UNIVERSITY OF Southampton

University of Southampton Research Repository
ePrints Soton

Copyright © and Moral Rights for this thesis are retained by the author and/or other copyright owners. A copy can be downloaded for personal non-commercial research or study, without prior permission or charge. This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the copyright holder/s. The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the copyright holders.

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given e.g.

AUTHOR (year of submission) "Full thesis title", University of Southampton, name of the University School or Department, PhD Thesis, pagination

http://eprints.soton.ac.uk

UNIVERSITY OF SOUTHAMPTON

COMPUTATIONAL ENGINEERING AND DESIGN GROUP

School of Engineering Sciences

Dimensional reduction and design optimization of gas turbine engine casings for tip clearance studies

by

Felix Stanley

Thesis for the degree of Doctor of Philosophy

October 2010

The objective of this research is to develop a design process that can optimize an engine casing assembly to reduce tip clearance losses. Performing design optimization on the casings that form a gas turbine engine's external structure is a very tedious and cumbersome process. The design process involves the conceptual, the preliminary and the detailed design stages. The redesign costs involved are high when changes are made to the design of a part in the detailed design stage. Normally a 2D configuration is envisaged by the design team in the conceptual design stage. Engine thrust, mass flow, operating temperature, materials and manufacturing processes available at the time of design, mass of the engine, loads and assembly conditions are a few of the many important variables that are taken into consideration when designing an aerospace component. The linking together of this information into the design process to achieve an optimal design using a quick robust method is still a daunting task. In this thesis, we present techniques to extract midsurfaces of complex 3D axisymmetric and non-axisymmetric geometries based on medial axis transforms. We use the proposed FE modeling technique for optimizing the geometry by designing a sequential workflow consisting of CAD, FE analysis and optimization algorithms within an integrated system. An existing commercial code was first used to create a midsurface shell model and the results showed that such models could replace 3D models for deflection studies. These softwares being black box codes could not be customized. Such limitations restrict their use in batch mode and development for research purposes. We recognized an immediate need to develop a bespoke code that could be used to extract midsurfaces for FE modeling. Two codes, Mantle-2D and Mantle-3D have been developed using Matlab to handle 3D axisymmetric and non-axisymmetric geometries respectively. Mantle-2D is designed to work with 2D cross-section geometry as an input while Mantle-3D deals with complex 3D geometries. The Pareto front (PF) of 2000 designs of the shell based optimization problem when superimposed on the PF of the solid based optimization, has provided promising results. A DoE study consisting of 200 designs was also conducted and results showed that the shell model differs in mass and deflection by $<1\%$ and $<5.0\%$ respectively. The time taken to build/solve a solid model varied between 45-75 minutes while the equivalent midsurface based shell model built using Mantle-2D required only 3-4 minutes. The Mantle-3D based dimensional reduction process for a complex non-axisymmetric solid model has also been demonstrated with encouraging results. This code has been used to extract and mesh the midsurface of a non-axisymmetric geometry with shell elements for use in finite element analysis. 101 design points were studied and the results compared with the corresponding solid model. The first 10 natural frequencies of the resulting shell model deviates from the solid model by $<4.0\%$ for the baseline design, while the mass and deflection errors were $<3.5\%$ and $<9.0\%$ for all 101 design points.

Acknowledgements

I express my profound gratitude to my teachers, family and friends for all their help, support and motivation without which this dissertation would not have been possible. I am grateful to my mentors, Prof. Andy J. Keane and Dr. Ivan I. Voutchkov for providing me with this wonderful opportunity to pursue a Ph.D and for their patient demeanor, constant encouragement and excellent guidance. I have always been in awe of their in-depth understanding and knowledge of the subject matter and feel honoured to have worked under their guidance. I would like to thank Dr. Ivan again for his help with the administrative support during the course of this research program.

I would like to thank Robert Fox, Richard Golder, Mike Benison and Tony Stocks at Rolls-Royce Group Plc for providing the support needed to enhance the quality of this work. I thank all my friends, especially Saleem and Surya for making my life outside of work relaxing and enjoyable. I greatly appreciate the financial support that I received from Rolls-Royce Group Plc, Civil Aerospace, UK.

Last, but not the least, for the unconditional love, emotional understanding and constant support of my parents Stanley and Shalat, my brother Frederick, sister-in-law Dhanya, my wife Bindu and son Ronan, I thank them. I take this opportunity to extend my heartfelt gratitude to Bindu and Ronan for standing by me during these three years of thick and thin and for being a pillar of hope and support in my time of need. To you, I dedicate this work!

Contents

List of Figures

List of Tables

Declaration of authorship

I, **Felix Stanley**, declare that the thesis entitled **Dimensional reduction and design optimization of gas turbine engine casings for tip clearance studies** and the work presented in the thesis are both my own, and have been generated by me as the result of my own research. I confirm that:

⋆ this work was done wholly or mainly while in candidature for a research degree at this University;

⋆ where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;

⋆ where I have consulted the published work of others, this is always clearly attributed;

⋆ where I have quoted from the work of of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;

⋆ I have acknowledged all main sources of help;

⋆ where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;

⋆ part of this work is under review as:

1. F Stanley, I. I. Voutchkov and A. J. Keane, "Gas turbine casing design optimization using the medial axis transform", *Computer Aided Design in Elsevier.*

Signature: ________________

Date: ________________

List of acronyms

MAT	Medial Axis Transform
CoG	Center of Gravity
CAD	Computer Aided Design
CAE	Computer Aided Engineering
SFC	Specific Fuel Consumption
FE	Finite Element
HM	Hypermesh
CAT	Chordal Axis Transform
Hexa	Hexahedral elements
Tet	Tetrahedral elements

Chapter 1

Introduction

1.1 Motivation

The flow of information between the conceptual design and the preliminary design phase is difficult in a large technologically driven company like Rolls-Royce. There is always more than one instance when the needs of one design team is over-ridden by the needs of another team, e.g. the stress team would want to add more material onto the high stress region while the weights group would want to reduce the weight of the engine and simultaneously the costing group will have limitations on the budget thereby trying to reduce the cost incurred in designing/manufacturing the engine. Work done by Boart *et al.* [1] shows how automating preprocessing activities can reduce lead time while improving quality. Engineering activities focus on the studied phenomena and an increased number of concepts result in improved ground for decisions. Rajagopal in [2] discusses how knowledge based engineering can be used to design the load path of a gas turbine engine component. The application of knowledge based engineering techniques while creating FE models has been discussed in [3] by Pinfold *et al.*

Performing design optimization on the casings that form a gas turbine engine's external structure is a very tedious and cumbersome process. The design process involves conceptual design, preliminary design and detailed design stages. The redesign costs involved are high when changes are made to the design of a part in the detailed design stage. Normally a 2D configuration is envisaged by the design team in the conceptual design stage. Engine thrust, mass flow, operating temperature, materials and manufacturing processes available at the time of design, mass of the engine, loads and assembly conditions are a few of the many important variables considered in designing an aerospace component. These tasks need a large amount of knowledge which is held by different groups/ teams within the designing company. The linking together of this knowledge into the design process to achieve an optimal design with the earliest possible robust method is still a daunting task. Phan *et al.* have developed a process around Pacelab [4] for formulation and implementation of an aircraft - system - subsystem interrelationship model as described in [5].

Engineering industries today have a lot of design activities that revolve around CAD and CAE environment. CAD and CAE as used in large aerospace companies are two different systems, each evolved from different backgrounds to cater to different needs which have resulted in different data representations. Detailed geometric models are created using CAD engines based on a datum design. These geometries are then imported into CAE preprocessors for meshing and finite element analysis. Automating the preprocessing of a CAD geometry for FE analysis however has its problems. The principal problem that faces academia and industry alike when moving files between the various CAD/CAE engines is loss of information which requires manual intervention to fix. Commercially available tools like CADfix [6] repair the geometry from a CAD engine before exporting to a CAE engine without loss of surfaces or curves. Beall *et al.* in [7] address the specific issues related to accessing CAD geometry for mesh generation.

