal{R}}_s = \boldsymbol{\mathcal{F}}_s^{int} - \boldsymbol{\mathcal{F}}_s^{ext} - \boldsymbol{R}_s = \boldsymbol{0} \tag{28}$$

$$\boldsymbol{\mathcal{R}}_p = \boldsymbol{\mathcal{F}}_p^{int} - \boldsymbol{\mathcal{F}}_p^{ext} - \boldsymbol{R}_p = \boldsymbol{0} \tag{29}$$

The system of equations in (26)-(29) cannot be solved as the reaction forces $\boldsymbol{R}_p$, $\boldsymbol{R}_m$ and $\boldsymbol{R}_s$ are unknowns. $\boldsymbol{R}_p$ can however be recovered from (29) once the system is solved.

Two solution methods in literature are symmetrising of the system to remove the slave DOFs [43] and an energy-based method that manipulates the system of equations utilising the fact that the boundary performs no work to remove the slave DOFs [45]. In the symmetrizing method, the reaction forces are not explicitly identified as part of the system, which implicitly enforces the requirement that zero work is done by the boundary. These methods are compared in APPENDIX A.

This derivation follows that of Kok and Wilke [45]. Since the master-slave relationship is known $\boldsymbol{u}_s = \boldsymbol{f}_{ms}(\boldsymbol{u}_m)$, $\boldsymbol{R_s}$ can be recovered from (28). However, to obtain a solution to the master-DOFs $\boldsymbol{u}_m$, the relationship between $\boldsymbol{R}_s$ and $\boldsymbol{R}_p$is required. This relationship is derived by considering that the MPC

reaction forces conduct no work on the system. It follows, therefore, that during an increment $\Delta \boldsymbol{u}_m$ and corresponding $\Delta \boldsymbol{u}_s$ (as it directly depends on $\Delta \boldsymbol{u}_m$), no work may be done by the reaction forces:

$$\Delta \boldsymbol{u}_m^T \boldsymbol{R}_m + \Delta \boldsymbol{u}_s^T \boldsymbol{R}_s = 0 \tag{30}$$

A linear Taylor series expansion of the MPC constraint $\boldsymbol{u}_s = \boldsymbol{f}_{ms}(\boldsymbol{u}_m)$ in (25) gives:

$$\Delta \boldsymbol{u}_s = \frac{d\boldsymbol{u}_s}{d\boldsymbol{u}_m} \Delta \boldsymbol{u}_m = \boldsymbol{P} \Delta \boldsymbol{u}_m \tag{31}$$

Note that $\boldsymbol{P} = \boldsymbol{P}(\boldsymbol{u}_m) = d\boldsymbol{u}_s / d\boldsymbol{u}_m$ is only a function of $\boldsymbol{u}_m$. Substituting (31) into (30) gives:

$$\begin{aligned} \Delta \boldsymbol{u}_m^T \boldsymbol{R}_m + (\boldsymbol{P} \Delta \boldsymbol{u}_m)^T \boldsymbol{R}_s &= 0 \\ \Delta \boldsymbol{u}_m{}^T (\boldsymbol{R}_m + \boldsymbol{P}^T \boldsymbol{R}_s) &= 0 \end{aligned} \tag{32}$$

For the nontrivial solution $\Delta \boldsymbol{u}_m \neq \boldsymbol{0}$ then $\boldsymbol{R}_m$ is related to $\boldsymbol{R}_s$ as:

$$\boldsymbol{R}_m = -\boldsymbol{P}^T \boldsymbol{R}_s \tag{33}$$

Substituting (28) into (33) for $\boldsymbol{R}_s$ and subsequently into (27) gives the reduced system of equations:

$$\boldsymbol{\mathcal{R}}_f = \boldsymbol{\mathcal{F}}_f^{int} - \boldsymbol{\mathcal{F}}_f^{ext} = \boldsymbol{0} \tag{34}$$

$$\boldsymbol{\mathcal{R}}_{ms} = \boldsymbol{\mathcal{F}}_m^{int} - \boldsymbol{\mathcal{F}}_m^{ext} + \boldsymbol{P}^T \left(\boldsymbol{\mathcal{F}}_s^{int} - \boldsymbol{\mathcal{F}}_s^{ext} \right) = \boldsymbol{0} \tag{35}$$

$$\boldsymbol{\mathcal{R}}_p = \boldsymbol{\mathcal{F}}_p^{int} - \boldsymbol{\mathcal{F}}_p^{ext} - \boldsymbol{R}_P = \boldsymbol{0} \tag{36}$$

$$\boldsymbol{u}_s = \boldsymbol{f}_{ms}(\boldsymbol{u}_m) \tag{37}$$

This system of equations can thus be solved since the number of unknowns and equations are now equal. Reaction forces $\boldsymbol{R}_m$ and $\boldsymbol{R}_s$ can be computed after (27) and (28) have been solved.

2.6.3. Lagrangian MPCs

For both linear and non-linear MPCs, the Lagrangian approach increases the size of the system by the number of constraint equations in addition to introducing an unknown Lagrange multiplier per constraint. This has the negative effect of potentially increased solution times. Furthermore, the addition of the constraint equations to the system results in increased bandwidth. It is also an indefinite system due to the introduction of zero elements on the diagonal. This was previously a great deterrent to the use of the Lagrangian approach. However, evolutions in linear algebra have resulted in improved computational efficiency [46, 47]. The system, therefore, cannot be solved using conventional $\mathbf{LU}$ or Cholesky $\mathbf{LL}^T$ decompositions, but rather requires the less efficient $\mathbf{LDL}^T$ decomposition [46].

The most significant advantage offered by the Lagrangian approach is its ease of implementation when compared to the MSEM. Since all equations are solved and no DOFs are eliminated, constraints that affect the same DOFs are handled without special treatment. This means that a complex selection algorithm is not required to determine the location of and specific manner in which a constrained DOF is handled. Even though the specification of constraints using the Lagrangian approach is easier than for the MSEM, care must still be taken to avoid matrix singularities resulting from linear dependency between constraints.

For non-linear MPCs, the Lagrangian approach only satisfies the constraint equations when the solution process is complete. However, this satisfaction of the constraints is exact to within the tolerance specified by the user. This relaxation of the constraints means that it is less likely to fail to converge when compared with the MSEM approach. It also means that high curvature in a constraint does not require special attention as in the MSEM approach.

Another feature worth noting is that the no-work requirement for MPCs is automatically satisfied using the Lagrangian approach. This is compared to the fact that this condition is explicitly imposed in the one formulation of the MSEM approach. The Lagrangian approach therefore presents itself as a good option to constrain boundary nodes to the boundary surface. The accurate consistent tangent will allow for obtaining accurate analytical sensitivities for the system.

2.6.3.1. Lagrangian system of equations

The system of residual equations in (19) to (22) are shown again in (38) to (41) for convenience for free $\boldsymbol{\mathcal{R}}_f$, boundary $\boldsymbol{\mathcal{R}}_c$ and prescribed $\boldsymbol{\mathcal{R}}_p$ DOFs and the MPC constraint equations $\boldsymbol{f}_{mpc}(\boldsymbol{\mathcal{X}}_c) = \boldsymbol{0}$.

$$\boldsymbol{\mathcal{R}}_f = \boldsymbol{\mathcal{F}}_f^{int} - \boldsymbol{\mathcal{F}}_f^{ext} = \boldsymbol{0} \tag{38}$$

$$\boldsymbol{\mathcal{R}}_c = \boldsymbol{\mathcal{F}}_c^{int} - \boldsymbol{\mathcal{F}}_c^{ext} = \boldsymbol{0} \tag{39}$$

$$\boldsymbol{\mathcal{R}}_p = \boldsymbol{\mathcal{F}}_p^{int} - \boldsymbol{\mathcal{F}}_p^{ext} - \boldsymbol{R}_P = \boldsymbol{0} \tag{40}$$

$$\boldsymbol{\lambda}^T \boldsymbol{f}_{mpc} = \boldsymbol{0} \tag{41}$$

Here $\boldsymbol{f}_{mpc}$ contributes additional equations to solve and are incorporated using the Lagrange multipliers $\boldsymbol{\lambda}$ in the form of $\boldsymbol{\lambda}^T \boldsymbol{f}_{mpc} = \boldsymbol{0}$. External forces associated with prescribed displacements are equal to zero, i.e. $\boldsymbol{\mathcal{F}}_p^{ext} = \boldsymbol{0}$ and are handled as reaction forces, $\boldsymbol{R}_P$.

The Lagrangian MPC method is well understood [42, 45] and is implemented here as presented by Kok and Wilke [45]. For the Lagrangian approach, the system potential energy is augmented with the MPC equations in (41). It is known that the potential energy for elastic media is given by the sum of the strain energy for the system or equivalently by work conducted by external forces. Consider a structure in its

initial configuration $\boldsymbol{\mathcal{X}}^i$ that ends in the final configuration $\boldsymbol{\mathcal{X}}^e = \boldsymbol{\mathcal{X}}^i + \boldsymbol{u}$, where $\boldsymbol{u} = \Delta\boldsymbol{\mathcal{X}}$. The potential energy in its discretised and partitioned form is then given by:

$$\Pi(\boldsymbol{u}_f, \boldsymbol{u}_c) = \oint_{\boldsymbol{x}_f^i}^{\boldsymbol{x}_f^e} \left(\boldsymbol{\mathcal{F}}_f^{int}\right)^T d\boldsymbol{u}_f + \oint_{\boldsymbol{x}_c^i}^{\boldsymbol{x}_c^e} \left(\boldsymbol{\mathcal{F}}_c^{int}\right)^T d\boldsymbol{u}_c - \left(\boldsymbol{\mathcal{F}}_f^{ext}\right)^T \boldsymbol{u}_f - (\boldsymbol{\mathcal{F}}_c^{ext})^T \boldsymbol{u}_c \tag{42}$$

By augmenting (42) with the MPC equations in (41) the Lagrangian functional is given as:

$$\mathcal{L}\left(\boldsymbol{u}_f, \boldsymbol{u}_c, \lambda\right) = \oint_{\boldsymbol{x}_f^i}^{\boldsymbol{x}_f^e} \left(\boldsymbol{\mathcal{F}}_f^{int}\right)^T d\boldsymbol{u}_f + \oint_{\boldsymbol{x}_c^i}^{\boldsymbol{x}_c^e} \left(\boldsymbol{\mathcal{F}}_c^{int}\right)^T d\boldsymbol{u}_c - \left(\boldsymbol{\mathcal{F}}_f^{ext}\right)^T \boldsymbol{u}_f - (\boldsymbol{\mathcal{F}}_c^{ext})^T \boldsymbol{u}_c + \boldsymbol{\lambda}\left(\boldsymbol{f}_{mpc}(\boldsymbol{\mathcal{X}}_c)\right) \tag{43}$$

The components of $\boldsymbol{\mathcal{R}}_p$ are ignored since the displacements are prescribed and the update displacement $\boldsymbol{u}_p = \boldsymbol{0}$. It is noted that $\boldsymbol{\lambda}$ is related to a reaction force that enforces the constraint when compared to $\left(\boldsymbol{\mathcal{F}}_f^{ext}\right)^T \boldsymbol{u}_f$ and $(\boldsymbol{\mathcal{F}}_c^{ext})^T \boldsymbol{u}_c$. Since $(\boldsymbol{\mathcal{F}}_c^{ext})^T \boldsymbol{u}_c = 0$ the potential energy from (42) is not altered, i.e. no additional work is done nor energy dissipated from the system.

Considering the first-order optimality condition, a stationary point is obtained for (43) as:

$$\frac{d\mathcal{L}}{d\boldsymbol{u}_f} = \boldsymbol{\mathcal{F}}_f^{int} - \boldsymbol{\mathcal{F}}_f^{ext} = \boldsymbol{0} \tag{44}$$

$$\frac{d\mathcal{L}}{d\boldsymbol{u}_c} = \boldsymbol{\mathcal{F}}_c^{int} - \boldsymbol{\mathcal{F}}_c^{ext} + \left(\frac{\mathrm{d}\boldsymbol{f}_{mpc}(\boldsymbol{\mathcal{X}}_c)}{d\boldsymbol{u}_c}\right)^{\mathrm{T}} \boldsymbol{\lambda} = \boldsymbol{0} \tag{45}$$

$$\frac{d\mathcal{L}}{d\boldsymbol{\lambda}} = \boldsymbol{f}_{mpc} = \boldsymbol{0} \tag{46}$$

The solution of (44)-(46) gives the equilibrium solution.

The unknowns that must be solved for are therefore the prescribed reaction forces $\boldsymbol{R}_p$, nodal displacements at the free $\boldsymbol{u}_f$ and MPC $\boldsymbol{u}_c$ dofs, and the Lagrange multipliers $\boldsymbol{\lambda}$ associated with the MPCs. $\boldsymbol{\lambda}$ is the total reaction force at the constraint and the components can be recovered from $\boldsymbol{f}_{mpc}$ and $\boldsymbol{\lambda}$.

2.7. Selection of truss system solution method

The solution to the truss equilibrium is, in fact, the minimisation of a higher-order energy problem $\Pi(\boldsymbol{\mathcal{X}})$ of which the truss equilibrium equations $\boldsymbol{F}(\boldsymbol{\mathcal{X}})$ are the first-order derivatives. Many optimisation techniques are available for obtaining the solution to the minimum of $\Pi(\boldsymbol{\mathcal{X}})$ [22] that include zero order, first order, and second-order methods.

There are two ways of obtaining the minimum. The first is to optimise based only on the cost function values. This is good for problems where sensitivities are inaccurate or unavailable [22]. The second is to optimise the based on the optimality conditions. For this scenario the optimality conditions are:

$$\frac{\mathrm{d}\Pi(\boldsymbol{\mathcal{X}})}{d\boldsymbol{\mathcal{X}}} = \boldsymbol{F}(\boldsymbol{\mathcal{X}}) = \mathbf{0} \tag{47}$$

$$\frac{\mathrm{dd}\Pi(\boldsymbol{\mathcal{X}})}{dd\boldsymbol{\mathcal{X}}} = \frac{d\boldsymbol{F}(\boldsymbol{\mathcal{X}})}{d\boldsymbol{\mathcal{X}}} > \mathbf{0} \tag{48}$$

This states that the first-order gradient must be equal to zero and the second-order (curvature) of the system must be positive. This is a root-finding problem where the solution to $\boldsymbol{F}(\boldsymbol{\mathcal{X}}) = \mathbf{0}$ is obtained.

Many methods are available to find the roots of a set of equations [22]. A simple and effective option is Newton's method, which utilises the sensitivities of the system to converge more quickly to the root. This method is generally used in non-linear FEA applications, as it efficiently obtains very accurate solutions to the non-linear problem. The Newton method utilizes the consistent tangent $\frac{d\boldsymbol{F}(\boldsymbol{\mathcal{X}})}{d\boldsymbol{\mathcal{X}}}$ of the truss equilibrium system $\boldsymbol{F}(\boldsymbol{\mathcal{X}})$ which in turn makes these sensitivities available to compute the mesh sensitivities $\frac{d\boldsymbol{\mathcal{X}}}{d\mathbf{x}}$. The Newton method will be used in the development of this mesher. This will be discussed in detail in §3.3.

2.8. Summary

DistMesh was discussed and shown to be a simple implementation of an unstructured meshing program. Given its truss structure implementation, it allows for easy modification to a gradient-based mesher that can offer analytical sensitivities of the mesh nodal coordinates $\boldsymbol{\mathcal{X}}$.

In §2.5, the applicability of using contact methods to describe the movement of nodes along a boundary was considered. The non-linear nature of the boundary force relationship was found to mean that any attempt at obtaining sensitivities would require that the system be re-converged for each sensitivity evaluated.

MPC methods were seen to exactly enforce constraints as opposed to approximately as with contact and penalty methods. MPC methods, therefore, allow the boundary nodes of the truss mesher to be constrained to the boundary analytically, thereby allowing for the boundary to be satisfied exactly.

Given that the analytical relationship between the MPC nodes and the boundary geometry is known, the MPC contributions can be used in obtaining analytical or semi-analytical sensitivities for mesh deformation. This will allow the sensitivities of the mesh to include the movement of nodes along the boundaries as well as internally. The exact manner in which this is accomplished is discussed further in Chapter 5.

MSEM and Lagrangian MPCs are shown to be best suited to the intent of this study which is to produce sensitivities for further use in gradient-based solvers for shape optimisation. The systems required to be solved are for the Lagrangian method is:

$$\frac{d\mathcal{L}}{d\boldsymbol{u}_f} = \boldsymbol{\mathcal{F}}_f^{int} - \boldsymbol{\mathcal{F}}_f^{ext} = \mathbf{0} \tag{49}$$

$$\frac{d\mathcal{L}}{d\boldsymbol{u}_c} = \boldsymbol{\mathcal{F}}_c^{int} - \boldsymbol{\mathcal{F}}_c^{ext} + \left(\frac{d\boldsymbol{f}_{mpc}(\boldsymbol{u}_c)}{d\boldsymbol{u}_c}\right)^{\mathrm{T}} \boldsymbol{\lambda} = \mathbf{0} \tag{50}$$

$$\frac{d\mathcal{L}}{d\boldsymbol{u}_\lambda} = \boldsymbol{f}_{mpc} = \mathbf{0} \tag{51}$$

and for the MSEM:

$$\boldsymbol{\mathcal{R}}_f = \boldsymbol{\mathcal{F}}_f^{int} - \boldsymbol{\mathcal{F}}_f^{ext} = \mathbf{0} \tag{52}$$

$$\boldsymbol{\mathcal{R}}_{ms} = \boldsymbol{\mathcal{F}}_m^{int} - \boldsymbol{\mathcal{F}}_m^{ext} + \boldsymbol{P}^T\left(\boldsymbol{\mathcal{F}}_s^{int} - \boldsymbol{\mathcal{F}}_s^{ext}\right) = \mathbf{0} \tag{53}$$

$$\boldsymbol{\mathcal{R}}_p = \boldsymbol{\mathcal{F}}_p^{int} - \boldsymbol{\mathcal{F}}_p^{ext} - \boldsymbol{R}_P = \mathbf{0} \tag{54}$$

In §2.7 Newton's method was shown to be a suitable solution method for solving these types of problems. It also requires analytical sensitivities for the system in order to obtain convergence. As will be discussed in Chapter 5, these sensitivities are required to obtain accurate mesh sensitivities $d\boldsymbol{X}/d\mathbf{x}$.

CHAPTER 3
NEWTON TRUSS AND MPC IMPLEMENTATION

"Engineering is the art of modelling materials we do not wholly understand, into shapes we cannot precisely analyse so as to withstand forces we cannot properly assess, in such a way that the public has no reason to suspect the extent of our ignorance."— Dr. AR Dykes

3.1. Introduction

In this chapter implementation of Newton's method to solve the truss equilibrium equations and the MPC equilibrium equations will be developed. The truss equilibrium equations in (17) (see §2.3) were first solved using the prescribed boundary approach given by (18).

In §3.2 the truss member force equations are developed and the numerous options for the force function discussed. Force function selection is shown in §3.2.2 and the testing is discussed in APPENDIX C. The truss equilibrium with the Newton-Raphson method is shown in §3.3, followed in §3.4 by the proof of the correct implementation of sensitivities for the Newton-Raphson method by displaying quadratic convergence.

Having proven the correct implementation of the truss formulations the implementation is expanded to include the MPC boundary nodes. The Newton-Raphson implementations for the Lagrangian and MSEM MPC methods are developed in §3.6 and §3.6.2 respectively. The proof of implementation for the two methods is demonstrated in §3.8. Evaluation of methods for solving the two resulting linear systems is then discussed in §3.7.

3.2. Truss member force function

The general truss force F_{truss}, or spring force equation acting on a node is given by:

$$-F_{int} = F_{truss} = k'_{truss}\frac{L_{des} - L_c}{L_e^*} = k'_{truss}\frac{\Delta L}{L_{des}^*} \tag{55}$$

where F_{int} is the systems internal load on the node at the end of the truss, L_{des} is the desired or free equilibrium length of the truss, L_c is the current length of the truss, k'_{truss} is the stiffness coefficient of the truss and L_e^* is a placeholder length. For real trusses or springs $L_e^* = L_{des}$. This is however not complete since this gives only the force of the truss and not the force components for each direction at either end.

3.2.1.1. Desired length function

At this point the function for determining the desired truss length is relevant. The desired lengths for trusses, L_{des}, can be calculated via a function $h(p_{mid})$ requiring the midpoint of the truss, p_{mid}, as an input. See §2.2.4. This then returns a desired length value to the truss:

$$L_{des} = h(p_{mid}) \tag{56}$$

It is noted that (56) is the same in principle as given in the original implementation of DistMesh by Persson and Strang [33]. For this implementation, it can be given as an absolute desired length, which would allow for better user control and removes the discontinuity associated with the scaling factor as shown in (15).

3.2.1.2. Truss system implementation

Given local nodal coordinates x_1, x_2 and x_3 at node 1 and x_4, x_5 and x_6 at node 2 corresponding to the X, Y, and Z dimensions respectively, the current length L_c of the truss member is calculated as:

$$L_c = \sqrt{(\Delta X)^2 + (\Delta Y)^2 + (\Delta \mathrm{Z})^2} \tag{57}$$

where:

$$\Delta \mathrm{X} = x_4 - x_1 \tag{58}$$

$$\Delta \mathrm{Y} = x_5 - x_2 \tag{59}$$

$$\Delta \mathrm{Z} = x_6 - x_3 \tag{60}$$

The unit vector components l, m, n of the truss are given respectively for the X, Y, and Z dimensions:

$$l = \frac{\Delta X}{L_c} = \frac{x_4 - x_1}{L_c} \tag{61}$$

$$m = \frac{\Delta Y}{L_c} = \frac{x_5 - x_2}{L_c} \tag{62}$$

$$n = \frac{\Delta \mathrm{Z}}{L_c} = \frac{x_6 - x_3}{L_c} \tag{63}$$

These unit vector components give the truss's nodal force contributions as:

$$F_{int}(\boldsymbol{\mathcal{X}}) = \begin{Bmatrix} -l \\ -m \\ -n \\ l \\ m \\ n \end{Bmatrix} F_{truss} \tag{64}$$

where the components of the vector F_{int} are force components acting on the nodes at either end of the truss. The top three components $\{-l \quad -m \quad -n\}^T$ act on node 1 and the bottom three $\{l \quad m \quad n\}^T$ act equally and opposite on node 2. This is similar to the standard truss FEM derivation method [30].

3.2.2. Truss member stiffness formulations

Since the truss system of the mesh is not required to satisfy some real physical laws, the truss stiffness k'_{truss} from (55) can be formulated as desired; several are considered here to facilitate comparison of truss stiffness variations.

3.2.2.1. Linear delta over the desired length of the truss

The simplest formulation considers a linear response with unit stiffness $k'_{truss} = 1$. This is obtained if the truss length is allowed to be the desired or undeformed length of the truss, $L_e^* = L_{des}$. This selection ensures that the truss strain $\epsilon = \frac{L_{des} - L_c}{L_{des}}$ has the same magnitude as the truss force and is, therefore, the true description for the compressed bar. Since L_{des} is a known constant, F_{truss} only depends on L_c in the numerator:

$$F_{truss} = \frac{L_{des} - L_c}{L_{des}} \tag{65}$$

3.2.2.2. Linear delta over the current length of the truss

Alternatively, L_e^* is chosen to be the current length L_c and $k_{truss} = 1$. The truss force equation becomes non-linear due to the function $L_c(u)$ in the denominator. The consequence of this is that the truss strain $\epsilon = \frac{L_{des} - L_c}{L_c}$ is no longer the true strain. This is the formulation used in DistMesh. In DistMesh the use of the current length in the denominator results in larger forces in compressed trusses. This allows the forward-Euler implementation to progress more quickly to a solution when the system has high compression. The modified formulation of the force response is then given by:

$$F_{truss} = \frac{L_{des} - L_c}{L_c} \tag{66}$$

3.2.2.3. Exponential over the desired length of the truss

Large discrepancies in truss length lead to numerical instability and the formation of sliver elements. To penalise such large discrepancies, exponentially non-linear response functions were created:

Selecting $k'_{truss} = e^{\left(S_1\left(\frac{L_{des} - L_c}{L_{des}}\right)^2\right)}$ and $L_e^* = L_{des}$ as the desired length with S_1 as the exponential scaling factor, the force response is given as:

$$F_{truss} = \frac{L_{des} - L_c}{L_{des}} e^{\left(S_1\left(\frac{L_{des}-L_c}{L_{des}}\right)^2\right)} \tag{67}$$

As $S_1 \to 0$ the truss stiffness $k'_{truss} \to 1$ and the formulation reduces to (65). This is evident in the response curves that follow in §3.2.2.5 for various values of S_1. When $S_1 < 0$ the impact of the exponential scaling is to create a softening stiffness in the truss.

3.2.2.4. Exponential scaled over the current length of the truss

Selecting $k'_{truss} = e^{\left(S_1\left(\frac{L_{des}-L_c}{L_c}\right)^2\right)}$ and $L_e^* = L_c$ as the current length, the force response is given as:

$$F_{truss} = \frac{L_{des} - L_c}{L_c} e^{\left(S_1\left(\frac{L_{des}-L_c}{L_c}\right)^2\right)} \tag{68}$$

3.2.2.5. Comparison of the force response curves for the various functions

Figure 10 shows the various force response curves given by the formulations discussed previously. These are plotted against the compression ratio of the truss $\frac{L_c}{L_{des}}$. A value of $\frac{L_c}{L_{des}} < 1$ represents a compressed truss, $\frac{L_c}{L_{des}} = 1$ indicates equilibrium and $\frac{L_c}{L_{des}} > 1$ signifies tension. The deviation from 1 represents the deviation from the desired length. In Figure 10, all of the stiffness curves for the various truss stiffness formulae (65) through (68) are plotted. Several values for the exponential scaling factor S_1 are used.

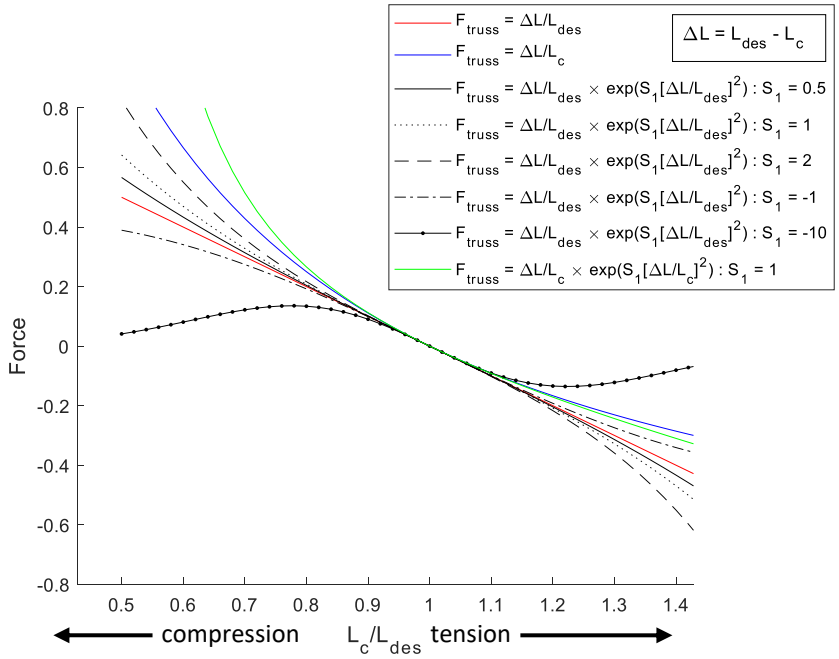


Figure 10 Truss force response for various truss stiffness formulae

It is evident from Figure 10 that trusses with the desired length L_{des} as the denominator, (65) and (67), have force responses that are symmetric in compression and tension. In contrast, the formulations with the current length L_c as the denominator, (66) and (68), have different force responses under compression to those under tension for a given deviation from the desired length. This results in the trusses being more difficult to compress than extend.

For all exponential formulations where $S_1 > 0$, the force response increases exponentially with greater deviations from the desired length resulting in an increased force magnitude compared to their linear counterparts. When $S_1 < 0$ the force response is less affected by increasing deviations from the desired length and begins to depart from expected behaviour. In particular, as shown by the curve for $S_1 = -10$, the force response initially increases, as expected, with increasing deviation from the desired length. There is however a point beyond which further compression or tensioning results in a decreased force response. This results in a condition where the maximum force the truss can support is limited, beyond which increasing deviations from the desired length will result in snap-through. In the curve for $S_1 = -10$ the point of maximum force corresponds to values of $L_c/L_{des} \approx 0.78$ and 1.22.

3.2.3. Selection of force function

The simplest force function given in §3.2.2.1 is selected. This performed overall the best of the functions investigated. A full discussion of the comparison is given in APPENDIX C. The function on average produced the best convergence properties and gave comparably the best element qualities (as defined in §1.4.3). It is repeated here for clarity:

$$F_{truss} = \frac{L_{des} - L_c}{L_{des}} \tag{69}$$

3.3. Newton truss implementation

The truss system will be solved by implementing Newton's method, which is known to be quadratically convergent. This method requires the calculation of the consistent tangent matrix comprising the truss sensitivities. The availability of these sensitivities is important for obtaining the mesh sensitivities $d\boldsymbol{\mathcal{X}}/d\mathbf{x}$ required for gradient-based shape optimisation. This will be discussed in detail in Chapter 5.

From the truss equilibrium system given in §2.3 (17) the internal forces $\boldsymbol{\mathcal{F}}^{int}$ for the truss system are given by $\boldsymbol{F}(\boldsymbol{\mathcal{X}})$ from (64). Since the truss members are not subject to any external forces $\boldsymbol{\mathcal{F}}^{ext}$, the system in (18) reduces to:

$$\boldsymbol{\mathcal{R}} = \boldsymbol{\mathcal{F}}^{int} = \boldsymbol{F}(\boldsymbol{\mathcal{X}}) = \mathbf{0} \tag{70}$$

This can then be solved iteratively using Newton's method [22]. To determine the update for the nodal coordinate, $\boldsymbol{\mathcal{X}}$, required to reduce the residuals to zero, the Newton step is given by:

$$\boldsymbol{\mathcal{X}}_{k+1} = \boldsymbol{\mathcal{X}}_k - \left(\frac{d\boldsymbol{F}}{d\boldsymbol{\mathcal{X}}}\right)_k^{-1} \boldsymbol{F}_k$$
$$\boldsymbol{\mathcal{X}}_{k+1} = \boldsymbol{\mathcal{X}}_k + \boldsymbol{u}_k \tag{71}$$

where $\frac{d\boldsymbol{F}}{d\boldsymbol{\mathcal{X}}}$ and $\boldsymbol{F}_k$ must be computed at every iteration. The system update $\boldsymbol{u} = \Delta\boldsymbol{\mathcal{X}}$ can therefore be solved as:

$$\left(\frac{d\boldsymbol{F}}{d\boldsymbol{\mathcal{X}}}\right)_k \boldsymbol{u}_k = -\boldsymbol{F}_k \tag{72}$$

The Newton iteration number subscript k will be dropped going forward to simplify reading. The consistent tangent $\frac{d\boldsymbol{F}}{d\boldsymbol{\mathcal{X}}}$ is assembled for the system as:

$$\frac{d\boldsymbol{F}}{d\boldsymbol{\mathcal{X}}} = \begin{bmatrix} \frac{dF_1}{d\mathcal{X}_1} & \frac{dF_1}{d\mathcal{X}_2} & \cdots & \frac{dF_1}{d\mathcal{X}_n} \\ \frac{dF_2}{d\mathcal{X}_1} & \frac{dF_2}{d\mathcal{X}_2} & \cdots & \frac{dF_2}{d\mathcal{X}_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{dF_n}{d\mathcal{X}_1} & \frac{dF_1}{d\mathcal{X}_2} & \cdots & \frac{dF_n}{d\mathcal{X}_n} \end{bmatrix} \tag{73}$$

3.3.1. Solution to internal nodes

Since certain DOFs are prescribed the system can be partitioned and solved as in (18):

$$\boldsymbol{\mathcal{R}}_f = \boldsymbol{F}_f^{int} = \boldsymbol{0}$$
$$\boldsymbol{\mathcal{R}}_p = \boldsymbol{F}_p^{int} - \boldsymbol{R}_p = \boldsymbol{0} \tag{74}$$

where the subscript f denotes free DOFs and subscript p denotes prescribed DOFs. $\boldsymbol{R}_p$ is the reaction force associated with the prescribed displacements as discussed in §2.4. The Newton update step (72) is partitioned as:

$$\begin{bmatrix} \frac{d\boldsymbol{F}_f}{d\boldsymbol{\mathcal{X}}_f} & \frac{d\boldsymbol{F}_f}{d\boldsymbol{\mathcal{X}}_p} \\ \frac{d\boldsymbol{F}_p}{d\boldsymbol{\mathcal{X}}_f} & \frac{d\boldsymbol{F}_p}{d\boldsymbol{\mathcal{X}}_p} \end{bmatrix} \begin{Bmatrix} \boldsymbol{u}_f \\ \boldsymbol{u}_p \end{Bmatrix} = - \begin{Bmatrix} \boldsymbol{\mathcal{R}}_f \\ \boldsymbol{\mathcal{R}}_p \end{Bmatrix} \tag{75}$$

Since the prescribed DOFs are known, the free DOF updates are solved as:

$$\frac{d\boldsymbol{F}_f}{d\boldsymbol{\mathcal{X}}_f}\boldsymbol{u}_f = -\boldsymbol{\mathcal{R}}_f - \frac{d\boldsymbol{F}_f}{d\boldsymbol{\mathcal{X}}_p}\boldsymbol{u}_p \tag{76}$$

Since there is no movement at this stage, prescribed displacement updates u_p are equal to zero, i.e. $\boldsymbol{u_p} = \boldsymbol{0}$, the system can be solved as:

$$\frac{d\boldsymbol{F}_f}{d\boldsymbol{X}_f}\boldsymbol{u}_f = -\boldsymbol{\mathcal{R}}_f \tag{77}$$

where the components $\boldsymbol{\mathcal{R}}_f$ and $\frac{d\boldsymbol{F}_f}{d\boldsymbol{X}_f}$ are first computed, followed by solution of $\boldsymbol{u}_f$ and the nodal coordinates $\boldsymbol{X}$ are updated. Every Newton iteration requires these computations to be completed due to the changing nodal coordinates. The reaction forces, $\boldsymbol{R}_p$, can be recovered from (74), if required, after solution of the system. This is not compulsory and is hence ignored in this implementation.

3.4. Internal node convergence investigation

3.4.1. 2D internal node investigation

An investigation was conducted to determine the effectiveness of the various truss formulations detailed in §3.2.2. and the Newton's method implementation. For the linear system, the built-in MATLAB 2018b "backslash" function was used to obtain a solution. The following initial conditions were created:

A circular distance function with unit radius was used to determine the distance of the point under consideration to the boundary:

$$f_{\mathrm{d}\Omega}\,(\boldsymbol{p}_i) = \|\boldsymbol{p}_i\|_2 - 1 \tag{78}$$

For 2D the coordinates of the i^{th} point are extracted from the mesh nodal coordinate vector $\boldsymbol{X}$ at the positions denoted in the subscripts as follows:

$$\boldsymbol{p_i} = \{\mathcal{X}_{i\times 2-1}, \mathcal{X}_{i\times 2}\} \tag{79}$$

Similarly, for 3D the coordinates of the point coordinates are given by:

$$\boldsymbol{p_i} = \{\mathcal{X}_{i\times 3-2}, \mathcal{X}_{i\times 3-1,}\, \mathcal{X}_{i\times 3}\} \tag{80}$$

The initial mesh was seeded using equilateral triangles with the initial uniform edge length size, L_0:

$$L_0 = 0.05 \tag{81}$$

Here, the initial seeding length L_0 should not be confused with the desired truss length L_{des}.

As shown in Figure 11, all nodes located outside of a tolerance of $-0.7L_0$ interior to the boundary of the circle were removed and the boundary was seeded using fixed nodes. This forced the trusses to remain inside of the boundary. Nodes were placed along the circumference with a spacing equal to L_0 as shown by the * symbols in Figure 11.

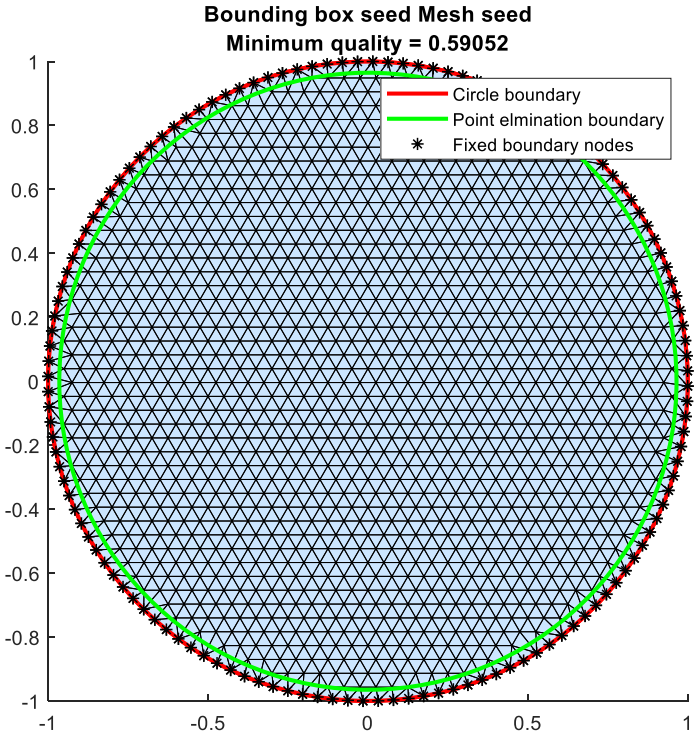


Figure 11 Initial seed mesh for truss convergence study in 2D

Having defined the initial seed mesh shown in Figure 11, the equilibrium for each of the considered truss formulations is solved. Several scaling factors for the deviation from desired truss length were analysed to allow for a good comparison between the formulations. The results of the study are discussed in §3.5 and see APPENDIX C for the full investigation.

3.4.2. 3D internal node investigation

To solve the linear system, the built-in MATLAB 2018b "backslash[6]" function was used to obtain a solution. This investigation was performed retrospectively once a high-quality surface mesh was obtained using the MPC implementation as depicted in Figure 12. These resulting surface nodes were then used to create the prescribed boundary nodes. An initial mesh was then created that was able to converge for both methods without the need for retriangulation. The desired length of $L_{des} = 0.2$ was used for the study and the internal domain was seeded with an initial length, $L_0 = 0.2$.

[6] The "backslash" function in MATLAB is a general solver for linear systems of the form $Ax = b$ or $Ax = B$ for x. The solver evaluates the system A for its various properties, positive definiteness, rank, condition, etc. Based on these results it selects suitable methods for solving the system. The "backslash" function will solve least squares solutions for under or over determined systems.

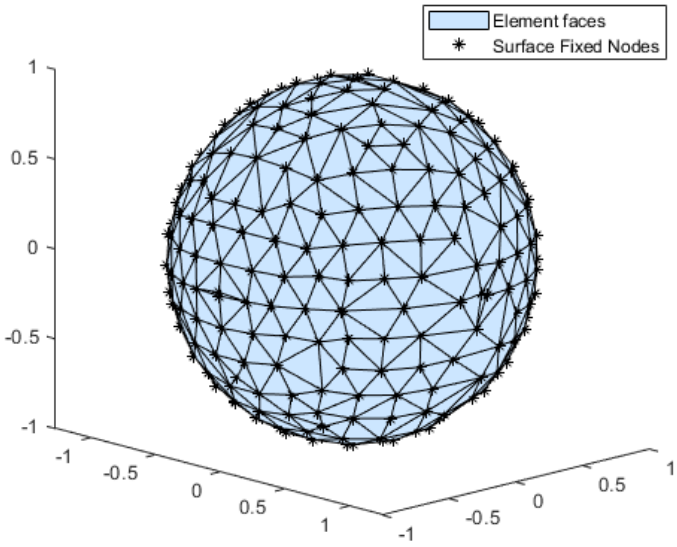


Figure 12 Initial seed mesh for truss convergence study in 3D

3.4.3. Verifying quadratic convergence of truss implementation

Demonstration of quadratic convergence is essential since it proves the correct implementation of the consistent tangent. Any errors in the consistent tangent will carry through and result in unacceptable errors in the sensitivities.

From Table 1 it is clear that both the 2D and 3D implementations of Newton's method converge quadratically. This is demonstrated by subsequent iterations of the norm of the update squaring approximately as the sequence converges $\frac{\|\boldsymbol{u}_f\|_k}{\|\boldsymbol{u}_f\|_{k-1}^2}$. A value between 0 and 1 shows the correct implementation of the method for the system of equations. At this point the consistent tangent is exact and since the boundary nodes are prescribed, the exact analytical sensitivities can be obtained for the internal mesh deformation [7].

Table 1 Convergence error norm results for Newton implementations for 2D and 3D

Iteration	2D $\|\boldsymbol{u}_f\|$	2D Convergence $\|\boldsymbol{u}_f\|_k/\|\boldsymbol{u}_f\|_{k-1}^2$	3D $\|\boldsymbol{u}_f\|$	3D Convergence $\|\boldsymbol{u}_f\|_k/\|\boldsymbol{u}_f\|_{k-1}^2$
1	4.59×10^{-1}	-	6.28×10^{-1}	-
2	1.06×10^{-1}	0.50	2.35×10^{-2}	0.06
3	4.85×10^{-3}	0.43	1.25×10^{-3}	2.26
4	3.87×10^{-5}	1.65	1.28×10^{-6}	0.82
5	7.74×10^{-10}	0.52	8.98×10^{-13}	0.55

3.5. Snap-through due to highly compressed systems

During the development process, high compression systems showed instability and were prone to nodes snapping through. Figure 13 shows the effects of the highly compressed system. In (a), the effect of slight compression is shown where only one node had snapped through; this could generally be resolved through remeshing. In (b), a highly compressed system where the trusses were compressed to 77% of their free lengths demonstrates the tendency towards severe distortion in one Newton step. It is for this reason that update step size control methods for the system were investigated and are discussed in APPENDIX F.

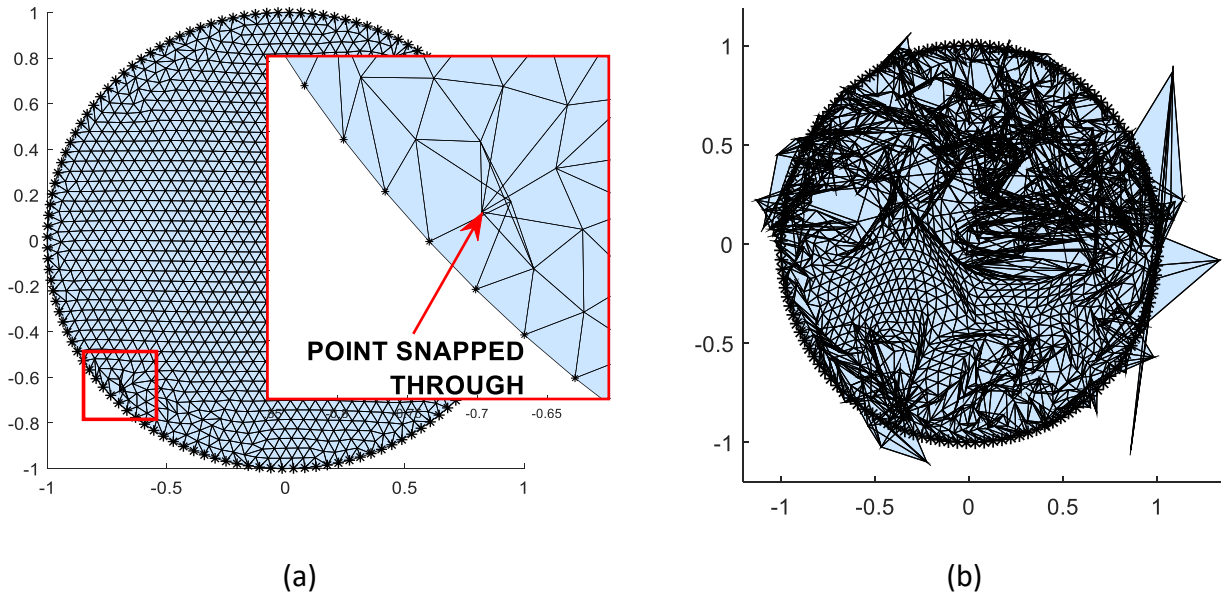


Figure 13 Examples of problems associated with highly compressed systems
(a) Snap through of a single node due to high compression in the element (b) General mesh snap-through due to unstable update with average truss compressions of 23%

Limiting the Newton update step size based on the largest single update was found to be able to control the system with reasonable effectiveness. The method employed by Persson and Strang in DistMesh [33] as described in §2.2.4 (15) proved to be insensitive to poor length scaling as expected. Additionally, this method benefitted from being able to propel the mesh to the boundary and corners. Its only drawback was that it introduced discrete length changes, even for uniform length systems, which impacts the Newton's method convergence rate. Line searching to minimise the energy of the system was found to be ineffective due to the presence of many local minima, introduced by progressively more snapped through nodes.

Since the method employed in DistMesh results in discrete changes to edge lengths, it encumbers the performance of Newton's method. This may be mitigated by permitting the allocation of desired lengths only after a predefined number of iterations. For this implementation, the Newton update step

size limiting method was used to retain the quadratic convergence properties. For the MSEM, this became a function of the master update $\boldsymbol{u}_m$. This is discussed further in §4.3.3.

3.6. MPC Implementations for rolling boundary nodes

Having proved the correct Newton's method implementation for the internal nodes, the limitation of fixed boundary nodes is removed. This is accomplished through the use of the Lagrangian and MSEM MPC methods discussed in §2.6. These methods allow the nodes to move along but not away from the boundary.

3.6.1. Implementation of Lagrangian MPC method

The Lagrangian MPC method is implemented according to the approach described by Kok *et al.* [45]. The set of residual equations for the implicit solution is given in §2.6.3.1 (44)-(46). Since all residuals, $\boldsymbol{\mathcal{R}}$, can be viewed as a function of the truss internal forces $\boldsymbol{F}^{int}(\boldsymbol{\mathcal{X}}(\boldsymbol{u}))$ and $\frac{d\boldsymbol{\mathcal{X}}}{d\boldsymbol{u}} = 1$, this system can be solved for the k^{th} iteration using Newton's method in the form of:

$$\begin{bmatrix} \frac{d\boldsymbol{F}_f^{int}}{d\boldsymbol{\mathcal{X}}_f} & \frac{d\boldsymbol{F}_f^{int}}{d\boldsymbol{\mathcal{X}}_c} & \boldsymbol{0} \\ \frac{d\boldsymbol{F}_{\partial\Omega}^{int}}{d\boldsymbol{\mathcal{X}}_f} & \frac{d\boldsymbol{F}_c^{int}}{d\boldsymbol{\mathcal{X}}_c} + \frac{d^2\boldsymbol{f}_{mpc}}{d\boldsymbol{\mathcal{X}}_c{}^2}\boldsymbol{\lambda} & \frac{d\boldsymbol{f}_{mpc}}{d\boldsymbol{\mathcal{X}}_c}^T \\ \boldsymbol{0} & \frac{d\boldsymbol{f}_{mpc}}{d\boldsymbol{\mathcal{X}}_{\partial\Omega}} & \boldsymbol{0} \end{bmatrix}_k \begin{Bmatrix} \boldsymbol{u}_f \\ \boldsymbol{u}_c \\ \Delta\boldsymbol{\lambda} \end{Bmatrix}_k = -\begin{Bmatrix} \boldsymbol{\mathcal{R}}_f \\ \boldsymbol{\mathcal{R}}_c + \left(\frac{d\boldsymbol{f}_{mpc}(\boldsymbol{\mathcal{X}}_c)}{d\boldsymbol{\mathcal{X}}_c}\right)^T \boldsymbol{\lambda} \\ \boldsymbol{f}_{mpc}(\boldsymbol{\mathcal{X}}) \end{Bmatrix}_k \tag{82}$$

where $\boldsymbol{\mathcal{X}}_c$ are the DOFs associated with the MPC constraints. For each iteration, the consistent tangent and force functions need to be computed followed by the solution to $\{\boldsymbol{u}_f \quad \boldsymbol{u}_c \quad \Delta\boldsymbol{\lambda}\}$. The system in (82) is solved repeatedly until no improvement of the nodal coordinates $\boldsymbol{\mathcal{X}}$ (to within some tolerance) can be made. This is done in the same manner discussed in §3.3 (71) and (72).