Design search and optimization of a task of this nature needs to allow for a wide range exploration over many possible combinations and configurations of all the relevant variables and modules. Therefore, a work flow needs to be constructed and customized to a specific need that can sequentially assemble all the processes into a program which can then be looped and tested as a design process. Integration of these tools can lead to the automation of some of the iterative processes involved in the design process. Much work has been done in the industry on the integration of the various parameters involved in the manufacturing side of the design process through the VERDI [8] project (Virtual Engineering for Robust manufacturing with Design Integration). Work has also been carried out by Voutchkov *et al.* [9] on the modelling and design of gas turbine engines during the VIVACE [10] (Value Improvement through a Virtual Aeronautical Collaborative Enterprise) project executed at the University of Southampton. Work on a CAD based parametric designer of gas turbine engine cross-section for use in multi-disciplinary optimization has been carried out by Dye *et al.* in [11]. On similar principles, an optimization work flow needs to be created to optimize a gas turbine engine structure that satisfies design objectives raised by concerns such as weight, stress, stiffness and fuel efficiency. This research program concentrates on software methods that will allow multiple goals in an aeroengine structural design to be studied in an efficient way.

1.2 Background

A twin spool aeroengine consists of a fan, compressor, combustor, high pressure turbine, low pressure turbine and an exhaust nozzle. A typical Rolls-Royce three spool engine cutout is as shown in Fig. 1.1 and has an intermediate spool (shaft) that drives the intermediate compressor and turbine stages. The rotating components e.g. the blades, disks, shafts and other accessories are assembled with bearing housings through a combination of ball and roller bearings located along the engine axis. The entire rotating assembly is housed in thin cylindrical shell structures which in turn are attached to the engine frames. The thrust generated by the engine is transmitted to the aircraft through mounting blocks in the frames.

The engine structural assembly consisting of frames, casings and mounts enables an engine to maintain a rigid shape within specified tolerances, while withstanding enormous internal pressures and temperatures. In addition to providing the needed strength, rigidity and heat resistance, engine structures need to be as light as possible to meet low weight-to-thrust ratio. An increase in engine performance is achieved by minimizing the clearances between the rotor and stator casings. It is therefore important to be able to predict accurately the axial and radial displacements of an engine's structural assembly. Chandrasekaran *et al.* [12] and Funatogawa *et al.* [13], discuss how FE analysis of the whole engine is conducted to obtain the displacements of the engine assembly for various flight conditions. The ovality (out of roundness) of the casing is detrimental to tip clearance. The overall deflection of the entire casing structure for a downward gravity load is as shown in Fig. 1.2 (reproduced from [13]). It can be seen from this figure that for a downward gravity load, the engine bends, opening up the tip clearances at the top and closing at the bottom.

The efficiencies of axial flow intermediate and high pressure compressors (IPC and HPC), are strongly affected by blade tip clearances. Fig. 1.3 shows such a blade tip clearance for an axial flow compressor. The greater this clearance, the more over tip leakage is created which results in subsequent loss of capacity and performance. The difference between rotor and casing movement during operation is influenced by both axisymmetric relative movements, e.g., centrifugal growth of the rotor/blade, thermal expansion, 'boiler' pressure, axial movements, etc., and asymmetric shape changes of the casing. Asymmetric deflections of the casing, in

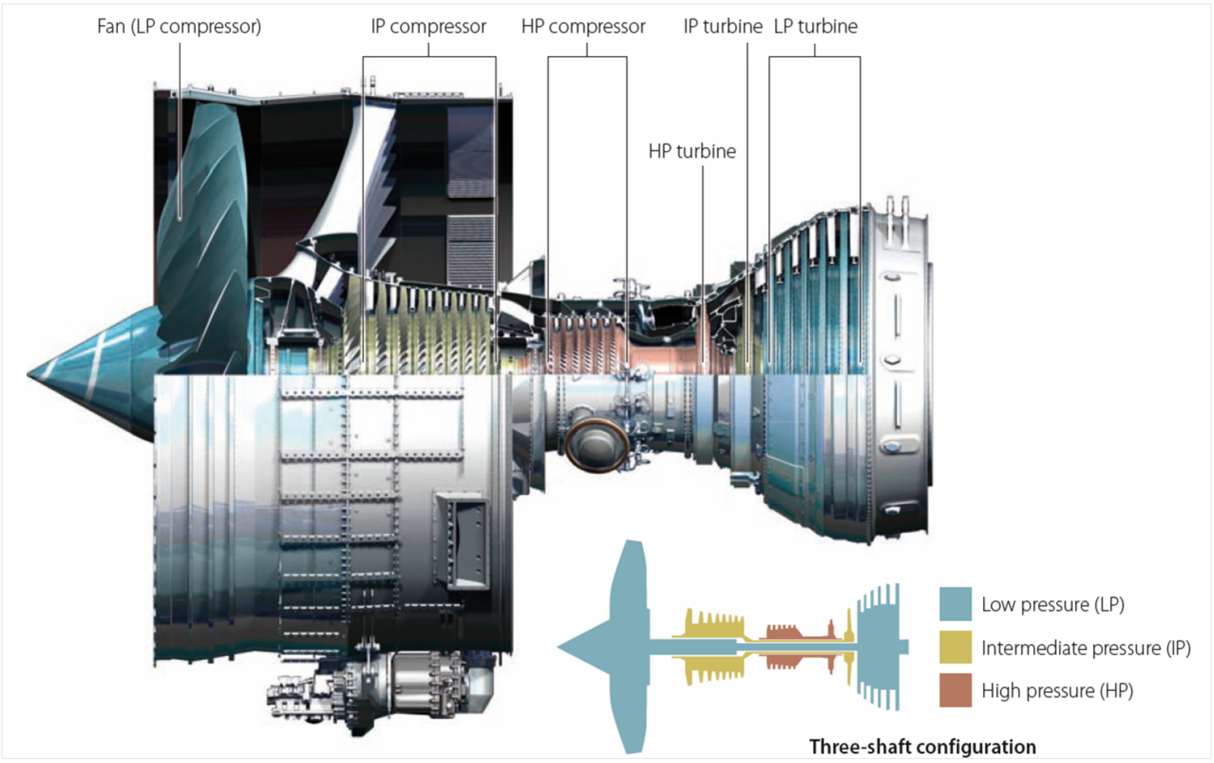


FIGURE 1.1: A typical Rolls-Royce three spool engine

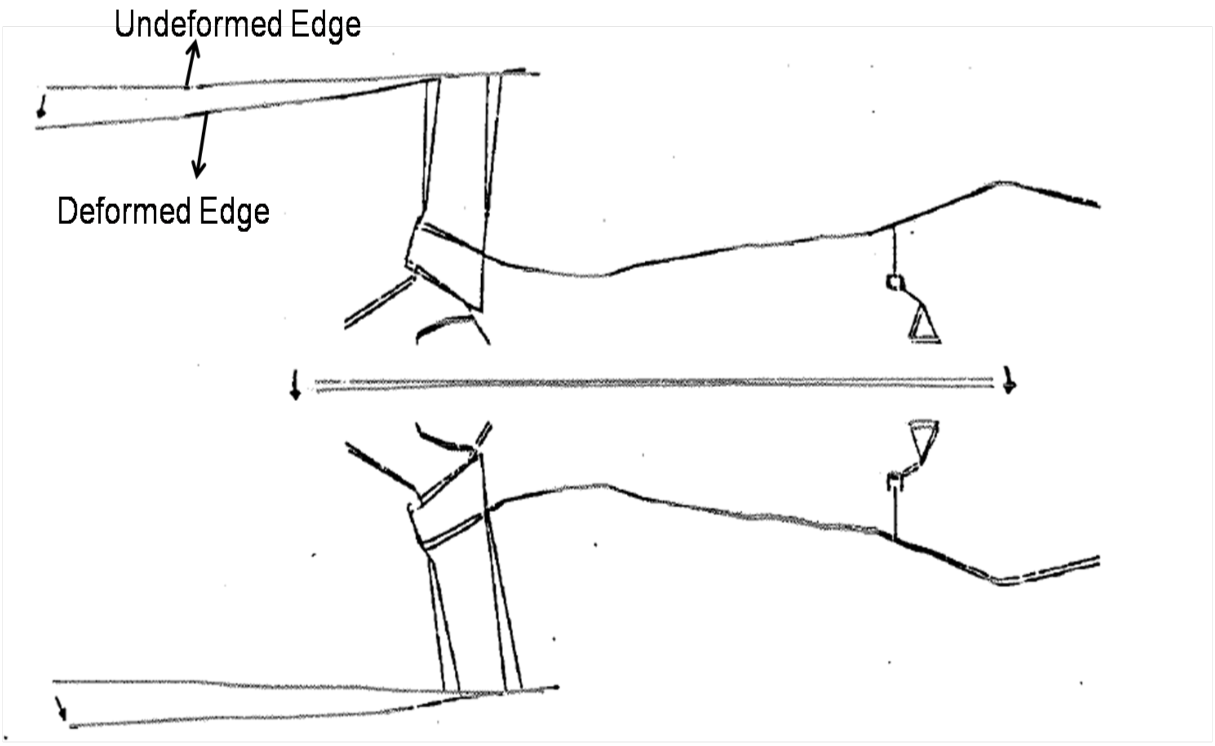


FIGURE 1.2: Changes in tip clearance along the engine's length for a downward gravity load

turn, are created by large thermal masses, such as bleed ports and/or by structural loading on the engine and non symmetric thermal loading. Isothermal structural loads are generated by thrust, inertia, gyro and aerodynamic loading resulting from aircraft manoeuvre and gust loads observed during operation.

Aeroengines are designed for their typical operating conditions, i.e. when the engine is hot. The engine's dimensions are different when it is cold and these differences must be allowed for in fabricating the parts. Therefore, there is a need for preparing the cold lay-up for the engine. If the engine structure is poorly designed without the displacement analysis of the complete engine assembly, a significant engine performance loss can result from blade rubs caused by structural deflections.

The ability of the engine casings to remain accurately circular under structural loading improves the efficiency of the compressors. Any reduction in blade tip clearance can be represented as a ratio of each individual stage clearance to blade height, summed across all the stages as a Root Mean Squared term (RMS). This RMS can be related to compressor efficiency, which is traded as an improvement to fuel efficiency of the engine i.e. a reduction in SFC (Specific Fuel Consumption). Typically for a large jet engine the effect of gaining 1% SFC on a twin engine aircraft would mean the capacity to carry 12 additional passengers or the aircraft range could be extended by 160 miles. This multiplied by the many flights an aircraft can make during its life, shows how tip clearances can influence the overall performance of the aircraft. Throughout a flight cycle the compressor casings would ideally be required to remain circular under the twin demands of supplying the required thrust and resisting aircraft loading. However, in practice this is never achieved since there is always an isothermal load that affects the casing shape; even the influence of normal gravitational load changes the casing shape. Fig. 1.4 shows how a casing may deflect under isothermal loading. The stiffness of the structure connecting the casings will depend on the asymmetric deflections of the compressors and greatly affect the overall SFC of the engine. To alleviate any eccentricities the structure could be made very stiff. This will invariably increase the weight and possibly the cost, which can also be related to SFC and using this as a common currency between tip clearance improvement and weight gain, objectives can be set for an optimisation study.

The tip displacement criterion as mentioned by DeCastro *et al.* [14] is based on the typical combined deformations of turbine components. This quantity is related to the clearance gap $g(t)$ as follows:

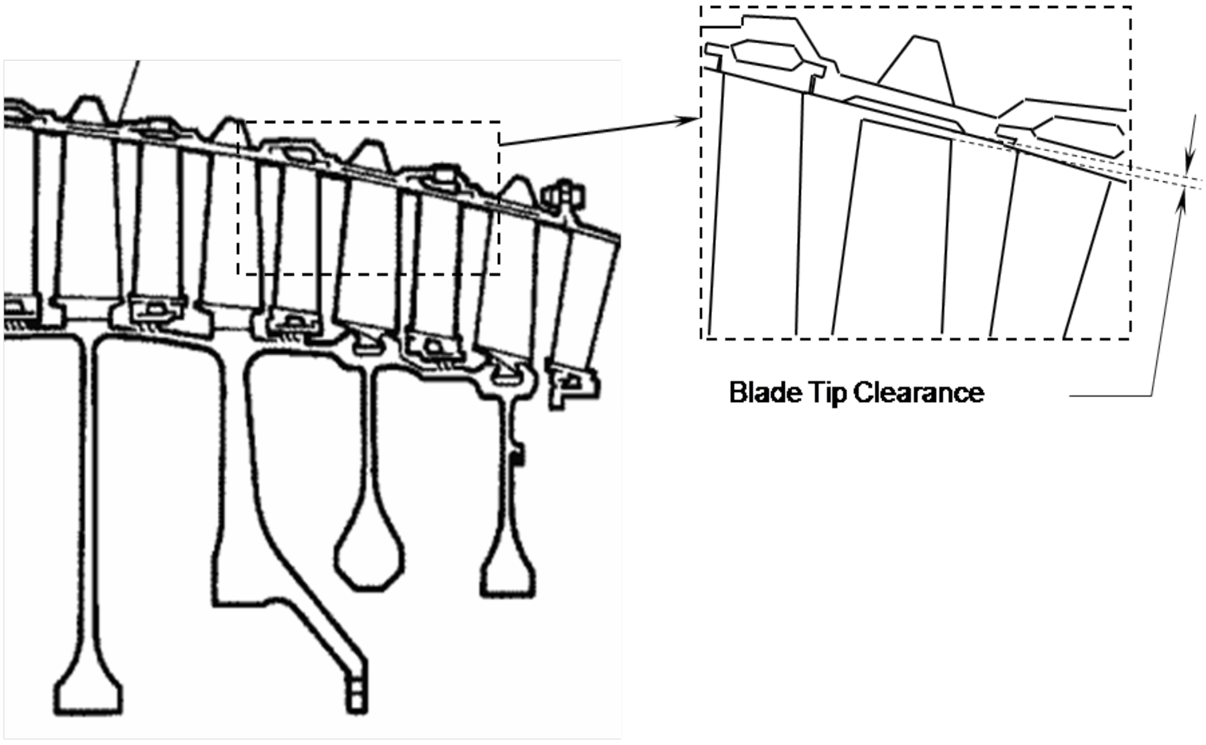


FIGURE 1.3: Intermediate pressure compressor and tip clearance

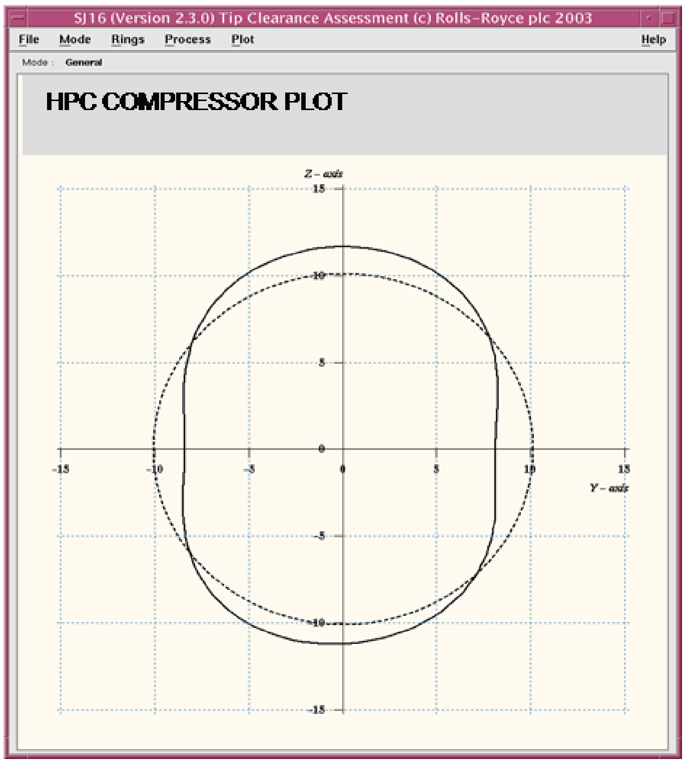

SJ16 (Version 2.3.0) Tip Clearance Assessment (c) Rolls-Royce plc 2003
File Mode Rings Process Plot Help
Mode : General
HPC COMPRESSOR PLOT
Z – axis
Y – axis
15
10
5
0
-5
-10
-15

FIGURE 1.4: Showing asymmetric deflection (solid line) of a compressor casing

$$g(t) = g_{cold} + \delta_{shroud}(t) + \delta_{case}(t) - \delta_{rotor}(t) \tag{1.1}$$

where g_{cold} is the cold clearance and $\delta_{case}(t)$, $\delta_{shroud}(t)$ and $\delta_{rotor}(t)$ are the outward thermal and mechanical deformations of the case, stator shroud and rotor assembly relative to the cold clearance respectively. As mentioned in [14], a clearance reduction of 10mils (1mil = One thousandth of an inch i.e. 0.0254 millimeter) results in an overall approximate SFC improvement of 1% over the mission. Optimization of the casing stiffness aims to reduce the tip clearance when designed for the worst loading conditions.

Design search and optimization is a term used to describe the application of a formal optimization software to the problem of engineering design. Design search indicates that the solution for a complex problem is not obvious, but can be looked for by an exploratory process which has no fixed end points i.e, design search is the over arching process while optimization is a tool. Rather, in engineering components (aerospace, automotive) the design team is faced with multiple and competing goals, various constraints and design tools that are often not integrated with each other, as explained by Keane *et al.* in [15].

In the aerospace industry, computational structural design generally makes use of:

⋆ A geometry manipulation engine that is primarily used for the purposes of generating geometric models. Some of the very well known commercial CAD engines in use in the industry today are Catia [16], Siemens-NX [17], Pro Engineer [18] and SolidWorks [19].

⋆ A meshing process that uses the geometry from CAD software to create a finite element model (FE model). The import of the geometry can be in different forms e.g. STEP, IGES, Parasolid or part files from Siemens-NX. These finite element models can then be used in a structural analysis.

⋆ A collection of finite element solvers are then used to analyze the finite element models generated. Some of the commercially available structural solvers used in the industry today are ANSYS [20], NASTRAN [21], LS-Dyna [22] etc.

⋆ A post-processing tool that can interpret the results obtained using these solvers needs to be used to understand the structural behaviour of the component being studied. Generally, all the solvers have their own preprocessors and post processors which meet each of their specific requirements.

- ⋆ These results can then be used to perform an optimization study using a tool like iSIGHT [23].
- ⋆ A process integration tool also needs to be used to bring together all the above mentioned tasks. The integration of these tasks is generally done in batch mode thereby allowing for a fully automated execution.

Various Finite Element (FE) approaches and schemes have been looked into in order to model the tip clearance variation problem. In this research program, tip clearance variations arising due to geometric modeling (inclusion/exclusion of features such as fillets, holes, bosses etc), FE approximation (element choice) and definition of loads and boundary conditions within the FE model play an important role in the output of the analysis. The choice of method eventually adopted depends upon four governing factors, i.e. parametrization of the geometric model using sufficient feature representation, the ease with which the FE model can be created in batch mode, the solution time required for each optimization run and finally the optimization scheme being used to arrive at the objectives defined, as mentioned by Lee *et al.* in [24].

Based on the work done by Leary *et al.* in [25], the problem of reducing the computational cost of expensive function optimization is tackled through the use of approximations to the expensive function. With these techniques as basis, a 3D FE model representing the 3D geometry with the necessary geometric features (High Fidelity model) can be created. Simultaneously a simpler FE model (Low Fidelity model) can also be created. Combining the low fidelity model with the more accurate but expensive high fidelity model can provide a good combination of high accuracy and low cost. A low fidelity model in this case would be a complete shell model or a combination of 1D and 2D elements which represent the solid model's stiffness appropriately. However, the time required to create these low fidelity models depends on whether they are created manually or by automation. If built manually, they can be cumbersome and time consuming while there is no existing code at the time of writing to automate the creation of a low fidelity model. However, work has been done by Young *et al.* [26] to automate the generation of 3D FE data based on medical imaging data.

In this thesis case studies to understand each of the above factors have been carried out. This has been done in order to find the best work flow to model tip clearance variation. The choice of element to be used to create a simplified FE model is based on the level of dimensional reduction of the 3D geometric model. However the level of complexity of a geometric model depends on the algorithms used to create the dimensionally reduced geometric model. For example too many complex features can cause midsurface or medial axis generating algorithms to fail.

1.3 Scope and layout of the thesis

The main scope of this work is to propose a method to obtain the optimum stiffness of a gas turbine casing assembly to realize reduced SFC and/or weight with an insight into how the structure performs. In order to optimize the stiffness of a part or an assembly, a set of geometries have to be searched to find a set of best designs that could help achieve this objective (optimum stiffness). The geometry is modified/updated based on the design objectives that have been set. The design objectives in this research work could be all or a combination of stiffness, weight, center of gravity, tip clearance and SFC. Changes in stiffness change the deflection of a component which results in change in tip clearance and SFC, i.e. increase in stiffness, reduces deflection hence reducing change in tip clearance and SFC. However, an increase in stiffness should be achieved with no or minimal addition to weight of the component. Deflection and weight are two objectives that need to be minimized to arrive at a set of best designs to achieve reduced SFC.

Such a design optimization problem generally comprises various stages, i.e. dimensional reduction of the geometry, multi-fidelity approaches for optimization and multi-objective optimization. A 3D geometric model needs to be dimensionally reduced to a 2D or 1D geometric model before creating the finite element model in order to reduce the time taken to analyze the finite element model using solvers like ANSYS or NASTRAN. Generally the analysis times for 3D FE models created using tetrahedral elements could be at least 10 times more than FE models created using shell or beam elements. The lesser the analysis time, the more extensive the design search will be with the available computer resources. However, the main concern when dimensionally reducing a complex 3D structure is the loss of stiffness, change in mass, center of gravity in the reduced model and the effort required to generate these reduced models. These important parameters have to be matched within a small tolerance when compared to the 3D model before use in any design search process. This thesis attempts to find a dimensionally reduced geometric model that can be built automatically for use in optimization without deviating too much in stiffness, mass or center of gravity from the original 3D geometric model. Current trends and issues in automatic mesh generation are discussed by Shimada in [27].