Since the truss system is self-contained no external forces exist, $\boldsymbol{\mathcal{F}}^{ext} = 0$, therefore:

$$\boldsymbol{\mathcal{R}}_f = \boldsymbol{\mathcal{F}}_f^{int} = \boldsymbol{F}_f^{int} \tag{83}$$

$$\boldsymbol{\mathcal{R}}_c = \boldsymbol{\mathcal{F}}_c^{int} = \boldsymbol{F}_c^{int} \tag{84}$$

Or, in the simplified form where $\boldsymbol{\mathcal{X}}' = \{\boldsymbol{\mathcal{X}}_f \quad \boldsymbol{\mathcal{X}}_{\mathrm{d}\Omega} \quad \boldsymbol{\lambda}\}^T$ and $\boldsymbol{u}' = \{\boldsymbol{u}_f, \boldsymbol{u}_c, \Delta\boldsymbol{\lambda}\}$:

$$\frac{d^2\mathcal{L}(\boldsymbol{\mathcal{X}}')}{d\boldsymbol{\mathcal{X}}'^2}u' = -\frac{d\mathcal{L}(\boldsymbol{\mathcal{X}}')}{d\boldsymbol{\mathcal{X}}'} \tag{85}$$

This system is solved iteratively until no meaningful change in any of $\boldsymbol{\mathcal{X}}_f$, $\boldsymbol{\mathcal{X}}_c$ and $\boldsymbol{\lambda}$ is obtained. In this derivation the second-order contribution of the MPCs, $\frac{d^2\boldsymbol{\mathcal{R}}_c^{mpc}}{d\boldsymbol{\mathcal{X}}_c{}^2}\boldsymbol{\lambda}$, is introduced into the system. This is required to obtain the exact consistent tangent which is required for quadratic convergence using the Newton method. The resulting consistent tangent matrix is symmetric and is made indefinite due to the zero values present on the diagonal.

3.6.2. Implementation of the MSEM MPC method

The MSEM is implemented using the approach discussed in §2.6.2.1 following that of Kok *et al.* [45]. The constraint DOFs $\boldsymbol{\mathcal{X}}_c$ are now partitioned into the master $\boldsymbol{\mathcal{X}}_m$ and slave $\boldsymbol{\mathcal{X}}_s$ DOFs. The set of residual equations in (34) - (37) can be solved using Newton's Method as follows:

$$\begin{bmatrix} \frac{d\boldsymbol{F}_f}{d\boldsymbol{\mathcal{X}}_f} & \frac{d\boldsymbol{F}_f}{d\boldsymbol{\mathcal{X}}_m} & \frac{d\boldsymbol{F}_f}{d\boldsymbol{\mathcal{X}}_s} \\ \widetilde{\boldsymbol{K}}_{mf} & \widetilde{\boldsymbol{K}}_{mm} & \widetilde{\boldsymbol{K}}_{ms} \end{bmatrix} \begin{Bmatrix} \boldsymbol{u}_f \\ \boldsymbol{u}_m \\ \boldsymbol{P}^T\boldsymbol{u}_m \end{Bmatrix} = -\begin{Bmatrix} \boldsymbol{\mathcal{R}}_f \\ \boldsymbol{\mathcal{R}}_{ms} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{\mathcal{F}}_m^{int} - \boldsymbol{\mathcal{F}}_m^{ext} \\ \boldsymbol{\mathcal{F}}_m^{int} - \boldsymbol{\mathcal{F}}_m^{ext} + \boldsymbol{P}^T\left(\boldsymbol{\mathcal{F}}_s^{int} - \boldsymbol{\mathcal{F}}_s^{ext}\right) \end{Bmatrix} \tag{86}$$

where:

$$\widetilde{\boldsymbol{K}}_{mf} = \frac{d\boldsymbol{F}_m}{d\boldsymbol{\mathcal{X}}_f} + \boldsymbol{P}^T\frac{d\boldsymbol{F}_s}{d\boldsymbol{\mathcal{X}}_f} \tag{87}$$

$$\widetilde{\boldsymbol{K}}_{mm} = \frac{d\boldsymbol{F}_m}{d\boldsymbol{\mathcal{X}}_m} + \frac{d(\boldsymbol{P}^T\boldsymbol{F}_s)}{d\boldsymbol{\mathcal{X}}_m} \tag{88}$$

$$\widetilde{\boldsymbol{K}}_{ms} = \frac{d\boldsymbol{F}_m}{d\boldsymbol{\mathcal{X}}_s} + \boldsymbol{P}^T\frac{d\boldsymbol{F}_s}{d\boldsymbol{\mathcal{X}}_s} \tag{89}$$

All parts of the system in (86) are computed and the system is solved for each Newton step as in §3.3 (71) and (72). Since the system is self-contained, no external forces exist, $\boldsymbol{\mathcal{F}}^{ext} = \boldsymbol{0}$, therefore:

$$\boldsymbol{\mathcal{R}}_f = \boldsymbol{\mathcal{F}}_f^{int} - \boldsymbol{\mathcal{F}}_f^{ext} = \boldsymbol{F}_f^{int} \tag{90}$$

$$\boldsymbol{\mathcal{R}}_{ms} = \boldsymbol{\mathcal{F}}_m^{int} - \boldsymbol{\mathcal{F}}_m^{ext} + \boldsymbol{P}^T\left(\boldsymbol{\mathcal{F}}_s^{int} - \boldsymbol{\mathcal{F}}_s^{ext}\right) = \boldsymbol{F}_m^{int} + \boldsymbol{P}^T\boldsymbol{F}_s^{int} \tag{91}$$

The term $\frac{d(\boldsymbol{P}^T\boldsymbol{F}_s)}{d\boldsymbol{\mathcal{X}}_m}$ is a second-order derivative since $\boldsymbol{P} = \boldsymbol{P}(\Delta\boldsymbol{\mathcal{X}}_m) = \frac{d\Delta\boldsymbol{\mathcal{X}}_s}{d\Delta\boldsymbol{\mathcal{X}}_m} = \boldsymbol{f}'_{ms}(\Delta\boldsymbol{\mathcal{X}}_m)$ of the MSEM MPC equation in (25). This is expanded as $\frac{d(\boldsymbol{P}^T\boldsymbol{F}_s)}{d\boldsymbol{\mathcal{X}}_m} = \frac{d\boldsymbol{P}^T}{d\boldsymbol{\mathcal{X}}_m}\boldsymbol{F}_s + \boldsymbol{P}^T\frac{d\boldsymbol{F}_s}{d\boldsymbol{\mathcal{X}}_m}$.

Grouping the terms then gives the system:

$$\begin{bmatrix} \boldsymbol{K}_{ff} & \boldsymbol{K}_{fm} \\ \boldsymbol{K}_{mf} & \boldsymbol{K}_{mm} \end{bmatrix} \begin{Bmatrix} \boldsymbol{u}_f \\ \boldsymbol{u}_m \end{Bmatrix} = -\begin{Bmatrix} \boldsymbol{F}_f \\ \boldsymbol{F}'_{ms} \end{Bmatrix} = -\begin{Bmatrix} \boldsymbol{F}_f \\ \boldsymbol{F}_m^{int} + \boldsymbol{P}^T\boldsymbol{F}_s^{int} \end{Bmatrix} \tag{92}$$

where:

$$\begin{aligned} \boldsymbol{K}_{ff} &= \frac{d\boldsymbol{F}_f}{d\boldsymbol{\mathcal{X}}_f} \\ \boldsymbol{K}_{fm} &= \frac{d\boldsymbol{F}_f}{d\boldsymbol{\mathcal{X}}_{\boldsymbol{m}}} + \frac{d\boldsymbol{F}_f}{d\boldsymbol{\mathcal{X}}_{\boldsymbol{s}}}\boldsymbol{P} \\ \boldsymbol{K}_{mf} &= \boldsymbol{K}_{fm}^T = \frac{d\boldsymbol{F}_{\boldsymbol{m}}}{d\boldsymbol{\mathcal{X}}_f} + \boldsymbol{P}^T\frac{d\boldsymbol{F}_{\boldsymbol{s}}}{d\boldsymbol{\mathcal{X}}_f} \\ \boldsymbol{K}_{mm} &= \frac{d\boldsymbol{F}_{\boldsymbol{m}}}{d\boldsymbol{\mathcal{X}}_m} + \frac{d\boldsymbol{P}^T}{d\boldsymbol{\mathcal{X}}_m}\boldsymbol{F}_s + \boldsymbol{P}^T\frac{d\boldsymbol{F}_s}{d\boldsymbol{\mathcal{X}}_m} + \frac{d\boldsymbol{F}_m}{d\boldsymbol{\mathcal{X}}_{\boldsymbol{s}}}\boldsymbol{P} + \boldsymbol{P}^T\frac{d\boldsymbol{F}_{\boldsymbol{s}}}{d\boldsymbol{\mathcal{X}}_s}\boldsymbol{P} \end{aligned} \tag{93}$$

The slave deformations are then recovered from the MPC equations:

$$\boldsymbol{u}_s = \boldsymbol{f}_{ms}(\boldsymbol{u}_m) \tag{94}$$

This derivation of the MSEM method includes the second-order consistent tangent contribution $\frac{d\boldsymbol{P}^T}{d\boldsymbol{\mathcal{X}}_m}\boldsymbol{F}_s$ in the term $\boldsymbol{K}_{mm}$required to make the consistent tangent exact. As discussed previously, the exact consistent tangent is required for quadratic convergence of the system under conditions involving non-linear MPC constraints. The consistent tangent is symmetric and has the potential to be positive definite. It will be shown in §4.3.4 that the inclusion of the second order term $\frac{d\boldsymbol{P}^T}{d\boldsymbol{\mathcal{X}}_m}\boldsymbol{F}_s$ can render the linear system indefinite due to certain master and slave selections.

3.7. Selection of linear system solution methods

Solver selection is dependent on the problem being solved. In the context of PDE problems, the essential rule is that a sparse direct method can be a good solver for a problem on a two-dimensional spatial domain, but is generally computationally infeasible for a three-dimensional problem [48]. The bandwidth (the largest distance of a non-zero value from the matrix diagonal) of the linear system is generally considerably narrower for 2D FEM problems when compared to 3D FEM. The bandwidth affects the “fill-in” associated with system decomposition methods [47]. Reordering schemes such as Minimum Degree [49] and Cuthill-Mckee [50] reduce the bandwidth of the linear system thereby reducing the fill-in of the decomposed system. This significantly improves the computational efficiency of sparse direct solvers [47]. Reordering schemes perform well on 2D FEM linear systems, resulting in a narrow bandwidth thereby making sparse direct solvers feasible for 2D FEM problems of very large sizes. 3D FEM linear systems cannot be reduced to as narrow-a-bandwidth, and therefore suffer from significant fill-in during decomposition of the system. Cuthill-McKee reordering was used to reorder the linear systems before solution using the sparse direct solvers.

The Lagrangian MPC method creates a symmetric indefinite system of linear equations [45]. For direct solvers of an indefinite matrix, the $\mathbf{LDL}^T$ and modified Cholesky solvers are good choices [46, 47, 51].

Literature indicates that Krylov subspace methods, specifically MINRES, are the preferred choice for large sparse symmetric indefinite saddle point systems [48]. The specific nature of the Lagrangian MPC system is that it is a saddle point system of the form [48, 52]:

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{B}^T \\ \boldsymbol{B} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{v} \\ \boldsymbol{w} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{s} \\ \boldsymbol{t} \end{Bmatrix} \tag{95}$$

For solving these types of problems, Benzi *et al.* [52] gave extensive coverage of various methods that improve solution times. One of the methods discussed is an augmentation method of the form:

$$\begin{bmatrix} \boldsymbol{A} + \boldsymbol{B}^T\boldsymbol{W}\boldsymbol{B} & \boldsymbol{B}^T \\ \boldsymbol{B} & \boldsymbol{0} \end{bmatrix} \begin{Bmatrix} \boldsymbol{v} \\ \boldsymbol{w} \end{Bmatrix} = \begin{Bmatrix} \boldsymbol{s} + \boldsymbol{B}^T\boldsymbol{W}\boldsymbol{t} \\ \boldsymbol{t} \end{Bmatrix} \tag{96}$$

where $\boldsymbol{W} = \gamma \boldsymbol{I}$ and the choice of $\gamma = \frac{||\boldsymbol{A}||_2^2}{||\boldsymbol{B}||_2^2}$ has been shown to generally perform well in practice [52].

For the symmetric positive definite matrices expected from the MSEM, Cholesky $\mathbf{L}\mathbf{L}^T$ factorisation method is recommended for direct solvers [46, 47]. For iterative solvers, the preconditioned conjugate gradient is identified as the preferred method [48]. As will be discussed in Chapter 4, the system is not always positive definite. To this end, the GMRES solver with a sparse incomplete $\tilde{\mathbf{L}}\tilde{\mathbf{U}}$ factorisation preconditioner will be used for the iterative solver and an $\mathbf{L}\mathbf{D}\mathbf{L}^T$ decomposition for the direct solver [48].

The direct solvers that decompose the system, allow for efficient solving of multiple right-hand-sides (RHS) [7]. This makes them an excellent choice for obtaining the mesh sensitivities $\frac{d\boldsymbol{x}}{d\mathbf{x}}$, since each parameter $\mathbf{x}$ requires a solution to its associated RHS. The iterative solvers can also be utilised in this manner in so-called block Krylov space methods [53]. There are, however, inherent difficulties that render the implementation of block Krylov space methods and their application challenging: these include the possible linear dependence of the various residuals; the much larger memory requirements (versus iterative solvers for single RHSs); and the drastic increase of the size of certain small ("scalar") auxiliary problems that must be solved in each step [53].

The systems were solved using the $\mathbf{L}\mathbf{D}\mathbf{L}^T$ direct solver for both Lagrangian and MSEM MPC implementations. For the iterative solver, GMRES with incomplete $\mathbf{L}\mathbf{U}$ preconditioner was used to solve the resulting MSEM linear system, and MINRES with an augmented system was used to solve the resulting Lagrangian system. The selection of these solvers was discussed in §3.7.

The solvers were implemented using built-in MATLAB functions. For the direct solvers the Cuthill-McKee reordering was implemented using `symrcm`, and the $\mathbf{L}\mathbf{D}\mathbf{L}^T$ decomposition using the `ldl` function followed by the use of `mldivide` to solve the back substitutions. For the iterative solvers, the functions `minres` and `gmres` built into MATLAB were used. The student license of MATLAB R2018b Update 2 was used [54].

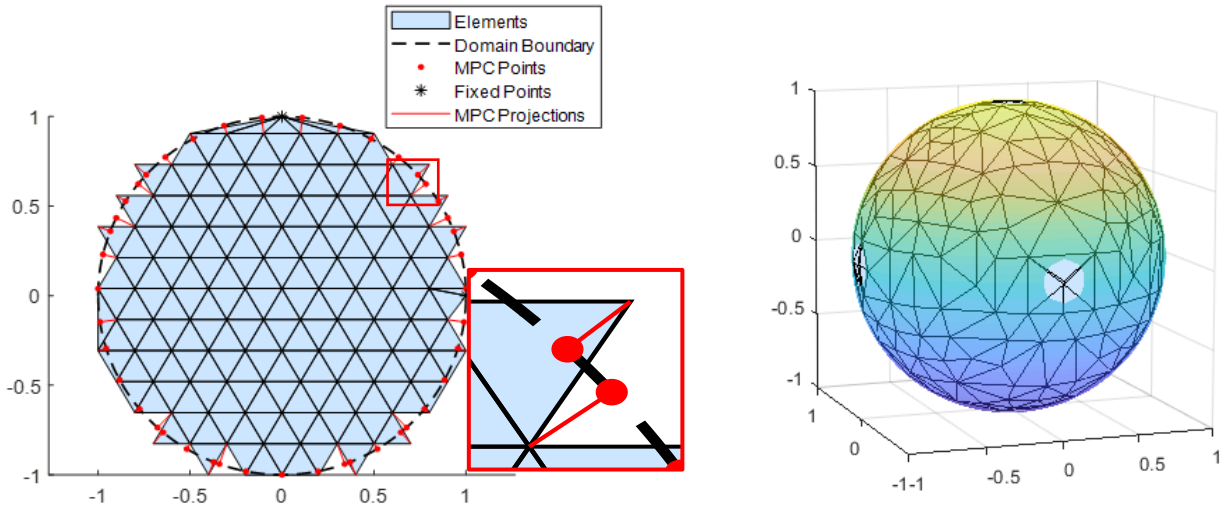
3.8. Verifying quadratic convergence of MPCs with second-order terms

3.8.1. Experimental setup

The MPC methods can be considered to be correctly implemented if quadratic convergence is obtained during the solution of the systems. To test this a circle and sphere with unit radius are used for 2D and 3D respectively. This is done since the first and second-order derivatives for the boundary are simple to obtain. This ensures the function derivatives were correctly implemented.

An initial mesh is created using the mapping discussed as part of the initial mesh setup in §3.4, with the exception that no nodes were seeded along the boundary. Instead, the external nodes on the initial mesh were selected as MPC nodes and displaced to the boundary along the shortest distance. Figure 14 (a) shows the projection lines for the MPC points. This is seen clearly in the magnified section. This "boundary initialisation" step is essential to improve the stability of the MSEM. Without this, the MSEM is not robust. This is discussed later in §4.3. The initial mesh for the 3D system inside of its boundary sphere is shown in (b).

Since the MPC boundary condition permits free movement of the nodes along the boundary, rigid body motion of the truss structure can occur and needs to be eliminated. Whilst the minimum norm solution can be obtained for the linear system, this can influence the convergence rate of the Newton's method implementation. For this study, two nodes were fixed on the boundary for 2D, and three for 3D, to prevent rigid body motion given that the MPC boundary nodes were not constrained along the surface.



(a) (b)

Figure 14 Initial mesh for verification of quadratic convergence for MPC implementation
(a) 2D circle radius 1 (b) 3D sphere radius 1

Nodes coming too close to one another were allowed to merge if the edge length L_c between those nodes is shorter than 2/3 of the desired edge length. The intent of this is to improve system stability

and avoid the negative consequences associated with excessive truss compression (see APPENDIX C and APPENDIX F). The systems were converged to a tolerance of $\|\boldsymbol{\mathcal{R}}\|_2 < 10^{-6}$.

3.8.2. Verification of MPC implementations

In Table 2, Table 3, and Figure 15 the convergence results for the implementation of the two MPC methods are shown. Table 2 shows the quadratic convergence for the 2D implementations. This is demonstrated by subsequent iterations of the norm of the update squaring approximately as the sequence converges $\frac{\|\boldsymbol{u_f}\|_k}{\|\boldsymbol{u_f}\|_{k-1}^2}$. A value between 0 and 1 indicates that quadratic convergence was achieved on a given step. The results for the 3D Lagrangian and MSEM implementations are shown in Table 3. This shows that the Newton method can obtain quadratic convergence for both 2D and 3D, indicating that the MPC methods have been correctly implemented with their second order contributions.

Table 2 Residual norm and ratio results for MPC implementations on the 2D unit circle

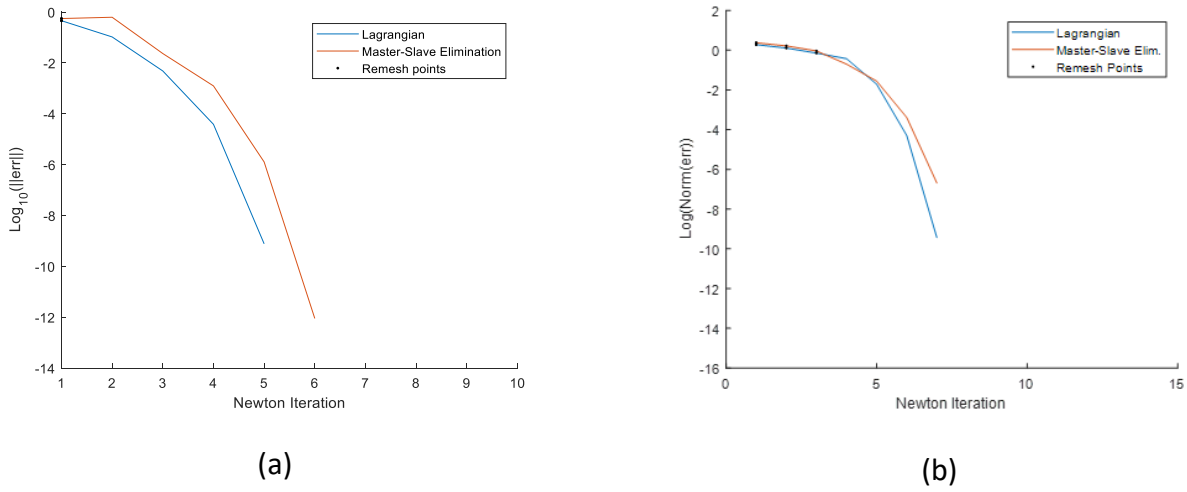
MPC Method Iteration (k)	Lagrangian (2D) $\|\boldsymbol{u_f}\|$	Lagrangian (2D) $\|\boldsymbol{u_f}\|_k/\|\boldsymbol{u_f}\|_{k-1}^2$	MSEM (2D) $\|\boldsymbol{u_f}\|$	MSEM (2D) $\|\boldsymbol{u_f}\|_k/\|\boldsymbol{u_f}\|_{k-1}^2$
1	4.59×10^{-1}	-	5.59×10^{-1}	-
2	1.06×10^{-1}	0.50	6.28×10^{-1}	2.01
3	4.85×10^{-3}	0.43	2.35×10^{-2}	0.06
4	3.87×10^{-5}	1.65	1.25×10^{-3}	2.26
5	7.74×10^{-10}	0.52	1.28×10^{-6}	0.82
6	-	-	8.98×10^{-13}	0.55

Table 3 Residual norm and ratio results for MPC implementations on the 3D unit sphere

MPC Method Iteration (k)	Lagrangian (3D) $\|\boldsymbol{u_f}\|$	Lagrangian (3D) $\|\boldsymbol{u_f}\|_k/\|\boldsymbol{u_f}\|_{k-1}^2$	MSEM (3D) $\|\boldsymbol{u_f}\|$	MSEM (3D) $\|\boldsymbol{u_f}\|_k/\|\boldsymbol{u_f}\|_{k-1}^2$
1	1.18	-	1.34	-
2	5.21×10^{-1}	0.37	5.51×10^{-1}	0.31
3	1.71×10^{-1}	0.63	2.58×10^{-1}	0.85
4	3.33×10^{-3}	0.11	5.86×10^{-2}	0.88
5	2.76×10^{-6}	0.25	2.28×10^{-3}	0.66
6	1.37×10^{-11}	1.80	6.48×10^{-6}	1.24
7	-	-	7.35×10^{-11}	1.75

The quadratic shape of the curve in Figure 15 provides a clear visual indication of quadratic convergence. The Lagrangian implementation is seen in (a) to require one less iteration than the MSEM for the 2D implementation, while both methods are seen to be equivalent in (b) for the 3D implementation. The black dots along the residual norm plots show the iterations where the system

was remeshed. Since initial surface uniformity was not good for the 3D implementation, several remeshes were required to obtain a truss system that met the criteria.



(a) (b)

Figure 15 Norm updates for MPC method implementations
(a) Convergence for 2D implementation on the unit circle (b) Convergence for 3D implementation on the unit sphere

3.9. Non-uniform mesh update

During the shape optimisation process, analyses are run repeatedly to evaluate the updated shape. If large errors in analysis results are anticipated, part of the domain may benefit from mesh refinement. Mesh refinement without the need for a complete remesh of the domain is achieved by either insertion of nodes, which is discrete, or mesh adaption, which maintains the smooth behaviour of the objective function. This was discussed in Chapter 1.

To meet the new refinement requirements of the optimisation process, the truss mesh can be updated between optimisation iterations without remeshing the entire domain. Figure 16 (a) shows the uniform mesh used for the k^{th} optimisation step and (b) shows the same mesh updated with adaptive edge sizing for use in the $(k+1)^{th}$ optimisation step. It is important to note that allowing the boundary nodes to move along the boundary enables large movements in the mesh, whilst maintaining overall mesh quality.

Tracer lines are given on the k^{th} mesh in (a) and are shown mapped to the $(k+1)^{th}$ mesh in (b). The undeformed tracer lines are shown as dotted lines in (b) to allow for direct comparison and show how the mesh has moved to accommodate the mesher's refinement requirement.

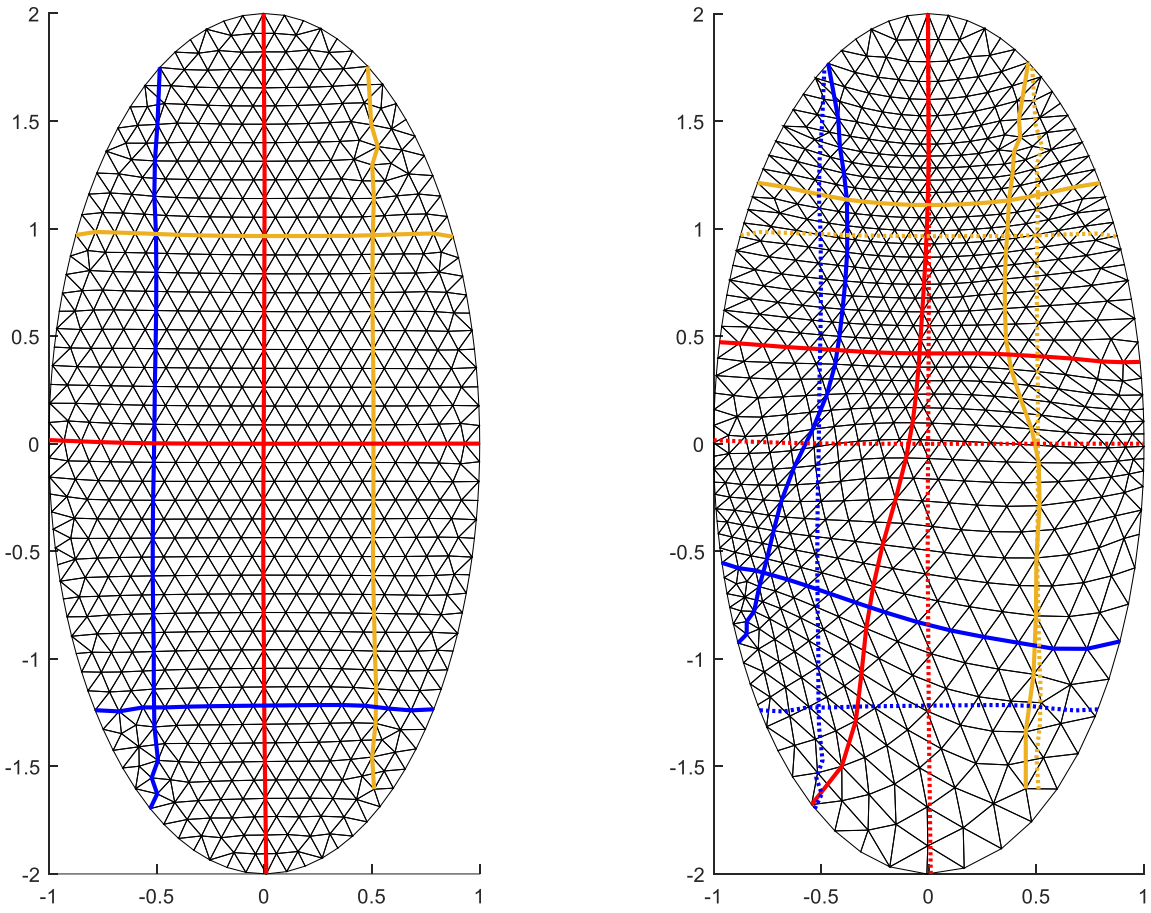


Figure 16 Mesh updated from a uniform to non-uniform sizing
(a) Uniform mesh at k^{th} optimisation step (b) Updated non-uniform mesh at $(k+1)^{th}$ optimisation step

3.10. Summary

Several truss stiffness functions were discussed in §3.2, with $F_{truss} = \frac{L_{des}-L_c}{L_{des}}$ being selected as the preferred implementation. This is discussed in detail in APPENDIX C.

In §3.3 the Newton implementation for the truss system was derived with the Lagrangian and MSEM implementations following in §3.6 and §3.6.2 respectively. The implementation of Newton's method for the trusses were shown to be correct in §3.4.3, and for the Lagrangian and MSEM implementations in §3.8. Both of the MPC methods obtained the expected quadratic convergence associated with Newton's method, hence verifying that the sensitivities are correct. Verification of the correct sensitivities was important since they feature in the chain of sensitivities required for shape optimisation as discussed in §1.3.2. It is noted briefly that the MSEM required one iteration more for both the 2D and 3D implementations on systems designed for easy convergence.

During development, high compression systems were seen to exhibit instability. Such systems tended node snap-through, resulting in inverted elements. In extreme cases, the entire mesh could be distorted beyond recovery. The effect of nodes snapping through was discussed in §3.5 with details of the investigation provided in APPENDIX F. Whilst the method employed by Persson and Strang showed the greatest insensitivity to system compression, it negated the quadratic convergence behaviour of the Newton solver. Consequently, the Newton update step size limit approach was selected for this mesher since it maintains quadratic convergence.

For each iteration of the Newton method, the consistent tangent and residual need to be calculated and the resulting linear system solved. In §3.7 linear system solvers were discussed. $\mathbf{LDL}^T$ was shown to be a good selection for the Lagrangian MPC system. For the MSEM $\mathbf{LDL}^T$ was selected over $\mathbf{LL}^T$ due to the possibility of the system becoming indefinite due to high curvature (see §4.3.4). For the iterative solvers, an augmented MINRES method was selected for the Lagrangian implementation and a GMRES with sparse incomplete $\tilde{\mathbf{L}}\tilde{\mathbf{U}}$ factorisation preconditioning is selected for the MSEM.

CHAPTER 4
MSEM AND LAGRANGIAN COMPARISON

"Only two things are infinite, the universe and human stupidity, and I'm not sure about the former." — Albert Einstein

4.1. Introduction

In this chapter comparison between the Lagrangian and MSEM approaches will be discussed. In §4.2 the challenges specific to the MSEM are demonstrated and discussed. This will discuss the occurrence of complex roots and the sensitivity to high boundary curvature.

In §4.3 several methods used to improve the performance of the MSEM are investigated and shown. These methods were designed to combat the issues discussed in §4.2.

In §4.4 the characteristics of the two MPC methods associated with scaling the problem in terms of the number of nodes is discussed. This is done in the context of this implementation for both the percentage nodes MPC present and the number of nodes in the system.

4.2. Discussion of challenges associated with the MSEM

Early investigations into the MSEM revealed that obtaining convergence would be difficult under certain circumstances because the constraint is required to be satisfied exactly at every Newton iteration. The challenges involved in obtaining convergence were investigated using an ellipsoid, thereby taking advantage of the simplified mathematics.

The Lagrange multiplier method was seen to be robust with the only challenges limited to high compressions and tension values, discussed in §3.5. This section, therefore, focuses exclusively on the basic challenges associated with the MSEM.

The ellipse and ellipsoid boundary definitions were chosen since they offer the simplest domain boundaries to resolve. They have smooth and continuous surfaces for nodes to move along and are easily manipulated to create high curvature regions. Furthermore, they force an interdependence between dimensions such that a single DOF does not have a solution in the real domain for all selections of the remaining DOFs. The nature of the convex shape is such that it removes complexities associated with concave domains for comparison between the methods.

4.2.1. Departure from the boundary and the creation of complex roots

The choice of an ellipse or ellipsoidal domain boundary meant that certain circumstances led to the calculation of an update of the master DOFs that prevented direct recovery of the associated slave DOF.

The presence of a complex component in the computed slave DOF implies that no solution exists that satisfies the master-slave relationship for the master updates. This is illustrated as follows.

Consider a given boundary node $X = \{1.9199\ ,\ \ 0.0560\}$ on an ellipse of the form:

$$\left(\frac{x_1}{2}\right)^2 + \left(\frac{x_2}{0.2}\right)^2 - 1 = 0 \tag{97}$$

For this investigation, the x_1 component was designated as the master. The update to the master DOF was given as $\Delta u_1 = 0.1$. It follows that the slave DOF update would then be given by:

$$\Delta u_2 = 0.2\sqrt{1 - \left(\frac{X_1 + \Delta u_1}{2}\right)^2} - X_2 \tag{98}$$

When the value of Δu_1 is too large, it makes the term $\left(\frac{X_1 + \Delta u_1}{2}\right)^2 > 1$ resulting in the value under the root sign becoming negative. This introduces the complex component to the slave update Δu_2 due to the node leaving the constraint surface. When solved, the update to the slave DOF can be seen to contain an imaginary component:

$$\Delta u_2 = -0.0560\ - (0.0283)i \tag{99}$$

The presence of the complex component, therefore, indicates that the master update step size Δu_1 is too large and some control must be implemented. It is evident that only the real component of the solution may be used in the update since there is no complex dimension present. The consequence of this is shown in Figure 17 where the master and slave update components are shown together with the total update step size.

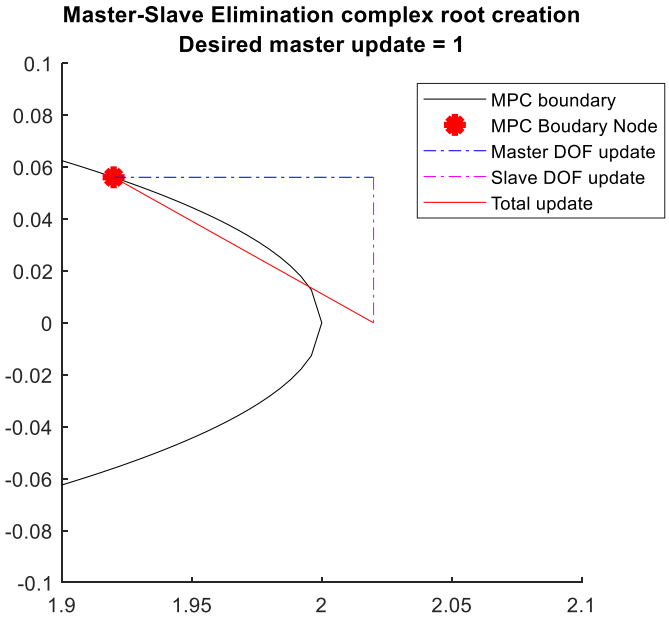


Figure 17 MSEM update resulting in complex roots at high curvature boundary in 2D

Such a scenario was found to be manageable and preferable to a solver exception error which prevented any further solution. The program was therefore modified to ignore the imaginary components of any update and is further modified to cut back the step size, see §4.3.3.

4.2.2. Large MSEM updates resulting in points leaving the domain

Only interior elements were kept where their centroids p_{cent} were inside of 10% of the local desired length from the boundary, $p_{cent} < -0.1 \times h(p_{cent})$. Any resulting nodes that were no longer part of the mesh were removed from the solution. The MSEM was seen to be prone to this "loss of nodes" due to updates that ignored imaginary components as described in §4.2.1. The presence of the complex update component indicates the nodes have left the constraint surface. This was common for regions of high curvature with accompanying increased truss energy which resulted in large master DOF updates followed by a complex root solution to the Slave DOF. The resulting node location was seen to be far outside of the domain as illustrated by Figure 18(b) and the resulting mesh in (d). The effect of the first update on the system employing Lagrangian MPCs (Figure 18 (a) and (c)) is also shown to highlight the extent of the challenge faced in using the MSEM compared to the Lagrangian implementation which allows for updates to be made that deviate from the constraint surface.

This is illustrated for the first update in a 3D system in Figure 18 for an ellipsoid of dimensions $X \times Y \times Z = 0.5 \times 2 \times 2$. The initial mesh condition for both solutions was identical.

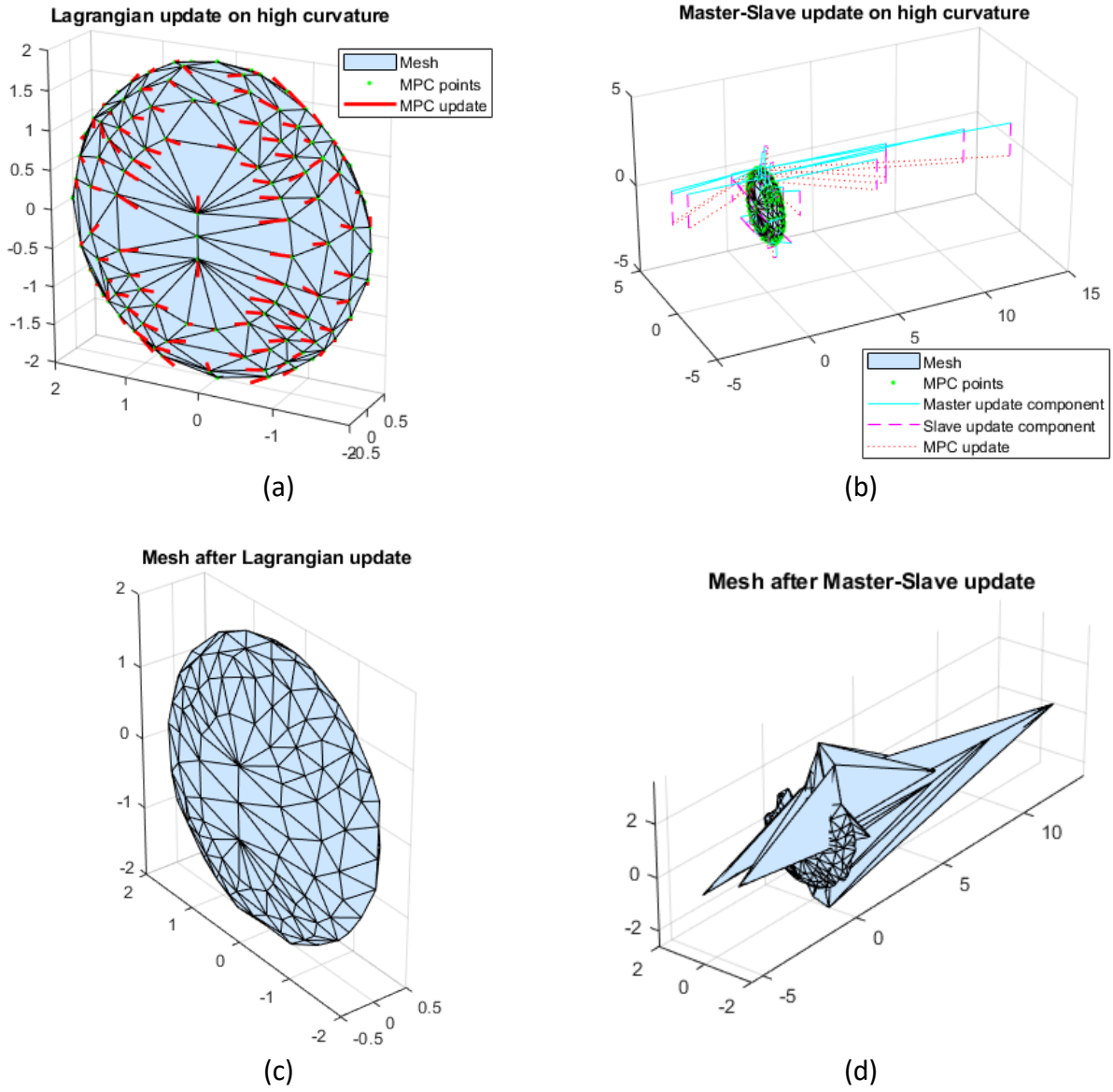


Figure 18 3D MPC updates for Lagrangian and MSEM on high curvature
(a) Update vectors for Lagrangian method on high curvature (b) Update vectors for MSEM on high curvature (c) mesh after update Lagrangian (d) Mesh after update MSEM

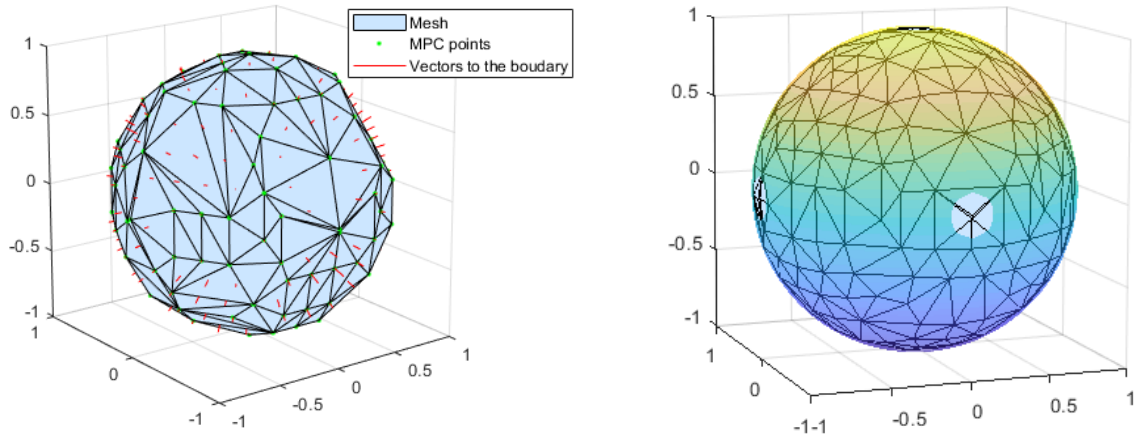
4.3. Enhancements required for MSEM stability

Two methods were considered to help stabilise the performance of the MSEM. The first was to snap the MPC nodes to the boundary and the second was to reduce the systems step size based on the largest master update size relative to the desired local edge size. This was done as detailed in APPENDIX F §F.1.1.

4.3.1. Snapping points to the boundary using the shortest distance

The first step taken to improve the solution of the MSEM involved snapping the points identified as being on the exterior of the initialised mesh, or near the constraint surface (within half of the local

desired length), to the constraint surface. This was done along the shortest distance to the boundary for each node. Figure 19 (a) shows the shortest distances to the boundary for the initial mesh and Figure 19 (b) shows the new nodal positions after the remesh. It is evident that this simple step significantly improves the surface mesh quality and initial domain representation:

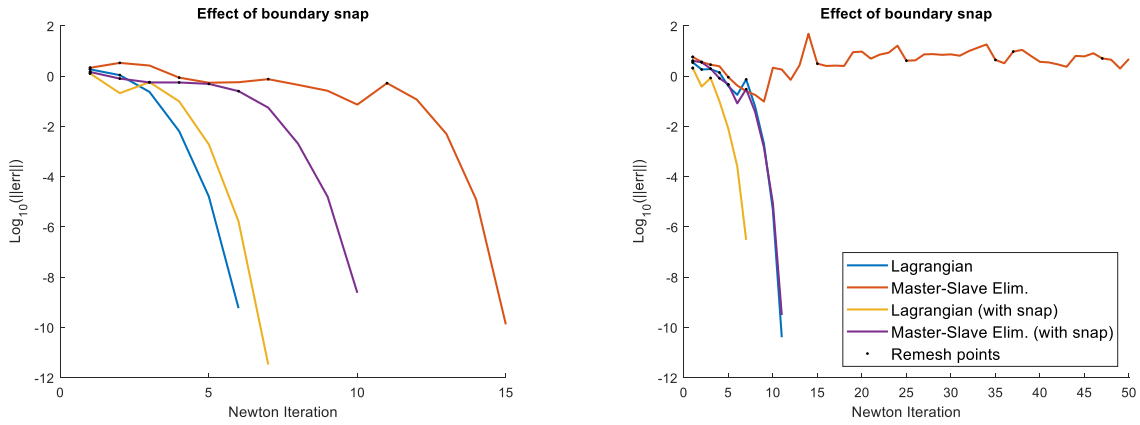


(a) (b)

Figure 19 Boundary snapping initial mesh on the sphere
(a) Showing shortest distance vectors to boundary (b) Mesh after with "snapped" boundary node locations

4.3.2. Results for improvement of MSEM using boundary snap update

The effectiveness of the boundary snap was demonstrated through its implementation in the mesh generator keeping all other conditions the same as detailed in §3.8.1. Figure 20 shows the effect of snapping points to the boundary. For performance reasons, the snapping was done on mesh initialising for both methods and after each remesh step for the MSEM. Figure 20 shows the effect of boundary snapping on (a) a unit sphere and (b) a $0.5 \times 2 \times 2$ ellipsoid. The residuals were converged to 1^{-6}.



(a) (b)

Figure 20 Norm updates for 3D MPC methods including boundary snap
(a) Unit sphere (b) $0.5 \times 2 \times 2$ ellipsoid

The cost of the iteration in which the surface nodes are snapped to the boundary is $O(n_{mpc})$ (assuming it is directly computable) compared to the cost for a Newton iteration which is $O(n^2)$. The MSEM benefited significantly from the boundary snap step. For a unit sphere, the MSEM was able to converge in 10 iterations with the boundary snapping compared to 15 without.

For the $0.5 \times 2 \times 2$ ellipsoid, the MSEM initially did not converge, however, the inclusion of boundary snapping allowed the MSEM to obtain convergence in the same number of iterations as the Lagrangian method implemented without the boundary snap.

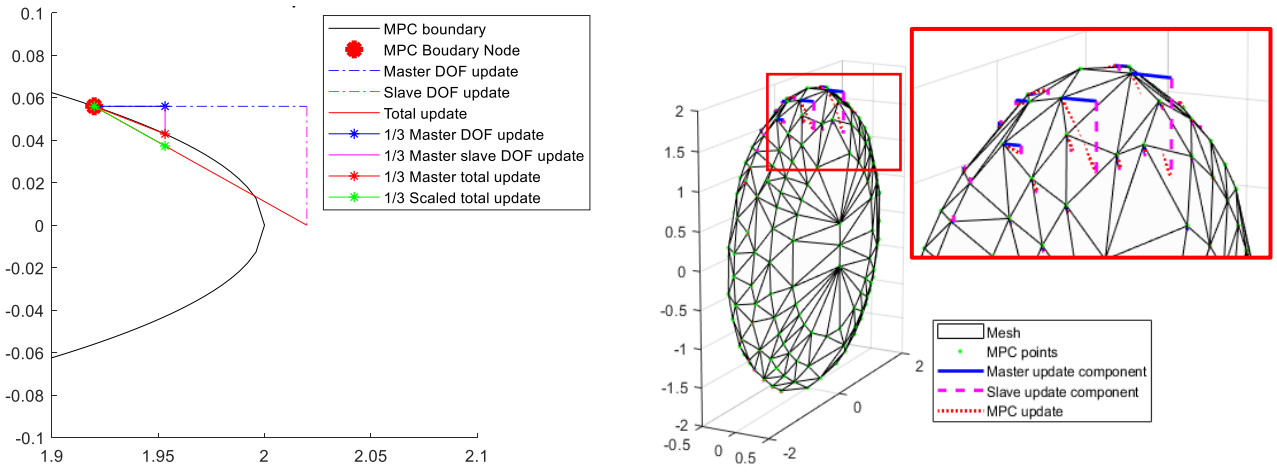
On the sphere that has constant curvature, the Lagrangian method took an additional step due to the introduction of boundary snapping. On the $0.5 \times 2 \times 2$ ellipsoid, significant improvement was demonstrated by reducing iterations from 11 down to 7. This equates to a 36% time saving for the system. It is evident that due to its simplicity and computational benefits, boundary snapping is included in the meshing program irrespective of the approach.

4.3.3. Update step size limit

Further stabilisation of the 3D implementation of the MSEM was accomplished by limiting the largest update step size for the master DOFs to the value α_{change}. This follows the same principle as discussed in detail in APPENDIX F §F.1.1 for the global update. All master DOF update values are evaluated against α_{change}. If $\max(\boldsymbol{u}_m) > \alpha_{change}$ then all DOF updates $\boldsymbol{u}$ are scaled down until $\max(\boldsymbol{u}_m) = \alpha_{change}$. In the event $\max(\boldsymbol{u}_m) < \alpha_{change}$, the full Newton step update $\boldsymbol{u}$ will be used.

In the initial consideration of the update step size limit, all DOFs were considered. For the implementation of the step size limit with respect to the MSEM, the updates on only the free and master DOFs are tested against the limit. The Slave DOFs are then calculated from the master positions. This ensures the points remain on the boundary.

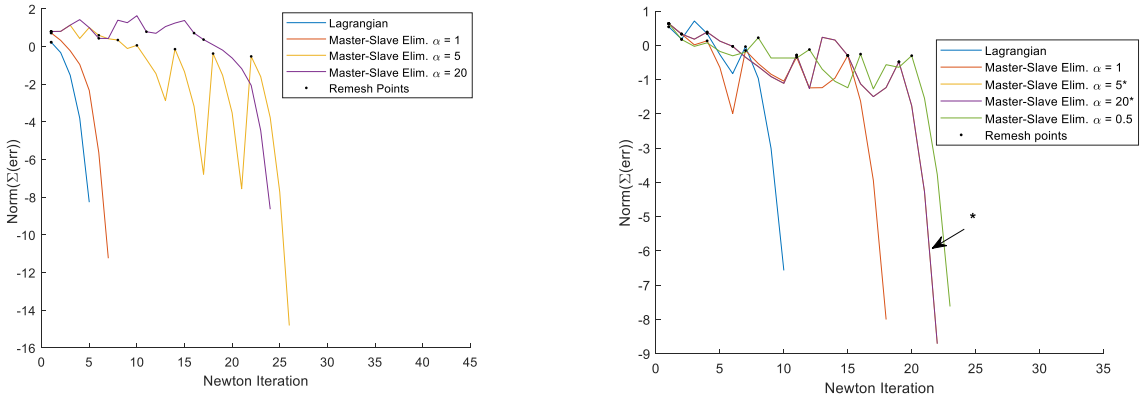
From Figure 21 (a) it is evident that the MPC update step size limit is easily implemented and is very effective in controlling the MSEM updates. The limiter value α_{change} should be related to the curvature, however, the simplified approach was able to achieve the desired result. The additional complexities associated with using the curvature to determine local α_{change} values were therefore avoided. In Figure 21 (b), an update limit, α_{change}, equal to half of the local element edge size can be seen to limit the largest master updates, thereby scaling the entire system update. The initial truss system used was the same as that in §4.2.2. Comparing Figure 18 (b) and Figure 21 (b) shows the improvement in implementation using this method.



(a) (b)

Figure 21 Update step size limit at high curvature boundary for MSEM
(a) 2D ellipse (b) 3D ellipsoid

For this study, no update limiter α_{change} was used with the Lagrangian method, since no effect was demonstrable for these problems. The effect of the step size limit on the stability of the 3D system employing the MSEM was investigated using three-step size limiting factors $\alpha_{change} = 1$, 5 and 20. The snapping of surface nodes to the boundary was only used for the initial mesh. In Figure 22 (a) for a $1 \times 2 \times 2$ ellipsoid, it is clear a significant convergence improvement was shown in using the update step size limit. It must, however, be noted that the effect in improving convergence, was not universally effective and higher curvature problems had varying results. The update limiter value α_{change} should be related to the local curvature for each constrained node, however, given the general effectiveness of the fixed size α_{change}, the curvature was not used. Figure 22 (b) shows the results of a $0.5 \times 2 \times 2$ ellipsoid. It is clear here that the step size limit had limited effect.



(a) (b)

Figure 22 Convergence for 3D MPC MSEM update control for various α_{change}
(a) $1 \times 2 \times 2$ ellipsoid (b) $0.5 \times 2 \times 2$ ellipsoid

The MSEM derived significant benefit from the stability offered by the update step size limiter α_{change} and it was therefore included in the implementation of the MSEM with a value of $\alpha_{change} = 1$. This was in addition to the use of boundary snapping after remeshing. Since this problem did not exist for the Lagrangian method, the limiter was implemented with a value of $\alpha = 10$ to stop severe distortions of the mesh due to high compression in the trusses (see §3.5).

4.3.4. Second-order contribution difficulties

Theoretically, the MSEM retains positive definiteness of the system. It was found however, that slave selection could impact the results associated with the curvature. The contributions to the matrix diagonal for the first order and second-order components are shown in Figure 23 (a). It is clear that the first-order components of the diagonal are all positive, whilst the second-order oscillates around zero. This was based on the selection of the slave DOF minimising the first-order components of the system. As seen in Figure 23 (b), the system is largely unaffected by the inclusion of the second-order term. There are however a few notable spikes below zero indicating negative diagonal terms. Negative diagonal terms mean that the matrix is no longer positive definite which necessitates an $\mathbf{LDL}^{\mathrm{T}}$ decomposition as opposed to a Cholesky $\mathbf{LL}^{\mathrm{T}}$ decomposition for the direct solver.

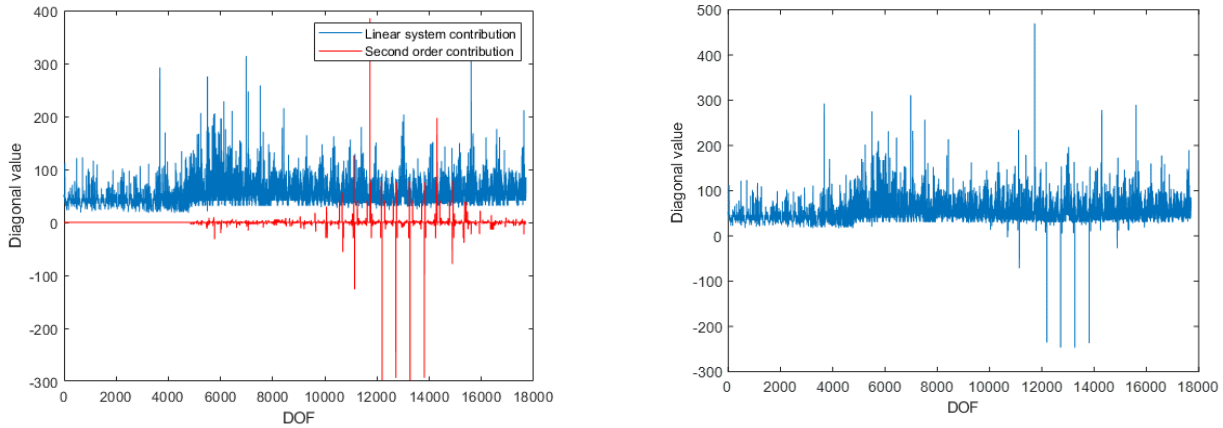


Figure 23 System diagonal contribution of MSEM
(a) First and second-order contributions (b) Complete system

To demonstrate the effect of the negative values on the system diagonal, the 2 x 0.5 x 0.5 ellipsoid problem is used. Following the approach outlined in [43], the slave DOF is selected to minimise the magnitude of the first order contributions to the matrix $\boldsymbol{P}$. The impact of slave DOF selection is demonstrated by using node number 3 444 with associated global DOFs 10 330, 10 331 and 10 332; and corresponding local DOFs 1, 2, and 3 (corresponding to the x, y and z displacements of node 3444). This node is selected since it presented a large negative second-order diagonal element for the consistent tangent as shown in Figure 23(a) and (b). This consistent tangent is selected as it caused the failure of the $\mathbf{LL}^{T}$ solver during investigation into the performance of the MSEM on the $2 \times 0.5 \times 0.5$ ellipsoid.

In Table 4 it is clear that choosing the slave local DOF as 3 yields the lowest average magnitude for the first order contributions $\frac{du_s}{du_{m_1}}$ and $\frac{du_s}{du_{m_2}}$ to the matrix $\boldsymbol{P}$. However, looking at the associated diagonal components $\frac{d^2u_s}{du_{m_1}du_{m_1}}$ and $\frac{d^2u_s}{du_{m_2}du_{m_2}}$, it is seen they are both large negative values at this node. Choosing the slave local DOF as either 1 or 2 would have produced positive diagonal elements. It is noted that selecting the local DOF 1 as the slave would have resulted in the lowest magnitude contributions of $\boldsymbol{P}$ while maintaining positive diagonal elements. It is therefore extremely important to select slave DOFs based on the curvature component for each node. This, however, requires the computation of all first and second-order derivatives for all DOFs selected as slaves, increasing the time costs for the implementation. Alternatively, having computed the first-order derivatives the diagonal second-order contributions $\frac{d^2u_s}{du_{m_1}du_{m_1}}$ and $\frac{d^2u_s}{du_{m_2}du_{m_2}}$ can be computed for each slave selection. This would be done from the slave selection with the lowest magnitude first-order contribution to the largest.

It is also important to note that this is the result for a single DOF constraint, having two constraints per node would further complicate selection.

Table 4 Example of effect of slave selection on curvature contributions

Slave DOF (local)	1	2	3
Master DOF1 (m_1)	2	1	1
Master DOF2 (m_2)	3	3	2
ds/dm_1	-1.27E-01	-7.85E+00	9.78E-01
ds/dm_2	1.02E+00	8.03E+00	1.24E-01
$d^2u_s/du_{m_1}du_{m_1}$ (Diagonal 1)	9.90E+02	4.80E+05	-9.35E+02
$d^2u_s/du_{m_1}du_{m_2}$	-1.30E+00	-4.96E+03	-1.19E+00
$d^2u_s/du_{m_2}du_{m_1}$	-1.30E+00	-4.96E+03	-1.19E+00
$d^2u_s/du_{m_2}du_{m_2}$ (Diagonal 2)	1.00E+03	4.80E+05	-9.26E+02

4.4. Comparison of Lagrange vs MSEM for scaled problems

As discussed in §3.7 the solution of the linear systems time contributions resulting from 2D FEM problems is somewhat inconsequential [48]. The scaling studies discussed here will be limited to 3D problems which are the primary focus of this study.

For this study, three systems of varying sizes are compared. These are designated small medium and large. The large system of 70848 DOFs was limited by the memory requirements associated with this implementation. This was to ensure that no inefficient memory caching on the disk occurred. The systems without the MPC contributions had the baseline properties as shown in Table 5.

For only the consistent tangent of a non-sparse matrix and 70848 DOFs results in 70 848×70 848×8bytes=40.1GB memory requirements. This is the worst-case scenario since there are many zeros values present. Since this was a sparse implementation the size is reduced to the number of non-zeros present in the decomposed matrices $\mathbf{L}$. This, however, varies based on system bandwidth, reordering scheme, and solving method. The study is conducted using an Intel Core i5 6600K @4.1GHz machine with 16GB DDR4-2400MHz RAM, running under Windows 10™ operating system. It must be noted that there was a 7.2GB memory overhead for the operating system, MATLAB, and ancillary programs. The memory requirements from this implementation peaked at 6.5GB in total.

In Table 5 the small, medium, and large system properties are shown. This table shows the values without the inclusion of any MPC contributions since the MPCs modify the consistent tangent. The number of non-zeros shown in Table 5 are for the consistent tangent only.

Table 5 Consistent tangent properties for scaling comparison

	No. Nodes	No. DOF	No. truss members	No. Non-zeros (excl. MPC)
Small	4 110	12 330	28 148	389 293
Medium	15 835	47 505	109 691	1 491 226
Large	23 616	70 848	163 876	2 215 552

The systems were then solved with 0, 20, 40, 60, 80, and 100% of nodes designated as MPC nodes. The 100% MPC constraint case would be associated with thin features (one element through the thickness) or surfaces.

In this set-up, the 100% MPC nodes problem was associated with a system that has very high curvature and the maximum curvature of the system was also dependant on the number of nodes in the system. This was due to constraining nodes to nested spheres in order to obtain the desired MPC %.

The same solvers as outlined in §3.7 were used and all times represented are taken as the average of ten runs. It must be noted that the results that follow are representative of this implementation and are used for discussion purposes primarily in the context of this mesher.

4.4.1. Comparison of system DOFs and non-zeros

Figure 24 shows how the number of non-zeros increases as more nodes are constrained. It is clear that the MSEM benefits from the elimination of the slave DOFs. For a 3D system, the number of DOFs requiring solving can be reduced by 33% since each MPC constrained node eliminates one of the three

DOFs associated with it. The number of non-zeros reduces proportionally. For 2D this benefit is increased to 50%. Further irrespective of whether one or two constraint equations are applied to a given node, one DOF will be removed from the problem.

For the Lagrangian approach, each constraint equation contributes an additional DOF requiring solution. This results in an increase of 33% DOFs to the system size for a single constraint applied to 100% of nodes. Until this point only a single constraint equation per node has been considered, i.e. the node has been constrained to a single surface. In the event a node is constrained to two surfaces, two constraint equations may be present. If this is the case for all nodes the number of DOFs increases by 66%. However, these two constraints can potentially be replaced by a single constraint to a line at the intersection of the two surfaces, if the user codes the constraints efficiently. The application of three constraints to a node can more efficiently be handled as a fixed node (in 3D).

Whilst there is the potential for a large increase in the number of DOFs for a given problem, the number of non-zeros in the system only increases by ≈6.5% for 100% of nodes with one constraint and ≈13% for two constraints applied to each node. As mentioned, efficient programming from a user can result in 1 constraint equation per node. This is what will be represented here.

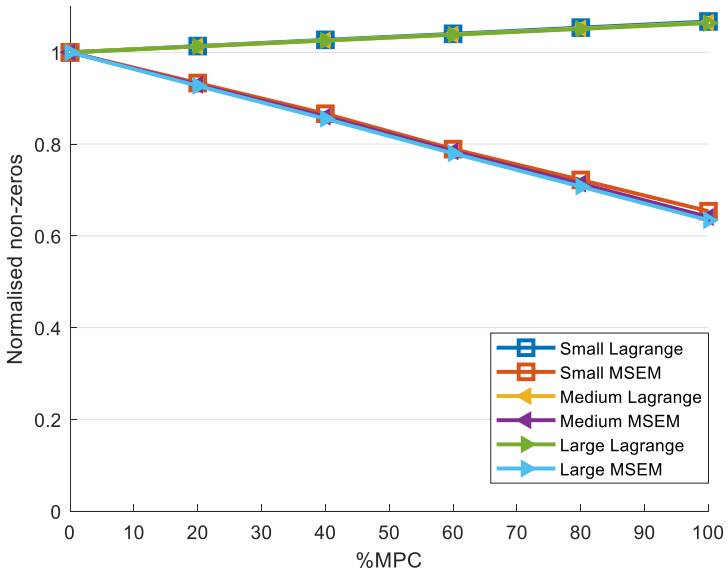


Figure 24 Normalised number of non-zeros comparison

Figure 25 shows the locations of non-zero values within the consistent tangent matrix. This is shown for the medium system and 80% of nodes constrained using MPCs. Figure 25 (a) shows the original truss system without the application of MPC constraints, (b) shows the system with the Lagrangian augmentation, and (c) shows the reduced system with the MSEM reduction. The system without MPCs (a) initially has 47 505 DOFs. The Lagrangian method adds an additional 12 668 DOFs to the system to a total of 60 173 DOFs and the MSEM reduces the system size by 12 668 to 34 837 DOFs.

Whilst the Lagrangian MPC implementation adds off-diagonal terms thereby increasing the bandwidth of the system (b) the MSEM reduces the system, and the bandwidth remains essentially unchanged (c). This is the case in this implementation since a nod's slave DOF is only constrained to the other DOFs of the same node. All of the associated matrix locations are already populated from the truss equations, which means no additional non-zeros are added to the system of equations. In both cases, reordering schemes can be employed to improve bandwidth.

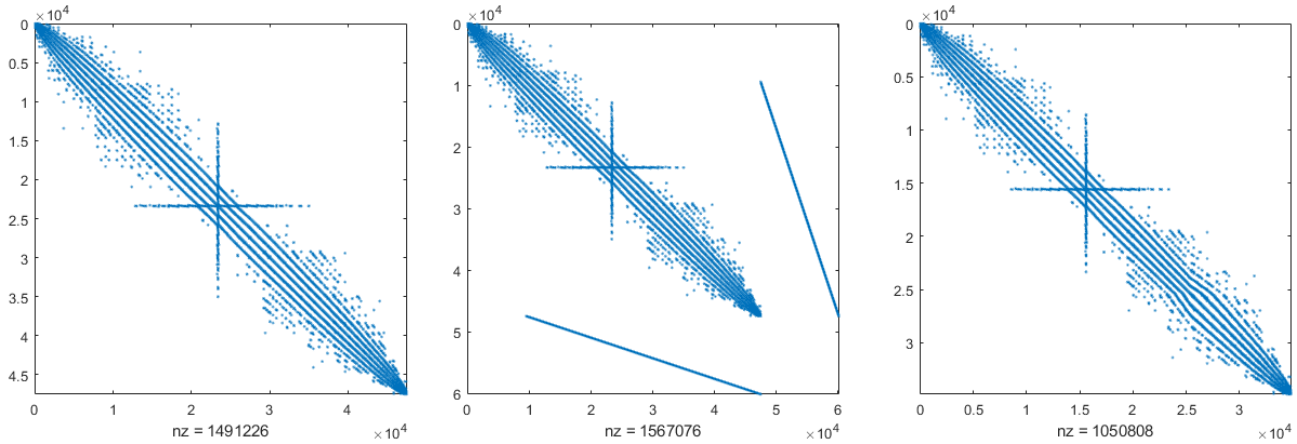


Figure 25 Matrix fill patterns for 80% MPC constraint
(a) Original system (b) Lagrangian MPC (c) MSEM MPC

For this truss system, it is clear that the MSEM MPC method significantly reduces the number of non-zeros and solve DOFs proportionally to the number of constrained nodes. The Lagrangian method increased the number of non-zeros by three times the number of constraint equations and the number of solve DOFs by the number of constraint equations. The MSEM would therefore present an advantage over the Lagrangian method, based on the storage and solution requirement of the linear system.

4.4.2. Time associated with truss assembly and MPC DOFs and non-zeros

In order to assemble the system to be solved, the truss member contributions, as well as the MPC contributions, need to be computed and assembled. For this implementation assembly of the linear system was done using vectors and the vectors assembled into a sparse matrix in a single step. This was the recommended approach for efficient sparse assembly implementations in MATLAB 2018b [54].

Figure 26 shows the time to compute and assemble the consistent tangent for the small medium and large systems for both MPCs and truss components. Notably, the MSEM MPC assembly contributes two to three times the amount of assembly time compared with the Lagrangian method. The comparatively large times associated with the MSEM are due to the more complex derivatives that need to be computed (see APPENDIX B) and the sparse matrix calculations that need to be performed (see §3.6.2 (92) and (93)) to obtain the final system of linear equations. For the 100% constrained scenario, the MSEM requires nearly the same amount of time to calculate and assemble the MPCs to the system as

it did to assemble the entire truss system. This is comparing the 23616 constraint equations assembled to the 163876 trusses assembled.

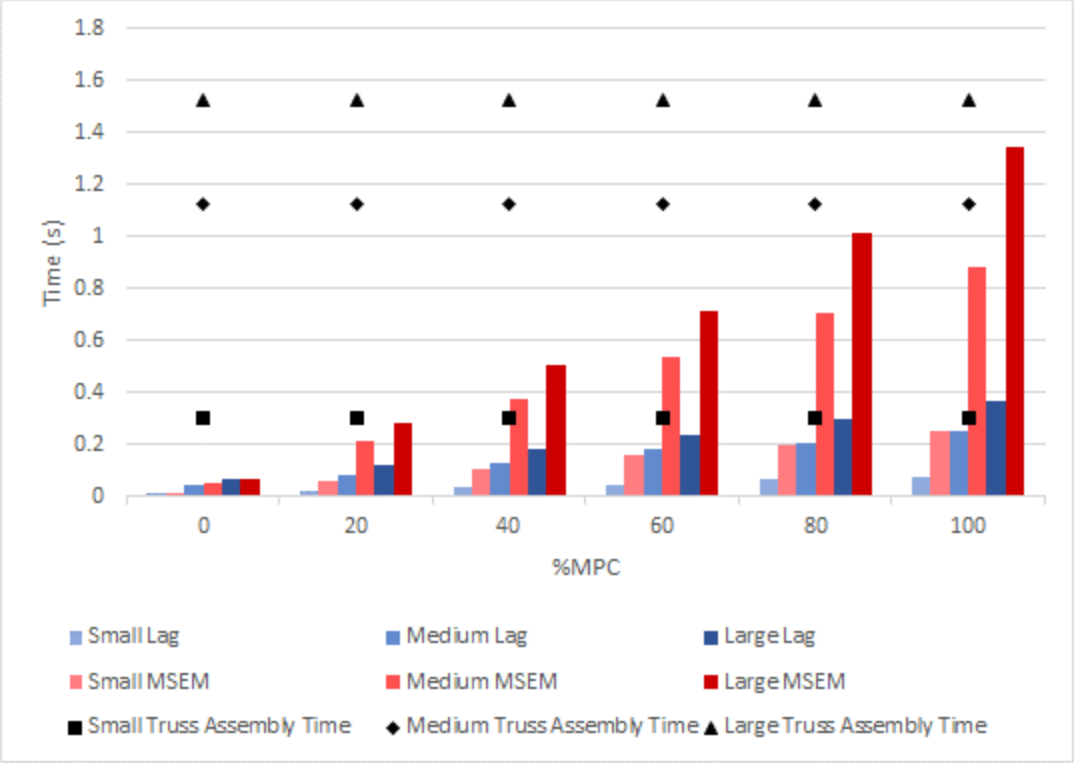


Figure 26 Truss and MPC computation and assembly times for small, medium and large systems

For the 40% MPC nodes case, the Lagrangian method contributes 12% additional time to assemble the matrix, and the MSEM contributes 33% additional time. Therefore, the MSEM requires 2.8 times more time than the Lagrangian MPCs. The total system MSEM assembly takes 18% longer to be calculated than the Lagrangian system. These times are shown in Table 6.

Table 6 Large system matrix assembly times 40% MPC constraint

	Lagrangian	MSEM	Ratio MSEM to Lagrangian
Truss assembly (s)	1.52	1.52	1
MPC contribution (s)	0.22	0.50	2.8
Total time for assembly (s)	1.71	2.03	1.2
MPC times as % of truss assembly times	12%	33%	

For the 100% constrained case, the MSEM time increased to 88% of the truss assembly time compared to 24% for the Lagrangian time. The MSEM assembly time is 3.7 times that of the Lagrangian MPCs assembly time. These specific values are shown in Table 7.

Table 7 Large system matrix assembly times 100% MPC constraint

	Lagrangian	MSEM	Ratio MSEM to Lagrangian
Truss assembly (s)	1.52	1.52	1
MPC contribution (s)	0.44	1.35	3.7
Total time for assembly (s)	1.99	2.99	1.5
MPC times as % of truss assembly times	24%	89%	

4.4.3. Computational cost of iterative vs direct solvers for the linear system solution

The resulting linear system was solved using direct and iterative solvers. This section shows the times associated with the solution to the linear system for a single Newton step. The selection of the direct and iterative solvers is discussed in §3.7.

Figure 27 shows the times associated with the solution of the linear system using the direct solvers. It is clear that the reduced number of DOFs associated with the MSEM has a significant benefit on the solution time for a given Newton step. The Lagrangian method computational cost increased by 7 times (at 100% constrained nodes) compared to the MSEM. This is due to the matrix fill in associated with the off-diagonal terms during the matrix decomposition. A more suitable reordering scheme may be employed to improve this.

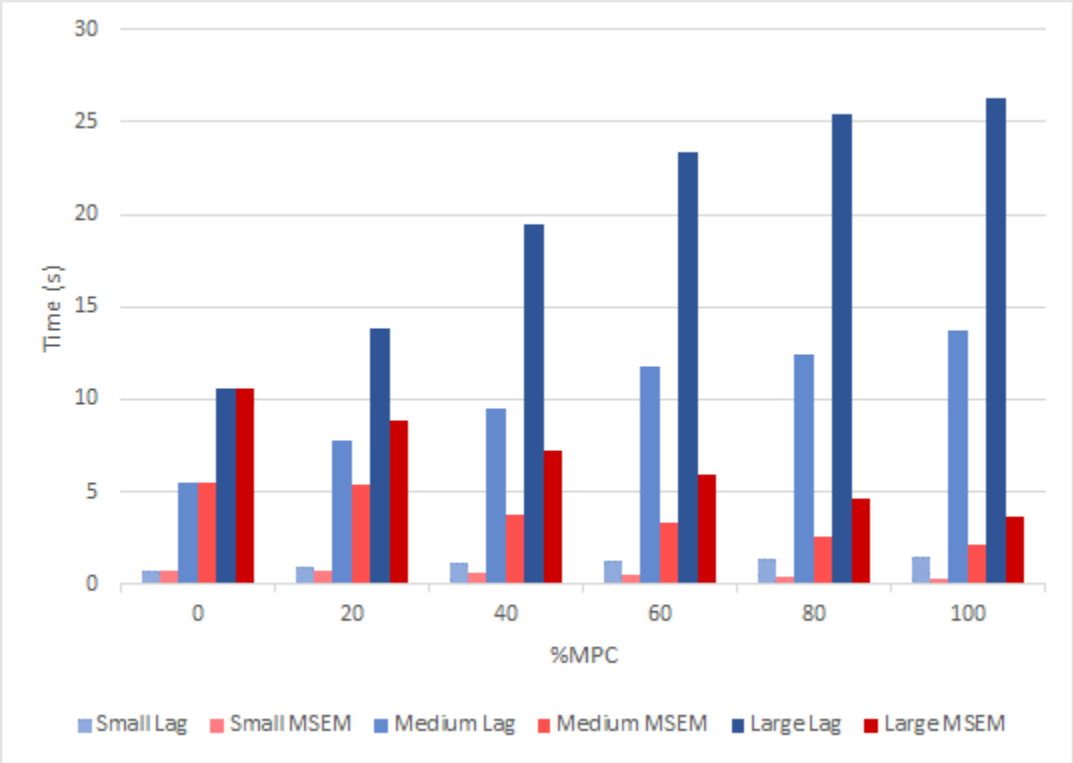


Figure 27 Single Newton step solution times of linear system using direct solvers

Figure 28 shows the times when the system assembly contributions, as discussed in §4.4.2, are included as part of the time for solving the system for a single Newton step. The impact at 100% constrained nodes for the large system, is reduced from a ratio of 7 times (for solution time only in Figure 27) to 4.2 times (assembly plus linear solution) for the Lagrangian vs MSEM times.

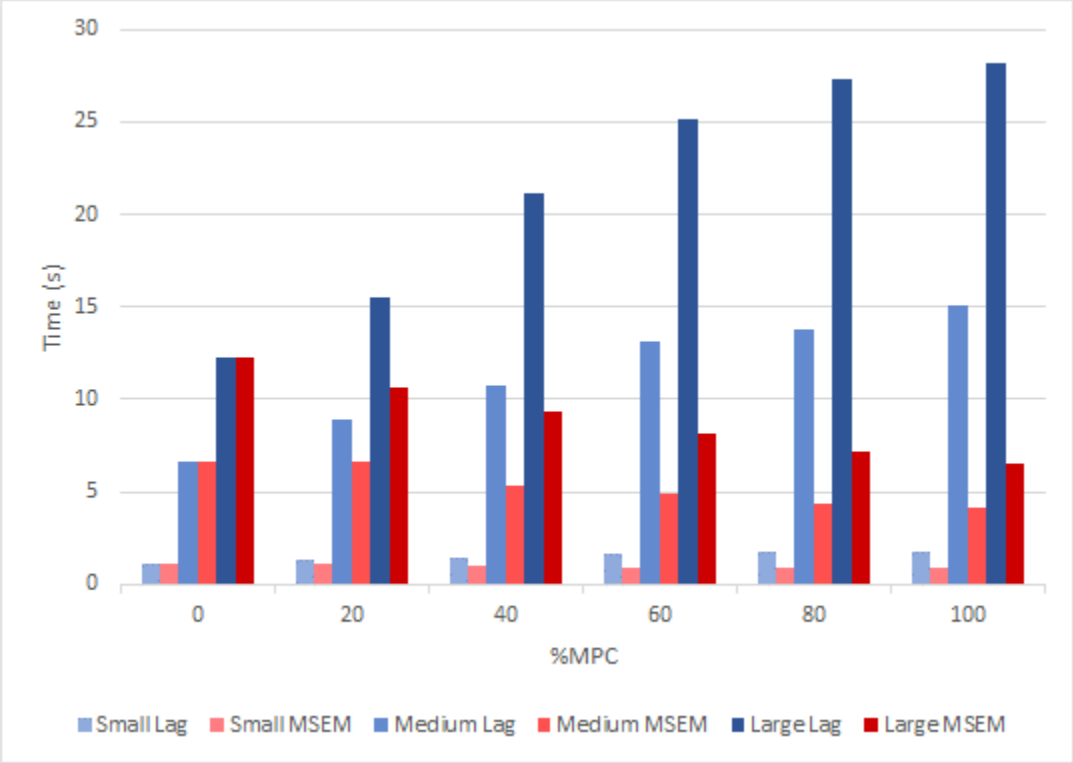


Figure 28 Single Newton step time for linear system assembly and solution with direct solvers

The iterative solver implementation used the completely assembled sparse matrix and was not implemented matrix-free for this study and a convergence tolerance of 10^{-8} was used. The iterative solvers presented significant savings for both MSEM and Lagrangian implementations. The method selected for the solution to the Lagrangian MPC saddle-point system was affected significantly by the introduction of the constraint equations. This was due to the introduction of quite high curvature contributions (compared to the truss contributions) into the problem.

This resulted in a 3.1 times increased cost for the 100% MPC constraint case for the large system for the Lagrangian implementation. The medium system fared better with a 1.9 times impact for the 100% constraint case. The MSEM implementation of GMRES for the linear system solve performed better with less of a time impact associated with the increasing percentage of MPC constrained nodes. This indicates that the solution method for the Lagrangian saddle point system would likely benefit from an improved preconditioning selection. These comparisons are shown in Figure 29. It is interesting to note that the Medium system solved in very similar times for all %MPC nodes.

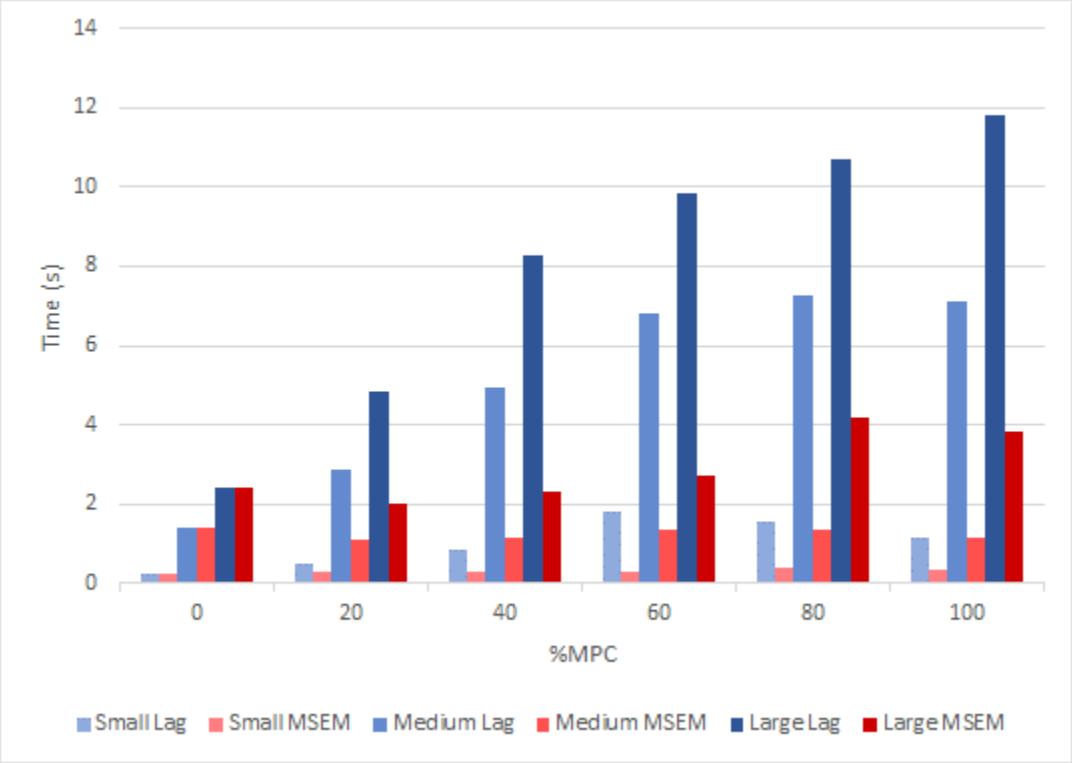


Figure 29 Single Newton step solution times of linear system using iterative solvers

However, incorporating the once-off system assembly time, shown in Figure 26, decreases the total time cost of the Lagrangian MPC implementation to approximately 2 times that of the MSEM for the 100% MPC constraint scenario on the large system. This is demonstrated in Figure 30.

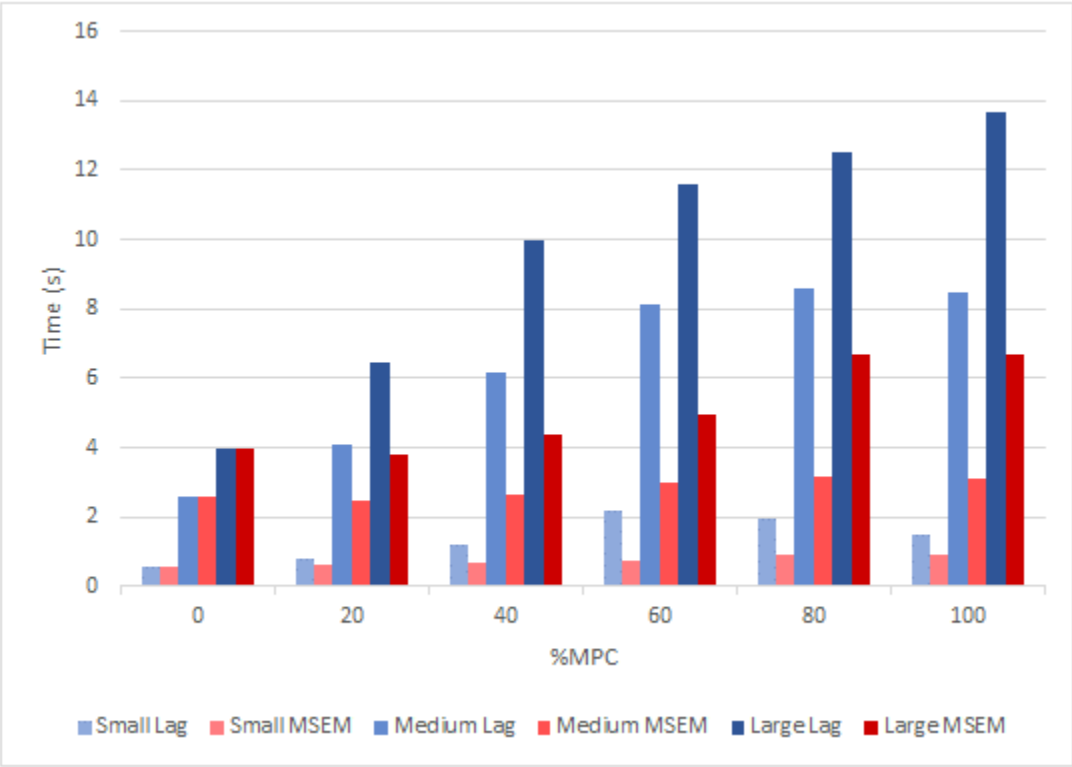


Figure 30 Single Newton step time for linear system assembly and solution with iterative solvers

In Figure 31 the ratios of iterative to direct solver times $\frac{t_{iterative}}{t_{direct}}$ are shown. These ratios are for the times associated with the complete computation and assembly, with solution of the resulting linear system included. From Figure 31 it is clear to see that out of a total number of 36 systems solved, only 5 solved faster using the direct solver. Of these, 4 were associated with the small system of equations, whilst 1 was for the large system of equations employing the MSEM.

On average the iterative solvers exhibited better performance than the direct solvers, with the Lagrangian implementation benefitting the most. For the small systems, the solution times were sufficiently low on the direct solvers that inconsistencies were observed on relative performance.

For the MSEM, as more DOFs were constrained, so the ratios of solution times increased. It was noted that the solution time ratio for the 100% MPC constraint on the large system for the MSEM solved faster with the direct versus the iterative solver. This was investigated since the ratios for the 0% to 60% MPC constraint scenarios for the large system correspond to those of the medium system. This was found to be due to the introduction of several large curvature components into the system resulting in difficulty for the iterative solver to converge to a solution.

The Lagrangian method with iterative solvers showed significant improvements in terms of a 40% time saving for the medium-sized system and approximately 55% time saving for the large system with higher percentages of constrained nodes. For the Lagrangian solutions, the % time saving increased with the system size for MPC constrained scenarios.

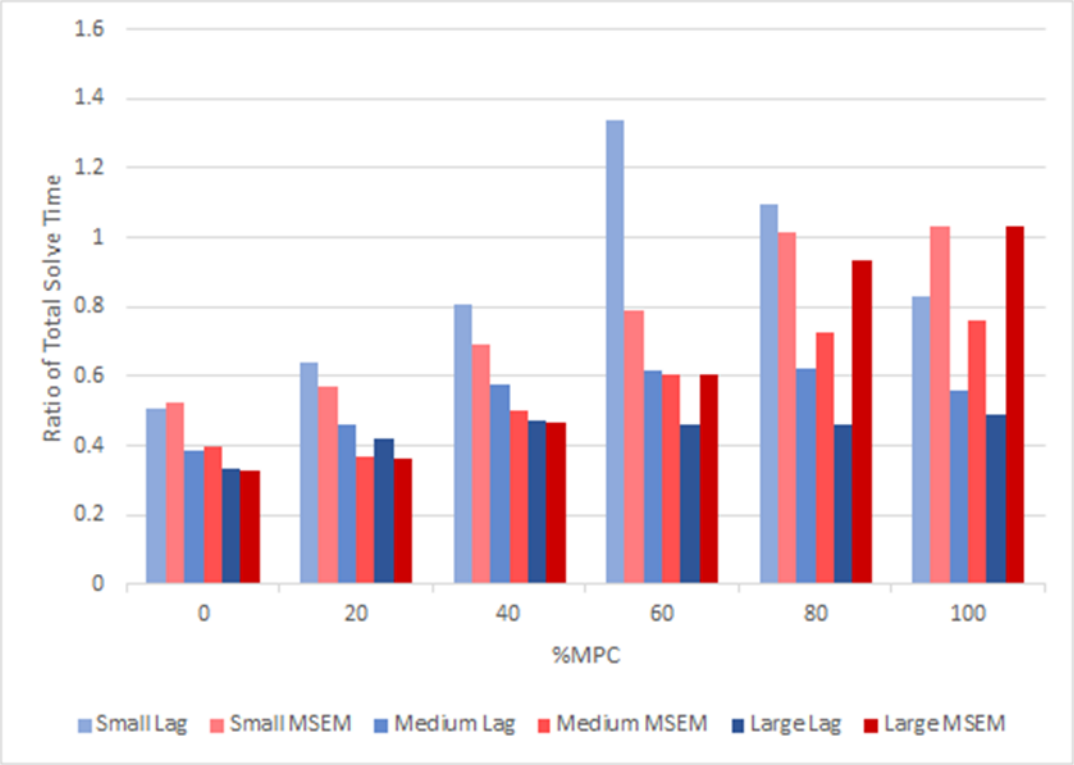


Figure 31 Ratio of times $t_{iterative}/t_{direct}$ for a single Newton step linear system assembly and solution

Given that the iterative solvers demonstrated superior performance, Figure 32 shows the ratios for the total time associated with the Lagrangian iterative implementation solution times versus the corresponding MSEM solution times. For 0% MPCs, the Lagrangian iterative solver performed slightly better than the MSEM implementation. In general, the MSEM benefitted from the reduced system size and a reduced number of non-zeros associated with the system. From the graph in Figure 32, it is easy to see that the general time implication of using the Lagrangian method was between 1.8-2.5 times for the large % MPC nodes.

The 60% constraint system demonstrates the largest time ratio on average. This is likely due to the significant decrease in system size favouring the MSEM and the fact that curvature is not sufficiently high so as to significantly reduce convergence in the MSEM solver.

The average time ratios between the Lagrangian and MSEM were 1.77, 2.18, and 1.87 for the small, medium, and large linear systems respectively. The average for the medium and large systems combined was 2.03. For the higher %MPC problems, 40% through to 100%, the average time ratio was 2.33. In conclusion, the MSEM exhibits superior performance per Newton iteration compared to the Lagrangian method.

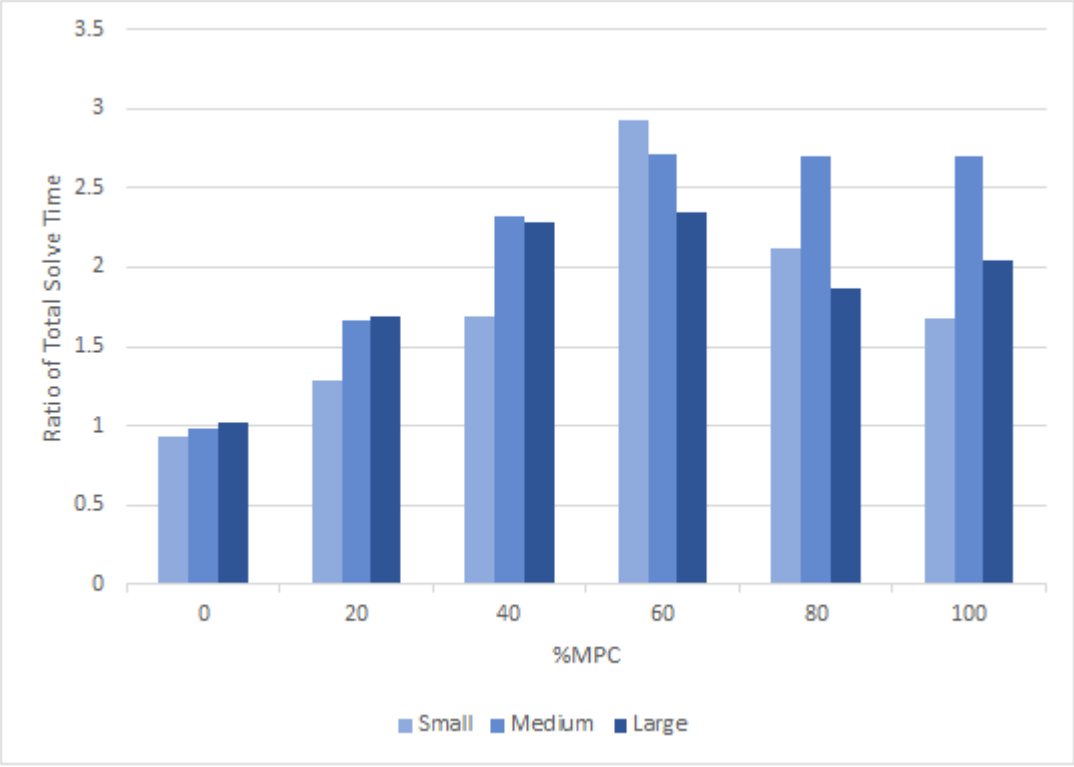


Figure 32 Ratio of times $t_{Langrangian}$ / t_{MSEM} for a single Newton step linear system assembly and solution

4.4.4. Total times for truss equilibrium

It is clear from §4.3.3 that the MSEM presented significant challenges in obtaining truss equilibrium. This resulted in many more required iterations to obtain the equilibrium solution for the truss system. For the two investigations, this ranged from 1 iteration longer to obtain convergence to problems that did not converge.

For the problem discussed in §4.3.3 (see Figure 22) for the moderate curvature $0.5 \times 2 \times 2$ ellipsoid, the best performance achieved for the MSEM was 18 iterations and the worst performance was 24 iterations. This is compared to 10 iterations for the Lagrangian implementation. This indicates a factor of 1.8 to 2.4 iteration impact for solving the MSEM over the Lagrangian. For higher curvature problems the MSEM solution was erratic and was often unable to resolve without significant "tuning" for the specific problem. These results are typical for the performance of the MSEM vs Lagrangian implementations.

From §4.4.3, the average time cost of the Lagrangian vs MSEM iterative solutions was 2.03 for the medium and large systems. The average ratio between solve times was 2.33 for the 40% to 100% MPC scenarios for the medium and large systems. Considering the increased number of iterations required and the ratios for iterative solver times and extending these to obtain a total estimated performance gives the results in Table 8. Table 8 shows the potential time implications for the scenarios described. This was calculated as:

$$R_{tot.\ Iter.time}\left(\frac{Lag}{MSEM}\right) = \frac{t_{Lag.\ Iter.}/t_{MESM\ iter.}}{Iter_{MSEM}/Iter_{Lag.}} = \frac{t_{Lag.} \times Iter_{Lag.}}{t_{MESM} \times Iter_{MSEM}} \qquad (100)$$

It is clear from the table that for this implementation the overall performance of the Lagrangian method compared to the MSEM ranges from 15% less time to 30% more time to obtain the solution. On average 6% additional time would be required to obtain the Lagrangian solution.

Table 8 Comparison of solve times and iterations

Ratio of single Newton iteration solution time $\frac{t_{Lagrangian}}{t_{MESM}}$	Ratio of no. iterations $\frac{Iter_{MSEM}}{Iter_{Lagrangian}}$	Ratio of total time for solution $R_{tot.\ time\ Iter}\left(\frac{Lag}{MSEM}\right)$	Lagrangian additional time compared to MSEM
2.03	1.8	1.13	13%
2.33	1.8	1.29	29%
2.03	2.4	0.85	-15%
2.33	2.4	0.97	-3%
2.18 (avg.)	2.10 (avg.)	1.06 (avg.)	6%

4.4.5. Matrix-free implementations

For a matrix-free implementation, the calculations associated with the total assembly times in §4.4.2 are directly proportional to the time required to produce a single matrix multiplication in the direct process. For the 100% MPC nodes case, it is clear that significant, approximately 52%, additional time will be required for the MSEM for a given iteration of the matrix-free iterative solver. It therefore follows that the Lagrangian MPC method presents a potentially faster matrix-free implementation. Furthermore, the complexities associated with the removal of DOFs in the MSEM (see §3.6.2 (92) and (93)), introduce associated difficulty in terms of DOF management and calculation during the matrix-free calculation.