There will always be inconsistencies when complex models are broken down into simpler representations. The difference between an original model and a dimensionally reduced model depends on the level of approximations made during the creation of the reduced model. The result of these approximations is the variation in stiffness, mass and center of gravity of a part when compared to the original model. This variation in the results cannot be avoided but can be minimized. For simple components, the approximations will not result in major changes in the reduced model's behaviour. However, when the model being dimensionally reduced gets complicated, the assumptions and approximations will lead to variations in results. For this research program, when comparing the stiffness, mass and center of gravity of the reduced model with the original solid model, a deviation of $<10\%$ is considered acceptable. Components that form the load bearing outer casings of an aeroengine have been considered for this work. This includes components like the intermediate case or the rear inlet case which are complex 3D non-axisymmetric geometries.

The remainder of the thesis is arranged as follows;

- ⋆ In Chapter 2, the existing literature on dimensional reduction process and their pros and cons are discussed.
- ⋆ Chapter 3 presents a case study conducted using a commercially available software with midsurface extraction capabilities. The code is used to build a midsurface based shell model and results compared with a solid model. The decision to represent 3D engine casings using shell elements was made based on work done in this chapter.
- ⋆ Chapter 4 looks at different modeling techniques to connect the discontinued midlines that are noticed towards the end of Chapter 3. If these discontinuties are not dealt with appropriately, the load path definition and the overall stiffness of the part will not be representative of the 3D model.
- ⋆ Chapter 5, a multiobjective optimization workflow for a simple stepped beam is set up. This exercise was instrumental in defining the requirements for setting up an optimization workflow in batch mode.
- ⋆ Based on the work done in Chapter 4 and 5, a code has been developed to extract midlines of 2D geometry without discontinuties and use it for design optimization in Chapter 6.

- ⋆ Chapter 7 deals with representing non-axisymmetric 3D geometries as midsurface based shell models. A code has been developed to achieve this dimensional reduction and in conjunction with work done in Chapter 6 can be used to build a complex system of models that represent a full gas turbine engine finite element model. A design of experiments survey of the resulting structural model is used to test the approach.
- ⋆ Chapter 8 summarizes the major findings and conclusions of this work. Interesting avenues of future research are also proposed.

Chapter 2

Dimensional Reduction of 3D models - Literature Survey

The behaviour of a structural component is influenced by factors such as geometry, distribution of loads, interaction with adjoining structures, material, etc. The behaviour of components can be very complex and to simulate this behaviour, complicated FE models need to be created. In design optimization, the need for complex models occurs when high fidelity models need to be used. The automated production of high fidelity 3D FE meshes using tetrahedral elements is now relatively well developed. Such models are however very large and time consuming to solve. Therefore, once the high fidelity model is in place, a low fidelity model can also be created which maybe useful for optimization purposes. Many techniques to reduce the computational cost of expensive structural optimization problems are discussed in [28], [29] and [30]. To do this, a low fidelity model needs to be generated from the geometric model retaining all the essential quantities that define the parameters and goals being optimized. The purpose of dimensional reduction of a complex 3D model is to basically reduce the analysis time taken by an FE solver. This allows for a larger number of competing designs to be studied.

Idealization, i.e. representing a component in its simplest form, is not unique to the FE method. The different elements available within the FE domain are often trying to idealize a 3D structure into a 1D or 2D or mixed dimensions. For instance, to idealize an ‘I’ section steel beam as a 1D beam means defining a beam element with appropriate section properties. The overall behaviour of the beam is adequately represented by this model, although detailed stresses such as those at joints in the real structure will not be correctly identified. There are three major approaches to geometric idealization:

- ⋆ Detail removal: The removal of certain geometric features whose effect on the results of the analysis is relatively minor.
- ⋆ Dimensional reduction: The removal of physical dimensions from the governing equations.
- ⋆ Symmetry: Taking advantage of symmetries in the geometry and the boundary conditions to reduce the mathematical size of the problem.

The advantage a symmetric geometry provides has been considered in this work because the geometry which forms the load bearing cases in an aeroengine can be idealized as axisymmetric or cyclic symmetric (as for Intermediate casing) when holes and bosses are ignored. However, these models can also be dimensionally reduced using midsurfaces. The mid section of a geometry's outer and inner surface is called the midsurface. When a 2D cross-section geometry is being considered, the midsection would be referred to as midlines. Several element formulations are available in the modern commercial FE solvers. e.g. the axisymmetric, plane, shell, shell-solid elements in ANSYS. These element types tend to produce more computationally efficient models, hence reducing analysis time and cost. But finite element model generation currently requires extensive manual interventions during the modeling process in order to use these elements for analysis. More powerful tools still need to be developed to enable the conversion of existing solid models to analysis models of appropriate complexity as mentioned by Armstrong *et al.* in [31] and [32]. A case study for the same is explained in Chapter 3.

There are numerous methods/techniques to mesh a geometry and update these geometries based on changes made later on in the design stage. In a simulation driven design process, the flow of information from one process to another, i.e. from CAD to CAE needs to be seamlessly integrated. Work in this area is of interest in the academic world. There are not many tools in the current CAD systems that allow for seamless integration of CAD and CAE, but attempts have been made by CAE giants like ANSYS into developing tools like "ANSYS Design Space" [33]. Such tools allow the designer to perform a preliminary analysis of the component that has been designed and take decisions based on their form (shape) and fit (how it fits to other parts). However, such tools cannot be easily used in a generic environment and possibly for large scale automation. Some of the techniques described below are widely used in academia and industry to dimensionally reduce a 3D model and also to update the mesh as changes are made to the geometry. There are many techniques which are prevalent in the industry as well as the academia today, which can be used to either generate or modify meshes in optimization cycles as and when the updated geometry is available. Each of these techniques have their pros and cons. However, the choice of the method that needs to be adopted tends to be specific to each task and cannot be easily generalized. One meshing technique might be very suitable for one type of geometry while it may not be the best for another. A few of the more common mesh generation/manipulation techniques are discussed in the following sections. A choice is made to use several of these methods for case studies as explained in Chapters 3 and 4.

2.1 3D mapping of an FE mesh and mesh sweeping using Cooper Algorithm

The 3D mapping of an FE mesh is widely practiced in the medical fraternity. This technique is particularly useful in orthopedic biomechanics where the 3D meshing of a human bone tends to be a very cumbersome and laborious process. By using this method, an already existing FE mesh can be superimposed and matched to another geometry as mentioned by Couteau in [34]. However, another technique of sweeping/extruding to create a 3D mesh using the 'Cooper Algorithm' is discussed by Miyoshi *et al.* in [35]. It can also be noted that for this method to work, the geometry should be broken down into sweepable volumes; such an assembly of sweepable volumes is shown in Fig. 2.1, reproduced from [35]. If one of the faces of a volume can be meshed and then extruded along a path, this geometry can be considered a mappable geometry. The two geometries shown in the figure consist of such sweepable sections. A major aspect of this method is that the mesh topology and its element count is not altered. However, its usefulness when handling geometry with complicated fillets is questionable.

HyperMorph, a module within HyperMesh (HM), is capable of performing 3D mapping of meshes. The mesh matching algorithm developed in [34], mesh created using the multi-axis Cooper Algorithm and Hypermorph are particularly useful when a geometry is scaled up or down without changing its central location. This method cannot be used when major changes in the geometry are made, e.g. addition/ removal of flanges in a fan case, addition/removal of struts in an intermediate case. Another major disadvantage is that it needs an initial mesh to perform the mesh matching. In an optimization cycle, the interest is in trying to generate an

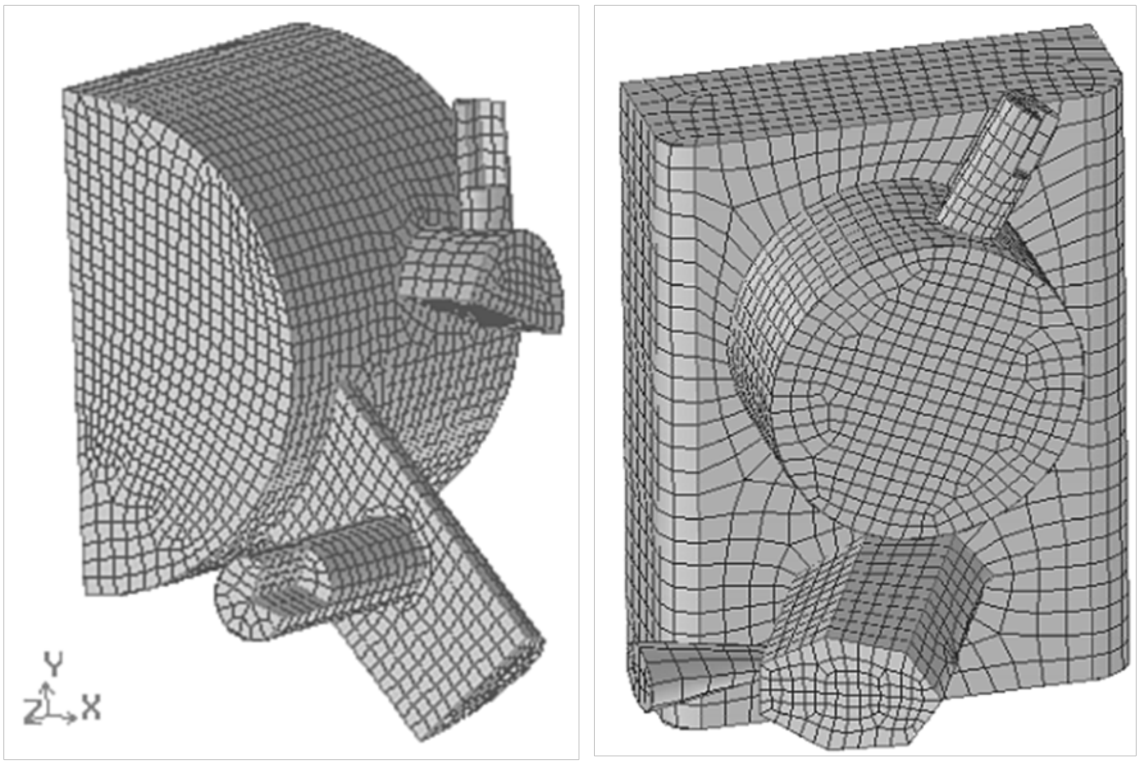


Figure 2.1: 3D mesh mapping and morphing

FE model which represents the actual structures with reduced elements and complexity. The mesh matching technique does not reduce the element count nor the complexity of the model. Rather it fits the existing mesh to scaled up/down geometries by retaining the same element and node count. Consequently, mesh matching/morphing is not useful for the research work that is being carried out in this thesis since the main aim is to reduce complexity and speed up the design search process.

2.2 Mixed dimensional coupling

In many FE analysis models there are usually regions that are particularly ideal candidates for dimensional reduction. The dimensional reduction technique relies on idealizing a 3D model into a model comprising a mix of 3D, 2D and 1D elements as shown in Fig. 2.2 (reproduced from the VIVACE reports [36]). In order to capture stress concentrations at critical features, it would be desirable to combine the reduced dimensional element types with higher dimensional elements in the whole global model. It is therefore, important to integrate into the analysis some scheme for coupling the element types that conform to the governing equations e.g. Elasticity/ plasticity. The greatest advantage of this technique is that the complex 3D model can be reduced to a simplified FE model based on engineering judgment and experience.

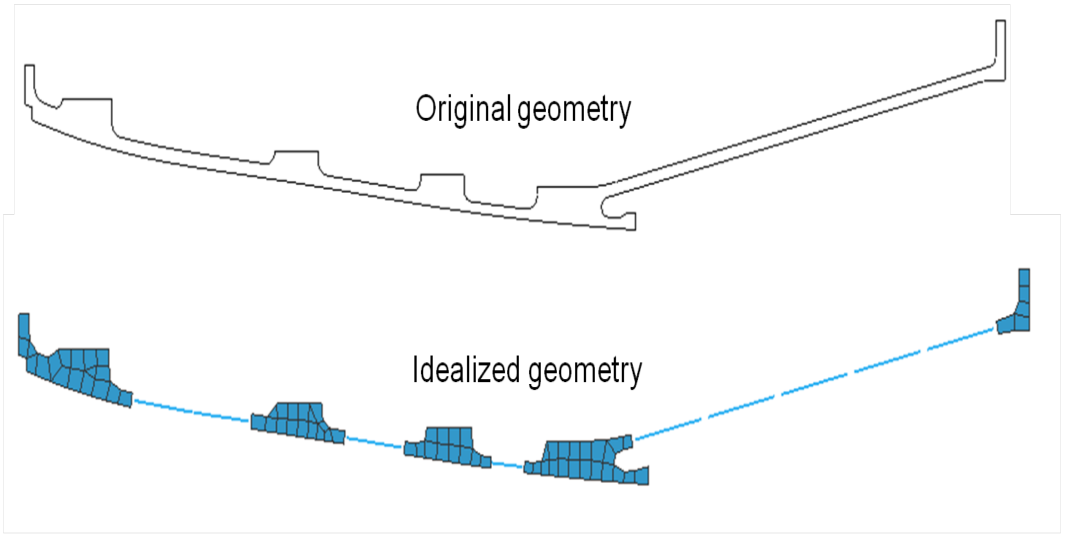


FIGURE 2.2: Idealization of an axisymmetric geometry using mixed dimensional couping

The general approach in a mixed dimensional coupling method is to try and connect the degrees of freedom of the various element formulations that have been used to generate an FE model. This is done by equating the work done on either side of the mixed dimensional interface. This is achieved by identifying the assumed variation of the stresses given by either beam, plate/ shell theory over the cross section of the interface to the solid. From this, multi-point constraint equations can be generated which provide compatible relationships for the nodal degrees of freedom between different element types. A detailed approach developed for this purpose is described by Armstrong *et al.* [37], [38], [39], Monaghan *et al.* [40] and by Bornival *et al.* in [41].

Coupling of beams and shells or coupling of shells and solids are probably two of the most commonly used mixed dimensional coupling techniques. A fan case structure, like the one discussed in Chapter 3, would consist of beam elements representing all the flanges and shell elements representing the casing. The spigot fits and radial fits, if any, are then modelled using beams which is a very cumbersome process. This is because a beam element would need to be offset appropriately to simulate these fits. Another major disadvantage of this technique is the many beam section properties that need to be generated to represent each of the flanges that constitute a structure like a full engine model. The coupling of beam and shell element however, is not very complicated when compared to the beam and solid or shell and solid combination. This is because the beam and shell elements both share the same degrees of freedom and are idealized as shown in Fig. 2.3 (reproduced from the VIVACE reports).