Assuming that the two iterative solvers for the MSEM and Lagrangian implementations are able to resolve a linear system in the same number of iterations, the percentage time difference between the two methods can be calculated as follows. Using the average 1.06 ratio $R_{tot.\ Iterative}$ obtained in §4.4.4 for the Lagrangian vs MSEM for iterative solver implementation and considering the relative time cost of assembly $\frac{t_{assem\ MSEM}}{t_{assem\ Lag.}}$ from §4.4.2 the expected difference for a matrix-free implementation is given as:

$$R_{Mat.\ Free}\left(\frac{Lag}{MSEM}\right) = \frac{t_{matrix\ free\ Lag.}}{t_{matrix\ free\ MSEM}} = \frac{R_{tot.\ Iterative}\left(\frac{Lag}{MSEM}\right)}{t_{assem\ MSEM}/t_{assem\ Lag.}} \tag{101}$$

This suggests that the Lagrangian implementation of a matrix-free solver may be as much as 30% more efficient than the MSEM matrix-free solver when 100% MPC constraints are present. This implies that the Lagrangian method is a better selection for matrix-free implementations.

Table 9 Comparison of solve times and iterations

Ratio of single Newton iteration solution time $\frac{t_{Lagrangian}}{t_{MESM}}$	Ratio of total time for solution $R_{tot.\ Iterative}\left(\frac{Lag}{MSEM}\right)$	Ratio of total time for linear system assembly $t_{assem\ MSEM}/t_{assem\ Lag.}$	Ratio of total time for linear system assembly $R_{Mat.\ Free}\left(\frac{Lag}{MSEM}\right)$
20	1.06	1.10	0.97
40	1.06	1.19	0.89
60	1.06	1.27	0.84
80	1.06	1.39	0.76
100	1.06	1.52	0.70

4.5. Early termination for mesh generation based on mesh criterion

An advantage of the MSEM having the boundary criterion exactly satisfied at each iteration is that it invites the use of a convergence criterion based on mesh statistics rather than a system equilibrium tolerance. A statistics-based criterion has the potential to achieve convergence at fewer Newton steps due to the option of solution termination at the end of any iteration since the constraint equations f_{mpc} are satisfied at the end of every step. This also means that no unnecessary boundary element distortion would result if the process is terminated early.

In the Lagrangian approach, the boundary is only satisfied upon solution of mesh equilibrium. This means that at early termination nodes may not be on the constraint surface and will not necessarily return until the solution has converged. This implies that early termination would result in a poorly approximated boundary for the system.

The MSEM advantage of termination at any iteration could be achieved with the Lagrangian method by including a boundary snap at the end of each iteration and evaluating the resulting system accordingly. This, however, would result in distorted boundary elements and would also introduce a discrete step into the Lagrangian solution process that will affect the convergence rate since the sensitivities are no longer continuous. This problem is easily resolved by determining whether the mesh criterion would be met “if” the nodes were snapped to the boundary and only snapping if this is the case.

For both systems, it is noted that early termination implies that static equilibrium for the truss system has not been obtained. Static equilibrium of the truss system is critical for the acquisition of accurate mesh sensitivities. This is because the mesh sensitivities should be a function of only control parameters and not of residual potential energy in the system.

For the MSEM the truss system sensitivities would still be exact for the mesh state at early termination, but there would still be an error associated with obtaining the mesh sensitivities $\frac{d\boldsymbol{X}}{d\boldsymbol{x}}$. However, for the boundary snapped mesh resulting from the early termination of the Lagrangian method, the sensitivities of the mesh would no longer be valid, and the computation of sensitivities would hence be inaccurate.

4.6. Summary

As discussed in §4.2, several challenges were seen to be unique to the MSEM, particularly when solution of the system resulted in large nodal position updates. In §4.3 two methods for minimising the impact of large updates on the MSEM implementation were investigated and shown to considerably improve the solution stability.

In the first method, nodes identified on the surface of the mesh were snapped to the boundary of the domain along the shortest distance. This reduced the likelihood of nodes leaving the domain in the first update and resulted in a smooth surface mesh with a more accurate representation of the boundary.

Additional implementation of MSEM update control (limiter) was found to permit the solution of systems that had previously failed to achieve convergence due to large updates. This significantly improved the stability and reliability of the method. The relative instability of the MSEM is associated with the fact that all constraints must be exactly satisfied for each iteration. As discussed in §4.1, the updates can, in turn, result in complex components in the slave-update.

The Lagrangian method was found to be the most stable and was largely unaffected by the non-linearities associated with regions of high curvature. This is because the boundary condition is only satisfied once equilibrium of the entire system has been found. This corresponds to the results obtained by Kok *et al*. [45]

In §4.5 the potential for early termination when using the MESM and Lagrangian methods was discussed. The MSEM could be terminated early with the only consequence being the reduced accuracy of mesh sensitivities $\frac{d\boldsymbol{X}}{d\mathbf{x}}$ associated with the truss system no longer being in equilibrium. For the Lagrangian method, the nodes would have to be discretely forced to the boundary thereby creating discontinuous sensitivities that would invalidate the computed mesh sensitivities.

The time comparisons done in §4.4 showed that for large percentages of MPC constrained nodes the MSEM presented a distinct advantage with respect to the linear system solution time associated with a single Newton step. This was especially noticeable for the direct solvers discussed in §4.4.3. where the Lagrangian linear system took 7.2 times longer to solve versus the MSEM system for the large system with 100% nodes constrained. The use of iterative solvers was however shown to reduce this discrepancy to twice the computational time cost for the Lagrangian method versus the MSEM.

For problems with moderate curvature, the MSEM requires significantly more iterations to obtain convergence. This was shown in §4.3.3. The MSEM required significant assistance to reduce the impact of curvature on the solution to the system. For the $0.5 \times 0.5 \times 2$ ellipsoid (moderate curvature) problem in §4.3.3 the update step size limiting methods were shown to reduce the number of Newton iterations to just under twice that of the Lagrangian implementation.

Total truss equilibrium solution time discussed in §4.4.4 for the implementation of the Lagrangian method was shown to require between 0.85 to 1.3 times that of the MSEM implementation. The Lagrangian method is expected to, on average, take 6% longer to obtain truss equilibrium for large percentages of MPCs and large systems of equations.

Matrix free implementations, discussed in §4.4.5, are typically associated with very large linear systems. To implement a matrix-free solver each iteration of the solver, for the linear system, requires recalculation for the assembly of the system. Given that the MSEM was up to 52% computationally more expensive in terms of assembly costs, there is the potential for the Lagrangian implementation to be as much as 30% faster.

Table 10 Comparison of pros and cons associated with MPC implementations

	Lagrangian	MSEM
Ease of implementation	Best	Worst
Robustness in obtaining solution	Best, no Lagrangian specific problems found	Worst Solution not guaranteed Requires significant heuristic control to converge Generally unstable Nodes leaving constraint surface
Newton steps to obtain solution	Best	Worst (1.8 times best-case scenario for moderate curvature system)
No of non-zeros (memory requirement)	Worst (up to 40% more)	Best
System assembly time	Best	Worst (up to 88% longer)
Direct solver solution time (single newton step)	Worst (up to 7 times as long)	Best
Iterative solver time (single newton step)	Worst (up to 2 times as long)	Best
Overall iterative solver times	Similar (0.85 to 1.3 ave. 1.06)	Similar
Matrix free implementation (extrapolated from given results)	Best	Worse (up to 1.3 times longer)

Table 10 summarises all the features of the two methods. Overall, the implementation of the Lagrangian MPC method was considerably easier than the MSEM. The MSEM was however shown to result in faster solve times for the linear system, except for the results extrapolated for the matrix-free implementation where it was predicted to perform worse.

Since the use of a mesher in shape optimisation is not the primary focus of the optimisation process, but rather a tool, it is essential that the mesher be simple and robust. The Lagrangian approach proved easier to implement and did not require sophisticated heuristics to obtain convergence. For the problems studied, the Lagrangian MPC method only experienced a 6% time penalty when using an iterative solver and could potentially be as much as 30% faster for a matrix-free implementation. The Lagrangian MPC method is therefore selected for implementation in the final mesher.

CHAPTER 5
SENSITIVITIES IMPLEMENTATION FOR MPCs

"Engineering problems are under-defined, there are many solutions, good, bad, and indifferent. The art is to arrive at a good solution. This is a creative activity, involving imagination, intuition, and deliberate choice." — Ove Arup

5.1. Introduction

At this point, it is important to distinguish for the reader that there are two FEAs in discussion. The first is the FEA of the truss system that is done to resolve the mesh nodal coordinates $\boldsymbol{\mathcal{X}}$. This FEA was geometrically non-linear and iterative. The second is the FEA used to analyse the structural shape used to compute the performance in the cost/objective and constraint functions of the shape optimisation problem and is given as $\mathbf{Ku} = \mathbf{f}$ for linear elastic analysis. The cost function is given as $\mathbb{C}\left(\boldsymbol{\mathcal{X}}\left(\Omega^*(\mathbf{x})\right)\right)$, where $\mathbf{x}$ is the set of control variables for the parametrised geometric domain and $\Omega^*(\mathbf{x})$ represents the computational domain which is split into the interior Ω and boundary $\partial\Omega$.

As discussed in §1.3.2, gradient-based shape optimisation using linear-static FEA requires stiffness matrix sensitivities $\frac{d\mathbf{K}}{d\mathbf{x}}$. Since the linear static system is partitioned into free f and prescribed p degrees of freedom and selecting $\mathbf{u}_p = \mathbf{0}$, only the $\frac{d\mathbf{K}_{ff}}{d\mathbf{x}}$ component is required. The final form required for the optimisation process is repeated here for easy reference:

$$\mathbf{K}_{ff}\,\frac{d\mathbf{u}_f}{d\mathbf{x}} = -\frac{d\mathbf{K}_{ff}}{d\mathbf{x}}\mathbf{u}_f \tag{102}$$

Additionally, the $\frac{d\mathbf{K}_{ff}}{d\mathbf{x}}$ components are not directly computable since the domain $\Omega^*(\mathbf{x})$ is discretised using a mesh. The stiffness matrix is related to the parameters $\mathbf{x}$ through the mesh $\boldsymbol{\mathcal{X}}$ as $\mathbf{K}\left(\boldsymbol{\mathcal{X}}(\mathbf{x})\right)$. It therefore requires the use of the chain rule to obtain the derivative:

$$\frac{d\mathbf{K}_{ff}}{d\mathbf{x}} = \frac{d\mathbf{K}_{ff}}{d\boldsymbol{\mathcal{X}}}\frac{d\boldsymbol{\mathcal{X}}}{d\mathbf{x}} \tag{103}$$

Obtaining the mesh sensitivity $\frac{d\boldsymbol{\mathcal{X}}}{d\mathbf{x}}$ will be the focus of this chapter. This will first be done numerically in §5.2.2 by resolving the mesh equilibrium, with free boundary nodes, for a given perturbed parameter and using a forward finite difference method to calculate the sensitivity. This will be followed in §5.2.3, which is the focus of this study; obtaining analytical mesh sensitivities with free boundary nodes.

In §5.3 through §5.5 comparison of the numerical mesh sensitivities for 2D and 3D are discussed. To ease visualisation, the focus will be on 2D mesh sensitivities, while 3D mesh sensitivities will be visualised as 2D as far as possible.

5.2. Mesh nodal coordinate sensitivities

5.2.1. Output of the sensitivities

Sensitivities are calculated as partial derivatives at each of the $i = 1 \dots n$ control variables x_i. These partial derivatives, from the individual perturbations on the control variables, are assembled into the $m \times n$ (where m is the number of nodal coordinates) sensitivity matrix $\frac{d\boldsymbol{\mathcal{X}}}{d\mathbf{x}}$:

$$\frac{d\boldsymbol{\mathcal{X}}}{d\mathbf{x}} = \left[\frac{d\boldsymbol{\mathcal{X}}}{d\mathrm{x}_{i=1}} \quad \frac{d\boldsymbol{\mathcal{X}}}{d\mathrm{x}_{i=2}} \quad \dots \quad \frac{d\boldsymbol{\mathcal{X}}}{d\mathrm{x}_{i=n}}\right] \tag{104}$$

The sensitivity associated with the i^{th} control variable $\frac{d\boldsymbol{\mathcal{X}}}{d\mathrm{x}_i}$ is calculated by perturbing only one variable x_i at a time. Sensitivities can be computed either numerically or analytically as discussed in §5.2.2 and §5.2.3 respectively.

5.2.2. Numerical sensitivities

After an adjustment of the domain boundary by a small value of $\Delta\mathrm{x}_i$, the mesh is resolved for equilibrium iteratively for each of the x_i control variables. Numerical sensitivities are determined using a forward finite difference step. Although this method is simple to implement, computational effort is high for large meshes and many control variables, since the full equilibrium of the system needs to be resolved for each variable.

The forward finite difference method is used to calculate the system sensitivities is as follows:

$$\frac{d\boldsymbol{\mathcal{X}}}{d\mathrm{x}_i} = \frac{\boldsymbol{\mathcal{X}}(\mathbf{x} + \Delta\mathrm{x}_i \times \boldsymbol{e}_i) - \boldsymbol{\mathcal{X}}(\mathbf{x})}{\Delta\mathrm{x}_i} \tag{105}$$

where $\boldsymbol{e}_i$ is the unit vector for the i^{th} component. Numerical sensitivities are easier to implement than the analytical sensitivities, but the analytical sensitivities are much faster to compute. The numerical sensitivities will be used to verify the correct implementation of the analytical sensitivities.

5.2.3. Analytical sensitivities

Since this study solves a geometrically non-linear truss problem using Newton's method and given the dependency of the mesh $\boldsymbol{\mathcal{X}}$ on the control variables $\mathbf{x}$ for the shape optimisation problem, the residual force functions $\boldsymbol{F}(\boldsymbol{\mathcal{X}})$ can be rewritten. The mesh nodal coordinates $\boldsymbol{\mathcal{X}}(\mathbf{x})$ are also split into the nodes associated with the interior of domain Ω and those with the domain boundary $\partial\Omega$:

$$\boldsymbol{F}\left(\boldsymbol{\mathcal{X}}^{\Omega}(\mathbf{x}), \boldsymbol{\mathcal{X}}^{\partial\Omega}(\mathbf{x})\right) = \mathbf{0} \tag{106}$$

The forces $\boldsymbol{F}$ can be partitioned into the nodes associated with the interior of domain Ω and those with the domain boundary $\partial\Omega$ giving:

$$\boldsymbol{F}\left(\boldsymbol{\mathcal{X}}^{\Omega}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega}(\mathbf{x})\right)=\begin{Bmatrix}\boldsymbol{F}_{\Omega}\left(\boldsymbol{\mathcal{X}}^{\Omega}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega}(\mathbf{x})\right)\\ \boldsymbol{F}_{\partial\Omega}\left(\boldsymbol{\mathcal{X}}^{\Omega}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega}(\mathbf{x})\right)\end{Bmatrix}=\mathbf{0} \tag{107}$$

Similarly, the mesh sensitivities are split into the sensitivity of the interior nodes $\frac{d\boldsymbol{\mathcal{X}}^{\Omega}}{d\mathbf{x}}$ and sensitivity of the boundary nodes $\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega}}{d\mathbf{x}}$:

$$\frac{d\boldsymbol{\mathcal{X}}}{d\mathbf{x}}=\begin{bmatrix}\frac{d\boldsymbol{\mathcal{X}}^{\Omega}}{d\mathbf{x}}\\ \frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega}}{d\mathbf{x}}\end{bmatrix} \tag{108}$$

To obtain the relationship between the nodal coordinates $\boldsymbol{\mathcal{X}}$ and control variables $\mathbf{x}$, the derivative of $\boldsymbol{F}\left(\boldsymbol{\mathcal{X}}^{\Omega}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega}(\mathbf{x})\right)$ in (107) is taken with respect to $\mathbf{x}$. The derivative of the system is taken of the truss system in equilibrium $\boldsymbol{F}=\mathbf{0}$ therefore giving $\frac{d\boldsymbol{F}}{d\boldsymbol{x}}=\mathbf{0}$**.** This gives the partitioned derivatives as:

$$\frac{d\boldsymbol{F}_{\Omega}}{d\mathbf{x}}=\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}}\frac{d\boldsymbol{\mathcal{X}}^{\Omega}}{d\mathbf{x}}+\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega}}{d\mathbf{x}}=\mathbf{0} \tag{109}$$

$$\frac{d\boldsymbol{F}_{\partial\Omega}}{d\mathbf{x}}=\frac{\partial\boldsymbol{F}_{\partial\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}}\frac{d\boldsymbol{\mathcal{X}}^{\Omega}}{d\mathbf{x}}+\frac{\partial\boldsymbol{F}_{\partial\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega}}{d\mathbf{x}}=\mathbf{0} \tag{110}$$

with the solution to the exterior and interior domains obtained from the system of equations in (109) and (110). If the location of the boundary nodes is a function of only the control variables $\mathbf{x}$, then $\frac{\partial\boldsymbol{F}_{\partial\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}}=\frac{\partial\boldsymbol{F}_{\partial\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}=\mathbf{0}$ and $\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega}}{d\mathbf{x}}$ is known, $\frac{d\boldsymbol{\mathcal{X}}^{\Omega}}{d\mathbf{x}}$ can then be solved from the reduced linear system:

$$\frac{\partial\boldsymbol{F}_{\boldsymbol{\Omega}}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}}\frac{d\boldsymbol{\mathcal{X}}^{\Omega}}{d\mathbf{x}}=-\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega}}\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega}}{d\mathbf{x}} \tag{111}$$

The decomposition of $\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}}$ only needs to be computed once to solve the multiple RHS associated with $\mathbf{x}$**.** This is the method implemented by Wilke *et al.* [7] since their boundary nodes were prescribed on the piece-wise linear boundary as a function of the full system. $\partial\boldsymbol{F}_{\Omega}/\partial\boldsymbol{\mathcal{X}}^{\Omega}$ is obtained from the last Newton step of the constrained boundary problem.

Since the MPC components reside on the boundary $\partial\Omega$ and are influenced by both the control variables $\mathbf{x}$ and the internal nodes Ω, the system in (109) and (110) needs to be resolved as a system of equations. Therefore, the nodal coordinates $\boldsymbol{\mathcal{X}}$ and force equilibrium equations $\boldsymbol{F}$ are split into the interior of domain Ω , the MPC $\partial\Omega_{\mathrm{c}}$ and prescribed $\partial\Omega_{\mathrm{p}}$ DOF. This is done the same as in way as (106) to (108) to give the equilibrium system as:

$$\boldsymbol{F}\left(\boldsymbol{\mathcal{X}}^{\Omega}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{c}}}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{p}}}(\mathbf{x})\right)=\begin{Bmatrix}\boldsymbol{F}_{\Omega}\left(\boldsymbol{\mathcal{X}}^{\Omega}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{c}}}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{p}}}(\mathbf{x})\right)\\ \boldsymbol{F}_{\partial\Omega_c}\left(\boldsymbol{\mathcal{X}}^{\Omega}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{c}}}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{p}}}(\mathbf{x})\right)\\ \boldsymbol{F}_{\partial\Omega_{\mathrm{p}}}\left(\boldsymbol{\mathcal{X}}^{\Omega}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{c}}}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{p}}}(\mathbf{x})\right)\end{Bmatrix}=\mathbf{0} \tag{112}$$

Taking the derivatives of (112) around the equilibrium point $\boldsymbol{F}=\mathbf{0}$ ($\therefore \frac{d\boldsymbol{F}}{d\mathbf{x}}=\mathbf{0}$) as done in (109)-(110) and assembling the resulting components into matrix form gives:

$$\begin{Bmatrix}\frac{d\boldsymbol{F}_{\Omega}}{d\mathbf{x}}\\ \frac{d\boldsymbol{F}_{\partial\Omega_c}}{d\mathbf{x}}\\ \frac{d\boldsymbol{F}_{\partial\Omega_p}}{d\mathbf{x}}\end{Bmatrix}=\begin{bmatrix}\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}} & \frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_c}} & \frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}\\ \frac{\partial\boldsymbol{F}_{\partial\Omega_c}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}} & \frac{\partial\boldsymbol{F}_{\partial\Omega_c}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_c}} & \frac{\partial\boldsymbol{F}_{\partial\Omega_c}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}\\ \frac{\partial\boldsymbol{F}_{\partial\Omega_p}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}} & \frac{\partial\boldsymbol{F}_{\partial\Omega_p}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{c}}}} & \frac{\partial\boldsymbol{F}_{\partial\Omega_p}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}\end{bmatrix}\begin{Bmatrix}\frac{d\boldsymbol{\mathcal{X}}^{\Omega}}{d\mathbf{x}}\\ \frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_c}}{d\mathbf{x}}\\ \frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}{d\mathbf{x}}\end{Bmatrix}=\begin{Bmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\end{Bmatrix} \tag{113}$$

Since $\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}{d\mathbf{x}}$ is non-zero and constant, the derivatives $\frac{\partial\boldsymbol{F}_{\partial\Omega_p}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}}=\frac{\partial\boldsymbol{F}_{\partial\Omega_p}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_c}}=\frac{\partial\boldsymbol{F}_{\partial\Omega_p}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}=0$. The linear system can, therefore, be rearranged to solve for the unknowns $\frac{d\boldsymbol{\mathcal{X}}^{\Omega}}{d\mathbf{x}}$ and $\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{c}}}}{d\mathbf{x}}$ to form:

$$\begin{bmatrix}\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}} & \frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_c}}\\ \frac{\partial\boldsymbol{F}_{\partial\Omega_c}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}} & \frac{\partial\boldsymbol{F}_{\partial\Omega_c}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_c}}\end{bmatrix}\begin{Bmatrix}\frac{d\boldsymbol{\mathcal{X}}^{\Omega}}{d\mathbf{x}}\\ \frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{c}}}}{d\mathbf{x}}\end{Bmatrix}=-\begin{bmatrix}\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}\\ \frac{\partial\boldsymbol{F}_{\partial\Omega_c}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}\end{bmatrix}\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}{d\mathbf{x}} \tag{114}$$

Whilst the system in (114) now distinguishes the MPC nodes, the MPC constraint contributions have not yet been included. The matrix on the LHS comprises the consistent tangent components available from the last Newton step. The LHS of the system may exist in the $\mathbf{LDL}^T$ decomposed state provided a sparse direct solver is used. The efficient computation of the sensitivities is enhanced by reusing the last step of the Newton solution, although this may not be practical for very large matrices.

5.2.3.1. Lagrangian approach to the boundary sensitivities problem

Considering that the constraint equations $\boldsymbol{f}_{mpc}$ are a function of both the constrained boundary nodes $\boldsymbol{\mathcal{X}}^{\partial\Omega_c}$ and control variables $\mathbf{x}$ in the form of $\boldsymbol{f}_{mpc}\left(\boldsymbol{\mathcal{X}}^{\partial\Omega_c},\mathbf{x}\right)$ and considering that $\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_c}}{d\boldsymbol{u_c}}=1$ and $\frac{d\boldsymbol{\mathcal{X}}^{\Omega}}{d\boldsymbol{u_f}}=1$, (44) to (46) can be rewritten as:

$$\frac{d\mathcal{L}}{d\boldsymbol{\mathcal{X}}^{\Omega}}=\boldsymbol{\mathcal{F}}_{\Omega}^{int}-\boldsymbol{\mathcal{F}}_{\Omega}^{ext}=\mathbf{0} \tag{115}$$

$$\frac{d\mathcal{L}}{d\boldsymbol{\mathcal{X}}^{\partial\Omega_c}}=\boldsymbol{\mathcal{F}}_{\partial\Omega_c}^{int}-\boldsymbol{\mathcal{F}}_{\partial\Omega_c}^{ext}+\left(\frac{\partial\boldsymbol{f}_{mpc}\left(\boldsymbol{\mathcal{X}}^{\partial\Omega_c},\mathbf{x}\right)}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_c}}\right)^{\mathrm{T}}\boldsymbol{\lambda}=\mathbf{0} \tag{116}$$

$$\frac{d\mathcal{L}}{d\boldsymbol{\lambda}}=\boldsymbol{f}_{mpc}\left(\boldsymbol{\mathcal{X}}^{\partial\Omega_c},\mathbf{x}\right)=\mathbf{0} \tag{117}$$

Since the set of prescribed displacements $\boldsymbol{\mathcal{X}}^{\partial\Omega_p}(\mathbf{x})$ is a function of the control variables these can be incorporated into the system following the same method described in §2.6.3.1 and considering $\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}{d\boldsymbol{u}_p} = 1$ giving:

$$\frac{d\mathcal{L}}{d\boldsymbol{\mathcal{X}}^{\partial\Omega_p}} = \boldsymbol{\mathcal{F}}^{int}_{\partial\Omega_p} - \boldsymbol{\mathcal{F}}^{ext}_{\partial\Omega_p} + \boldsymbol{R}_{\partial\Omega_p} \tag{118}$$

Considering the internal forces are a function of the trusses then $\boldsymbol{\mathcal{F}}^{int}_{\Omega} = \boldsymbol{F}_{\Omega}$, $\boldsymbol{\mathcal{F}}^{int}_{\partial\Omega_c} = \boldsymbol{F}_{\partial\Omega_c}$ and $\boldsymbol{\mathcal{F}}^{int}_{\partial\Omega_p} = \boldsymbol{F}_{\partial\Omega_p}$, and external forces $\boldsymbol{\mathcal{F}}^{ext}_{\Omega} = \boldsymbol{\mathcal{F}}^{ext}_{\partial\Omega_c} = \boldsymbol{\mathcal{F}}^{ext}_{\partial\Omega_p} = 0$. Taking the derivative of (115) to (118) with respect to the control variables $\mathbf{x}$ and noting $\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}{d\mathbf{x}}$ is known, then following the logic of (112) to (114) it can be shown:

$$\begin{bmatrix} \frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\Omega}} & \frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_c}} & \mathbf{0} \\ \frac{\partial \boldsymbol{F}_{\partial\Omega_c}}{\partial \boldsymbol{\mathcal{X}}^{\Omega}} & \frac{\partial \boldsymbol{F}_{\partial\Omega_c}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_c}} + \frac{\partial^2 \boldsymbol{f}_{mpc}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_c{}^2}}\boldsymbol{\lambda} & \frac{\partial \boldsymbol{f}_{mpc}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_c}}^T \\ \mathbf{0} & \frac{\partial \boldsymbol{f}_{mpc}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_c}} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \frac{d\boldsymbol{\mathcal{X}}^{\Omega}}{d\mathbf{x}} \\ \frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_c}}{d\mathbf{x}} \\ \frac{d\boldsymbol{\lambda}}{d\mathbf{x}} \end{Bmatrix}$$
$$= -\begin{Bmatrix} \mathbf{0} \\ \left(\frac{\partial^2 \boldsymbol{f}_{mpc}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_c}\partial\mathbf{x}}\right)^T \boldsymbol{\lambda} \\ \frac{\partial \boldsymbol{f}_{mpc}}{\partial \mathbf{x}} \end{Bmatrix} - \begin{Bmatrix} \frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_p}} \\ \frac{\partial \boldsymbol{F}_{\partial\Omega_c}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_p}} \\ \mathbf{0} \end{Bmatrix} \frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}{d\mathbf{x}} \tag{119}$$

where $\boldsymbol{\lambda}$ is included and treated as it would be for any Newton step update since it is required to maintain equilibrium of the system. The prescribed components of the consistent tangent $\frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_p}}$ are retained since $\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}{d\mathbf{x}}$ is a non-zero and its contribution to the RHS of the system is required. All contributions of the equilibrium matrix on the LHS are available from the consistent tangent calculated during the last Newton step. The RHS terms $\left(\frac{\partial^2 \boldsymbol{f}_{mpc}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_c}\partial\mathbf{x}}\right)^T \boldsymbol{\lambda}$, $\frac{\partial \boldsymbol{f}_{mpc}}{\partial \mathbf{x}}$ and $\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}{d\mathbf{x}}$ need to be recomputed for each partial derivative of the control variables $\mathbf{x}$. The components $\frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_p}}$ and $\frac{\partial \boldsymbol{F}_{\partial\Omega_c}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_p}}$ only need to be computed at initialisation of the sensitivity calculation. Again, this implies that a single decomposition of the LHS is leveraged to solve multiple RHSs, one RHS for each control variable (x_i) perturbed, to allow us to compute direct sensitivities.

5.3. Verification of the implementation in 2D

For the Lagrangian analytical sensitivities, the RHS terms $\left(\frac{\partial^2 \boldsymbol{f}_{mpc}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_c}\partial \mathbf{x}}\right)^T$, $\frac{\partial \boldsymbol{f}_{mpc}}{\partial \mathbf{x}}$ and $\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_\mathrm{p}}}{d\mathbf{x}}$ for the analytical sensitivities were calculated numerically using a finite-difference step. This makes the resulting sensitivities *selectively* semi-analytical, as opposed to more traditional semi-analytical sensitivities where the entire residual in (115) to (117) would be recomputed for the finite difference step. This removed the complexity associated with obtaining these components to allow verification of the analytical sensitivity derivation. These sensitivities will be referred to as analytical further.

To verify the analytical sensitivities the results are compared to the numerically computed sensitivity. For the comparisons that follow the same $\Delta \mathrm{x}_i$ perturbation sizes were used for both the selective analytical and numerical sensitivities. The numerical sensitivities for the system were solved to a residual norm of $\|\boldsymbol{\mathcal{R}}\|_2 < 10^{-10}$.

The control parameters of the ellipse used in these studies are shown in Figure 33. The ellipse was chosen as it is a simple boundary that can easily be intuitively interpreted for the sensitivities. It allows large changes in parameter sensitivity and curvature through manipulation of only a few variables. This allows for easier discussion around the effectiveness of the method by removing unnecessary complexities associated with complex boundaries.

Parameters x_1 and x_2 control the semi-axes of the ellipse. The height is governed by and x_1 and the width by x_2. An interior control point is also shown located by x_3 and x_4. The interior control point is used to demonstrate the effect of direct local manipulation of the mesh $\boldsymbol{\mathcal{X}}$. The ellipse without the interior control point is denoted $f_{ellip}(\mathrm{x}_1, \mathrm{x}_2)$ whereas the ellipse with the interior control point is denoted $f_{ellip}(\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, \mathrm{x}_4)$.

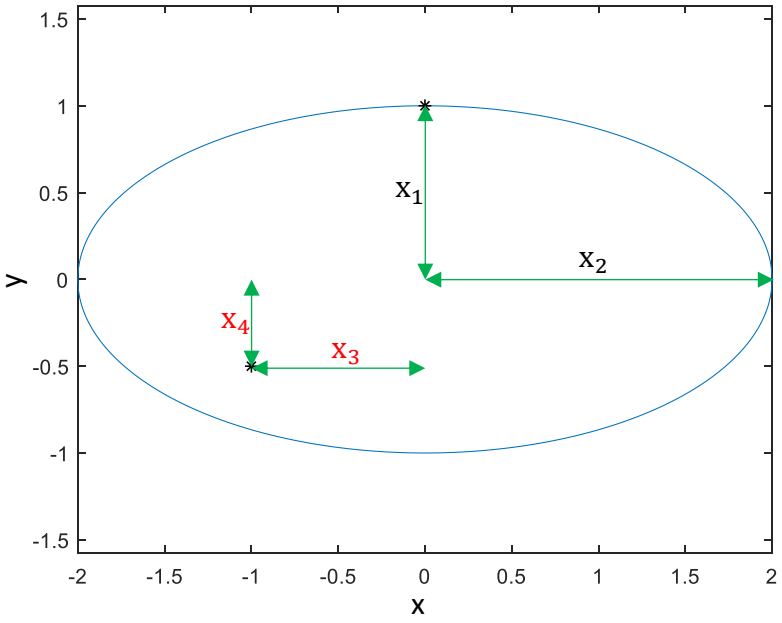


Figure 33 Ellipse parameterisation

For $f_{ellip}(\mathrm{x}_1, \mathrm{x}_2)$ the errors between the sensitivities obtained for each DOF in the system were compared for aspect ratios $1:2, 1:4, 1:8, 1:16$, and $1:32$ while the area was kept constant for the initialised ellipses. For each of these aspect ratios, the sensitivities were evaluated and compared for perturbations $\Delta \mathrm{x}_i = 10^{-3},\ 10^{-4},\ 10^{-5},\ 10^{-6},\ 10^{-7},\ 10^{-8}$ and 10^{-9}. The same perturbations were used for both the analytical and numerical sensitivities for the comparisons.

The absolute errors between the numerical and analytical sensitivities at each j^{th} nodal coordinate $d\mathcal{X}_\mathrm{j}$ was calculated as:

$$ERROR_{sensitivites} = \left| \frac{d\mathcal{X}_\mathrm{j}}{d\mathrm{x}_\mathrm{i}}_{numerical} - \frac{d\mathcal{X}_\mathrm{j}}{d\mathrm{x}_\mathrm{i}}_{analytical} \right| \tag{120}$$

and the norm of the sensitivity error as:

$$\left\| \left\{ \frac{d\boldsymbol{\mathcal{X}}}{d\mathrm{x}_\mathrm{i}} \right\}_{numerical} - \left\{ \frac{d\boldsymbol{\mathcal{X}}}{d\mathrm{x}_\mathrm{i}} \right\}_{analytical} \right\|_2 \tag{121}$$

Figure 34 (a) and (b) show the norm of the sensitivity errors (121) in the x- and y- directions. The legend shows the various ellipse aspect ratios $\mathrm{x}_1 : \mathrm{x}_2$. The sensitivities associated with x_2 are shown respectively for the x- and y- DOFs in (c) and (d). As the ellipse flattens out ($\mathrm{x}_1 : \mathrm{x}_2 \rightarrow 1:32$) the sensitivity error increases in Figure 34 (a), (b) and (c). In Figure 34 (a) and (b) this is due to the curvature becoming more and more sensitive to the height change x_1. Since forward finite difference sensitivities can only exactly capture linear changes, the increase in curvature reduces the method's accuracy.

For all four sets of sensitivities, the error between the methods decreases linearly with the perturbation size from 10^{-2} down to 10^{-6}. Furthermore, the errors are of the same order of magnitude as the

perturbation. This indicates that the MPC analytical sensitivity method is returning the correct sensitivities. Below a perturbation of 10^{-6} the errors start to deviate from the expected errors as the numerical roundoff limits are reached. The round-off errors affect the numerical sensitivities calculation causing the errors to deviate from the expected.

In §4.3.4 the contribution of the second-order term to the sensitivity was shown and discussed. The second-order term is included here in the computation of sensitivities, and it is noted that as the curvature increases so does the magnitudes of the contributions to the consistent tangent. These increasing magnitudes affect the numerical accuracy for smaller perturbations. This is caused by round off errors associated during the numerical sensitivity's computation. This is most noticeable in Figure 34 (c) and (d) for the 1:32 ratio ellipse where the error begins to increase below perturbations of 10^{-4}.

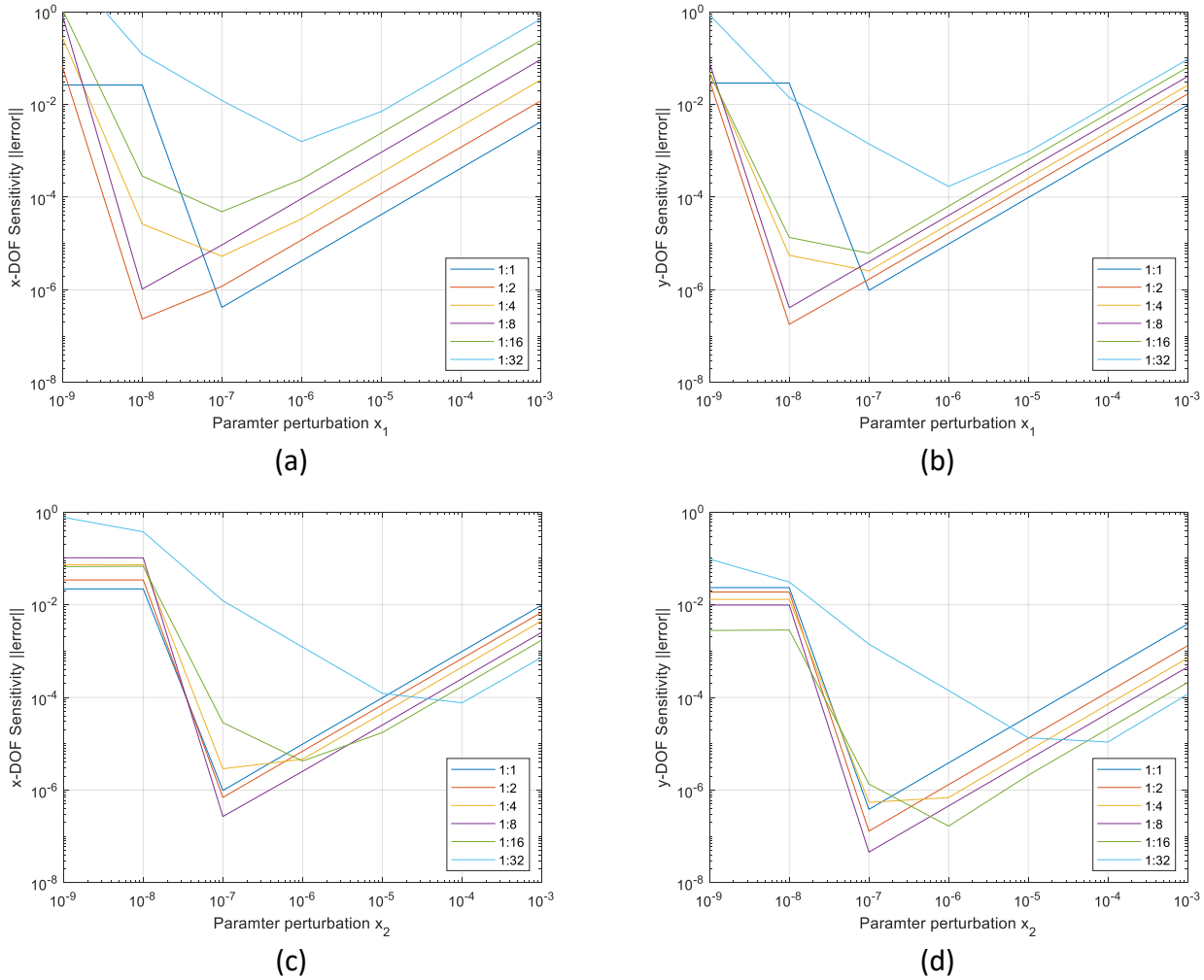


Figure 34 Numerical vs analytical sensitivity errors for various aspect ratio ellipse
(a) $\mathbf{x_1}$, x- direction. (b) $\mathbf{x_1}$, y- direction (c) $\mathbf{x_2}$, x-direction (d) $\mathbf{x_2}$, y-direction

For an ellipse with an aspect ratio of $1{:}\,4$ ($\mathrm{x_1}{:}\,\mathrm{x_2}$) the average errors (calculated as the average of the errors in (120)) and error norms (121) are shown in Figure 35 (a) and (b) respectively for various mesh densities denoted by various desired truss lengths for the same domain. The parameter and direction

combination with the largest deviations in error is shown; this was for x_2 in the x-direction. It is clear from Figure 35 (a) and (b) that the decrease in the desired length has little effect on the errors between the numerical and analytical sensitivities. Again, computational limits associated with numerical round-off cause perturbations below 10^{-6} to reduce the accuracy of the sensitivities. For this problem using a value between 10^{-3} and 10^{-6} yields the best results for the numerical sensitivities. Errors associated with the analytical sensitivity implementation only became noticeable below perturbations of 10^{-9}. For the numerical sensitivities, it is likely that the selection of the perturbation value will be problem dependant. Whereas the analytical sensitivities are more robust and a perturbation of between 10^{-8} and 10^{-6} would be a good selection.

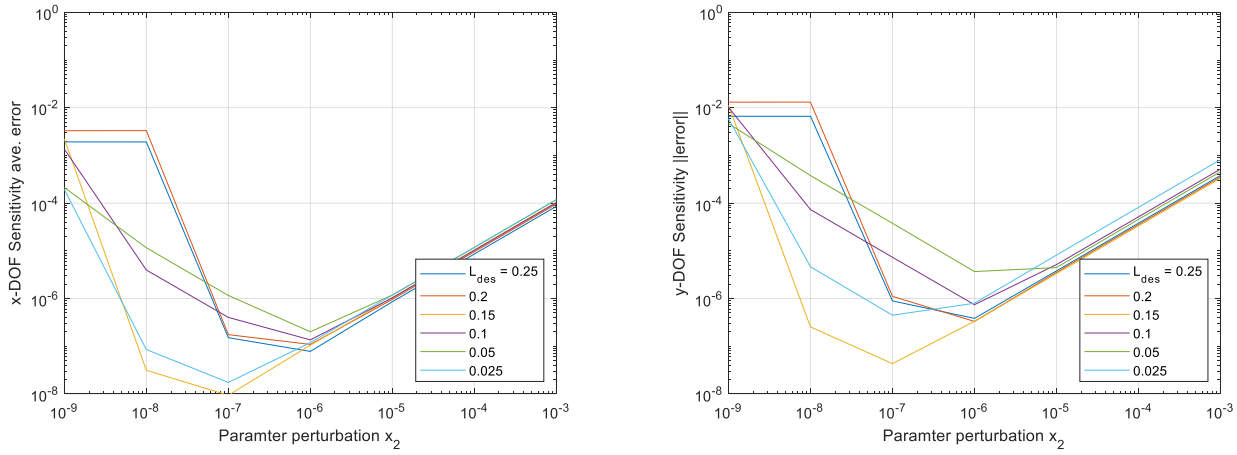


Figure 35 Mesh sensitivity errors for x_2 (x-direction) of ellipse $x_1 = 1$ and $x_2 = 4$, for various L_{des} (a) x-DOF average error (b) y-DOF error norm

Figure 36 shows the sensitivity errors for the parameters x_1 and x_2 in each of the x- and y-DOFs. This is shown for an ellipse with a $1:4$ aspect ratio and perturbations of 10^{-6} for both numerical and analytical. From the plots, it is seen that the errors are not constant throughout the domain. The parameter x_1 is shown to have the greatest effect on the errors between the numerical and analytical sensitivities due to the curvature being more sensitive to x_1 when compared to x_2. It is interesting to note that the largest errors in the sensitivities lie on the portion of the boundary with lower curvature. Again, the effect of a small change in x_1 has a large percentage effect on the portion of the boundary with lower curvature in those regions. This indicates the rate-of-change of the curvature, in response to the perturbations, are the most non-linear in those regions. These studies demonstrate where the first-order Taylor series expansion for the sensitivities is the least accurate. It must be noted that only the top centre point was fixed in this study. This is most noticeable in the x_1 y-DOF image (top, right).

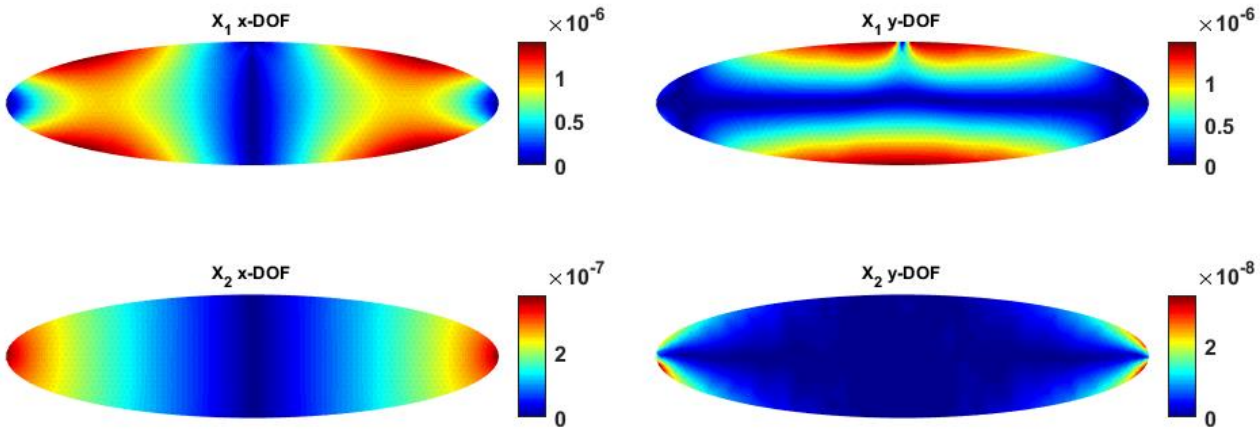


Figure 36 Errors between Lagrangian numerical and analytical sensitivities (ellipse 1:4 and perturbations 10^{-6})

5.4. Mesh response for 2D

In Figure 37 the sensitivity magnitudes for an ellipse $f_{ellip}(\mathrm{x}_1, \mathrm{x}_2)$ are shown. This demonstrates the smooth sensitivities throughout the domain in response to the parameter x_1 and x_2.

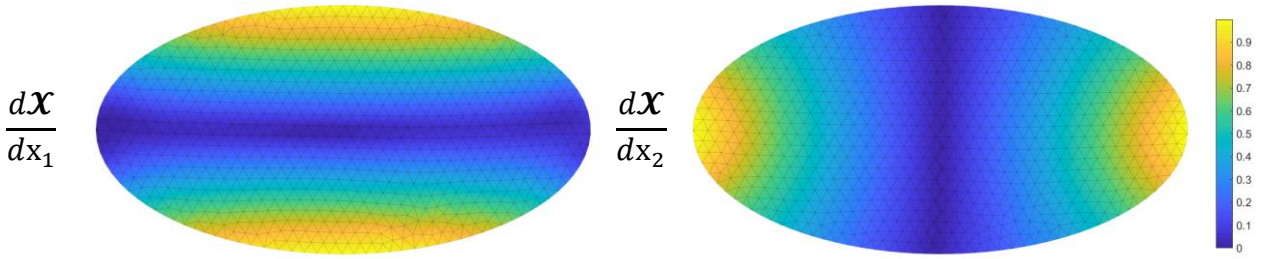


Figure 37 Sensitivity magnitude visualisation for 1:2 ellipse with no interior fixed node

In Figure 38 the sensitivities magnitudes for an ellipse with an interior fixed node $f_{ellip}(\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, \mathrm{x}_4)$ are shown (as described in Figure 33, the location is shown with the red arrow). Again, it is clear from the smooth gradient throughout the domain, that the sensitivities do describe the effect throughout the domain as desired. The $\frac{d\boldsymbol{\mathcal{X}}}{d\mathrm{x}_3}$ and $\frac{d\boldsymbol{\mathcal{X}}}{d\mathrm{x}_4}$ plots show that perturbing even a single node causes a smooth sensitivity response throughout the domain. This is desirable from a shape optimisation point of view as stated in §1.5.4.