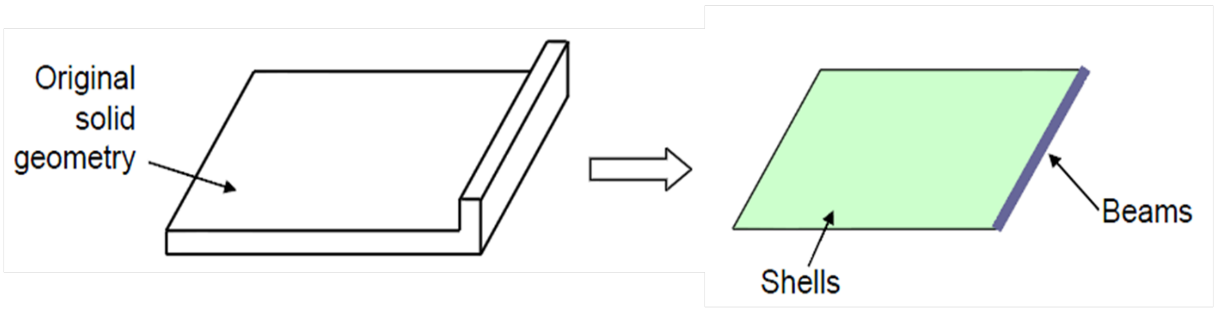
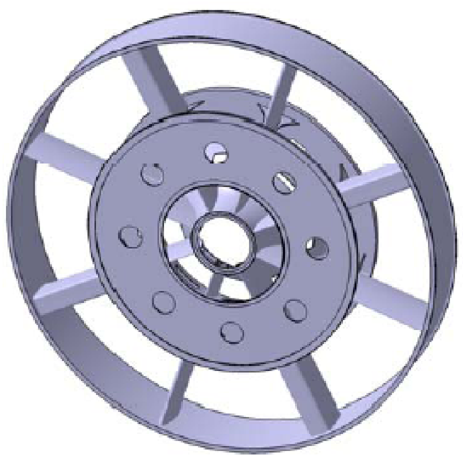
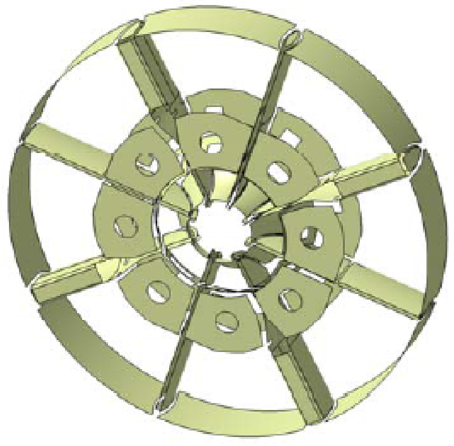


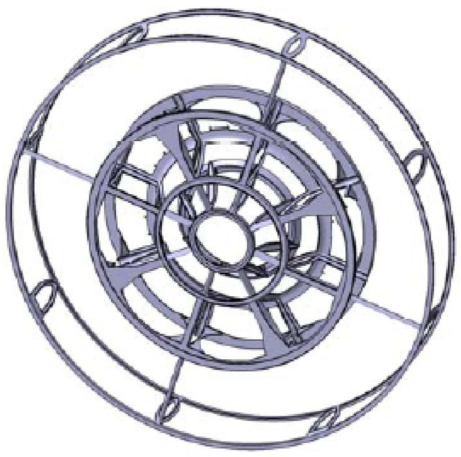
FIGURE 2.3: Idealization of a simple geometry using shell and beam elements

When coupling shell elements and solid elements, care has to be taken at all the interfaces between the two element types. This is primarily because the solid elements have only translational degrees of freedom (degrees of Freedom in XYZ directions) while the shell elements have both translational and rotational degrees of freedom (degrees of Freedom in X, Y, Z, RotX, RotY, RotZ). An example of this modeling technique is as shown in Fig. 2.4 (reproduced from the VIVACE reports). The connection between the two element types in this case is then made by the use of constraint equations. As discussed earlier, the main purpose of mix dimensional coupling is to reduce the complexity of the FE model. But as can be seen in Fig. 2.5(a) (reproduced from [36]), accurate modeling of the shell-solid interface is a non-trivial problem when handling such dimensional reductions in an automated workflow. The distance until which the solid section should be extended as shown in Fig. 2.5(b) before representing the model with shell elements still requires manual judgment and is difficult to generalize. This is one of the main reasons why this method of dimensional reduction has not been considered for this research program.

The benefits gained when an assembly of complex structures as in a full aeroengine model is automatically converted from a 3D model to a simpler beam and shell model is questionable. This is because the process for creating a mixed dimensional model needs to be automated. The way in which these algorithms decide the boundaries for the 3D, 2D and 1D elements are based on the thickness or the variation of stress function over any cross section. Models which are subjected to complex loading and boundary conditions and with


(a) The original 3D solid model


(b) Regions that can be represented with shell elements


(c) Regions that need to be retained as solids

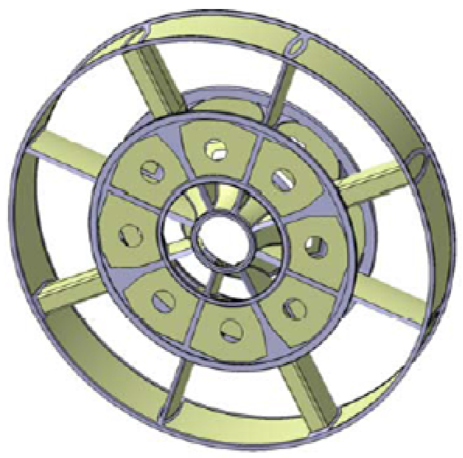
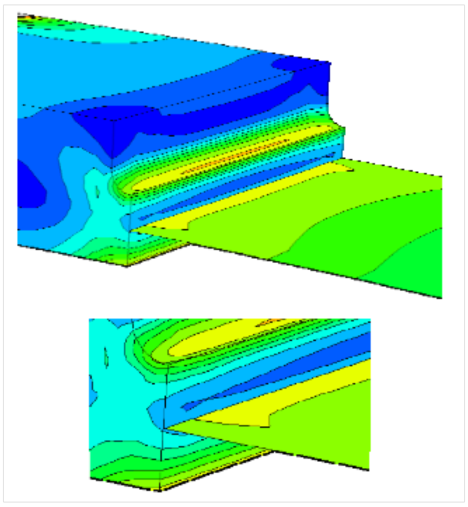

(d) Mixed dimensional model using shells and solids

FIGURE 2.4: Idealization of a complex geometry using solid and shell elements

complex changes in geometry provide a significant challenge while trying to create a mixed dimensional FE model with some or complete automation.

2.3 Midsurfaces using face pairing

There is a definite need for a dimensional reduction capability that is more powerful and easier to use than those currently available in the market. Such a capability should deliver an automated scheme for handling cases that have traditionally caused problems for algorithms in this field. A technique based on extraction of the midsurfaces has been proposed by


(a) Stress concentration at the shell-solid interface

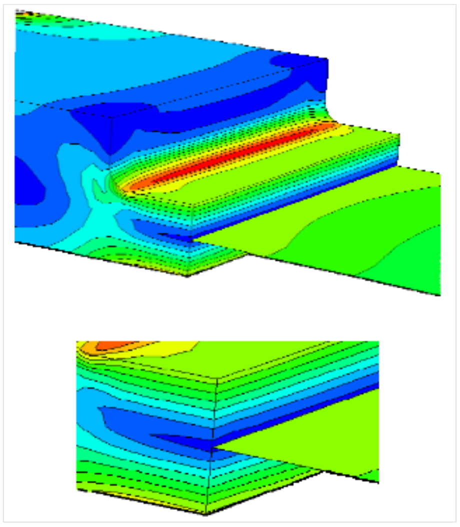

(b) Extend solids into shells to capture stress concentrations

Figure 2.5: Modeling the shell-solid interface

Rezayat [42] and Hanmin Lee *et al.* [43]. This approach combines geometric and topological information to get a discrete definition with reduced dimensions and is represented in Fig. 2.6. Though the method appears to be similar to the continuous medial surface discussed in the next subsection, there are some fundamental differences between the two. The face pairing method does not create branches since it deals with surfaces and not edges like the medial axis transform does. Since no branches are created, the face paired midsurface preserves the volume of the solid model. The branches duplicate mass and volume which is undesirable. Since this algorithm uses geometric reasoning to define shape, it reflects the form of the part better than the medial axis method. The biggest disadvantage however is when handling sections where there is no visible face to pair and extract the midsurface i.e, the joint in a T-section as shown in Fig. 2.6. The midsurfaces extracted need to be extended to meet before being stitched to form a usable dimensionally reduced model. Though the face pairing script can be designed to handle simple sections, the problem gets worse with complicated 3D geometries.