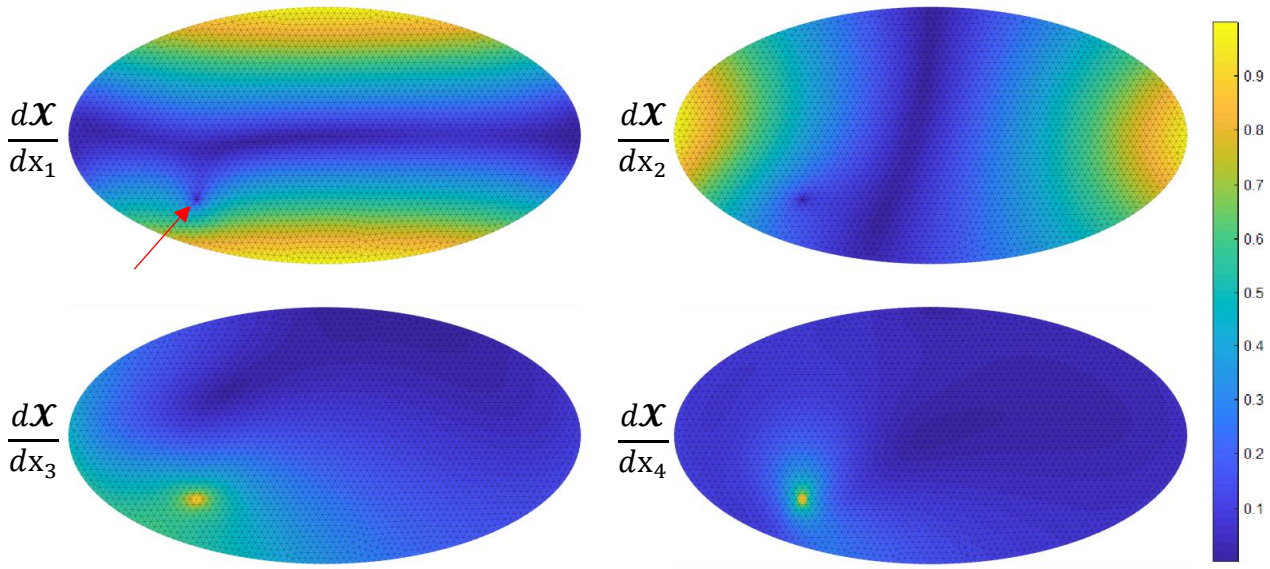


Figure 38 Sensitivity magnitude visualisation for 1:2 ellipse with interior fixed node

In Figure 39, for $f_{ellip}(\mathrm{x}_1, \mathrm{x}_2)$, a non-uniform mesh of varying element sizes is depicted; the mesh sensitivities are again smooth. It is interesting to note that the refined mesh areas (A), having shorter edge lengths and in turn higher stiffnesses, affect the sensitivity gradient in those regions. This is seen by the variation shown at (B). This can be compared with the symmetrical sensitivities for the uniform mesh shown in Figure 37.

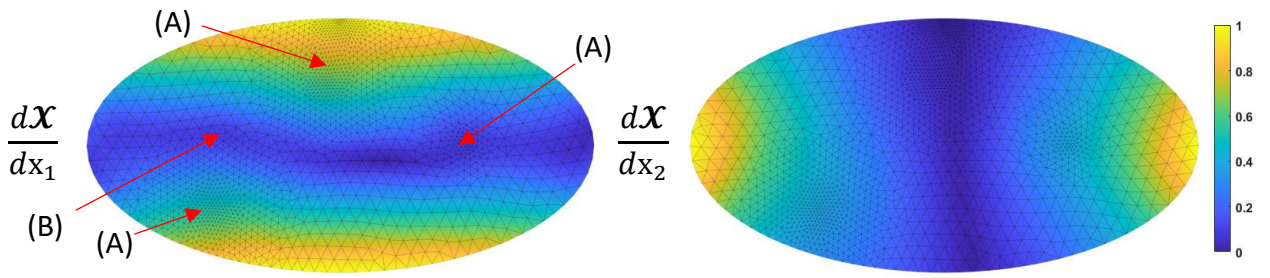


Figure 39 Sensitivity magnitude visualisation for ellipse 1:2 for non-uniform mesh

An alternative way to visualise sensitivities is to plot finite mesh updates when an internal or boundary node is displaced. In Figure 40 the internal node of $f_{ellip}(\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, \mathrm{x}_4)$ is displaced from $(-1, -0.5)$ to (-0.9,-0.5) by a value of $\Delta\mathrm{x}_3 = 0.1$. This perturbation shows the effect throughout the domain of the ellipse. The freeing of the boundary nodes, through the use of MPCs, allows the nodes to move smoothly along the boundary in response to the parameter update. This allows the mesh to maintain higher-quality elements in the update step. This is shown clearly in the magnified area by the movement of a node from (1) to (2). Figure 40 shows the new mesh estimation from only the sensitivity update $\boldsymbol{\mathcal{X}}_{\boldsymbol{n+1}} = \boldsymbol{\mathcal{X}}_{\boldsymbol{n}} + \frac{d\boldsymbol{\mathcal{X}}_{\boldsymbol{n}}}{d\mathrm{x}_3}\Delta\mathbf{x}_3$.

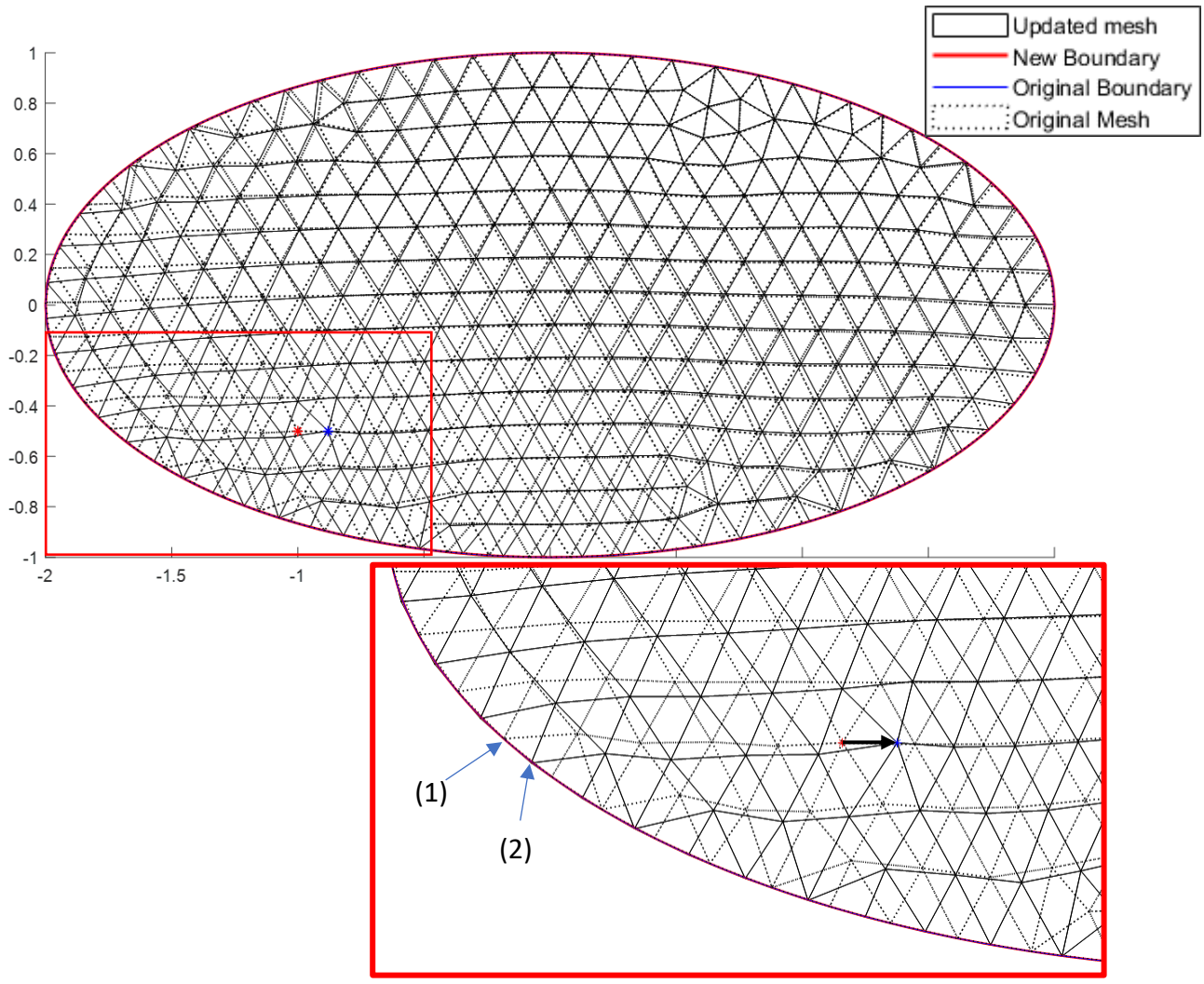


Figure 40 Mesh due to update of interior fixed node by parameter x_3

For the ellipse without the internal node $f_{ellip}(x_1, x_2)$, an update of the mesh for a change of +20% in x_1 and -10% in x_2 is shown in Figure 41. The mesh update shown is again an estimation of the updated mesh, based only on the prediction offered by the sensitivities $\boldsymbol{\mathcal{X}_{n+1}} = \boldsymbol{\mathcal{X}_n} + \left[\frac{d\mathcal{X}_n}{d\mathbf{x}}\right]\begin{Bmatrix}\Delta\mathbf{x}_1\\ \Delta\mathbf{x}_2\end{Bmatrix}$.

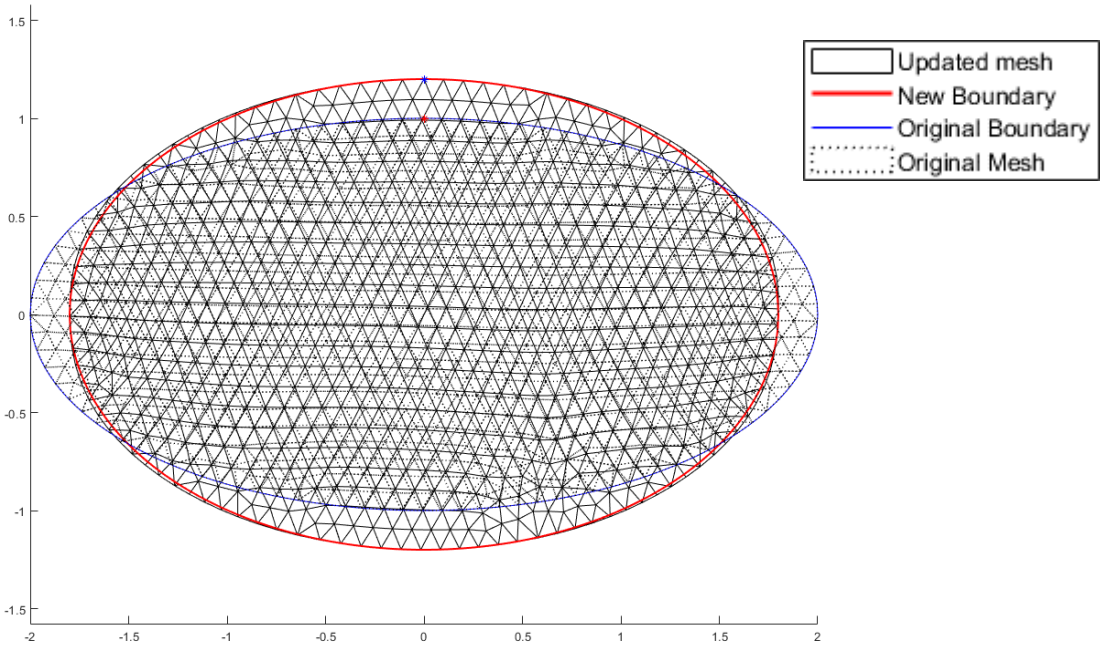


Figure 41 Non-uniform mesh deformation due to 20% height and 10% width adjustment

For an update of a non-uniform mesh, Figure 42 shows that even for a large update of +20% in x_1 and -10% in x_2, the mesh deforms evenly throughout the domain and the relative mesh sizing is retained. The update in x_1 at the top of the ellipse is more than 10 times larger than the average local element size and the mesh responds smoothly to the change.

The lower right magnification of Figure 42 shows the estimated mesh has deviated from the new expected boundary. This is due to the deformation in the boundary being non-linear and the update is based on the first order mesh sensitivities.

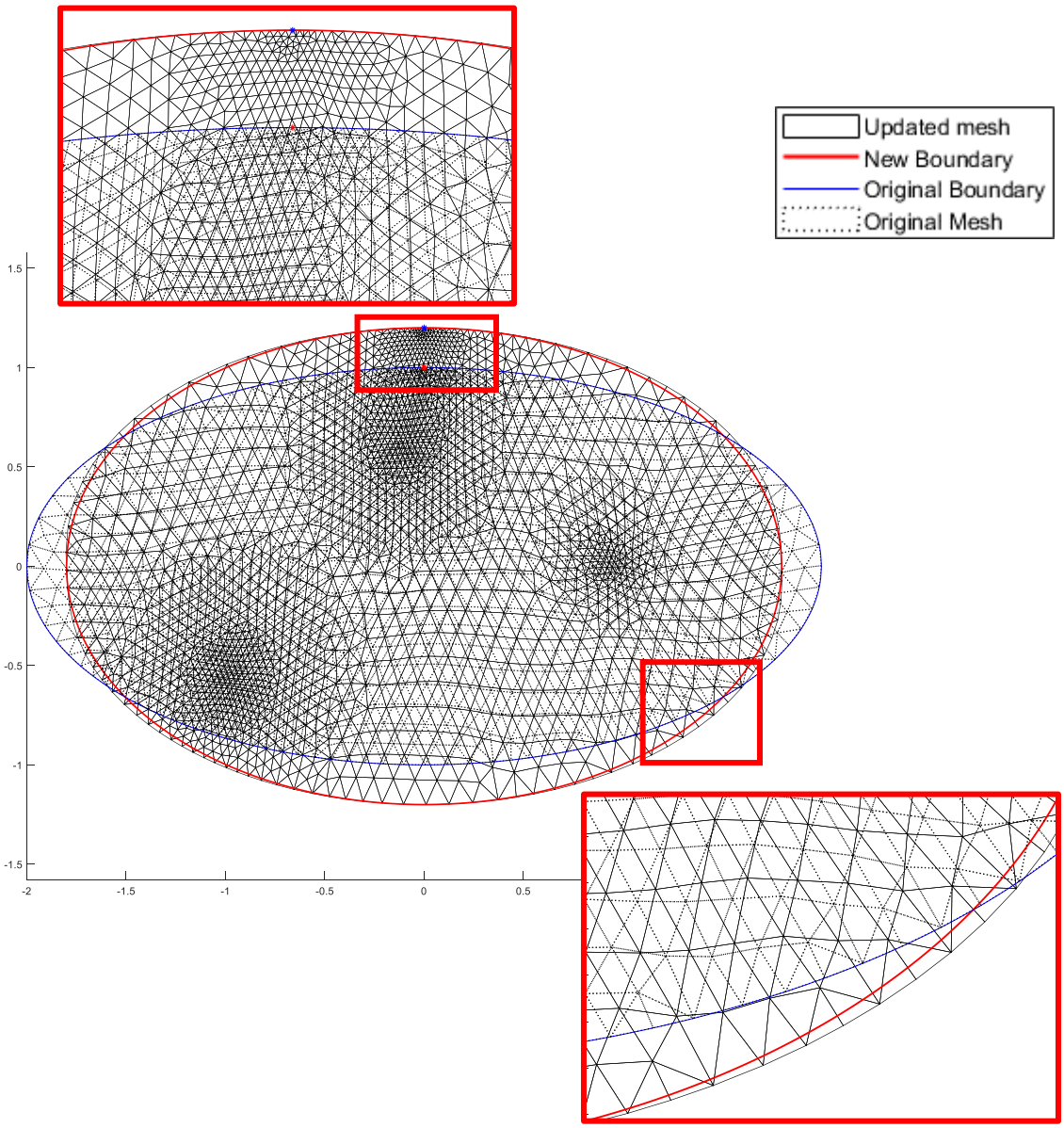


Figure 42 Non-uniform mesh deformation due to 20% height and 10% width adjustment

5.5. Verification of the implementation in 3D

Verification of correct implementation was obtained in the same manner as discussed in §5.3. The ellipsoid was defined using three parameters x_1, x_2 and x_3, corresponding to the three semi-axes and x_4 corresponding to the set of 3 nodes on the top surface of the sphere. The parameter x_4 is used to show the effect of direct local manipulation of the mesh.

Accuracy of the analytical implementation was verified by comparison with the numerical sensitivities obtained for ellipsoid configurations with aspect ratios equal to 1:1:1, 1:4:4 and 1:16:16 and perturbations of 10^{-3} and 10^{-5} were used for both the numerical and analytical methods. Table 11 shows the average of the sensitivity errors (120) between numerical and analytical implementations for the Lagrangian MPC approach for the various curvature problems:

Table 11 3D Lagrangian analytical sensitivity errors for ellipsoids 1:1:1, 1:4:4 and 1:16:16

$deps.$	10^{-3}			10^{-5}		
Parameter	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$
1:1:1	1.40×10^{-4}	1.38×10^{-4}	1.34×10^{-4}	1.41×10^{-6}	1.38×10^{-6}	1.34×10^{-6}
1:4:4	7.95×10^{-4}	2.65×10^{-5}	2.73×10^{-5}	8.64×10^{-6}	3.89×10^{-6}	3.90×10^{-6}
1:16:16	3.59×10^{-3}	3.95×10^{-6}	3.98×10^{-6}	3.60×10^{-5}	5.30×10^{-8}	5.31×10^{-8}

From Table 11 it is evident that the analytical implementation is returning accurate sensitivities. The analytical method shows average errors in the region of the perturbation value. Generally, the average sensitivity error is lower than the perturbation. This indicates that the method has been correctly implemented. As the curvature increases, it is noted that the sensitivity error associated with the x_1 parameter increases slightly above the perturbation magnitude. This is due to the same principle discussed in §5.3. The curvature does not change as rapidly in response to x_2 and x_3 (since their magnitude is 4 times and 16 times larger than x_1 for the 1:1:4 and 1:16:16 ellipsoids respectively), hence the error between methods decreases more than expected.

In Figure 43 a sphere was deformed by +20%, -20% and -10% in the three parametric directions x_1, x_2 and x_3 respectively. Deformation from the original mesh shows the mesh is updated as expected throughout the domain.

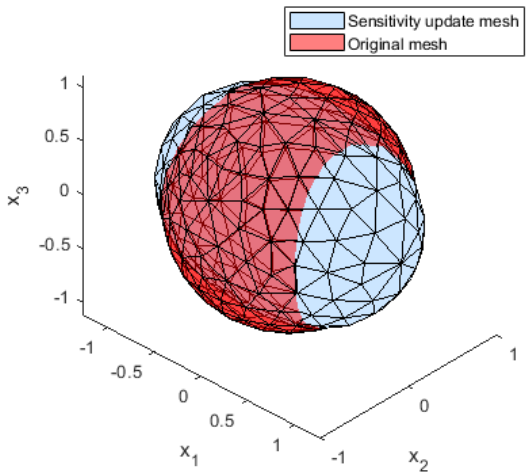


Figure 43 Ellipsoid with update estimation using Lagrangian analytical sensitivities

The sensitivities throughout the domain are shown for each parameter in Figure 44. The front quarter of the sphere has been removed to show the sensitivities in the interior of the domain. As was seen for the 2D mesh sensitivities, the 3D mesh sensitivities are smooth throughout the domain and the whole domain is shown to be influenced by a given boundary change. Figure 44 (a) through (c) shown the semi-axis sensitivities for x_1, x_2 and x_3 respectively. Figure 44 (d) shows the sensitivities for a set of three points on the upper surface controlled by the parameter x_4.

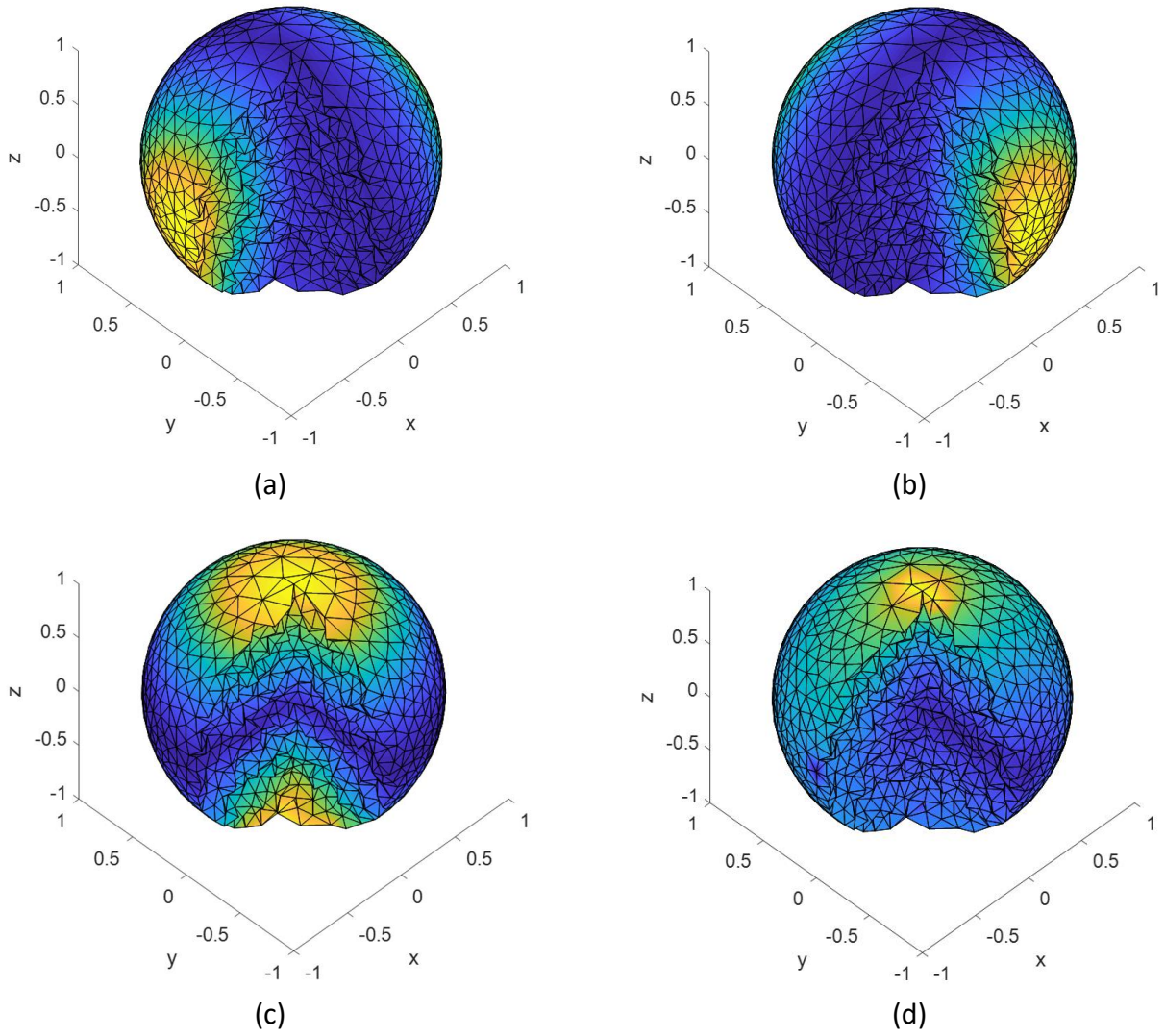


Figure 44 Mesh sensitivity responses for 3D sphere with front quarter removed

(a) $\frac{dX}{dx_1}$ **(b)** $\frac{dX}{dx_2}$ **(c)** $\frac{dX}{dx_3}$ **(d)** $\frac{dX}{dx_4}$

In Figure 45 a movement in $x_4 = 0.075$ is shown. The magnified portion of the image shows the three nodes on a line that were controlled by x_4. Figure 45 shows the displacement of three nodes along the surface from the initial position at (A) to the prescribed location (B). It is clear that the nodes can slide along the boundary to maintain the overall properties of the domain and report accurate sensitivities.

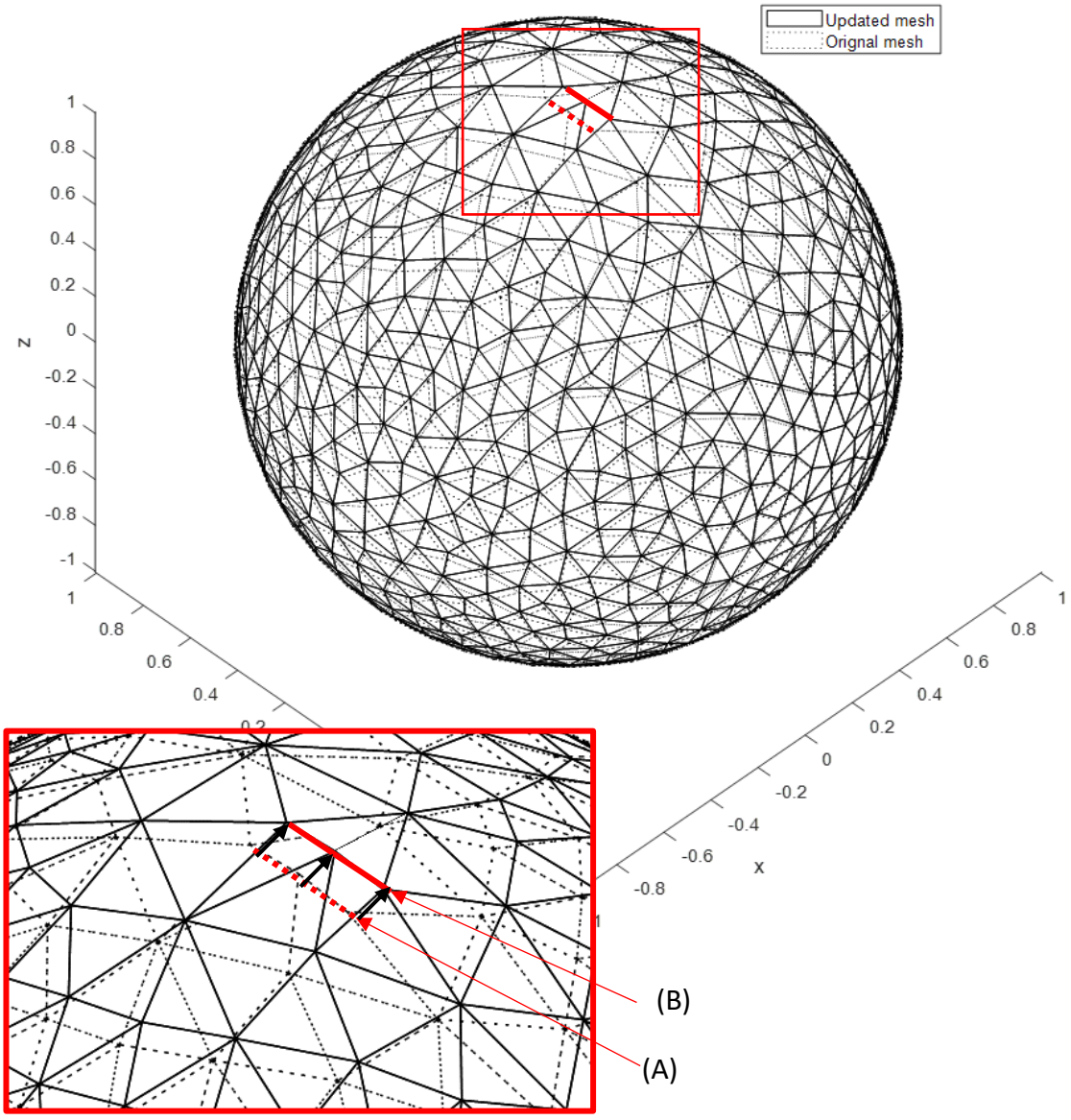
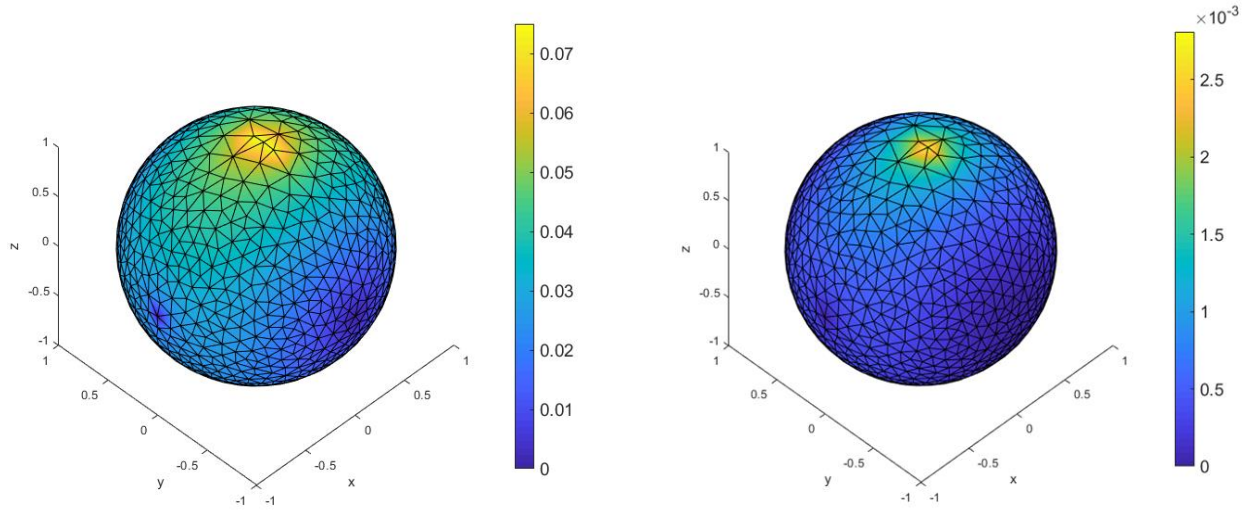


Figure 45 3D mesh update for $\mathcal{X} + \frac{d\mathcal{X}}{d\mathbf{x_4}} \times \mathbf{0.075}$

Figure 46 shows the magnitude of the movement of the surface nodes corresponding to the mesh movement in Figure 45 (a). It is clear that the node movement scales between 0 and 0.075 as is expected given the sensitivity profile in Figure 44. In (b) the resulting deviation from the surface of the geometry is shown. It is clear that the deviation from the surface is significantly less than the update magnitude of 0.075. The largest deviation is at the perturbed nodes where they deviate from the surface by 0.0028.



(a) (b)

Figure 46 Movement of surface nodes for $\mathbf{x_4 = 0.075}$ mesh estimation

(a) Update magnitude of surface nodes (b) Magnitude of deviation of nodes from the surface

5.6. Demonstration on concavity and non-linearity

Whilst the simplistic ellipse and ellipsoid have been used for discussion below demonstrates the methods effectiveness on a high curvature, convex and domain. The domain boundary has the closed form solution of:

$$f_{\partial\Omega} = -\mathrm{x}_4\boldsymbol{x}'^2 - \mathrm{x}_5\boldsymbol{y}'^{\boldsymbol{2}} - \mathrm{x}_6\boldsymbol{z}'^{\boldsymbol{2}} + \mathrm{x}_7\boldsymbol{x}'^{\boldsymbol{4}} + \mathrm{x}_8\boldsymbol{x}'^{\boldsymbol{4}} + \mathrm{x}_9\boldsymbol{x}'^{\boldsymbol{4}} + 1000 \tag{122}$$

Where:

$$\boldsymbol{x}' = \frac{\boldsymbol{x} - \mathrm{x}_1}{\mathrm{x}_{10}} \tag{123}$$

$$\boldsymbol{y}' = \frac{\boldsymbol{y} - \mathrm{x}_2}{\mathrm{x}_{11}} \tag{124}$$

$$\boldsymbol{z}' = \frac{\boldsymbol{z} - \mathrm{x}_3}{\mathrm{x}_{12}} \tag{125}$$

The initial domain has the variables $\Delta\mathbf{x} = \{0, 0, 0, 50, 50, 50, 1, 1, 1, 1, 1, 1\}^T$. An update to the domain of $\Delta\mathbf{x} = \{20, 5, 5, 3, 0, -4, -0.1, 0.1, 0, 0.1, 1, 0\}^T$ is applied and the system resolved. This is shown in Figure 47. Since the domain is highly non-linear and the sensitivities are linear, there is some discrepancy when changes of 10% are made to the domain. The translation components x_1 through x_3 give the exact result as expected. The predicted boundary is not exact since the remaining parameters are non-linearly related to the dimensions x, y and z.

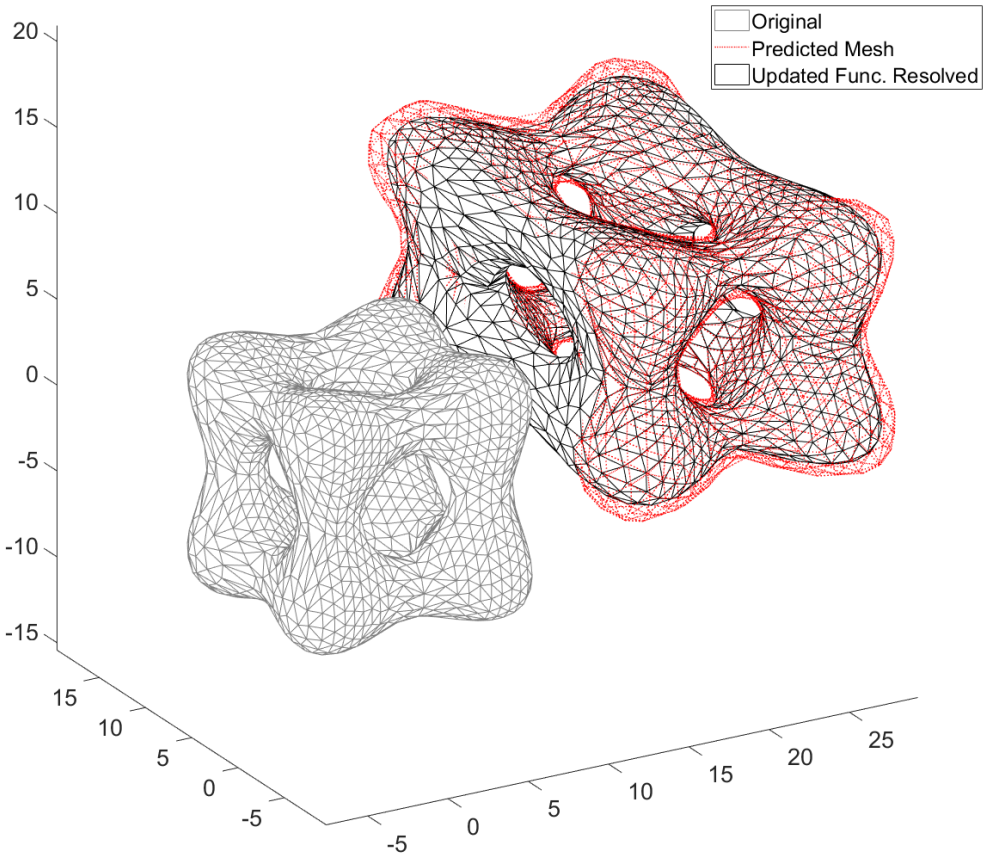


Figure 47 Predicted and resolved boundary for non-linear domain

5.7. Summary

In this chapter, the method for obtaining analytical sensitivities, that account for free movement of boundary nodes, was developed. It was shown to obtain the correct results when compared to numerical sensitivities. The numerical sensitivities were successfully implemented using a forward finite difference step to allow for comparison with analytical sensitivities.

The analytical sensitivities were successfully implemented using the Lagrangian MPC method. This allowed the nodes to move freely along the boundary as well as relaxing surface nodes from predefined positions. This reduces the need for remeshing between iterations as discussed by Wilke *et al.* [7].

Comparison of the numerical and analytical sensitivities for 2D and 3D showed that the errors between the two methods were on the same order of magnitude as the perturbation used to obtain the sensitivity. This was limited (for these problems) to perturbations of greater than 10^{-6} since numerical precision limited the accuracy of the finite-difference sensitivities, causing errors between the methods to increase.

The largest curvature cases had the largest effect on errors between the two methods. The parameter controlling the short dimension in both verifications produced an increased error with increased curvature. This was due to the relative change in the value of the parameter increasing. The increase in

the number of points (reduced edge length) was shown to not affect the accuracy of the system sensitivities appreciably.

The analytical sensitivities were demonstrated to act over the entire domain as desired for shape sensitivities. As a result, these sensitivities were able to predict large geometry changes of 20%. This was successfully demonstrated for both 2D and 3D. Further, the ability of the sensitivities to update along the boundary nodes for small perturbations was demonstrated and quantified for both 2D and 3D.

CHAPTER 6
CONCLUSION

"First rule of engineering; beware prototypes. Along with, avoid anything made by an engineer who doesn't have all his own fingers." — Simon R. Green

6.1. Discussion

In this study, a prototype mesher that outputs global mesh sensitivities was developed. The mesher was based on a non-linear finite element truss system that was solved iteratively using Newton's method to obtain static equilibrium of the system. The mesh sensitivities for the system were obtained using an analytical method. The accuracy of these sensitivities was demonstrated for 2D and 3D using smooth and continuous ellipse and ellipsoid geometries respectively.

The importance of accurate mesh sensitivities for obtaining accurate shape sensitivities for the gradient-based optimisation was highlighted and the preference of these sensitivities to describe mesh deformation throughout the domain was also discussed. The meshing method described herein was developed in this context and based on the forward-Euler truss based mesher DistMesh [33]. The conversion to a non-linear FEM based truss mesher was successfully completed and shown to obtain quadratic convergence.

The mesher was initially designed to utilise prescribed boundary nodes. From the outset, it was noted that having only prescribed boundary nodes was a limitation and that allowing the nodes to follow the boundary freely was important. This was especially true for the 3D implementation where prescribing boundary nodes over non-linear surfaces quickly presented significant difficulties.

To overcome the limitations associated with prescribing boundary nodes, traditional contact methods, and multipoint constraint methods (MPC) were investigated. Since the geometrical domain boundary needs to be "exactly" satisfied by the discretisation algorithm, of the methods investigated two methods were selected that satisfied this criterion. These were the Lagrangian and MSEM MPC methods. Both methods were successfully implemented for both 2D and 3D meshing platforms and shown to obtain the quadratic convergence associated with the correct implementation of Newton's method.

Examination of the two MPC methods revealed differences in the robustness and computational resources required to solve the truss system. The MSEM exhibited significant challenges in obtaining a converged solution, particularly for geometries with medium-to-high curvature. These problems, such as the introduction of complex roots during recovery of the slave DOF; impacted the convergence capabilities of the MSEM. Means of controlling these difficulties were developed and associated

improvement in convergence rates was shown. The Lagrangian method was shown to be far more robust than the MSEM.

The MSEM showed significant benefits in solution time in comparison with the Lagrangian method per Newton iteration. For the direct solver implementations, the Lagrangian method took up to 3.1 times longer to solve a single Newton iteration. For the iterative solver, the difference was reduced to a maximum of ∼2.5 times with 100% of nodes constrained as MPCs. This was due to the reduction of DOFs from the solved linear system in the MSEM. In contrast, the Lagrangian method augmented the linear system and increased the number of DOFs. The higher the percentage of MPC nodes constrained, the greater the discrepancy between the two methods. The associated increase in solution time was particularly prevalent for the direct solver of up to 3.1 times to solve the Lagrangian system for a single Newton iteration. The time impact associated with the additional DOFs of the Lagrangian method was significantly reduced by employing indirect solvers. The discrepancy was bought down to an average of 2.3 for large systems where 40% or more nodes participate in the MPCs.

The largest difference in resulting system size arises from the case where 100% of nodes in the system are MPC constrained. When coupled with the results from the iterative solver times and the additional Newton iterations required by the MSEM, the Lagrangian method required ∼6% longer solution times to resolve truss equilibrium. It was noted that the MSEM was often unable to converge or could require a substantial number more iterations to obtain convergence for medium to high curvature geometries.

The time associated with computing and assembling the MPC linear system for one Newton step was as much as 60% longer for the MSEM. In a matrix-free iterative solver, these calculations are performed for every iteration of the solver and the time associated with these calculations dominates the time associated with each iteration of the solver and, correspondingly, the time to solve one Newton Iteration. Based on the information obtained during this study, this implies that the Lagrangian MPC method will likely prove a faster implementation for matrix-free iterative solver implementations. Calculations showed the improved performance of the Lagrangian could be as much as 30% more efficient for matrix-free implementations under conditions where 100% of MPC nodes are constrained and 3% where 20% nodes are constrained. The Lagrangian method was selected as the preferred method for the boundary node control.

The method for obtaining the mesh sensitivities was then developed for the Lagrangian method. These were shown to be correct through comparison with numerical sensitivities produced using the forward finite difference method. The sensitivities were shown to allow for boundary node movement and to be able to describe mesh movement throughout the domain. The ability of the sensitivities to describe the behaviour of non-uniform meshes was also demonstrated to be accurate, as well as the ability of the mesh to adapt to large deformations in geometry.

Overall the mesher was shown to accomplish the objectives of this study. The main objectives were the ability to discretise non-linear domains and to produce reliable mesh sensitivities for use in gradient-based shape optimisation.

6.2. Suggestions for future work

Future development of this mesher could include:

1. In the current implementation, the length control functions discretely assign truss lengths to the system. For a non-uniform mesh, investigate the possibility of incorporating the gradients of the truss length assignment functions, to improve convergence in the regions of local mesh refinements. This would require alteration of the Newton truss implementation in §3.3 to incorporate the effect of $\boldsymbol{L}_{des}$ as the function $\boldsymbol{L}_{des}(\boldsymbol{X})$. This could potentially utilise RBF fields since they are smooth and continuous with analytical sensitivities easily computable. This will, however, require an efficient differentiation of RBF functions. This could then potentially also be used in the implementation of the sensitivities.
2. Implement a curvature-based adaptive refinement strategy to facilitate accurate domain discretisation with minimal input from the user.
3. Implement an adaptive meshing strategy that uses the error estimate fields to dictate refinement.
4. Implement the mesher rather as a sensitivities "wrapper" that can utilise standard NURBS geometry descriptions and a pre-meshed domain. Thereafter use this implementation to return an optimised mesh and mesh sensitivities. In this application, the MSEM may be robust enough. The acquisition of the MSEM sensitivities is developed in APPENDIX L to facilitate this.
5. Improve edge addition and removal to not require full retriangulation of the mesh for local mesh refinement.
6. Include faux "cross struts" in simplexes to reduce the likelihood of degenerating elements during updates. These could potentially use non-linear truss functions to retain a "minimum".
7. Include a "quality" enforcement strategy through altering the truss stiffness coefficient k_{truss} throughout the domain.

Alternatives:

1. Use the MPC boundary method with the stiffness matrix obtained from the system FEA to obtain internal nodal coordinate sensitivities.

LIST OF REFERENCES

[1] P. Le Tallec and E. Laporte, “Numerical Methods in Sensitivity Analysis and Shape Optimisation,” Springer Science + Business Media, New York, 2002.

[2] G. Allaire, F. Jouve and A.-M. Toader, “Structural optimisation using sensitivity analysis and a level-set method,” *Journal of Computational Physics,* vol. 194, pp. 363-393, 2004.

[3] O. Pironneau, Optimal Shape Design for Elliptic Systems, New York: Springer-Verlag, 1984.

[4] J. Simon, “Diffferentiation with respect to the domain in boundary value problems,” *Numerical Functional Analysis and Optimization vol. 2,* vol. 2, pp. 649-487, 1980.

[5] J. Sienz and E. Hinton, “Reliable Structural Optimization With Error Estimation, Adaptivity and Robust Sensitivity Analysis,” *Computers & Structures,* vol. 64, pp. 31-63, 1997.

[6] J. Haslinger and R. A. E. Mäkinen, Introduction to shape optimization: Theory, approximation, and computation, Philadelphia: SIAM Society for Industrial and Applied Mathematics, 2003.

[7] D. N. Wilke, S. Kok and A. A. Groenwald, “A quadratically convergent unstructured remeshing strategy for shape optimisation,” *International Journal for Numerical Methods,* vol. 65, no. 1, pp. 1-17, 2006.

[8] M. Garcia and C. Gonzalez, “Shape optimisation of continuum structures via evolution strategies and fixed grid finite element analysis,” *Structural and Multidisciplinary Optimization,* vol. 26, no. 1-2, p. 92–98, 2004.

[9] Y. M. Xie and G. Steven, “A simple evolutionary procedure for structural optimization,” *Computers & Structures,* vol. 49, no. 5, pp. 885-896, 1993.

[10] L. Rao and H. Chen, “The Technique of the Immersed Boundary Method: Application to Solving Shape Optimization Problem,” *Journal of Applied Mathematics and Physics,* vol. 5, no. 02, pp. 329-340, 2017.

[11] S. Hällgren, L. Pejryd and J. Ekengren, “(Re)Design for Additive Manufacturing,” in *26th CIRP Design Conference*, 2016.

[12] J. Plocher and A. Panesar, “Review on design and structural optimisation in additive manufacturing: Towards next-generation lightweight structures,” *Materials & Design,* vol. 183, 2019.

[13] R. T. Haftka and R. V. Gradhi, “Structural Shape Optimization - A Survey,” *Computer Methods in Applied Mechanics and Engineering,* vol. 57, pp. 91-106, 1986.

[14] Q. Li, G. P. Steven, O. M. Querin and Y. M. Xie, “Evolutionary Shape Optimisation for Stress Minimisation,” *Mechanics Research Communication,* vol. 26, no. 6, pp. 657-664, 1999.

[15] M. P. Bendsøe and N. Kikuchi, “Generating Optimal Topologies in Structural Design Using a Homogenization Method,” *Computer Methods in Applied Mechanics and Engineering,* vol. 71, no. 2, pp. 197-224, 1988.

[16] C. Mattheck, “Engineering Components Grow Like trees,” *MAterialwissenschaft und Werk stofftechnic,* vol. 21, pp. 143-168, 1990.

[17] T. Hughes, J. Cottrell and Y. Bazilevs, “Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement,” *Computer Methods in Applied Mechanics and Engineering,* vol. 194, no. 39-41, pp. 4135-4195, 2005.

[18] J. S. Dokken, S. W. Funke, A. Johansson and S. Schmidt, “Shape optimization using the finite element method on multiple meshes with Nitsche coupling,” 26 June 2018. [Online]. Available: https://arxiv.org/pdf/1806.09821.pdf. [Accessed 23 November 2109].

[19] A. de. Boear, M. van der Schoot and H. Bijl, “Mesh deformation based on radial basis function interpolation,” *Computers & Structures,* vol. 85, no. 11-14, pp. 784-795, 2007.

[20] P. Coulier and E. Darve, “Efficient mesh deformation based on radial basis function interpolation by means of the inverse fast multipole method,” *Computer Methods in Applied Mechanics and Engineering,* vol. 308, pp. 286-309, 2016.

[21] C. Degand and F. Charbel, “A three-dimensional torsional spring analogy method for unstructured dynamic meshes,” *Computers & Structures,* vol. 80, no. 3-4, pp. 305-316, 2002.

[22] J. S. Arora, Introduction to Optimum Design, 4th ed., London: Academic Press, 2016.

[23] E. Haug, E. Choi and E. Komkov, Design Sensitivity Analysis of Structural Systems, New York: Academic Press, 1986.

[24] K. Choi and E. Haug, “Shape design sensitivity analysis of elastic structures,” *Journal of Structural Mechanics,* vol. 11, no. 2, pp. 231-269, 1983.

[25] K. Bandara, T. Rüberg and F. Cirak, “Shape optimisation with multiresolution subdivision surfaces and immersed finite elements,” *Computer Methods in Applied Mechanics and Engineering,* vol. 300, no. 1, pp. 510-539, 2016.

[26] M. de' Michieli Vitturi and B. F., "A discrete gradient-based approach for aerodynamic shape optimisation in turbulent viscous flow," *Finite elements in Design,* vol. 43, no. 1, pp. 68-80, 2006.

[27] N. Olaff, J. Rasmussen and E. Lund, "A Method of "Exact" Numerical Differentiation for Error Elimination in Finite-Element-Based Semi-Analytical Shape Sensitivity Analyses," *Mechanics Based Design of Structures and Machines,* vol. 21, no. 1, pp. 1-66, 1993.

[28] O. Zienkiewics and D. Philips, "An automatic mesh generation scheme for plane and curved surfaces by 'isoparametric' coordinates," *International Journal for Numerical Methods in Engineering,* vol. 3, no. 4, pp. 519-528, 1971.

[29] Altair, "Altair Hypermesh Homepage," 2019. [Online]. Available: https://www.altair.com/hypermesh/. [Accessed 10 August 2019].

[30] T. R. Chandrupatla and A. D. Belegundu, Introduction to Finite Elements in Engineering, 3rd ed., Upper Saddle River: Prentice-Hall, 2002.

[31] O. C. Zienkiewics, R. L. Taylor and J. Z. Zhu, The Finite Element Method: Its Basis and Fundamentals, 7th ed., Oxford, UK: Elsevier, 2013.

[32] V. D. Liseikin, Grid Generation Methods, 2nd ed., Springer, 2010.

[33] P.-O. Persson and G. Strang, "A Simple Mesh Generator in Matlab," *SIAM Review,* vol. 46, pp. 329-345, 2004.

[34] R. Neely, "FAST CHAPTER 16 SURFERU," NASA Langley, 3 August 1999. [Online]. Available: https://www.nas.nasa.gov/Software/FAST/RND-93-010.walatka-clucas/htmldocs/chp_16.surferu.html. [Accessed 31 January 2020].

[35] Livermore Software Technology Corporation, "Mesh refinement & adaptive meshing tools," 2019. [Online]. Available: http://www.lstc.com/applications/icfd/features/adaptive_meshing. [Accessed 10 September 2019].

[36] R. Schneiders, "Robert Schneiders," 5 December 2018. [Online]. Available: http://www.robertschneiders.de/meshgeneration/software.html. [Accessed 2 November 2019].

[37] S. J. Owen, "A Survey of Unstructured Mesh Generation Technology," *7th International Meshing Roundtable,* vol. 3, 2000.

[38] B. Klingner, "Tetrahedral Mesh Improvement," Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, 20018.

[39] C. H. Bischof, W. T. Jones and A. Mauer, “Experiences with the Application of the ADIC Automatic Differentiation tool to the CSCMDO 3-D Volume Grid Generation Code,” in *34th AIAA Aerospace Sciences Meeting and Exhibit*, 1996.

[40] W. T. Jones and J. Samareh-Abolhassani, “A Grid Generation System for Multidisciplinary Design Optimization,” in *Proceedings of the Workshop on Surface Modeling, Grid Generation, and Related Issues in CFD Solutions, pages 11-21, NASA-CP3291*, 1995.

[41] N.-H. Kim, “Chapter 5: Finite Element Analysis for Contact Problems,” in *Introduction to Nonlinear Finite Element Analysis*, New York, Springer, 2015, pp. 367-422.

[42] C. A. Felippa, Introduction to Finite Element Methods; Chapter 9: MultiFreedom Constraints II;, 2004 ed., Boulder, Colorado: University of Colorado, 2004.

[43] C. A. Felippa, Introduction to Finite Element Methods; Chapter 8: MultiFreedom Constraints I, 2004 ed., Boulder, Colorado: University of Colorado, 2004.

[44] S. Kok and D. N. Wilke, “Understanding linear and non-linear multi-point constraints in finite element methods,” in *The South African Conference on Computational and Applied Mechanics*, Somerset West, 2014.

[45] S. Kok and D. N. Wilke, “Complete consistent tangent contributions for nonlinear finite element multi-point constraint strategies,” in *The South African Conference on Computational and Applied Mechanics*, 2016.

[46] M. Benzi, “Preconditioning Techniques for Large Linear Systems: A Survey,” *Journal of Computational Physics,* vol. 182, pp. 418-477, 2002.

[47] N. I. M. Gould and J. A. Scott, “A Numerical Evaluation of Sparse Direct Solvers for the Solution of Large Sparse Symmetric Linear Systems of Equations,” *ACM Transactions on Mathematical Software,* vol. 33, no. 2, p. Article 10, 2007.

[48] A. J. Wathen, “Preconditioning,” *Acta Numerica,* vol. 24, pp. 329-376, 2015.

[49] H. M. Markowitz, “The Elimination Form of the Inverse and Its Application to Linear Programming,” *Management Science,* vol. 3, no. 3, pp. 255-269, 1957.

[50] E. Cuthill and J. McKee, “Reducing the bandwidth of sparse symmetric matrices,” in *24th ACM National Conference*, 1969.

[51] S. H. Cheng, “Symmetric Indefinite Matrices: Linear System Solvers and Modified Inertia Problems,” 1998. [Online]. Available:

https://www.maths.manchester.ac.uk/~higham/links/theses/cheng98.pdf. [Accessed 04 December 2019].

[52] M. Benzi, G. H. Golub and J. Liesen, "Numerical solution of saddle point problems," *Acta Numerica,* p. 1–137, 2005.

[53] M. H. Gutknecht, "Block Krylov Space Methods for Linear Systems with Multiple Right-Hand Sides: An Introduction," in *Seminar for Applied Mathematics*, Zurich, 20006.

[54] MathWorks, "MATLAB R2018b Update 2 (9.5.0.1033004)".

[55] R. L. Burden and J. D. Faires, Numerical Analysis, 9th ed., Brooks/Cole, 2011.

[56] K. M. Miettine, Nonlinear Multiobjective Optimization, Illustrated ed., Springer Science & Business Media, 1999.

[57] K.-H. Chang, Design Theory and Methods using CAD/CAE: The Computer Aided Engineering Design Series, 1st ed., London: Academic Press, 2015.

[58] G. E. P. Box and K. B. Wilson, "On the Experimental Attainment of Optimum Conditions," *Journal of the Royal Statistical Society. Series B,* vol. 13, no. 1, pp. 1-45, 1951.

[59] G. K. Rose, "Computational Methods for Nonlinear Systems Analysis With Applications in Mathematics and Engineering," Old Dominion University, 2017.

[60] L. Piegl and W. Tiller, The NURBS Book, 2nd ed., Berlin: Spring-Verlag, 1997.

[61] D. N. Wilke and S. Kok, "Design optimization of multi-point constraints in structures," in *11th World Congress on Structural and Multidisciplinary Optimization*, Sydney, 2015.

APPENDIX A COMPARISON OF THE MSEM IMPLEMENTATIONS

In literature, there are two distinct approaches to the derivation of the MSEM. The first approach highlighted uses a transformation to generate the system of equations to be solved (see [43]). The second uses an energy method that identifies that no work may be done by the constraint (see [44]). For this discussion, linear FEM will be considered to facilitate comparison between the two MSEM approaches.

The standard form of the linear FEM is $\mathbf{Ku} = \mathbf{f}$. This can be partitioned into the free $\mathbf{u}_f$, master $\mathbf{u}_m$, slave $\mathbf{u}_s$ and prescribed $\mathbf{u}_p$ DOFs. The vector $\mathbf{f}$ represents external forces applied at the nodes and $\boldsymbol{R}$ represents boundary reaction forces at the master, slave, and prescribed DOFs:

$$\begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{fm} & \mathbf{K}_{fs} & \mathbf{K}_{fp} \\ \mathbf{K}_{mf} & \mathbf{K}_{mm} & \mathbf{K}_{ms} & \mathbf{K}_{mp} \\ \mathbf{K}_{sf} & \mathbf{K}_{sm} & \mathbf{K}_{ss} & \mathbf{K}_{sp} \\ \mathbf{K}_{pf} & \mathbf{K}_{pm} & \mathbf{K}_{ps} & \mathbf{K}_{pp} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_f \\ \mathbf{u}_m \\ \mathbf{u}_s \\ \mathbf{u}_p \end{Bmatrix} = \begin{Bmatrix} \mathbf{f}_f \\ \mathbf{f} + \boldsymbol{R}_m \\ \mathbf{f}_s + \boldsymbol{R}_s \\ \boldsymbol{R}_p \end{Bmatrix} \tag{126}$$

Since the prescribed DOFs are known, the system can be modified to require solution to only the unknown DOFs $\mathbf{u}_f$,$\mathbf{u}_m$ and $\mathbf{u}_s$ and the unknown reaction forces $\boldsymbol{R}_m$ and $\boldsymbol{R}_s$ on the RHS of the above equation. The reaction forces $\boldsymbol{R}_p$ are recovered from (126) after the reduced system is solved.

In the implementation of the transformation method, the reaction forces $\boldsymbol{R}_m$ and $\boldsymbol{R}_s$ are notably missing from the initial formulation. Though ignored in these implementations they will be carried through and highlighted grey. The system can be rearranged to show only the unprescribed DOFs as:

$$\begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{fm} & \mathbf{K}_{fs} \\ \mathbf{K}_{mf} & \mathbf{K}_{mm} & \mathbf{K}_{ms} \\ \mathbf{K}_{sf} & \mathbf{K}_{sm} & \mathbf{K}_{ss} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_f \\ \mathbf{u}_m \\ \mathbf{u}_s \end{Bmatrix} = \begin{Bmatrix} \mathbf{F}_f \\ \mathbf{F}_m + \boldsymbol{R}_m \\ \mathbf{F}_s + \boldsymbol{R}_s \end{Bmatrix} \tag{127}$$

where:

$$\begin{aligned} \mathbf{F}_f &= \mathbf{f}_s - \mathbf{K}_{fp}\mathbf{u}_p \\ \mathbf{F}_m &= \mathbf{f}_m - \mathbf{K}_{mp}\mathbf{u}_p \\ \mathbf{F}_s &= \mathbf{f}_s - \mathbf{K}_{sp}\mathbf{u}_p \end{aligned} \tag{128}$$

Symmetrizing or transformation method

For the transformation approach given in [43], all unprescribed DOFs $\mathbf{u}_{fms}$ (free, slave and master) are equated to the reduced DOF vector $\hat{\mathbf{u}}$ of the free and master DOFs by a transformation matrix $\boldsymbol{T}$ as follows:

$$\mathbf{u} = \boldsymbol{T}\hat{\mathbf{u}} + \hat{\boldsymbol{d}} \tag{129}$$

or in the element-based form:

$$\begin{Bmatrix}\{\mathbf{u}_f\}\\ \{\mathbf{u}_m\}\\ \{\mathbf{u}_s\}\end{Bmatrix} = \begin{bmatrix}[\boldsymbol{I}] & \mathbf{0}\\ \mathbf{0} & [\boldsymbol{I}]\\ \mathbf{0} & [\boldsymbol{P}]\end{bmatrix}\begin{Bmatrix}\{\mathbf{u}_f\}\\ \{\mathbf{u}_m\}\end{Bmatrix} + \begin{Bmatrix}\mathbf{0}\\ \mathbf{0}\\ \boldsymbol{d}\end{Bmatrix} \tag{130}$$

Given the standard form of the linear FEA problem $\mathbf{Ku} = \mathbf{f}$, the system then becomes:

$$\mathbf{K}_{fms}\{\boldsymbol{T}\hat{\mathbf{u}} + \hat{\boldsymbol{d}}\} = \mathbf{f} \tag{131}$$

The system is fully ranked at this point, but not symmetric. It was well understood in the development of FEM and the MSEM approach that the mathematical systems associated with material physics must be symmetric. Symmetry of the MSEM system was therefore enforced by pre-multiplying the equation by the transpose of the transformation matrix $\boldsymbol{T}^T$ yielding the modified system:

$$\hat{\mathbf{K}}\hat{\mathbf{u}} = \hat{\mathbf{f}} \tag{132}$$

where:

$$\begin{aligned}\hat{\mathbf{K}} &= \boldsymbol{T}^T\mathbf{K}\boldsymbol{T}\\ \hat{\mathbf{f}} &= \boldsymbol{T}^T\{\mathbf{f} - \mathbf{K}\boldsymbol{d}\}\end{aligned} \tag{133}$$

The transformation matrix $\boldsymbol{T}$ was essentially created in such a way that when the system is transformed, the displacements associated with the slave DOFs are removed. This implies that they are "prescribed" in the transformed domain. This symmetrizing step retains congruency of the system. After solving the modified system in (132), the solution to the original problem is recovered using (129).

Multiplying (132) out gives the system:

$$\hat{\mathbf{K}} = \boldsymbol{T}^T\mathbf{K}\boldsymbol{T} = \begin{bmatrix}\mathbf{K}_{ff} & \left(\mathbf{K}_{fm} + \mathbf{K}_{fs}\boldsymbol{P}\right)\\ \left(\mathbf{K}_{mf} + P^T\mathbf{K}_{sf}\right) & \begin{pmatrix}\mathbf{K}_{mm} + \mathbf{K}_{ms}\boldsymbol{P}\\ +\boldsymbol{P}^T\mathbf{K}_{sm} + \boldsymbol{P}^T\mathbf{K}_{ss}\boldsymbol{P}\end{pmatrix}\end{bmatrix} \tag{134}$$

and:

$$\begin{aligned}\hat{\mathbf{f}} &= \begin{bmatrix}[\boldsymbol{I}] & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & [\boldsymbol{I}] & \boldsymbol{P}^T\end{bmatrix}\begin{Bmatrix}\mathbf{F}_f - \mathbf{K}_{fs}\boldsymbol{d}\\ \mathbf{F}_m + \boldsymbol{R}_m - \mathbf{K}_{ms}\boldsymbol{d}\\ \mathbf{F}_s + \boldsymbol{R}_s - \mathbf{K}_{ss}\boldsymbol{d}\end{Bmatrix}\\ &= \begin{Bmatrix}\mathbf{F}_f - \mathbf{K}_{fs}\boldsymbol{d}\\ \mathbf{F}_m + \boldsymbol{P}^T\boldsymbol{F}_s - \mathbf{K}_{ms}\boldsymbol{d} - \boldsymbol{P}^T\mathbf{K}_{ss}\boldsymbol{d} + (\boldsymbol{R}_m + \boldsymbol{P}^T\boldsymbol{R}_s)\end{Bmatrix}\end{aligned} \tag{135}$$

It is clear that an accurate description of the system requires inclusion and resolution of the MPC reaction forces ($\boldsymbol{R}_m$ and $\boldsymbol{R}_s$) at the constraints. However, the transformation method implementations reviewed make it clear that the reaction forces $\boldsymbol{R}_m$ and $\boldsymbol{R}_s$ are not included as unknowns in the initial

system in (126). The RHS is given by external forces $\mathbf{f}$ only, thereby implying that the reaction forces are equal to zero ($\boldsymbol{R}_m = \mathbf{0}$ and $\boldsymbol{R}_s = \mathbf{0}$).

However, when the reaction forces are included and carried through (as is physically correct), the term $\boldsymbol{R}_m + \boldsymbol{P}^T\boldsymbol{R}_s$ is found in the RHS of the system. Fortuitously, the inadvertent selection of $\boldsymbol{R}_m = \mathbf{0}$ and $\boldsymbol{R}_s = \mathbf{0}$ makes the term zero, leaving the only unknowns as $\mathbf{u} = \begin{Bmatrix}\mathbf{u}_f \\ \mathbf{u}_m\end{Bmatrix}$.

The term $\boldsymbol{R}_m + \boldsymbol{P}^T\boldsymbol{R}_s$ can indeed be demonstrated to be zero with a non-trivial solution ($\boldsymbol{R}_m \neq \mathbf{0}$ and $\boldsymbol{R}_s \neq \mathbf{0}$), this, however, requires an understanding of the fact that the constraint boundary does no work on the system other than that done by the prescribed displacement portion $\boldsymbol{d}$. Such an understanding is absent in the transformation approach.

Energy method

Since the finite element methods are a form of energy methods, it has been identified (see [44] and [45]) that for the MSEM MPC boundary conditions to be correctly derived no work can be done by the constraint, except by that which is associated with the prescribed displacement. The system therefore contains the DOF unknowns $\mathbf{u}_f$, $\mathbf{u}_m$, $\mathbf{u}_s$ and the unknown reactions $\boldsymbol{R}_m$ and $\boldsymbol{R}_p$. The reaction force $\boldsymbol{R}_p$ is recovered at the solution of the system.

Substituting (25) into (128) and manipulating gives:

$$\begin{bmatrix}\mathbf{K}_{ff} & \left(\mathbf{K}_{fm} + \mathbf{K}_{fs}\boldsymbol{P}\right) \\ \mathbf{K}_{mf} & \left(\mathbf{K}_{mm} + \mathbf{K}_{ms}\boldsymbol{P}\right)\end{bmatrix}\begin{Bmatrix}\mathbf{u}_f \\ \mathbf{u}_m\end{Bmatrix} = \begin{Bmatrix}\mathbf{F}_f - \mathbf{K}_{fs}\boldsymbol{d} \\ \mathbf{F}_m + \boldsymbol{R}_m - \mathbf{K}_{ms}\boldsymbol{d}\end{Bmatrix} \tag{136}$$

$\boldsymbol{R}_m$ is however still an unknown present in the RHS. By identifying that the constraint boundary should produce no work except for that associated with the non-zero prescribed displacement, the following equation is given:

$$\mathbf{u}_m{}^T\boldsymbol{R}_m + \mathbf{u}_s{}^T\boldsymbol{R}_s = \boldsymbol{d}\boldsymbol{R}_s \tag{137}$$

Substituting the MPC equations in (25) for $\mathbf{u}_s{}^T$:

$$\mathbf{u}_m{}^T\boldsymbol{R}_m + (\boldsymbol{P}\mathbf{u}_m + \boldsymbol{d})^T\boldsymbol{R}_s = \boldsymbol{d}^T\boldsymbol{R}_s \tag{138}$$

The equation can be rearranged for the variable $\boldsymbol{u}_m$ as:

$$\mathbf{u}_m{}^T(\boldsymbol{R}_m + \boldsymbol{P}^T\boldsymbol{R}_s) = \mathbf{0} \tag{139}$$

For this to hold true for the non-trivial solution ($\mathbf{u}_m \neq \mathbf{0}$, $\boldsymbol{R}_m \neq \mathbf{0}$ and $\boldsymbol{R}_s \neq \mathbf{0}$):

$$\boldsymbol{R}_m = -\boldsymbol{P}^T \boldsymbol{R}_s \tag{140}$$

The reaction force $\boldsymbol{R}_s$ can be solved from the last equation in (127). It can be shown that substituting the solution to $\boldsymbol{R}_s$ into (140) and also into (136) followed by grouping the displacement terms $\mathbf{u}_m$ and $\mathbf{u}_f$ gives:

$$\begin{bmatrix} \mathbf{K}_{ff} & \left(\mathbf{K}_{fm} + \mathbf{K}_{fs}\boldsymbol{P}\right) \\ \left(\mathbf{K}_{mf} + \boldsymbol{P}^T\mathbf{K}_{sf}\right) & \left(\begin{matrix}\mathbf{K}_{mm} + \mathbf{K}_{ms}\boldsymbol{P} \\ +\boldsymbol{P}^T\mathbf{K}_{sm} + \boldsymbol{P}^T\mathbf{K}_{ss}\boldsymbol{P}\end{matrix}\right) \end{bmatrix} \begin{Bmatrix} \mathbf{u}_f \\ \mathbf{u}_m \end{Bmatrix} = \begin{Bmatrix} \mathbf{F}_f - \mathbf{K}_{fs}\boldsymbol{d} \\ \mathbf{F}_m + \boldsymbol{P}^T\mathbf{F}_s - \mathbf{K}_{ms}\boldsymbol{d} - \boldsymbol{P}^T\mathbf{K}_{ss}\boldsymbol{d} \end{Bmatrix} \tag{141}$$

This equation in (141) is seen to be the same resulting system of equations as in (132) - (135), with the notable exception that the reaction force term $\boldsymbol{R}_m + \boldsymbol{P}^T\boldsymbol{R}_s$ is not present in the final system in (141).

To summarise, in the transformation method, symmetry of the system was enforced through a pre-multiplication that resulted in symmetry. This was done since it was understood that the system "should" be symmetric. This however never properly handles the presence of the reaction forces at the MPC boundaries. In contrast, the energy method approach uses direct algebraic manipulation and the requirement that no work is done at the constraint other than that done by the prescribed displacement, to recover a solvable system of equations. The system of equations is inherently symmetric as a result of the manipulations as opposed to simply enforced. It is, however, important to note the comparative ease of implementing the transformation method compared to the energy method: there are far fewer elements to manage and less rearranging of the resulting system is required.

APPENDIX B DEVELOPMENT OF DERIVATIVES FOR AN ELLIPSE

B.1. Development of derivatives

An ellipse was used to describe the domain boundary for the development of the first and second-order derivatives. This was chosen since the ellipse is simple to define as well as smooth and continuous to the second order. A domain described by an ellipsoid also facilitates modification to enable further investigation into problems with high curvature or high aspect ratios. The equation for an ellipsoid in three dimensions is given as:

$$f_{\partial\Omega}(\boldsymbol{u}) = \left(\frac{x_{i,1}}{a}\right)^2 + \left(\frac{x_{i,1}}{b}\right)^2 + \left(\frac{x_{i,3}}{c}\right)^2 - 1 = 0 \tag{142}$$

B.2. First-order partial derivative (Lagrangian)

The first-order partial derivative is used in the Lagrangian approach:

$$\frac{df_{\partial\Omega}}{d\boldsymbol{x}} = \left\{2\frac{x_{i,1}}{a^2} \quad 2\frac{x_{i,2}}{b^2} \quad 2\frac{x_{i,3}}{b^2}\right\}^T \tag{143}$$

B.3. Second-order partial derivatives (Lagrangian)

The second-order partial derivative is used in the Lagrangian approach:

$$\frac{d^2f_{\Omega}}{d\boldsymbol{x}^2} = \begin{bmatrix} {}^2/_{a^2} & 0 & 0 \\ 0 & {}^2/_{b^2} & 0 \\ 0 & 0 & {}^2/_{c^2} \end{bmatrix} \tag{144}$$

B.4. First-order derivative (MSEM)

In the equation below, the master DOFs are represented by$x_{i,m1}$ and $x_{i,m2}$ with corresponding constants given by c_{m1} and c_{m2} respectively. The slave DOF is represented by $x_{i,s}$ and constant c_s:

$$\frac{d\boldsymbol{x}_s}{d\boldsymbol{x}_m} = \left\{-\frac{c_s^2\ x_{i,m1}}{c_{m1}^2\ x_{i,s}} \quad -\frac{c_s^2 x_{i,m2}}{c_{m2}^2\ x_{i,s}}\right\}^T \tag{145}$$

B.5. Second-order derivative (MSEM)

In the equation below, the master DOFs are represented by $x_{i,m1}$ and $x_{i,m2}$ with corresponding constants given by c_{m1} and c_{m2} respectively. The slave DOF is represented by $x_{i,s}$ and constant c_s:

$$\frac{d^2\boldsymbol{x}_s}{d\boldsymbol{x}_{\boldsymbol{m}}^2} = \begin{bmatrix} -\dfrac{\left(c_s^4\left(c_{m2}^2 - x_{i,m2}^2\right)\right)}{c_{m1}^2 c_{m2}^2 x_{i,s}^3} & -\dfrac{c_s^4 x_{i,m2} x_{i,m1}}{c_{m2}^2 c_{m1}^2 x_{i,s}^3} \\ -\dfrac{c_s^4 x_{i,m1} x_{i,m2}}{c_{m1}^2 c_{m2}^2 x_{i,s}^3} & -\dfrac{\left(c_s^4\left(c_{m1}^2 - x_{i,m1}^2\right)\right)}{c_{m1}^2 c_{m2}^2 x_{i,s}^3} \end{bmatrix} \quad (146)$$

APPENDIX C TRUSS EFFECTIVENESS

C.1. Truss effectiveness and resilience of the formulations to scaling L_c/L_{des} in 2D

An investigation was conducted to determine the effectiveness of the various truss formulations detailed in §3.2.2. and the Newton's method implementation. The following initial conditions were created:

A circular distance function with unit radius was used to determine the distance of the point under consideration to the boundary:

$$f_{d\Omega}(\boldsymbol{p}_i) = \|\boldsymbol{p}_i\|_2 - 1 \tag{147}$$

For 2D the coordinates of the point are given by:

$$\boldsymbol{p_i} = \{\mathcal{X}_{i\times 2\,-1}, \mathcal{X}_{i\times 2}\} \tag{148}$$

For 3D the coordinates of the point are given by:

$$\boldsymbol{p_i} = \{\mathcal{X}_{i\times 3-2}, \mathcal{X}_{i\times 3-1,}\, \mathcal{X}_{i\times 3}\} \tag{149}$$

The initial mesh was seeded using equilateral triangles with the initial uniform edge length size L_0:

$$L_0 = 0.05 \tag{150}$$

Here, the initial seeding length L_0 should not be confused with the desired truss length L_{des}.

As shown in Figure C-1, all nodes located outside of a tolerance of $-0.7L_0$ interior to the boundary of the circle were removed and the boundary was seeded using fixed nodes which forced the triangles to remain inside of the boundary. Nodes were placed along the circumference with a spacing equal to L_{des} as shown by the * in Figure C-1.

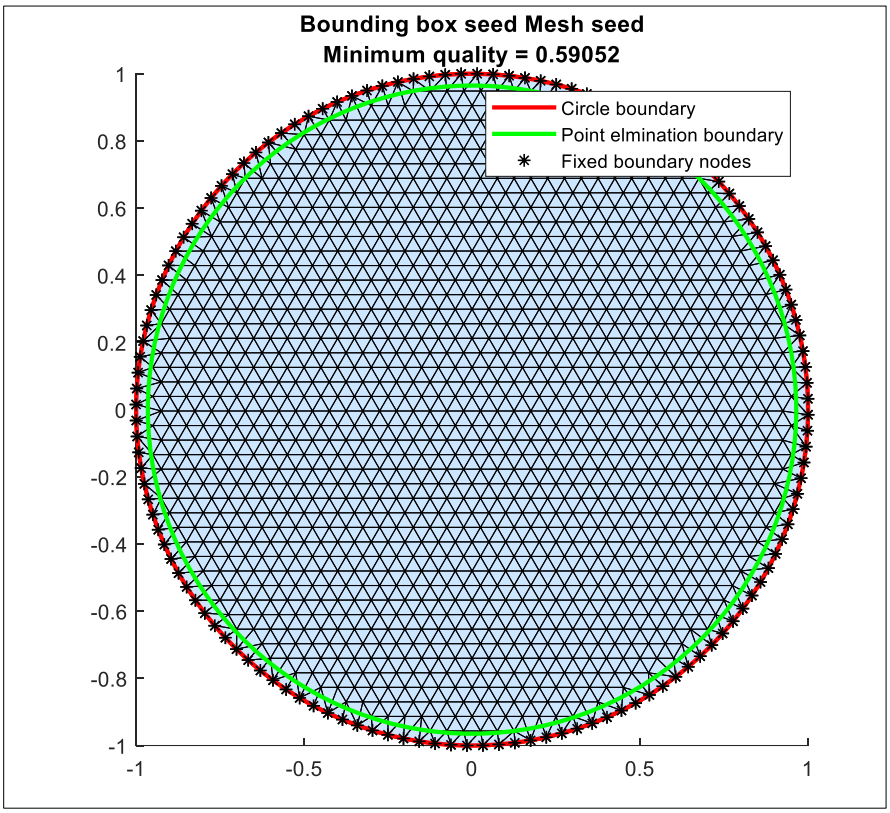


Figure C-1 Initial seed mesh for truss convergence study in 2D

Having defined the initial seed mesh shown in Figure C-1, equilibrium for each of the considered truss formulations is solved. Several scaling factors for the deviation from desired truss length were analysed to allow for a good comparison between the formulations. The desired length L_{des} was scaled to obtain a certain compression ratio $R_{comp} = \frac{L_c}{L_{des}}$ as shown in Figure 10 of §3.2.2.5:

$$L_{des} = \frac{L_0}{R_{comp}} \tag{151}$$

Remeshing was permitted for any iteration where a node was found to have moved a distance of more than 10% of the local desired truss length, L_{des}, from its previous position. The norm of the error squared was used as the solution criterion for this study with the convergence tolerance given as:

$$\|\boldsymbol{u}\|_2 < 1 \times 10^{-8} \tag{152}$$

C.2. Truss effectiveness and resilience of the formulations to scaling L_c/L_{des} in 3D

This investigation was performed retrospectively once a high-quality surface mesh was obtained using aspects of the MPC implementation as depicted in Figure C-2. The surface nodes were then fixed on the boundary. An initial mesh was then created that was able to converge for both methods without

the need for retriangulation. The mesh was verified by confirming that quadratic convergence was obtained for all truss formulations when $\frac{L_c}{L_{des}} = 1$.

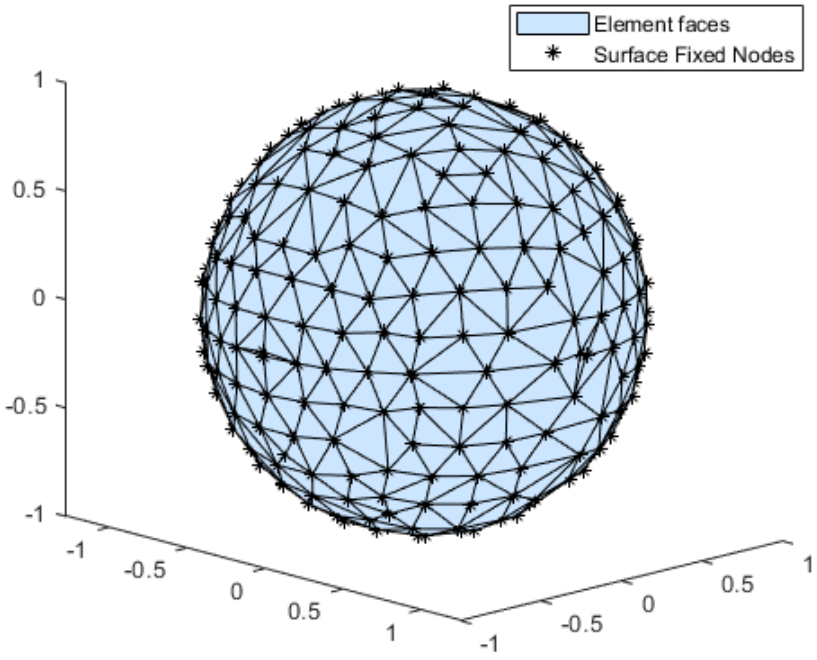


Figure C-2 Initial seed mesh for truss convergence study in 3D

Results are provided for scaling factors $\frac{L_c}{L_{des}}$ of 0.8, 1, 1.2, and 2 as a means of highlighting the similarities in convergence response for the 2D and 3D truss formulations.

C.3. Newton implementation results for 2D and various truss formulations

Investigations were performed for $\frac{L_c}{L_{des}} = 0.7\ , 0.8\ , 1.0\ , 1.2$ and 2. Table C-1 details results for $\frac{L_c}{L_{des}} = 1$, see APPENDIX D for results corresponding to the other ratios. Figure C-3 provides a visual representation of the quadratic convergence achieved for each ratio.

Table C-1 2D truss formulation convergence comparison results for L_c/L_{des} = 1

$F_{truss} =$	$\frac{\Delta L}{L_{des}}$	$\frac{\Delta L}{L_c}$	$\frac{\Delta L}{L_{des}} e^{S_1\left[\frac{\Delta L}{L_{des}}\right]^2}$			$\frac{\Delta L}{L_c} e^{S_1\left[\frac{\Delta L}{L_c}\right]^2}$	
S_1	-	-	0.5	1	2	-0.5	1
Iterations	5	8	5	5	6	5	8
Avg final $\frac{L_c}{L_{des}}$	1.013	1.013	1.013	1.013	1.013	1.013	1.013
Min-quality	0.671	0.675	0.675	0.679	0.687	0.667	0.685
Remeshes	2	5	3	3	3	3	5
Norm updates Iteration No.	0	0	0	0	0	0	0
1	3.90E-01	4.47E-01	3.41E-01	3.03E-01	2.48E-01	4.55E-01	4.12E-01
2	3.00E-02	7.18E-02	7.54E-02	1.08E-01	1.42E-01	4.65E-02	4.45E-02
3	4.22E-04	4.24E-02	2.76E-03	8.97E-03	3.21E-02	1.30E-03	4.20E-02
4	2.58E-07	2.30E-02	5.80E-06	6.93E-05	1.57E-03	1.20E-06	2.34E-02
5	3.38E-13	3.85E-03	8.66E-11	9.22E-09	5.37E-06	1.96E-12	3.84E-03
6	-	8.15E-05	-	-	1.07E-10	-	1.04E-04
7	-	6.64E-08	-	-	-	-	1.17E-07
8	-	8.73E-14	-	-	-	-	3.93E-13

Figure C-3 shows the effect of the compression ratio $\frac{L_c}{L_{des}} = 1, 0.7, 0.8, 1.2$ and 2 on system convergence. From these graphs, it is clear that the formulation $F_{truss} = \frac{\Delta L}{L_{des}}$ has the best overall performance with either the fastest or near fastest convergence rates. It is also clear that the compressed state $\frac{L_c}{L_{des}} < 1$ has greater difficulty in converging when compared to tension systems with $\frac{L_c}{L_{des}} > 1$. It is noted that the high tension system $\frac{L_c}{L_{des}} = 2$ improved the convergence rate for the formulation $F_{truss} = \frac{\Delta L}{L_{des}}$ by one iteration.

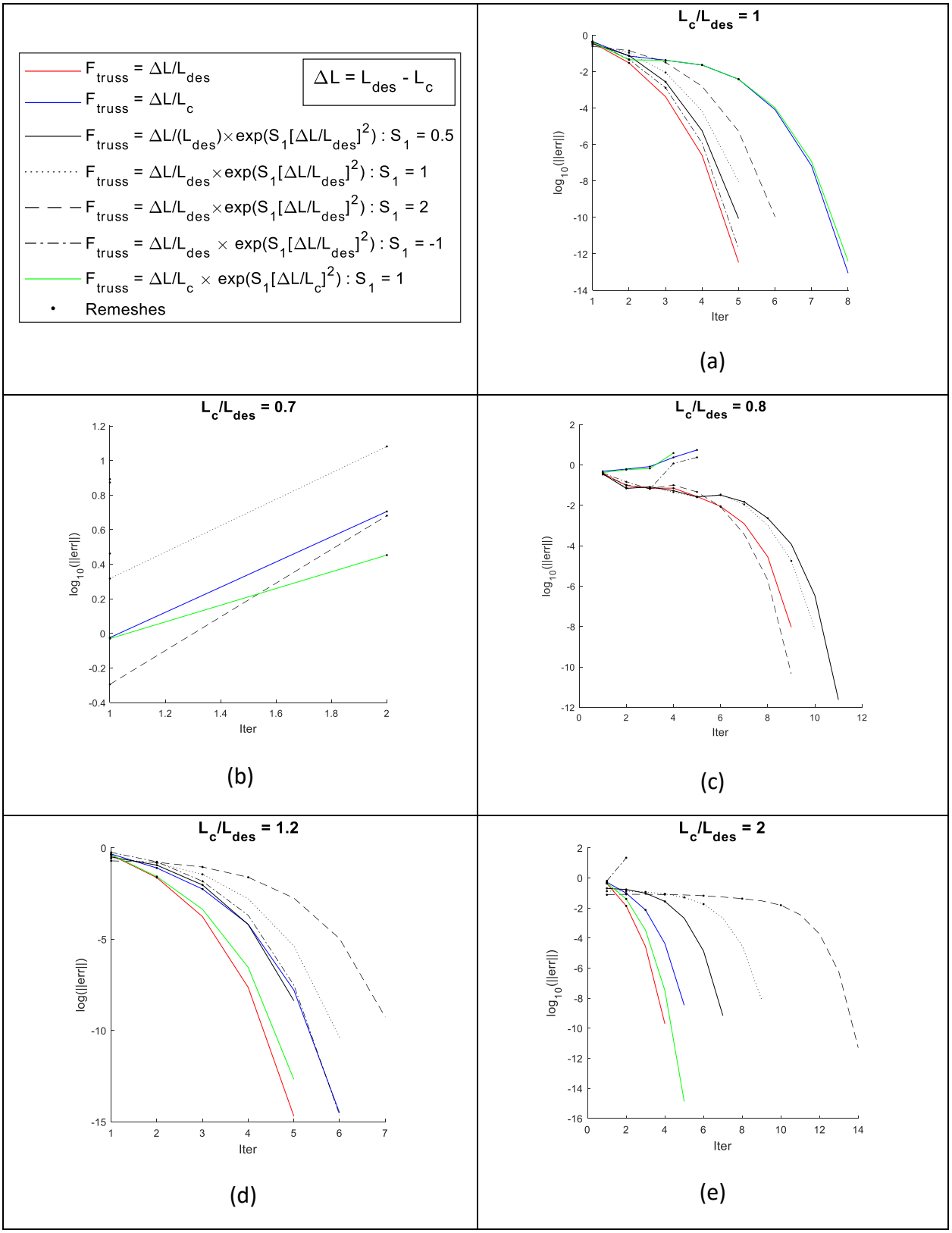


Figure C-3 Truss formulation convergence in 2D for various scaling factors L_c/L_{des}

C.4. Newton implementation results for 3D

Figure C-4 shows the 3D convergence response for $\frac{L_c}{L_{des}} = 1, 0.8$ and 2 in (a) (b) and (c) respectively. It is clear that for $\frac{L_c}{L_{des}} = 1$ all truss results converge quadratically with roughly the same effectiveness. For the compressed state, $\frac{L_c}{L_{des}} = 0.8$ and for the tension state the formulation $F_{truss} = \frac{\Delta L}{L_{des}}$ were seen to result in the best performance. The formulation $F_{truss} = \frac{\Delta L}{L_c}$ was shown to be susceptible to negative effects associated with compression.

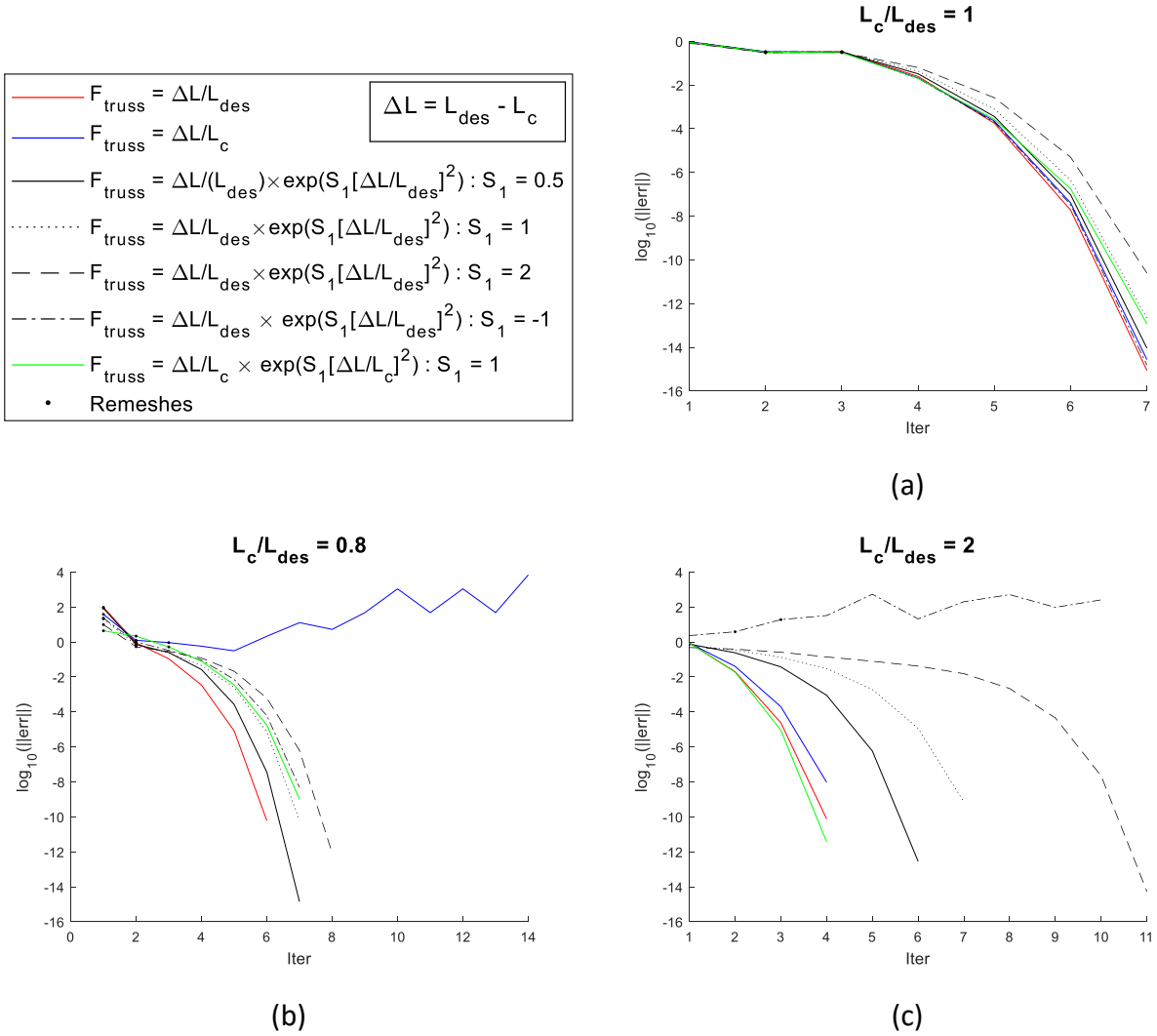


Figure C-4 Truss formulation convergence in 3D for various scaling factors L_c/L_{des}

C.5. Evaluation of truss method implementation

§C.3 and §C.4 show that quadratic convergence was achieved by all truss formulations considered as indicated by the reduction in error by a factor of at least 10^{-2} in subsequent steps. Newton's method and the truss formulations were therefore verified as correctly implemented. The number of iterations

required to obtain a solution are presented in Table C-2 for easy comparison. The minimum triangle (element) quality is given in Table C-3.

Table C-2 Truss formulation in 2D: comparison of number of required iterations

$F_{truss} =$	Scale L_c/L_{des}	$\frac{\Delta L}{L_{des}}$	$\frac{\Delta L}{L_c}$	$\frac{\Delta L}{L_{des}} e^{S_1 \left[\frac{\Delta L}{L_{des}}\right]^2}$			$\frac{\Delta L}{L_c} e^{S_1 \left[\frac{\Delta L}{L_c}\right]^2}$	
S_1	-	-	-	0.5	1	2	-0.5	1
Required Iterations (convergence failed)	0.7	(2)	(3)	(2)	(4)	(3)	(2)	(3)
	0.8	9	(6)	11	10	9	(6)	(5)
	0.9	5	7	6	6	6	5	9
	1	5	8	5	5	6	5	8
	1.1	5	6	5	6	7	5	5
	1.2	5	6	5	6	7	6	5
	1.5	4	5	6	7	9	(2)	5
	2	4	5	7	9	14	(3)	5

Table C-3 Truss formulation in 2D: comparison of minimum triangle quality

$F_{truss} =$	Scale L_c/L_{des}	$\frac{\Delta L}{L_{des}}$	$\frac{\Delta L}{L_c}$	$\frac{\Delta L}{L_{des}} e^{S_1 \left[\frac{\Delta L}{L_{des}}\right]^2}$			$\frac{\Delta L}{L_c} e^{S_1 \left[\frac{\Delta L}{L_c}\right]^2}$	
S_1	-	-	-	0.5	1	2	-0.5	1
Avg Quality (convergence failed)	0.7	(NaN)	(2.00E-08)	(8.04E-09)	(1.56E-06)	(5.51E-07)	(1.02E-06)	(4.63E-09)
	0.8	0.736	(1.98E-08)	0.731	0.730	0.720	(2.60E-06)	(5.69E-06)
	0.9	0.682	0.702	0.681	0.681	0.680	0.682	0.697
	1	0.671	0.675	0.675	0.679	0.687	0.667	0.685
	1.1	0.662	0.673	0.668	0.674	0.685	0.654	0.652
	1.2	0.654	0.658	0.663	0.669	0.679	0.663	0.649
	1.5	0.641	0.621	0.653	0.656	0.654	(1.85E-07)	0.635
	2	0.625	0.607	0.645	0.641	0.626	(1.59E-06)	0.607

The relative convergence rates for all truss formulations considered for 3D are shown in Figure C-4 (a) through (c). The number of iterations required to obtain a solution are presented in Table C-4 for easy comparison. The minimum tetrahedral (element) quality is given in Table C-5.

Table C-4 Truss formulation in 3D: comparison of number of required iterations

$F_{truss} =$	Scale L_c/L_{des}	$\frac{\Delta L}{L_{des}}$	$\frac{\Delta L}{L_c}$	$\frac{\Delta L}{L_{des}} e^{S_1\left[\frac{\Delta L}{L_{des}}\right]^2}$			$\frac{\Delta L}{L_c} e^{S_1\left[\frac{\Delta L}{L_c}\right]^2}$	
S_1	-	-	-	0.5	1	2	-0.5	1
Required Iterations (convergence failed)	0.8	12	20	13	13	14	(12)	20
	1	12	12	12	12	12	12	12
	1.2	5	5	5	5	8	5	4
	2	4	4	6	7	11	(25)	4

Table C-5 Truss formulation in 3D: comparison of average tetrahedral quality

$F_{truss} =$	Scale L_c/L_{des}	$\frac{\Delta L}{L_{des}}$	$\frac{\Delta L}{L_c}$	$\frac{\Delta L}{L_{des}} e^{S_1\left[\frac{\Delta L}{L_{des}}\right]^2}$			$\frac{\Delta L}{L_c} e^{S_1\left[\frac{\Delta L}{L_c}\right]^2}$	
S_1	-	-	-	0.5	1	2	-0.5	1
Avg Quality (convergence failed)	0.8	0.567	0.593	0.500	0.567	0.601	(0.442)	0.686
	1	0.871	0.873	0.872	0.872	0.871	0.871	0.873
	1.2	0.875	0.875	0.875	0.875	0.874	0.875	0.875
	2	0.877	0.877	0.875	0.872	0.866	NaN	0.877

Table C-2 through Table C-5 show that, for a convergence tolerance of 10^{-8}, the truss formulation $F_{truss} = \frac{\Delta L}{L_{des}}$ converges either on par with, or better than, any of the other formulations. This is not unexpected since the formulation $F_{truss} = \frac{\Delta L}{L_{des}}$ is derived directly from Hooke's law without modification and represents a linear spring.

This has significant implications: since each iteration in the Newton's method calculation has the same computational time cost for a given number of degrees of freedom, the fewer the number of iterations required to achieve convergence the less computational time is required. It is therefore concluded that the $F_{truss} = \frac{\Delta L}{L_{des}}$ formulation represents the most efficient option due to its best performance overall.

On average, the results for the quality of triangles (2D) or tetrahedra (3D) were relatively consistent across all methods. For the 2D mesher, results corresponding to elements of the lowest quality were shown. Low element quality was due to circumstances where snap-through had occurred and points departed from the domain. For the 3D mesher and the scaling factor $\frac{L_c}{L_{des}} = 0.8$, all average element qualities were similar and considerably lower than for other values of $\frac{L_c}{L_{des}}$. This is due to snap through in some cases and the development of sliver elements in other cases.

Trusses in a compressed state given by $\frac{L_c}{L_{des}} < 1$ showed high levels of instability. This was evidenced by the increased number of iterations required to obtain a solution. However, where convergence was

obtained, as in the 2D mesher for $\frac{L_c}{L_{des}} = 0.8$, the best element quality results were obtained. In contrast, high levels of tension only resulted in convergence problems for $\frac{L_c}{L_{des}} \geq 1.5$ and then only on the truss formulations that demonstrated snap-through as a characteristic of the force reaction curve. This shows that the truss system is more resilient under conditions of high tension than it is to compression.

When the scaling factor $\frac{L_c}{L_{des}} = 1$ was used as the scaling factor, the average triangle or tetrahedral quality was very similar for both 2D and 3D across all formulations. This is because all formulations have similar responses if only a small deviation from $\frac{L_c}{L_{des}} = 1$ occurs.

It is evident that the force response formulation derived directly from Hooke's law produces the best results in terms of efficiency and element quality. The poorest results were attributed to the exponential equations, particularly and not unsurprisingly, when a tendency towards snap-through was evident in the force reaction curve.

APPENDIX D 2D TRUSS STIFFNESS TABLES

Table D-6 2D truss formulation convergence comparison results for L_c/L_{des} = 0.7

$F_{truss} =$	$\frac{\Delta L}{L_{des}}$	$\frac{\Delta L}{L_c}$	$\frac{\Delta L}{L_{des}} e^{S_1 \left[\frac{\Delta L}{L_{des}}\right]^2}$			$\frac{\Delta L}{L_c} e^{S_1 \left[\frac{\Delta L}{L_c}\right]^2}$	
S_1	-	-	0.5	1	2	-0.5	1
Iterations	2	3	2	4	3	2	3
Avg final $\frac{L_c}{L_{des}}$	NaN	1.548	1.125	0.777	0.789	0.822	NaN
Min-quality	NaN	2.00E-08	8.04E-09	1.56E-06	5.51E-07	1.02E-06	4.63E-09
Remeshes	2	3	2	4	3	2	3
Norm updates Iteration No.							
1	7.46E+00	9.47E-01	2.90E+00	2.08E+00	5.08E-01	7.80E+00	9.34E-01
2	NaN	5.07E+00	NaN	1.21E+01	4.80E+00	NaN	2.84E+00
3	-	NaN	-	4.17E+01	NaN	-	NaN
4	-	-	-	NaN	-	-	-

Table D-7 2D truss formulation convergence comparison results for L_c/L_{des} = 0.8

$F_{truss} =$	$\frac{\Delta L}{L_{des}}$	$\frac{\Delta L}{L_c}$	$\frac{\Delta L}{L_{des}} e^{S_1 \left[\frac{\Delta L}{L_{des}}\right]^2}$			$\frac{\Delta L}{L_c} e^{S_1 \left[\frac{\Delta L}{L_c}\right]^2}$	
S_1	-	-	0.5	1	2	-0.5	1
Iterations	9	6	11	10	9	6	5
Avg final $\frac{L_c}{L_{des}}$	0.811	1.212	0.811	0.811	0.811	1.022	0.985417638
Min-quality	0.736	1.98E-08	0.731	0.730	0.720	2.60E-06	5.69E-06
Remeshes	6	6	8	8	6	6	5
Norm updates Iteration No.							
1	3.83E-01	4.86E-01	3.68E-01	3.55E-01	3.36E-01	4.02E-01	4.21E-01
2	9.67E-02	6.23E-01	6.89E-02	7.67E-02	1.04E-01	1.47E-01	5.84E-01
3	7.04E-02	8.35E-01	8.26E-02	7.79E-02	6.88E-02	6.50E-02	6.72E-01
4	7.12E-02	2.37E+00	5.33E-02	4.51E-02	1.00E-01	1.19E+00	3.88E+00
5	2.73E-02	7.88E+00	2.65E-02	2.50E-02	4.63E-02	2.38E+00	NaN
6	8.93E-03	NaN	3.23E-02	3.44E-02	8.54E-03	NaN	-
7	1.24E-03	-	1.50E-02	1.12E-02	3.72E-04	-	-
8	2.85E-05	-	2.31E-03	1.03E-03	2.01E-06	-	-
9	9.27E-09	-	1.20E-04	1.79E-05	4.56E-11	-	-
10	-	-	3.45E-07	7.46E-09	-	-	-
11	-	-	2.38E-12	-	-	-	-

Table D-8 2D truss formulation convergence comparison results for L_c/L_{des} = 1.2

$F_{truss} =$	$\frac{\Delta L}{L_{des}}$	$\frac{\Delta L}{L_c}$	$\frac{\Delta L}{L_{des}} e^{S_1 \left[\frac{\Delta L}{L_{des}}\right]^2}$			$\frac{\Delta L}{L_c} e^{S_1 \left[\frac{\Delta L}{L_c}\right]^2}$	
S_1	-	-	0.5	1	2	-0.5	1
Iterations	5	6	5	6	7	6	5
Avg final $\frac{L_c}{L_{des}}$	1.216	1.215	1.216	1.216	1.216	1.215	1.216
Min-quality	0.654	0.658	0.663	0.669	0.679	0.663	0.649
Remeshes	2	3	3	3	4	3	2
Norm updates Iteration No.							
1	4.01E-01	4.58E-01	3.14E-01	2.60E-01	1.93E-01	5.63E-01	4.23E-01
2	2.34E-02	7.84E-02	1.12E-01	1.51E-01	1.61E-01	1.72E-01	2.65E-02
3	1.70E-04	5.51E-03	9.27E-03	3.53E-02	8.89E-02	1.48E-02	4.30E-04
4	2.27E-08	6.50E-05	6.31E-05	1.65E-03	2.44E-02	2.03E-04	2.83E-07
5	2.00E-15	1.61E-08	4.24E-09	4.24E-06	1.77E-03	3.03E-08	2.08E-13
6	-	3.02E-15	-	4.16E-11	1.06E-05	2.31E-15	-
7	-	-	-	-	5.38E-10	-	-

Table D-9 Truss formulation convergence comparison results for L_c/L_{des} = 2

$F_{truss} =$	$\frac{\Delta L}{L_{des}}$	$\frac{\Delta L}{L_c}$	$\frac{\Delta L}{L_{des}} e^{S_1 \left[\frac{\Delta L}{L_{des}}\right]^2}$			$\frac{\Delta L}{L_c} e^{S_1 \left[\frac{\Delta L}{L_c}\right]^2}$	
S_1	-	-	0.5	1	2	-0.5	1
Iterations	4	5	7	9	14	3	5
Avg final $\frac{L_c}{L_{des}}$	2.025	2.025	2.026	2.026	2.026	2.547	2.025
Min-quality	0.625	0.607	0.645	0.641	0.626	1.59E-06	0.607
Remeshes	2	3	4	6	6	3.00E+00	2
Norm updates Iteration No.							
1	4.24E-01	5.18E-01	2.01E-01	1.31E-01	7.55E-02	6.33E-01	4.68E-01
2	1.36E-02	9.65E-02	1.70E-01	1.32E-01	8.11E-02	2.19E+01	4.00E-02
3	2.68E-05	7.31E-03	9.63E-02	1.16E-01	8.06E-02	NaN	3.32E-04
4	1.96E-10	4.28E-05	2.73E-02	8.71E-02	7.83E-02	-	3.24E-08
5	-	3.36E-09	2.09E-03	5.04E-02	7.37E-02	-	1.31E-15
6	-	-	1.35E-05	1.80E-02	6.54E-02	-	-
7	-	-	6.86E-10	2.14E-03	5.34E-02	-	-
8	-	-	-	3.09E-05	4.18E-02	-	-
9	-	-	-	8.07E-09	2.98E-02	-	-
10	-	-	-	-	1.52E-02	-	-
11	-	-	-	-	3.54E-03	-	-
12	-	-	-	-	1.83E-04	-	-
13	-	-	-	-	5.31E-07	-	-
14	-	-	-	-	4.97E-12	-	-

APPENDIX E 3D TRUSS STIFFNESS TABLES

Table E-10 3D truss formulation convergence comparison results for L_c/L_{des} = 0.8

$F_{truss} =$	$\frac{\Delta L}{L_{des}}$	$\frac{\Delta L}{L_c}$	$\frac{\Delta L}{L_{des}} e^{S_1 \left[\frac{\Delta L}{L_{des}}\right]^2}$			$\frac{\Delta L}{L_c} e^{S_1 \left[\frac{\Delta L}{L_c}\right]^2}$	
S_1	-	-	0.5	1	2	-0.5	1
Iterations	12	20	13	13	14	14	15
Avg final $\frac{L_c}{L_{des}}$	0.963	0.911	0.953	0.944	0.954	0.941	0.950
Min-quality	0.567	0.593	0.500	0.567	0.601	0.442	0.686
Remeshes	2	3	2.00E+00	2	2.00E+00	3.00E+00	3
Norm updates Iteration No.	0	0	0	0	0	0	0
1	9.72E+01	3.87E+01	8.61E+01	2.11E+01	9.92E+00	2.33E+01	4.40E+00
2	8.92E-01	1.24E+00	7.22E-01	6.83E-01	5.33E-01	3.55E+00	2.18E+00
3	1.09E-01	8.83E-01	2.46E-01	2.42E-01	2.83E-01	1.78E+00	5.24E-01
4	3.47E-03	6.27E-01	2.48E-02	4.63E-02	1.17E-01	1.52E+01	8.05E-02
5	8.30E-06	2.34E-01	2.27E-04	2.56E-03	2.13E-02	2.66E+01	3.54E-03
6	6.06E-11	1.26E-01	2.95E-08	7.22E-06	6.54E-04	1.71E+37	1.77E-05
7	-	1.56E-02	1.38E-15	6.26E-11	7.08E-07	-	1.03E-09
8	-	3.53E-04	-	-	1.04E-12	-	-
9	-	3.82E-07	-	-	-	-	-
10	-	5.09E-13	-	-	-	-	-

Table E-11 3D truss formulation convergence comparison results for L_c/L_{des} = 1

$F_{truss} =$	$\frac{\Delta L}{L_{des}}$	$\frac{\Delta L}{L_c}$	$\frac{\Delta L}{L_{des}} e^{S_1 \left[\frac{\Delta L}{L_{des}}\right]^2}$			$\frac{\Delta L}{L_c} e^{S_1 \left[\frac{\Delta L}{L_c}\right]^2}$	
S_1	-	-	0.5	1	2	-0.5	1
Iterations	12	12	12	12	12	12	12
Avg final $\frac{L_c}{L_{des}}$	1.001	1.001	1.001	1.001	1.001	1.001	1.001
Ave-Quality	0.871	0.873	0.872	0.872	0.871	0.871	0.873
Remeshes	2	2	2	2	2	2	2
Norm updates Iteration No.	0	0	0	0	0	0	0
1	9.44E-01	9.53E-01	9.11E-01	8.81E-01	8.28E-01	9.81E-01	9.16E-01
2	3.17E-01	3.36E-01	3.12E-01	3.04E-01	3.01E-01	3.36E-01	3.25E-01
3	3.37E-01	3.19E-01	3.22E-01	3.22E-01	3.09E-01	3.30E-01	3.05E-01
4	2.47E-02	2.11E-02	3.27E-02	4.31E-02	6.33E-02	2.02E-02	2.03E-02
5	1.78E-04	2.26E-04	3.64E-04	7.91E-04	2.63E-03	2.01E-04	2.70E-04
6	1.88E-08	4.02E-08	9.46E-08	4.46E-07	4.98E-06	3.31E-08	1.77E-07
7	8.74E-16	2.74E-15	9.06E-15	2.09E-13	2.48E-11	1.53E-15	1.18E-13

Table E-12 3D truss formulation convergence comparison results for L_c/L_{des} = 1.2

$F_{truss} =$	$\frac{\Delta L}{L_{des}}$	$\frac{\Delta L}{L_c}$	$\frac{\Delta L}{L_{des}} e^{S_1 \left[\frac{\Delta L}{L_{des}}\right]^2}$			$\frac{\Delta L}{L_c} e^{S_1 \left[\frac{\Delta L}{L_c}\right]^2}$	
S_1	-	-	0.5	1	2	-0.5	1
Iterations	-	-	-	-	8	-	4
Avg final $\frac{L_c}{L_{des}}$	1.205	1.205	1.205	1.205	1.197	1.205	1.205
Ave-Quality	0.875	0.875	0.875	0.875	0.874	0.875	0.875
Remeshes	0	0	0	0	1	0	0
Norm updates Iteration No.	0	0	0	0	0	0	0
1	8.98E-01	9.03E-01	8.53E-01	8.14E-01	7.48E-01	9.53E-01	8.91E-01
2	4.71E-02	5.45E-02	1.01E-01	1.57E-01	2.41E-01	7.41E-02	3.64E-02
3	2.45E-04	3.55E-04	2.28E-03	8.30E-03	1.45E-01	6.19E-04	1.79E-04
4	1.48E-08	3.08E-08	1.21E-06	2.30E-05	1.59E-02	1.01E-07	8.30E-09
5	7.18E-16	9.93E-16	4.93E-13	2.24E-10	2.50E-04	4.84E-15	-
6	-	-	-	-	1.14E-07	-	-
7	-	-	-	-	3.94E-14	-	-

Table E-13 3D truss formulation convergence comparison results for L_c/L_{des} = 2

$F_{truss} =$	$\frac{\Delta L}{L_{des}}$	$\frac{\Delta L}{L_c}$	$\frac{\Delta L}{L_{des}} e^{S_1 \left[\frac{\Delta L}{L_{des}}\right]^2}$				$\frac{\Delta L}{L_c} e^{S_1 \left[\frac{\Delta L}{L_c}\right]^2}$
S_1	-	-	0.5	1	2	-0.5	1
Iterations	4	4	6	7	11	25	4
Avg final $\frac{L_c}{L_{des}}$	2.009	2.009	1.994	1.995	1.998	1.266	2.009
Ave-Quality	0.877	0.877	0.875	0.872	0.866	NaN	0.877
Remeshes	0	0	0	0	0	2.00E+00	0
Norm updates Iteration No.	0	0	0	0	0	0	0
1	8.77E-01	8.71E-01	7.56E-01	6.49E-01	4.86E-01	2.33E+00	8.79E-01
2	2.00E-02	4.06E-02	2.42E-01	3.48E-01	3.92E-01	3.87E+00	1.91E-02
3	2.42E-05	2.00E-04	3.79E-02	1.34E-01	2.60E-01	NaN	9.05E-06
4	7.65E-11	9.54E-09	9.05E-04	3.06E-02	1.40E-01	-	3.75E-12
5	-	-	5.82E-07	1.94E-03	7.97E-02	-	-
6	-	-	2.85E-13	1.15E-05	4.20E-02	-	-
7	-	-	-	6.95E-10	1.55E-02	-	-
8	-	-	-	-	2.20E-03	-	-
9	-	-	-	-	4.53E-05	-	-
10	-	-	-	-	2.18E-08	-	-
11	-	-	-	-	5.37E-15	-	-

APPENDIX F STUDY OF TRUSS STEP SIZE LIMIT METHODS

When an update step becomes too large, the possibility exists that the accompanying mesh distortion can push nodes outside of the boundary resulting in severe mesh distortion rather than improving mesh quality. This effect was seen after the third iteration for a truss formulation with the scaling factor L_c/L_{des} = 0.77 implemented without an update step size limit in Figure F-5.

To assuage the effects of L_c/L_{des} scaling factors associated with high compression and tension (discussed in APPENDIX C), update step size limits are required to reduce the negative effects of large updates. The effects of poor scaling factors include the impaired performance of the mesh generator, snap-through, and the resultant severe mesh distortion shown in Figure F-5.

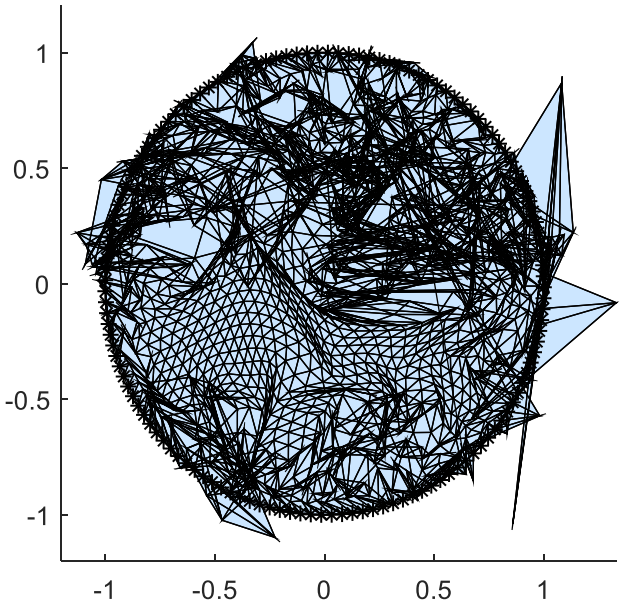


Figure F-5 Mesh distortion due to large update with L_c/L_{des} = 0.77

F.1. Update step size limit methods

F.1.1. Newton step α update fixed size limiter

During the implementation of Newton's method, conditions were found where the Newton update step size, Δu, could be too large depending on the L_c/L_{des} scaling factor. A method of reducing the size of the update is therefore required.

Given the expected update $\Delta \boldsymbol{X}$, and the expected change in length that any one truss would experience during a given step ΔL, the relative change in length for each truss is obtained by normalising the change in length:

$$\delta L = \left| \frac{\Delta L_{bar}}{L_c} \right| \tag{153}$$

This relative change in length, δL, for each truss is evaluated against the maximum allowable change, δ_{limit}. The allowable update proportion α is then calculated as the smaller of the ratio of δ_{limit} over the maximum relative change in length for the entire truss system, $\max(\delta \boldsymbol{L})$, and 1:

$$\alpha = \min\left(\frac{\delta_{limit}}{\max(\delta \boldsymbol{L})}, 1\right) \tag{154}$$

This implies that, until $\max(\delta \boldsymbol{L}) < \delta_{limit}$, the Newton step size will be limited to the fraction α. However, once the criterion is met the full Newton step size as given in (71) will apply. The modified form of the Newton update step is therefore given as:

$$\boldsymbol{\mathcal{X}}_{k+1} = \boldsymbol{\mathcal{X}}_{\boldsymbol{k}} + \alpha \Delta \boldsymbol{\mathcal{X}} \tag{155}$$

The intent of the modified Newton step is to restrict the change in truss length, based on the largest change in length, to prevent the negative effects associated with potentially large update sizes.

F.1.2. Newton step size: Line search method

It is recognised that the mesh generator is a form of optimisation problem. A more suitable step size could therefore be determined using a line search method, see [22]. The line search function for an allowable change in truss length, α is given by:

$$\Phi(\alpha) = \min\left(\Pi(\boldsymbol{\mathcal{X}}_{\boldsymbol{k}} + \alpha \Delta \boldsymbol{\mathcal{X}})\right) \tag{156}$$

where Π is the total system energy function that needs to be minimised and $\boldsymbol{\mathcal{X}}_{\boldsymbol{k}}$ and $\Delta \boldsymbol{\mathcal{X}}$ are constants. (156) can be solved for a value of α that minimises the total energy of the system. The new system state is then solved from (155).

F.1.3. Length normalisation (energy normalisation)

In DistMesh, desired truss lengths were generated as relative lengths rather than absolute lengths. This had the benefit of ensuring that the trusses would not, on average, be highly compressed or highly tensioned; a condition that results in high system potential energy and introduces instability. This was tested for suitability in the Newton implementation, where the relative desired truss length is then given by:

$$L_{des} = F_{scale}\left(\frac{\Sigma L_i^2}{\Sigma h(\mathcal{X}_i)}\right)^{\frac{1}{nD}} h(\boldsymbol{\mathcal{X}}) \tag{157}$$

A disadvantage of this method is that L_{des} has a discrete change whenever desired lengths are recalculated. This can influence the convergence rate of a solver; however, the effect can be mitigated by only updating the desired lengths every few iterations.

F.2. Convergence step size limit variants investigation

Three experiments were run to determine the effectiveness of a particular method for controlling the update step size, namely: fixed alpha update value; line search method and length normalisation. The results of these investigations are detailed in APPENDIX G, APPENDIX I, and APPENDIX H for the three methods respectively. Conclusions are presented in §F.3.

F.2.1. Newton update step: Fixed size α update control experiment

Convergence comparisons were done for control of the update step size using the update limiter method discussed in §F.1.1. Applicable values considered for the allowable change in truss length were $\alpha = 20, 5,\ 1, 0.5, 0.1, 0.05, 0.01$ for scaling factors $L_c/L_{des} = 0.77, 0.79, 0.8, 1, 10$. Initial conditions pertaining to the 2D truss effectiveness investigation detailed in APPENDIX C were used.

F.2.2. Newton update step: Line search method

Convergence comparisons were done for the line search method discussed in §F.1.2. Applicable values for the initial allowable change in truss length were $\alpha_0 = 1$ and 0.1 for scaling factors $L_c/L_{des} = 0.5$, $0.6,\ 0.7,\ 0.8, 1, 2, 5, 10$. Initial conditions pertaining to the 2D truss effectiveness investigation detailed in §APPENDIX C were used.

F.2.3. Length normalisation (energy normalisation) experiment on uniform mesh

Convergence comparisons were done for the length normalisation (energy normalisation) method discussed in §F.1.3. Applicable values for the scaling factor were $F_{scale} = 0.1, 0.8, 0.9, 1, 1.1, 1.2, 1.5$ for ratios $L_c/L_{des} = 0.1$ and 10. Only extreme ratios were compared due to the resilience of this method to poorly scaled trusses. Initial conditions pertaining to the 2D truss effectiveness investigation detailed in APPENDIX C were used.

F.3. Conclusion on update step size limit

The tendency of trusses under high compression to experience snap-through was revealed in §3.2.2.5 through the plotting and comparison of force curves for various force response formulae. Under these conditions, increased loading results in increased force up to a point, with further loading resulting in a decreased force response due to snap through.

The investigation conducted in §3.5 showed that the more compressed a system was ($L_c/L_{des} \leq 0.8$), the more likely the mesher was to fail to achieve convergence or to require a large number of iterations to solve in the cases where convergence was obtained. This held true for both the 2D and the 3D meshers.

In §3.6 the possibility of using an update step size limit to counteract the negative effects of highly compressed systems was introduced. The investigation conducted in this section highlighted the difficulties in managing compressed systems due to the many local minima present at various update step sizes. The system given by $L_c/L_{des} = 0.5$ was shown to be inherently unstable due to high

compression and therefore the most difficult to manage. The size of the update step required to solve the system was shown to have a significant effect on system stability, thereby leading to the conclusion that it should be carefully calculated.

It is worth noting that highly tensioned systems did not display convergence problems to the same extent. This does not, however, mean that tension should be favoured over compression. In this investigation, the systems under compression were shown to produce meshes with the highest values for element quality. Further, compressed systems play an important role in driving interior nodes to the boundary. This was the approach used by Persson and Strang.

Three approaches to calculating an update step size for improved stability in systems under compression were investigated: Fixed update size limiter, the line search method, and the length (system energy) normalisation method. The fixed update size method simply limits the maximum allowable step size to some multiple of the length of the truss. This was shown to facilitate convergence on systems that had previously failed but did not result in significant improvement over the unmodified Newton's method in terms of convergence rate.

The line search method was able to improve the stability of the compressed systems by preventing severe degeneration. This method was, however, unable to stabilise the very compressed systems enough to achieve convergence.

The energy normalisation method was found to produce the best results in terms of convergence and element quality. It was shown to be impervious to both severe compression and tension since the energy in each truss is normalised across the system energy. In terms of element quality, slight compression was shown to be desirable with the particular value $F_{scale} = 1.2$ giving the best results. This happened to correlate to Person and Strang's recommended value. For best convergence and acceptable element quality, the systems given by the range $0.8 \leq F_{scale} \leq 1.1$ were shown to meet these requirements.

APPENDIX G RESULTS OF FIXED-SIZE UPDATE LIMITER

Table G-14 Fixed α update: convergence comparison results for L_c/L_{des} = 0.79

$F_{truss} = \frac{\Delta L}{L_{des}}$							
α	20	5	1	0.5	0.1	0.05	0.01
Iterations	6	45	14	11	16	36	51
Avg L_c/L_{des}	NaN	0.865	0.801	0.801	0.801	0.801	0.801
Minimum Quality	NaN	0.000	0.740	0.741	0.742	0.721	0.600
Remeshes	6	45	11	8	8	12	1
Norm updates Iteration No.	0	0	0	0	0	0	0
1	3.87E-01	3.87E-01	3.87E-01	3.87E-01	3.87E-01	3.87E-01	3.87E-01
2	1.83E-01	1.83E-01	1.83E-01	2.12E-01	3.67E-01	3.74E-01	3.84E-01
3	1.01E+00	1.01E+00	7.51E-02	1.09E-01	3.33E-01	3.82E-01	3.82E-01
4	3.21E+00	4.73E-01	1.62E-01	1.12E-01	2.73E-01	3.47E-01	3.79E-01
5	4.46E+00	3.42E+00	1.52E-01	5.43E-02	2.12E-01	3.18E-01	3.78E-01
6	NaN	1.05E+01	6.34E-02	5.12E-02	1.55E-01	2.88E-01	3.76E-01
7	-	1.40E+01	1.20E-01	1.92E-02	1.30E-01	2.58E-01	3.76E-01
8	-	3.37E+00	7.86E-02	1.43E-02	9.39E-02	2.28E-01	3.77E-01
9	-	6.41E+00	4.10E-02	8.73E-04	7.21E-02	1.98E-01	3.80E-01
10	-	4.45E+00	1.96E-02	7.74E-06	7.93E-02	1.70E-01	3.87E-01
11	-	7.22E+00	2.93E-03	1.83E-09	4.71E-02	1.56E-01	4.07E-01
12	-	8.70E+00	2.29E-04	-	2.63E-02	1.39E-01	4.82E-01
13	-	1.02E+01	1.74E-06	-	3.10E-03	1.19E-01	1.66E+00
14	-	2.44E+00	5.77E-11	-	1.21E-04	9.91E-02	5.52E-01
15	-	8.03E+00	-	-	4.42E-07	8.67E-02	1.37E+00
16	-	7.96E+00	-	-	5.00E-12	7.01E-02	5.76E-01
34	-	1.08E+02	-	-	-	2.29E-04	1.31E+00
35	-	5.23E+00	-	-	-	1.62E-06	5.93E-01
36	-	4.72E+00	-	-	-	8.79E-11	1.61E+00
44	-	4.72E+00	-	-	-	-	2.29E+01
45	-	2.11E+02	-	-	-	-	5.07E-01
46	-	-	-	-	-	-	9.78E+00
47	-	-	-	-	-	-	4.77E-01
48	-	-	-	-	-	-	3.99E+00
49	-	-	-	-	-	-	4.90E-01
50	-	-	-	-	-	-	2.53E+00

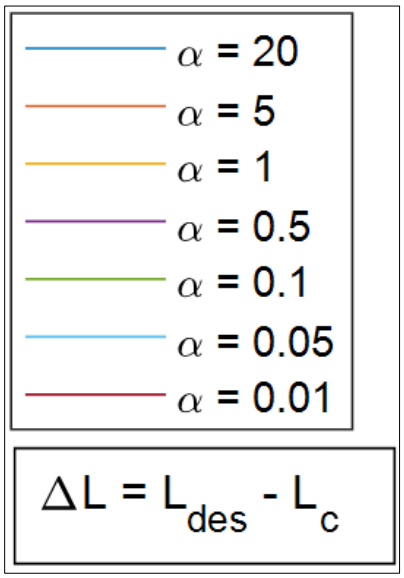


Figure G-6 Legend for following graphs, for reference

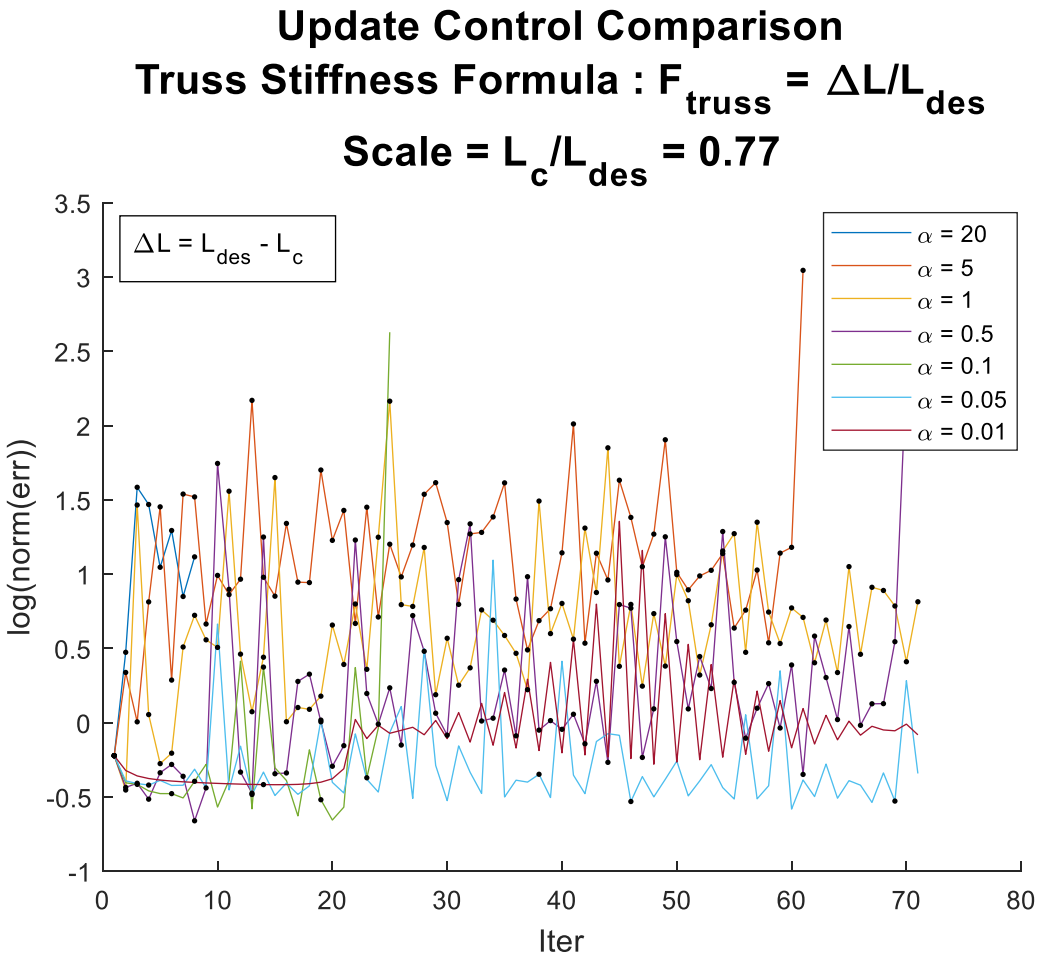


Figure G-7 Fixed α update convergence plot: $L_c/L_{des} = 0.77$

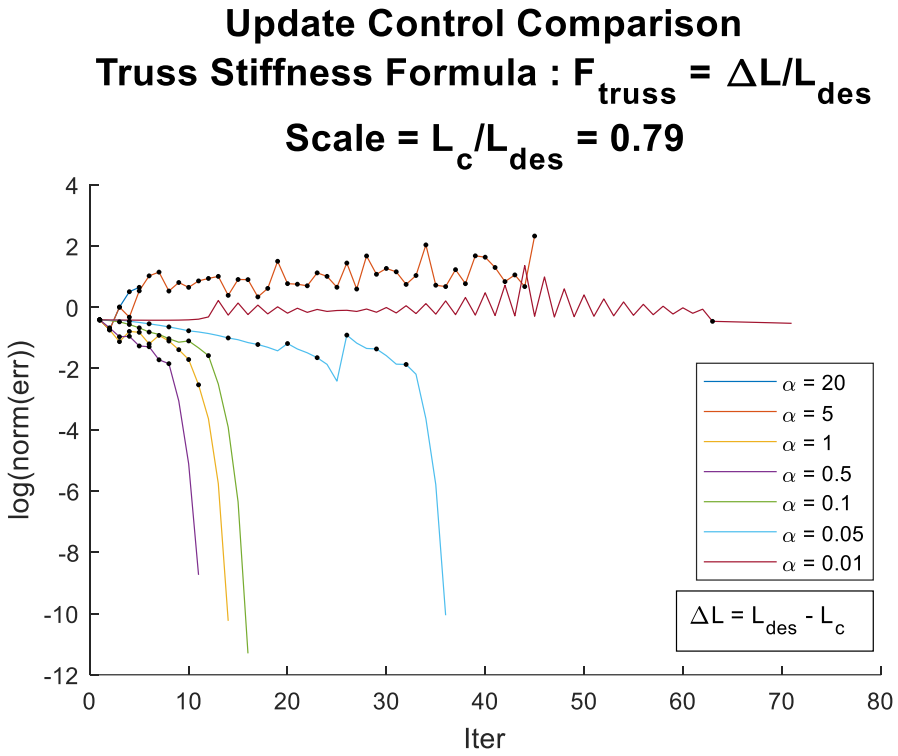


Figure G-8 Fixed α update convergence plot: $L_c/L_{des} = 0.79$

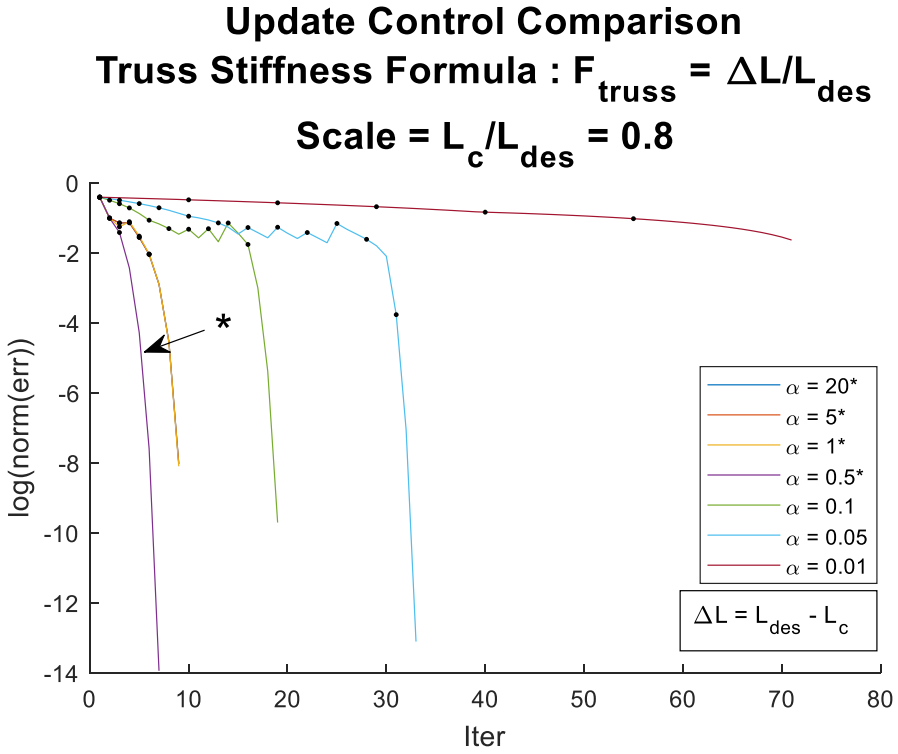


Figure G-9 Fixed α update convergence plot: $L_c/L_{des} = 0.8$

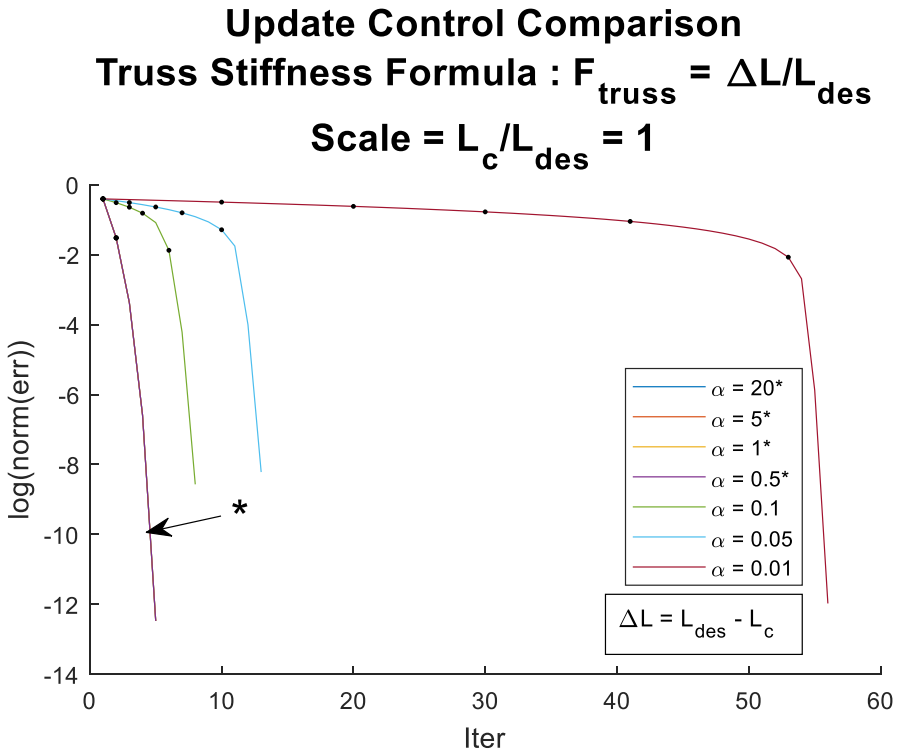


Figure G-10 Fixed α update convergence plot: $L_c/L_{des} = 1$

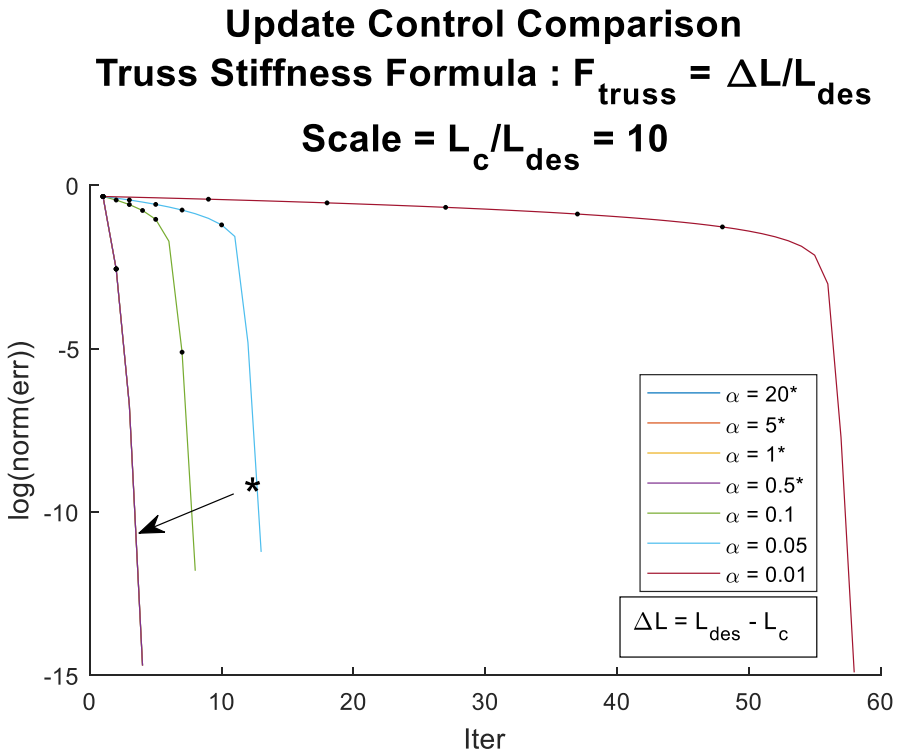


Figure G-11 Fixed α update convergence plot: $L_c/L_{des} = 10$

G.1. Evaluation of fixed-size update limiter

Table G-15 Fixed α update: comparison of number of required iterations

$F_{truss} =$	Scale L_c/L_{des}	$\frac{\Delta L}{L_{des}}$						
α	-	20	5	1	0.5	0.1	0.05	0.01
Required Iterations (convergence failed) [Exceeded max iterations]	0.77	(9)	[51]	[51]	[51]	(25)	[51]	[51]
	0.79	(6)	(45)	14	11	16	36	[51]
	0.8	9	9	9	7	19	33	[101]
	1	5	5	5	5	8	13	56
	10	4	4	4	4	8	13	58

Table G-16 Fixed α update: comparison of minimum triangle quality

$F_{truss} =$	Scale L_c/L_{des}	$\frac{\Delta L}{L_{des}}$						
α	-	20	5	1	0.5	0.1	0.05	0.01
Minimum quality (convergence failed) [Exceeded iterations]	0.77	NaN	[3.24E-09]	[0.069]	[0.432]	(0.623)	[0.638]	[0.601]
	0.79	NaN	(8.73E-06)	0.740	0.741	0.742	0.721	[0.600]
	0.8	0.736	0.736	0.736	0.721	0.736	0.736	[0.709]
	1	0.671	0.671	0.671	0.671	0.671	0.671	0.671
	10	0.587	0.587	0.587	0.587	0.587	0.587	0.587

From Figure G-6 through Figure G-11 and corresponding data presented in Table G-14 and Table G-15, it is evident that the introduction of the update step size limit improves the stability of the algorithms, especially for the highly compressed systems given by $L_c/L_{des} = 0.77$ and 0.79. Using small values of $\alpha < 1$ assisted in achieving convergence of systems that had previously failed. This achievement is however offset by the increased number of iterations required.

From Table G-16 it is evident that the quality of the mesh was not affected in any meaningful manner. This demonstrates the fact that the α update step size limit affects convergence only.

APPENDIX H RESULTS OF LINE SEARCH

Table H-17 Line search: convergence comparison results for various L_c/L_{des} and $\alpha_0 = 1$

$F_{truss} = \frac{\Delta L}{L_{des}}$							
L_c/L_{des}	0.5	0.6	0.7	0.8	1	2	5
Iterations	6	11	26	9	5	4	5
Avg L_c/L_{des}	NaN	NaN	0.745	0.811	1.013	2.025	5.063
Minimum Quality	NaN	NaN	0.242	0.736	0.671	0.625	0.595
Remeshes	6	11	26	6	2	2	2
Norm updates Iteration No.	0	0	0	0	0	0	0
1	1.45E+00	3.33E-01	1.88E-01	4.07E-01	4.12E-01	4.36E-01	4.50E-01
2	2.39E+00	5.40E-01	2.28E-01	4.97E-02	1.52E-02	3.84E-03	1.23E-03
3	1.70E+00	9.41E-01	1.59E-01	3.41E-02	2.10E-04	6.41E-06	2.43E-07
4	2.00E+00	8.62E-01	1.71E-01	3.41E-02	2.77E-08	5.15E-09	1.28E-08
5	1.18E+00	9.82E-01	1.80E-01	3.83E-02	5.69E-15	-	1.29E-15
6	NaN	1.20E+00	2.52E-01	9.89E-03	-	-	-
7	-	6.75E-01	2.79E-01	1.78E-04	-	-	-
8	-	1.03E+00	2.39E-01	1.75E-07	-	-	-
9	-	9.32E-01	1.53E-01	2.27E-09	-	-	-
10	-	1.06E+00	1.90E-01	-	-	-	-
11	-	NaN	1.87E-01	-	-	-	-
12	-	-	2.09E-01	-	-	-	-
18	-	-	1.33E-01	-	-	-	-

Table H-18 Line search: convergence comparison results for various L_c/L_{des} and $\alpha_0 = 0.1$

$F_{truss} = \frac{\Delta L}{L_{des}}$							
L_c/L_{des}	0.5	0.6	0.7	0.8	1	2	5
Iterations	6	26	26	14	5	4	12
Avg L_c/L_{des}	NaN	0.718	0.745	0.811	1.013	2.025	5.063
Minimum Quality	NaN	0.000	0.272	0.736	0.671	0.625	0.595
Remeshes	6	26	25	6	2	2	2
Norm updates Iteration No.	0	0	0	0	0	0	0
1	1.45E+00	3.33E-01	1.88E-01	4.07E-01	4.12E-01	4.36E-01	4.50E-01
2	2.38E+00	5.37E-01	2.19E-01	4.97E-02	1.52E-02	3.84E-03	1.24E-03
3	2.43E+00	7.72E-01	2.40E-01	3.41E-02	2.10E-04	6.48E-06	2.56E-08
4	1.47E+00	7.30E-01	2.06E-01	3.41E-02	2.70E-08	1.05E-10	2.42E-08
5	2.41E+00	8.87E-01	2.33E-01	3.83E-02	4.11E-11	-	2.17E-08
6	NaN	1.06E+00	1.66E-01	9.89E-03	-	-	1.85E-08
12	-	5.26E-01	8.38E-02	1.14E-08	-	-	9.61E-09
13	-	8.40E-01	1.60E-01	1.08E-08	-	-	-
14	-	8.74E-01	1.57E-01	9.20E-09	-	-	-
15	-	9.44E-01	2.09E-01	-	-	-	-
18	-	1.16E+00	1.24E-01	-	-	-	-

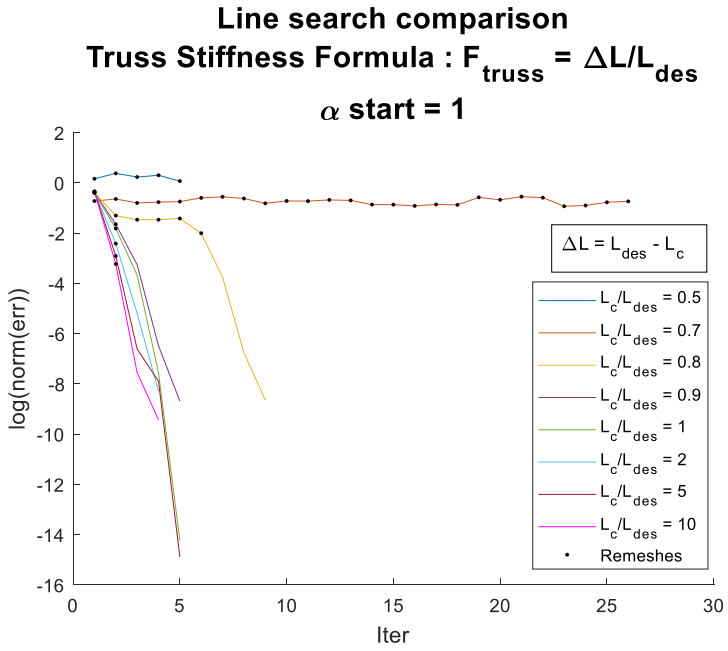


Figure H-12 Line search comparison for various L_c/L_{des} and initial step size = 1

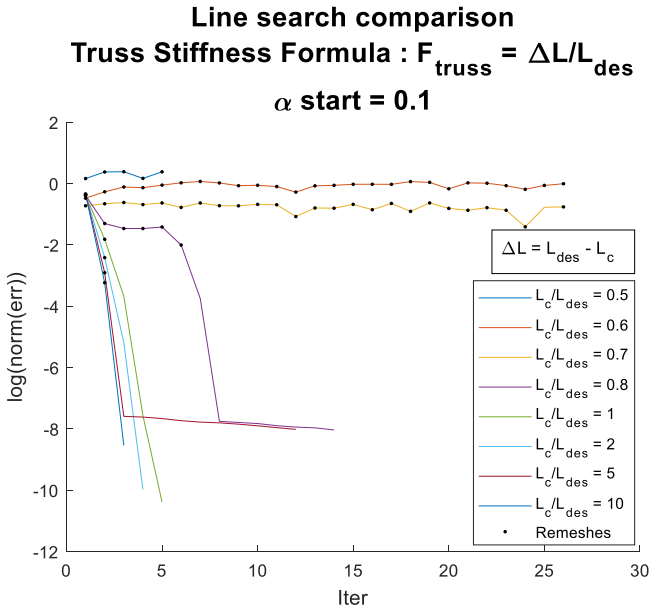


Figure H-13 Line search comparison for various L_c/L_{des} and initial step size = 0.1

H.1. Evaluation of line search update

Figure H-12 and Figure H-13 with corresponding data presented in Table H-17 and Table H-18, it is evident that the update step size limit calculated using a line search improves system stability. The line-search is known to be a fundamental component of many optimisation algorithms. It is therefore unsurprising that its implementation as a means of determining the optimum update size results in improved system stability. It was however found that whilst stability was improved for systems under high compression, in that they did not drastically diverge, convergence was no better than that of the standard Newton's method for slight compressions and in tension. This lack of improvement in terms of convergence is due to the existence of several local minima caused by the lower energy states of snapped-through truss locations as depicted in Figure H-14.

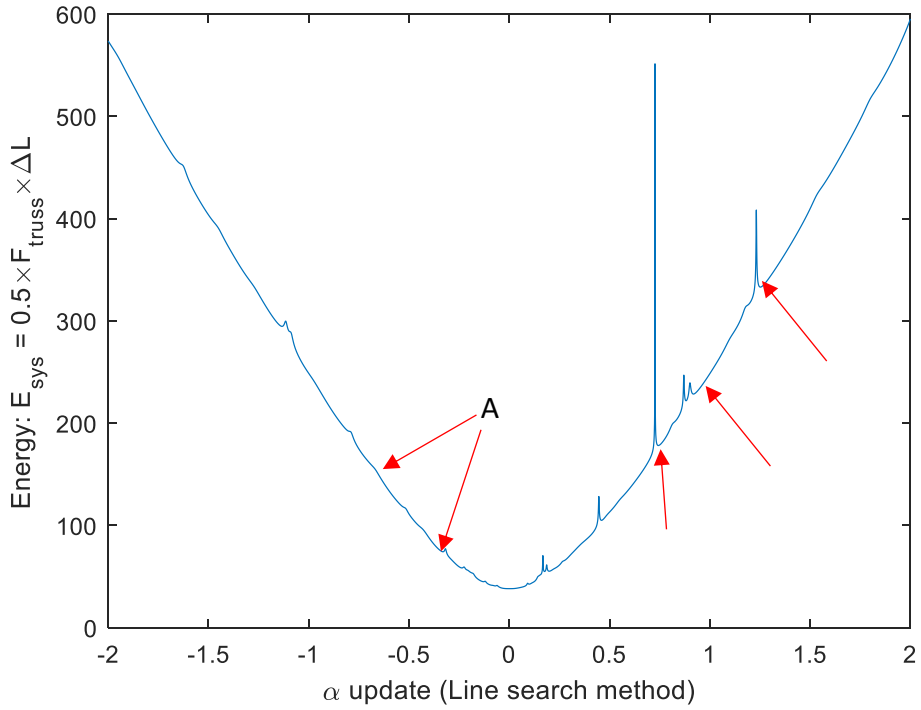


Figure H-14 Highly compressed system showing multiple local minima due to snap through

Under conditions of high compression given by $L_c/L_{des} = 0.5$, no local minima associated with snap-through were noted. Snap through was indicated by slight inflection points on the line search curve. Figure H-15 shows an inflection point for the magnified approximate update step size limit of $\alpha = 0.044$. A straight line is plotted next to the energy curve to highlight the inflection. Since this system energy function does not manifest the energy reduction associated with snap-through as distinct minima, the algorithm was unable to achieve convergence to the expected result. The result of the first update can be seen in Figure H-16.

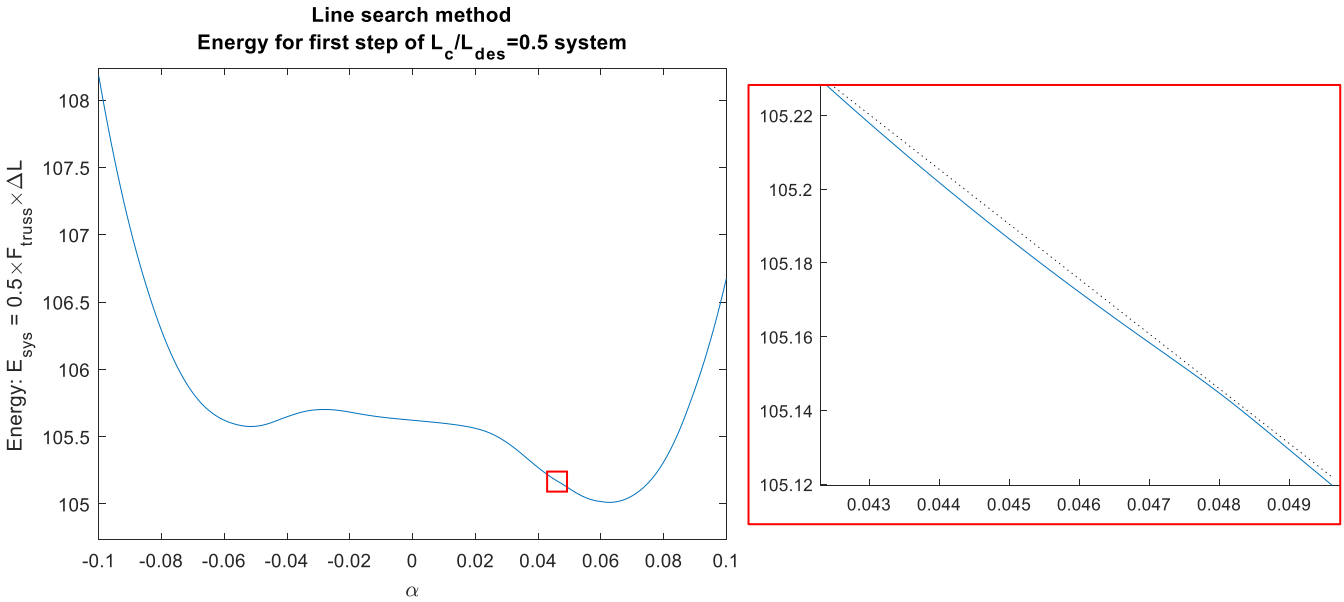


Figure H-15 System energy local minima first iteration $L_c/L_{des} = 0.5$

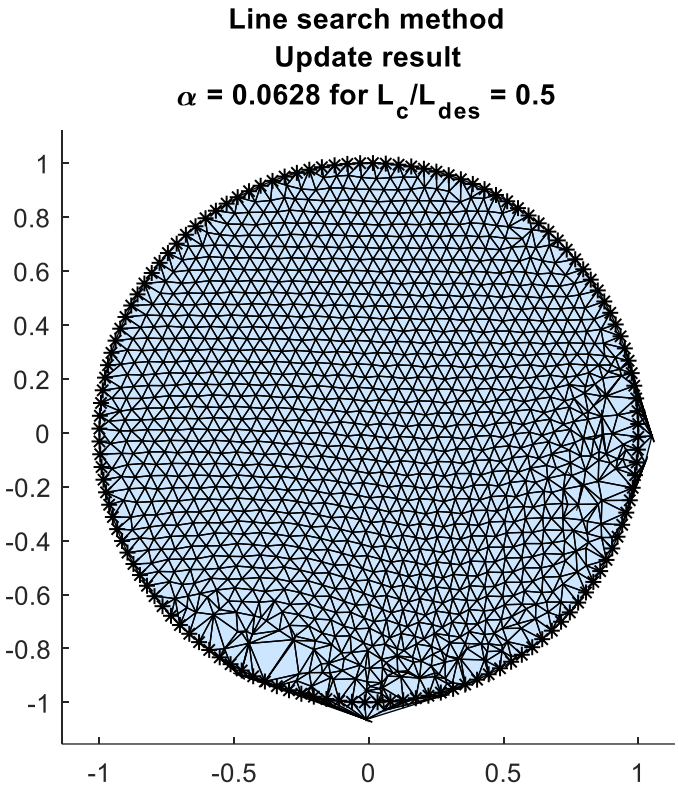


Figure H-16 First mesh update for $L_c/L_{des} = 0.5$ and $\alpha = 0.0628$

APPENDIX I RESULTS OF LENGTH (SYSTEM ENERGY) NORMALISATION

Table I-19 Length (system energy) normalisation: convergence results for L_c/L_{des} = 0.1

$F_{truss} =$	$F_{scale}\frac{\Delta L}{L_{scale}}$						
F_{scale}	0.1	0.8	0.9	1	1.1	1.2	1.5
Iterations	4	5	5	5	5	8	9
Avg L_c/L_{des}	0.101	0.101	0.101	0.101	0.101	0.101	NaN
Minimum Quality	0.587	0.652	0.662	0.672	0.682	0.720	NaN
Remeshes	2	2	2	2	2	5	9
Norm updates Iteration No.							
1	4.51E-01	4.02E-01	3.95E-01	3.89E-01	3.84E-01	3.81E-01	2.41E+00
2	2.80E-03	2.30E-02	2.66E-02	3.11E-02	3.78E-02	5.49E-02	5.24E+01
3	1.98E-07	1.57E-04	2.65E-04	4.70E-04	9.70E-04	3.88E-02	7.66E+01
4	2.43E-13	3.27E-08	9.38E-08	3.53E-07	2.22E-06	1.27E-02	1.55E+02
5	-	1.75E-11	5.89E-11	1.42E-10	1.06E-09	1.85E-03	3.64E+01
6	-	-	-	-	-	5.93E-05	6.50E+01
7	-	-	-	-	-	1.25E-07	3.21E+01
8	-	-	-	-	-	1.34E-09	1.28E+02
9	-	-	-	-	-	-	NaN

Table I-20 Length (system energy) normalisation: convergence results for L_c/L_{des} = 10

$F_{truss} =$	$F_{scale}\frac{\Delta L}{L_{scale}}$						
F_{scale}	0.1	0.8	0.9	1	1.1	1.2	1.5
Iterations	4	5	5	5	5	8	8
Avg L_c/L_{des}	10.125	10.129	10.130	10.132	10.135	10.139	NaN
Minimum Quality	0.587	0.652	0.662	0.672	0.682	0.720	NaN
Remeshes	2	2	2	2	2	5	8
Norm updates Iteration No.							
1	4.51E-01	4.02E-01	3.95E-01	3.89E-01	3.84E-01	3.81E-01	2.41E+00
2	2.80E-03	2.30E-02	2.66E-02	3.11E-02	3.78E-02	5.49E-02	5.24E+01
3	1.98E-07	1.57E-04	2.65E-04	4.70E-04	9.70E-04	3.88E-02	7.66E+01
4	2.43E-13	3.27E-08	9.38E-08	3.53E-07	2.22E-06	1.27E-02	1.55E+02
5	-	1.75E-11	5.89E-11	1.42E-10	1.06E-09	1.85E-03	3.62E+01
6	-	-	-	-	-	5.93E-05	1.21E+02
7	-	-	-	-	-	1.25E-07	1.94E+01
8	-	-	-	-	-	1.34E-09	NaN

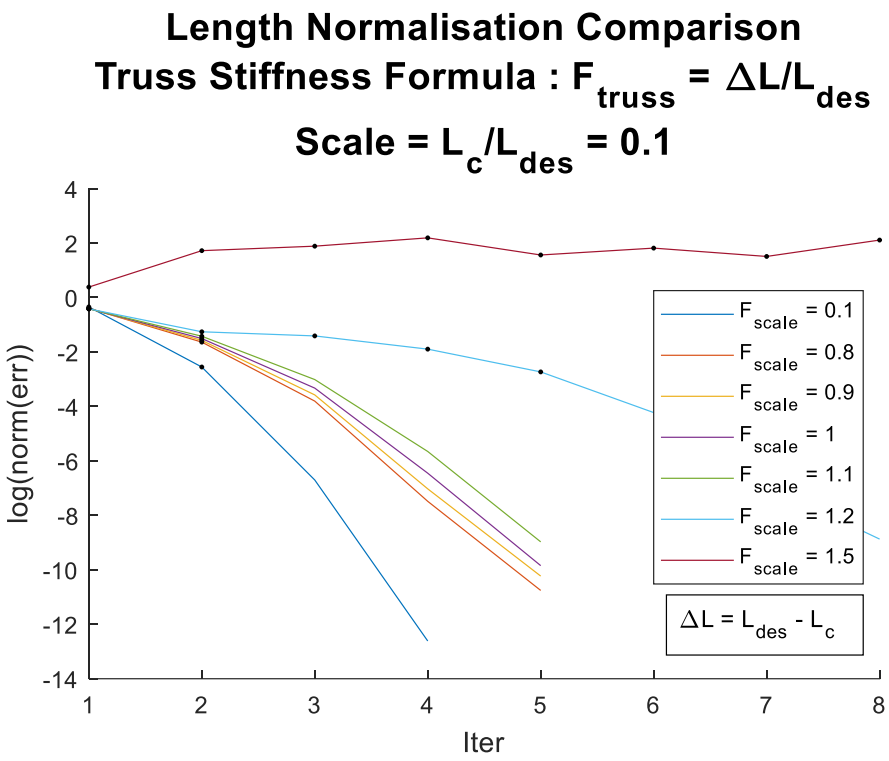


Figure I-17 Length (system energy) normalisation convergence plot: $L_c/L_{des} = 0.1$

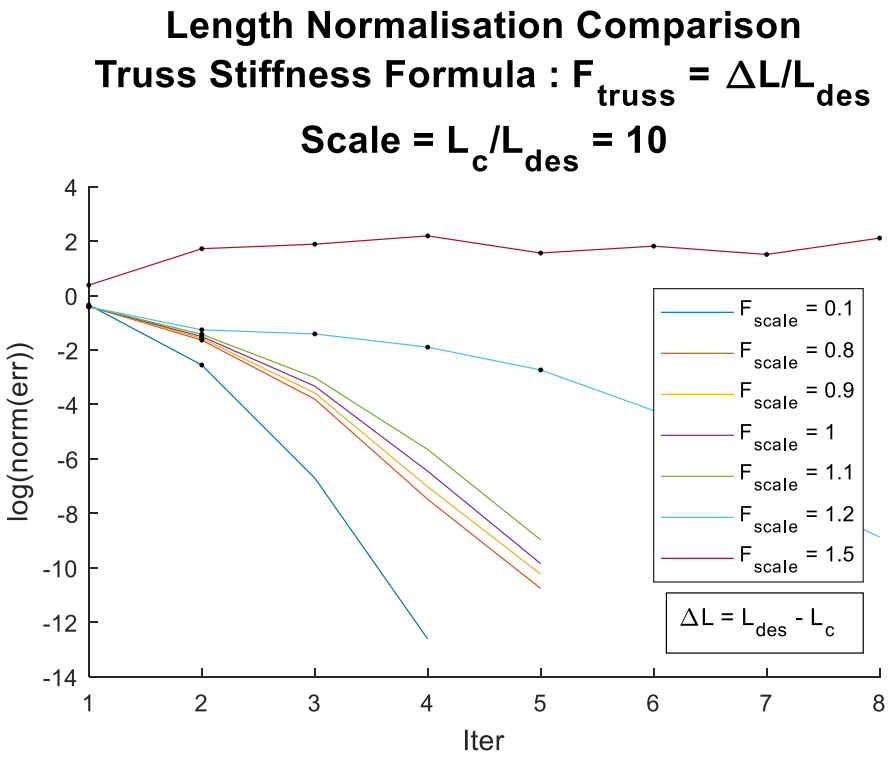


Figure I-18 Length (system energy) normalisation convergence plot: $L_c/L_{des} = 10$

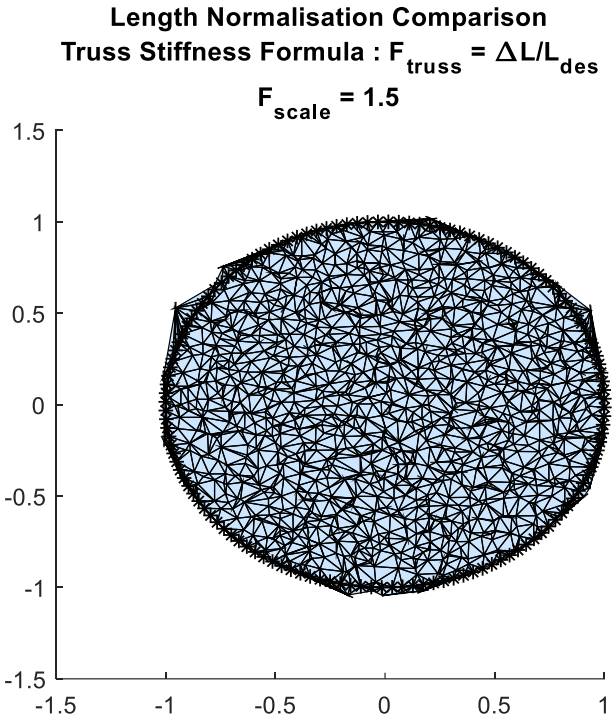


Figure I-19 Highly compressed state showing snap though and failure of mesh

I.1. Evaluation of length (system energy) normalisation

Figure I-17 and Figure I-18 with corresponding data presented in Table I-19 and Table I-20, it is evident that the length (system energy) normalisation method is extremely successful at improving system stability. It was for this reason that results used to demonstrate its effectiveness were given only for the system under compression given by $L_c/L_{des} = 0.1$ and the system under tension given by $L_c/L_{des} = 10$.

The mathematic associated with this method indicated that convergence results could be expected to be independent of the scaling factor. Comparison of the results presented in Table I-19 and Table I-20 for corresponding values of F_{scale} show that that the scaling factor has nearly no effect, thereby confirming correct implementation of the length normalisation method.

For values of $F_{scale} < 1.2$ this method results in excellent convergence characteristics. For higher values of F_{scale}, the system becomes unstable as would be expected due to increased compression.

Figure I-19 is interesting since it shows the effect of snap-through due to a system with high compression. The mesh is considered to have failed due to multiple points having moved outside of the boundary. This is one of the negative effects associated with large updates that an update step size limit successfully addresses.

APPENDIX J 2D UPDATE STEP SIZE LIMIT TABLES – FIXED UPDATE α CONTROL

Table J-21 α update: truss formulation convergence results for L_c/L_{des} = 0.77

$F_{truss} = \frac{\Delta L}{L_{des}}$							
α	20	5	1	0.5	0.1	0.05	0.01
Iterations	9	51	51	51	25	51	51
Ave L_c/L_{des}	NaN	0.843	0.797	0.786	0.782	0.781	0.781
Minimum Quality	NaN	0.000	0.069	0.432	0.623	0.638	0.601
Remeshes	9	51	51	51	7	6	2
Norm updates Iteration No.	0	0	0	0	0	0	0
1	6.00E-01	6.00E-01	6.00E-01	6.00E-01	6.00E-01	6.00E-01	6.00E-01
2	2.98E+00	2.18E+00	3.52E-01	3.67E-01	3.94E-01	4.06E-01	4.77E-01
3	3.84E+01	1.01E+00	2.92E+01	3.91E-01	3.85E-01	3.90E-01	4.39E-01
4	2.94E+01	6.50E+00	1.13E+00	3.06E-01	3.48E-01	3.81E-01	4.21E-01
5	1.11E+01	2.84E+01	5.31E-01	4.61E-01	3.33E-01	4.09E-01	4.11E-01
6	1.97E+01	1.94E+00	6.24E-01	5.24E-01	3.33E-01	3.78E-01	4.05E-01
7	7.07E+00	3.46E+01	3.23E+00	4.36E-01	3.12E-01	3.80E-01	4.00E-01
8	1.31E+01	3.31E+01	5.28E+00	2.19E-01	4.03E-01	4.87E-01	3.97E-01
9	NaN	4.62E+00	3.61E+00	3.64E-01	5.24E-01	3.64E-01	3.93E-01
10	-	9.81E+00	3.21E+00	5.56E+01	2.71E-01	4.61E+00	3.91E-01
24	-	5.15E+00	1.77E+01	9.78E-01	9.18E-01	3.42E-01	9.61E-01
25	-	1.59E+01	1.46E+02	1.72E+00	4.25E+02	8.59E-01	8.49E-01
26	-	9.58E+00	6.23E+00	7.08E-01	NaN	1.29E+00	8.89E-01
27	-	1.57E+01	6.06E+00	5.26E+00	-	3.09E-01	9.32E-01
28	-	3.45E+01	1.51E+01	3.02E+00	-	3.15E+00	8.30E-01
29	-	4.13E+01	1.54E+00	1.16E+00	-	5.16E-01	1.04E+00
48	-	1.86E+01	5.42E+00	1.24E+00	-	3.17E-01	5.26E-01
49	-	8.02E+01	2.41E+00	1.78E+01	-	4.16E-01	5.42E+00
50	-	1.03E+01	9.92E+00	3.52E+00	-	5.52E-01	5.43E-01

Table J-22 α update: truss formulation convergence results for L_c/L_{des} = 0.79

$F_{truss} = \frac{\Delta L}{L_{des}}$							
α	20	5	1	0.5	0.1	0.05	0.01
Iterations	6	45	14	11	16	36	51
Ave L_c/L_{des}	NaN	0.865	0.801	0.801	0.801	0.801	0.801
Minimum Quality	NaN	0.000	0.740	0.741	0.742	0.721	0.600
Remeshes	6	45	11	8	8	12	1
Norm updates Iteration No.	0	0	0	0	0	0	0
1	3.87E-01	3.87E-01	3.87E-01	3.87E-01	3.87E-01	3.87E-01	3.87E-01
2	1.83E-01	1.83E-01	1.83E-01	2.12E-01	3.67E-01	3.74E-01	3.84E-01
3	1.01E+00	1.01E+00	7.51E-02	1.09E-01	3.33E-01	3.82E-01	3.82E-01
4	3.21E+00	4.73E-01	1.62E-01	1.12E-01	2.73E-01	3.47E-01	3.79E-01
5	4.46E+00	3.42E+00	1.52E-01	5.43E-02	2.12E-01	3.18E-01	3.78E-01
6	NaN	1.05E+01	6.34E-02	5.12E-02	1.55E-01	2.88E-01	3.76E-01
7	-	1.40E+01	1.20E-01	1.92E-02	1.30E-01	2.58E-01	3.76E-01
8	-	3.37E+00	7.86E-02	1.43E-02	9.39E-02	2.28E-01	3.77E-01
9	-	6.41E+00	4.10E-02	8.73E-04	7.21E-02	1.98E-01	3.80E-01
10	-	4.45E+00	1.96E-02	7.74E-06	7.93E-02	1.70E-01	3.87E-01
11	-	7.22E+00	2.93E-03	1.83E-09	4.71E-02	1.56E-01	4.07E-01
12	-	8.70E+00	2.29E-04	-	2.63E-02	1.39E-01	4.82E-01
13	-	1.02E+01	1.74E-06	-	3.10E-03	1.19E-01	1.66E+00
14	-	2.44E+00	5.77E-11	-	1.21E-04	9.91E-02	5.52E-01
15	-	8.03E+00	-	-	4.42E-07	8.67E-02	1.37E+00
16	-	7.96E+00	-	-	5.00E-12	7.01E-02	5.76E-01
34	-	1.08E+02	-	-	-	2.29E-04	1.31E+00
35	-	5.23E+00	-	-	-	1.62E-06	5.93E-01
36	-	4.72E+00	-	-	-	8.79E-11	1.61E+00
44	-	4.72E+00	-	-	-	-	2.29E+01
45	-	2.11E+02	-	-	-	-	5.07E-01
46	-	-	-	-	-	-	9.78E+00
47	-	-	-	-	-	-	4.77E-01
48	-	-	-	-	-	-	3.99E+00
49	-	-	-	-	-	-	4.90E-01
50	-	-	-	-	-	-	2.53E+00

Table J-23 α update: truss formulation convergence results for L_c/L_{des} = 0.8

$F_{truss} = \frac{\Delta L}{L_{des}}$							
α	20	5	1	0.5	0.1	0.05	0.01
Iterations	9	9	9	7	19	33	101
Ave L_c/L_{des}	0.811	0.811	0.811	0.811	0.811	0.811	0.811
Minimum Quality	0.736	0.736	0.736	0.721	0.736	0.736	0.709
Remeshes	6	6	6	3	10	12	9
Norm updates Iteration No.							
1	3.83E-01	3.83E-01	3.83E-01	3.83E-01	3.83E-01	3.83E-01	3.83E-01
2	9.67E-02	9.67E-02	9.67E-02	9.67E-02	3.13E-01	3.49E-01	3.76E-01
3	7.04E-02	7.04E-02	5.45E-02	3.75E-02	2.49E-01	3.15E-01	3.69E-01
4	7.12E-02	7.12E-02	7.65E-02	3.64E-03	1.89E-01	2.82E-01	3.62E-01
5	2.73E-02	2.73E-02	2.95E-02	5.17E-05	1.31E-01	2.51E-01	3.56E-01
6	8.93E-03	8.93E-03	9.11E-03	2.50E-08	8.43E-02	2.21E-01	3.49E-01
7	1.24E-03	1.24E-03	1.25E-03	1.13E-14	6.60E-02	1.91E-01	3.43E-01
8	2.85E-05	2.85E-05	2.72E-05	-	4.85E-02	1.62E-01	3.36E-01
9	9.27E-09	9.27E-09	7.88E-09	-	3.35E-02	1.33E-01	3.30E-01
10	-	-	-	-	4.66E-02	1.10E-01	3.22E-01
16	-	-	-	-	1.71E-02	5.20E-02	2.83E-01
17	-	-	-	-	9.76E-04	3.72E-02	2.77E-01
18	-	-	-	-	4.06E-06	2.64E-02	2.71E-01
19	-	-	-	-	1.95E-10	5.30E-02	2.65E-01
29	-	-	-	-	-	1.57E-02	2.05E-01
30	-	-	-	-	-	7.89E-03	1.99E-01
31	-	-	-	-	-	1.69E-04	1.93E-01
32	-	-	-	-	-	8.72E-08	1.87E-01
33	-	-	-	-	-	7.74E-14	1.81E-01
108	-	-	-	-	-	-	8.15E-02
109	-	-	-	-	-	-	7.49E-02
110	-	-	-	-	-	-	6.92E-02
111	-	-	-	-	-	-	6.40E-02
112	-	-	-	-	-	-	5.94E-02

Table J-24 α update: truss formulation convergence results for L_c/L_{des} = 1

$F_{truss} = \frac{\Delta L}{L_{des}}$							
α	20	5	1	0.5	0.1	0.05	0.01
Iterations	5	5	5	5	8	13	56
Ave L_c/L_{des}	1.013	1.013	1.013	1.013	1.013	1.013	1.013
Minimum Quality	0.671	0.671	0.671	0.671	0.671	0.671	0.671
Remeshes	2	2	2	2	5	5	6
Norm updates Iteration No.							
1	3.90E-01	3.90E-01	3.90E-01	3.90E-01	3.90E-01	3.90E-01	3.90E-01
2	3.00E-02	3.00E-02	3.00E-02	3.00E-02	3.04E-01	3.47E-01	3.81E-01
3	4.22E-04	4.22E-04	4.22E-04	4.22E-04	2.26E-01	3.06E-01	3.73E-01
4	2.58E-07	2.58E-07	2.58E-07	2.58E-07	1.52E-01	2.67E-01	3.64E-01
5	3.38E-13	3.38E-13	3.38E-13	3.38E-13	8.20E-02	2.29E-01	3.56E-01
6	-	-	-	-	1.32E-02	1.92E-01	3.48E-01
7	-	-	-	-	6.06E-05	1.56E-01	3.40E-01
8	-	-	-	-	2.71E-09	1.21E-01	3.32E-01
9	-	-	-	-	-	8.59E-02	3.24E-01
10	-	-	-	-	-	5.12E-02	3.16E-01
11	-	-	-	-	-	1.75E-02	3.08E-01
12	-	-	-	-	-	9.84E-05	3.00E-01
13	-	-	-	-	-	6.09E-09	2.92E-01
37	-	-	-	-	-	-	1.17E-01
38	-	-	-	-	-	-	1.10E-01
39	-	-	-	-	-	-	1.03E-01
40	-	-	-	-	-	-	9.57E-02
41	-	-	-	-	-	-	8.87E-02
42	-	-	-	-	-	-	8.17E-02
50	-	-	-	-	-	-	2.78E-02
51	-	-	-	-	-	-	2.13E-02
52	-	-	-	-	-	-	1.48E-02
53	-	-	-	-	-	-	8.41E-03
54	-	-	-	-	-	-	2.04E-03
55	-	-	-	-	-	-	1.33E-06
56	-	-	-	-	-	-	1.07E-12

Table J-25 α update: truss formulation alpha update convergence results for L_c/L_{des} = 10

$F_{truss} = \frac{\Delta L}{L_{des}}$							
α	20	5	1	0.5	0.1	0.05	0.01
Iterations	4	4	4	4	8	13	58
Ave L_c/L_{des}	10.124	10.125	10.125	10.125	10.125	10.125	10.125
Minimum Quality	0.587	0.587	0.587	0.587	0.587	0.587	0.587
Remeshes	2	2	2	2	6	5	6
Norm updates Iteration No.							
1	4.51E-01	4.51E-01	4.51E-01	4.51E-01	4.51E-01	4.51E-01	4.51E-01
2	2.73E-03	2.73E-03	2.73E-03	2.73E-03	3.51E-01	4.01E-01	4.41E-01
3	1.84E-07	1.84E-07	1.84E-07	1.84E-07	2.55E-01	3.52E-01	4.31E-01
4	1.99E-15	1.99E-15	1.99E-15	1.99E-15	1.68E-01	3.04E-01	4.21E-01
5	-	-	-	-	9.00E-02	2.59E-01	4.12E-01
6	-	-	-	-	1.93E-02	2.15E-01	4.02E-01
7	-	-	-	-	7.75E-06	1.73E-01	3.92E-01
8	-	-	-	-	1.58E-12	1.34E-01	3.82E-01
9	-	-	-	-	-	9.66E-02	3.72E-01
10	-	-	-	-	-	6.09E-02	3.63E-01
11	-	-	-	-	-	2.70E-02	3.53E-01
12	-	-	-	-	-	1.51E-05	3.44E-01
13	-	-	-	-	-	5.97E-12	3.34E-01
14	-	-	-	-	-	-	3.25E-01
15	-	-	-	-	-	-	3.15E-01
16	-	-	-	-	-	-	3.06E-01
50	-	-	-	-	-	-	3.94E-02
51	-	-	-	-	-	-	3.28E-02
52	-	-	-	-	-	-	2.63E-02
53	-	-	-	-	-	-	1.99E-02
54	-	-	-	-	-	-	1.35E-02
55	-	-	-	-	-	-	7.20E-03
56	-	-	-	-	-	-	9.50E-04
57	-	-	-	-	-	-	1.92E-08
58	-	-	-	-	-	-	1.23E-15

APPENDIX K UPDATE STEP SIZE LIMIT TABLES - LENGTH CONTROL

Table K-26 Length control: truss formulation convergence results for L_c/L_{des} = 0.1

$F_{truss} = \frac{\Delta L}{L_{des}}$							
F_{scale}	0.1	0.8	0.9	1	1.1	1.2	1.5
Iterations	4	5	5	5	5	8	9
Ave L_c/L_{des}	0.101	0.101	0.101	0.101	0.101	0.101	NaN
Minimum Quality	0.587	0.652	0.662	0.672	0.682	0.720	NaN
Remeshes	2	2	2	2	2	5	9
Norm updates Iteration No.							
1	4.51E-01	4.02E-01	3.95E-01	3.89E-01	3.84E-01	3.81E-01	2.41E+00
2	2.80E-03	2.30E-02	2.66E-02	3.11E-02	3.78E-02	5.49E-02	5.24E+01
3	1.98E-07	1.57E-04	2.65E-04	4.70E-04	9.70E-04	3.88E-02	7.66E+01
4	2.43E-13	3.27E-08	9.38E-08	3.53E-07	2.22E-06	1.27E-02	1.55E+02
5	-	1.75E-11	5.89E-11	1.42E-10	1.06E-09	1.85E-03	3.64E+01
6	-	-	-	-	-	5.93E-05	6.50E+01
7	-	-	-	-	-	1.25E-07	3.21E+01
8	-	-	-	-	-	1.34E-09	1.28E+02
9	-	-	-	-	-	-	NaN

Table K-27 Length control: truss formulation convergence results for L_c/L_{des} = 10

$F_{truss} = \frac{\Delta L}{L_{des}}$							
F_{scale}	0.1	0.8	0.9	1	1.1	1.2	1.5
Iterations	4	5	5	5	5	8	8
Ave L_c/L_{des}	10.125	10.129	10.130	10.132	10.135	10.139	NaN
Minimum Quality	0.587	0.652	0.662	0.672	0.682	0.720	NaN
Remeshes	2	2	2	2	2	5	8
Norm updates Iteration No.							
1	4.51E-01	4.02E-01	3.95E-01	3.89E-01	3.84E-01	3.81E-01	2.41E+00
2	2.80E-03	2.30E-02	2.66E-02	3.11E-02	3.78E-02	5.49E-02	5.24E+01
3	1.98E-07	1.57E-04	2.65E-04	4.70E-04	9.70E-04	3.88E-02	7.66E+01
4	2.43E-13	3.27E-08	9.38E-08	3.53E-07	2.22E-06	1.27E-02	1.55E+02
5	-	1.75E-11	5.89E-11	1.42E-10	1.06E-09	1.85E-03	3.62E+01
6	-	-	-	-	-	5.93E-05	1.21E+02
7	-	-	-	-	-	1.25E-07	1.94E+01
8	-	-	-	-	-	1.34E-09	NaN

APPENDIX L MSEM APPROACH TO THE BOUNDARY SENSITIVITIES PROBLEM

For the MSEM approach, the constraint $\partial\boldsymbol{\Omega}_{\mathrm{c}}$ in (114) is partitioned to into the master $\partial\boldsymbol{\Omega}_m$ and slave $\partial\boldsymbol{\Omega}_s$ components.

$$\begin{aligned}\boldsymbol{F}\left(\boldsymbol{\mathcal{X}}^{\Omega}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{m}}}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_s}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{p}}}(\mathbf{x})\right)&\\ =\begin{Bmatrix}\boldsymbol{F}_{\Omega}\left(\boldsymbol{\mathcal{X}}^{\Omega}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{m}}}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_s}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{p}}}(\mathbf{x})\right)\\ \boldsymbol{F}_{\partial\Omega_m}\left(\boldsymbol{\mathcal{X}}^{\Omega}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{m}}}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_s}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{p}}}(\mathbf{x})\right)\\ \boldsymbol{F}_{\partial\Omega_{\mathrm{p}}}\left(\boldsymbol{\mathcal{X}}^{\Omega}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{m}}}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_s}(\mathbf{x}),\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{p}}}(\mathbf{x})\right)\end{Bmatrix}=\mathbf{0}\end{aligned} \tag{158}$$

The set of residual equations for the MSEM that need to be satisfied are given in (34) through (37) and restated here:

$$\boldsymbol{\mathcal{R}}_f=\boldsymbol{\mathcal{F}}_f^{int}-\boldsymbol{\mathcal{F}}_f^{ext}=\mathbf{0} \tag{159}$$

$$\boldsymbol{\mathcal{R}}_{ms}=\boldsymbol{\mathcal{F}}_m^{int}-\boldsymbol{\mathcal{F}}_m^{ext}+\boldsymbol{P}^T\left(\boldsymbol{\mathcal{F}}_s^{int}-\boldsymbol{\mathcal{F}}_s^{ext}\right)=\mathbf{0} \tag{160}$$

$$\boldsymbol{\mathcal{R}}_p=\boldsymbol{\mathcal{F}}_p^{int}-\boldsymbol{\mathcal{F}}_p^{ext}-\boldsymbol{R}_P=\mathbf{0} \tag{161}$$

$$\boldsymbol{u}_s=\boldsymbol{f}_{ms}(\boldsymbol{u}_m) \tag{162}$$

As was done in §5.2.3, taking the derivatives of (158) in (159)though (161), around the equilibrium point $\boldsymbol{F}=\mathbf{0}$ ($\therefore\frac{d\boldsymbol{F}}{d\mathbf{x}}=\mathbf{0}$) as done in (109)-(110) and assembling the resulting components into matrix form gives:

$$\begin{bmatrix}\frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}} & \frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_m}} & \frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_s}} & \frac{\partial\boldsymbol{F}_{\Omega}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}\\ \frac{\boldsymbol{K}_{\partial\Omega_{\mathrm{m}}}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}} & \frac{\boldsymbol{K}_{\partial\Omega_{\mathrm{m}}}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{m}}}} & \frac{\boldsymbol{K}_{\partial\Omega_{\mathrm{m}}}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_s}} & \frac{\boldsymbol{K}_{\partial\Omega_{\mathrm{m}}}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}\end{bmatrix}\begin{Bmatrix}\frac{d\boldsymbol{\mathcal{X}}^{\Omega}}{d\mathbf{x}}\\ \frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_m}}{d\mathbf{x}}\\ \frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_s}}{d\mathbf{x}}\\ \frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}{d\mathbf{x}}\end{Bmatrix}=\begin{Bmatrix}\mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\\ \mathbf{0}\end{Bmatrix} \tag{163}$$

Where:

$$\frac{\boldsymbol{K}_{\partial\Omega_{\mathrm{m}}}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}}=\frac{\partial\boldsymbol{F}_{\partial\Omega_{\mathrm{m}}}}{\partial\boldsymbol{\mathcal{X}}^{\Omega}}+\frac{\partial\left(P^T\boldsymbol{F}_{\partial\Omega_{\mathrm{s}}}\right)}{\partial\boldsymbol{\mathcal{X}}^{\Omega}} \tag{164}$$

$$\frac{\boldsymbol{K}_{\partial\Omega_{\mathrm{m}}}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{m}}}}=\frac{\partial\boldsymbol{F}_{\partial\Omega_{\mathrm{m}}}}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_m}}+\frac{\partial\left(P^T\boldsymbol{F}_{\partial\Omega_{\mathrm{s}}}\right)}{\partial\boldsymbol{\mathcal{X}}^{\partial\Omega_m}} \tag{165}$$

$$\frac{\boldsymbol{K}_{\partial\Omega_{\mathrm{m}}}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_s}} = \frac{\partial \boldsymbol{F}_{\partial\Omega_{\mathrm{m}}}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_s}} + \frac{\partial\left(P^T \boldsymbol{F}_{\partial\Omega_s}\right)}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_s}} \tag{166}$$

$$\frac{\boldsymbol{K}_{\partial\Omega_{\mathrm{m}}}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_p}} = \frac{\partial \boldsymbol{F}_{\partial\Omega_{\mathrm{m}}}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_p}} + P^T \frac{\partial \boldsymbol{F}_{\partial\Omega_{\mathrm{s}}}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_p}} \tag{167}$$

Allowing that the boundary function as a function of the master DOFs $\boldsymbol{\mathcal{X}}^{\partial\Omega_m}(\mathbf{x})$ and the offset $\boldsymbol{D}(\mathbf{x})$ such that:

$$\boldsymbol{\mathcal{X}}^{\partial\Omega_s}(\mathbf{x}) = f_{ms}\left(\boldsymbol{\mathcal{X}}^{\partial\Omega_m}(\mathbf{x}), \boldsymbol{\mathcal{X}}^{\partial\Omega_p}(\mathbf{x}), \boldsymbol{D}(\mathbf{x})\right) \tag{168}$$

Taking the derivative of (168) with respect to $\mathbf{x}$ and knowing $\frac{\partial f_{ms}}{d\boldsymbol{\mathcal{X}}^{\partial\Omega_m}} = \boldsymbol{P}$ yields:

$$\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_s}}{d\mathbf{x}} = \boldsymbol{P}\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_m}}{d\mathbf{x}} + \frac{\partial f_{ms}}{\partial \boldsymbol{D}}\frac{d\boldsymbol{D}}{d\mathbf{x}} \tag{169}$$

Substituting (169) into (163) yields the system:

$$\begin{bmatrix} \frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\Omega}} & \frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_m}} + \frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_s}}\boldsymbol{P} & \frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_s}} & \frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_p}} \\ \frac{\boldsymbol{K}_{\partial\Omega_{\mathrm{m}}}}{\partial \boldsymbol{\mathcal{X}}^{\Omega}} & \frac{\boldsymbol{K}_{\partial\Omega_{\mathrm{m}}}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_{\mathrm{m}}}} + \frac{\boldsymbol{K}_{\partial\Omega_{\mathrm{m}}}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_s}}\boldsymbol{P} & \frac{\boldsymbol{K}_{\partial\Omega_{\mathrm{m}}}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_s}} & \frac{\boldsymbol{K}_{\partial\Omega_{\mathrm{m}}}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_p}} \end{bmatrix} \begin{Bmatrix} \frac{d\boldsymbol{\mathcal{X}}^{\Omega}}{d\mathbf{x}} \\ \frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_m}}{d\mathbf{x}} \\ \frac{\partial f_{ms}}{\partial \boldsymbol{D}}\frac{d\boldsymbol{D}}{d\mathbf{x}} \\ \frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}{d\mathbf{x}} \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{Bmatrix} \tag{170}$$

Since $\frac{\partial f_{ms}}{\partial \boldsymbol{D}}\frac{d\boldsymbol{D}}{d\mathbf{x}}$ and $\frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}{d\mathbf{x}}$ are knowns and noting the $\boldsymbol{P} = \boldsymbol{P}\left(\boldsymbol{\mathcal{X}}^{\partial\Omega_p}\right)$, the system can be rearranged to give:

$$\begin{bmatrix} \frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\Omega}} & \frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_m}} + \frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_s}}\boldsymbol{P} \\ \frac{\partial \boldsymbol{F}_{\partial\Omega_{\mathrm{m}}}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_p}} + \boldsymbol{P}^T\frac{\partial \boldsymbol{F}_{\partial\Omega_{\mathrm{s}}}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_p}} & \frac{\partial \boldsymbol{F}'m}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_m}} \end{bmatrix} \begin{Bmatrix} \frac{d\boldsymbol{\mathcal{X}}^{\Omega}}{d\mathbf{x}} \\ \frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_m}}{d\mathbf{x}} \end{Bmatrix} = -\begin{bmatrix} \frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_s}} & \frac{\partial \boldsymbol{F}_{\Omega}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_p}} \\ \frac{\partial \boldsymbol{F}'_{\partial\Omega_{\mathrm{m}}}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_s}} & \frac{\partial \boldsymbol{F}'_{\partial\Omega_{\mathrm{m}}}}{\partial \boldsymbol{\mathcal{X}}^{\partial\Omega_p}} \end{bmatrix} \begin{Bmatrix} \frac{\partial f_{ms}}{\partial \boldsymbol{D}}\frac{d\boldsymbol{D}}{d\mathbf{x}} \\ \frac{d\boldsymbol{\mathcal{X}}^{\partial\Omega_p}}{d\mathbf{x}} \end{Bmatrix} \tag{171}$$

Where:

$$\frac{\boldsymbol{K}_{\partial\Omega_{\mathrm{m}}}}{\partial \boldsymbol{\mathcal{X}}^{\Omega}} = \frac{\partial \boldsymbol{F}_{\partial\Omega_{\mathrm{m}}}}{\partial \boldsymbol{\mathcal{X}}^{\Omega}} + \frac{\partial\left(P^T \boldsymbol{F}_{\partial\Omega_{\mathrm{s}}}\right)}{\partial \boldsymbol{\mathcal{X}}^{\Omega}} \tag{172}$$

$$\frac{\partial \boldsymbol{F'}_m}{\partial \boldsymbol{X}^{\partial\Omega_m}} = \frac{\partial \boldsymbol{F}_{\partial\Omega_\mathrm{m}}}{\partial \boldsymbol{X}^{\partial\Omega_m}} + \boldsymbol{P}^T \frac{\partial \boldsymbol{F}_{\partial\Omega_\mathrm{s}}}{\partial \boldsymbol{X}^{\partial\Omega_m}} + \frac{\partial \boldsymbol{P}^T}{\partial \boldsymbol{X}^{\partial\Omega_m}} \boldsymbol{F}_{\partial\Omega_\mathrm{s}} + \frac{\partial \boldsymbol{F}_{\partial\Omega_\mathrm{m}}}{\partial \boldsymbol{X}^{\partial\Omega_s}} \boldsymbol{P} + \boldsymbol{P}^T \frac{\partial \left(\boldsymbol{F}_{\partial\Omega_s}\right)}{\partial \boldsymbol{X}^{\partial\Omega_s}} \boldsymbol{P} \tag{173}$$

$$\frac{\partial \boldsymbol{F'}_{\partial\Omega_\mathrm{m}}}{\partial \boldsymbol{X}^{\partial\Omega_s}} = \frac{\partial \boldsymbol{F}_{\partial\Omega_\mathrm{m}}}{\partial \boldsymbol{X}^{\partial\Omega_s}} + \boldsymbol{P}^T \frac{\partial \boldsymbol{F}_{\partial\Omega_s}}{\partial \boldsymbol{X}^{\partial\Omega_s}} \tag{174}$$

$$\frac{\partial \boldsymbol{F'}_{\partial\Omega_\mathrm{m}}}{\partial \boldsymbol{X}^{\partial\Omega_p}} = \frac{\partial \boldsymbol{F}_{\partial\Omega_\mathrm{m}}}{\partial \boldsymbol{X}^{\partial\Omega_p}} + \boldsymbol{P}^T \frac{\partial \boldsymbol{F}_{\partial\Omega_\mathrm{s}}}{\partial \boldsymbol{X}^{\partial\Omega_p}} \tag{175}$$

If the system is in a natural state of static equilibrium, then $\boldsymbol{F}_{\partial\Omega_\mathrm{s}} = \boldsymbol{0}$ and the term $\frac{\partial \boldsymbol{P}^T}{\partial \boldsymbol{X}^{\partial\Omega_m}} \boldsymbol{F}_{\partial\Omega_\mathrm{s}} = \boldsymbol{0}$. This means the consistent tangent of (171) reduces to match the linear static implementation of the MSE method as discussed in APPENDIX A (141). Again, the RHS terms $\frac{\partial f_{ms}}{\partial \boldsymbol{D}} \frac{d\boldsymbol{D}}{d\mathbf{x}}$ and $\frac{d\boldsymbol{X}^{\partial\Omega_p}}{d\mathbf{x}}$ can be computed using a finite difference method.