"The general technique for the extraction of midsurface using face pairing as mentioned in [42] involves:

- ⋆ Pairing of surfaces to find two or more surfaces between which a midsurface can be created. This process chooses surfaces which lie opposite to each other and define

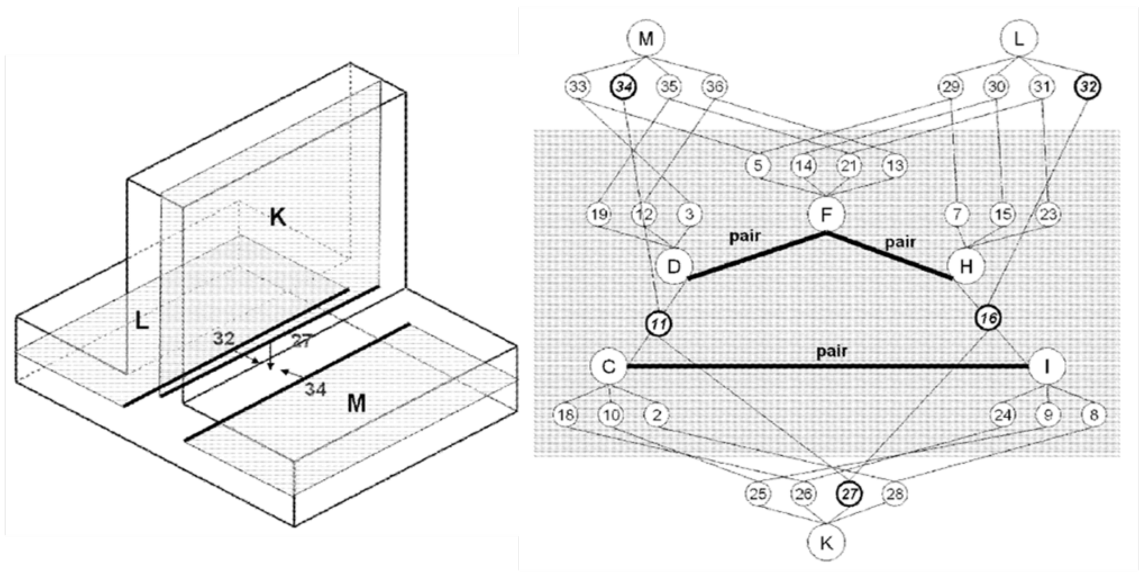


Figure 2.6: Midsurface extraction using the face pairing method

a clear midsurface. This pairing can be done either manually or automatically. For analysis applications pairing along the thinnest material direction is desired and this can be done automatically. The automatic pairing step first assembles a list of phase pairs belonging to the solid model and these are eventually used to generate midsurface patches.

⋆ Creating an adjacency graph based on the topology and size of surfaces: The geometry and topology of the solid model, along with graph theory, are used to identify features and determine the shape and topology of the midsurface.

⋆ Generating midsurface patches by geometric interpolation and 2D Boolean operations: To generate the complete midsurface, the midsurface patches associated with the active parent faces on the part must be identified.

⋆ Once the patches are generated, 'sewing' operators extend and stitch the midsurface patches in appropriate manner based on adjacency graphs. This operator attempts to extend and join two midsurface patches without generating an additional surface.

The major advantage of the midsurface abstraction technique is its feature recognition capabilities. Feature recognition enables us to represent parts in terms of high level functionally significant entities (holes, pins) rather than low level geometric entities (faces, edges). Such high level representation is indispensable in a concurrent design environment where feedback on manufacturability and ability to assemble must be communicated to all parties involved in a clear and intuitive manner."

The face pairing approach involves constructing the 3D midsurface for a part model by connecting or sewing the midsurface patches obtained for pairs of surfaces. The major disadvantage extracting midsurfaces using face pairing is that it works using a pairing strategy that has thus far required human intervention. Connecting various midsurface patches require 3D Boolean operations. The algorithm proposed by Rezayat [42] appears to work only for objects that are a combination of few geometric configurations such as L-sections and T-sections. A few approaches as mentioned by Muthuganapathy *et al.* in [44] involve generating the midcurves of faces first and then obtaining the topology of the midsurface using midcurves. Pairs of face corresponding to each midsurface patch are identified using the midcurves. However, generation of the midcurve has been addressed only for restricted domains. Correct pairing of the edge pairs is required to generate the midcurve. These approaches might have difficulty in handling general thin wall solids as the midcurve may not always resemble the midsurface, for example when a lateral surface is sub-divided into multiple surfaces due to the presence of fillets. Stolt[45] proposes a method where a complex geometry is divided into sections and the midline of each section extracted and assembled to create a midsurface. However, for this method to work, a midline without any branches needs to be created which is a non-trivial problem.

One commercial package which can effectively extract midsurfaces from 3D solid geometric models is HyperMesh (HM). It is however, noteworthy to mention that HM being a commercially available package, operates in a black box environment. This prevents the end user from knowing what algorithms HM uses to extract midsurfaces. For this research program, the steps followed to extract a midsurface need to be automated in order to work in batch mode. However, based on the inputs required to extract midsurfaces or midlines in HM, it is presumed that it uses either the face pairing or the graph based midsurface extraction methods.

2.4 Medial axis transform

Understanding shapes is of great importance in pattern analysis and machine intelligence. The medial axis or skeleton has been introduced as a generic and compact representation of a shape while maintaining its topology. Skeletonization aims at reducing the dimensionality of the shape such that it can be represented with less information than the original one. Topology and geometry alone are not sufficient to describe an engineering part and the function of its component, additionally some indication of the relationships and connectivity between features such as steps, notches and slender parts is required. A significant amount of research has been done by Armstrong *et al.* in [46] and Krishnan in [47] and [48] on a shape description tool called the Medial Axis Transform (MAT). MAT is difficult to apply in a robust and stable way since a small perturbation to the boundary can cause a larger change to the MAT. The MAT provides an alternative representation of geometric models that has many useful properties for analysis modeling. Applications include decomposition of general solids into sub regions for map meshing, identification of slender regions for dimensional reduction and recognition of small features for suppression. The goal of an automated FE modeling system is to accept a general problem definition as input and to return results of prescribed accuracy. A general problem definition will include a geometric model of the component to be analyzed with all the relevant attributes such as loading, restraints, material properties, etc. However, in many cases, the geometric model is too complex a domain to analyze in a realistic time frame when carrying out design search processes.

MAT of a 2D object provides an additional skeleton like representation of the shape of the model based on the geometric proximity of its boundary elements. It can be described as the locus of the center of a maximal inscribed disc as it rolls around the object's interior, where a disc is maximal if it is contained within the object but not within any other disc. The radius of this inscribed disc can be obtained at each point on the medial axis, the combination of the locus and the associated radius function is known as MAT and is as shown in Fig. 2.7. The points where three or more 1D entities of the medial axis meet are called branch points. A big advantage, is that the original component can be completely recovered from the MAT. It has also been proven that the union of the entire infinite maximal diameter circles is equal to the original object. Therefore, MAT can be used as the 1D representation of 2D objects. A medial point exists at an equal distance from at least two points in the boundary; this is also referred to as the bisector property. Many algebraic methods have been developed based on this property and they fundamentally rely on the fact that the algebraic form is explicitly known for each surface of the medial axis of the solid. Thus the radius function which provides the local thickness, can be defined as the shortest distance to the boundary of the object for each point on the medial axis. This definition can also be extended in defining the medial surface of 3D objects giving a 3D to 2D transformation. Katz *et al.* in

[49] discuss how the MAT can be used to its full potential by separating the substance from the connection information of an object.

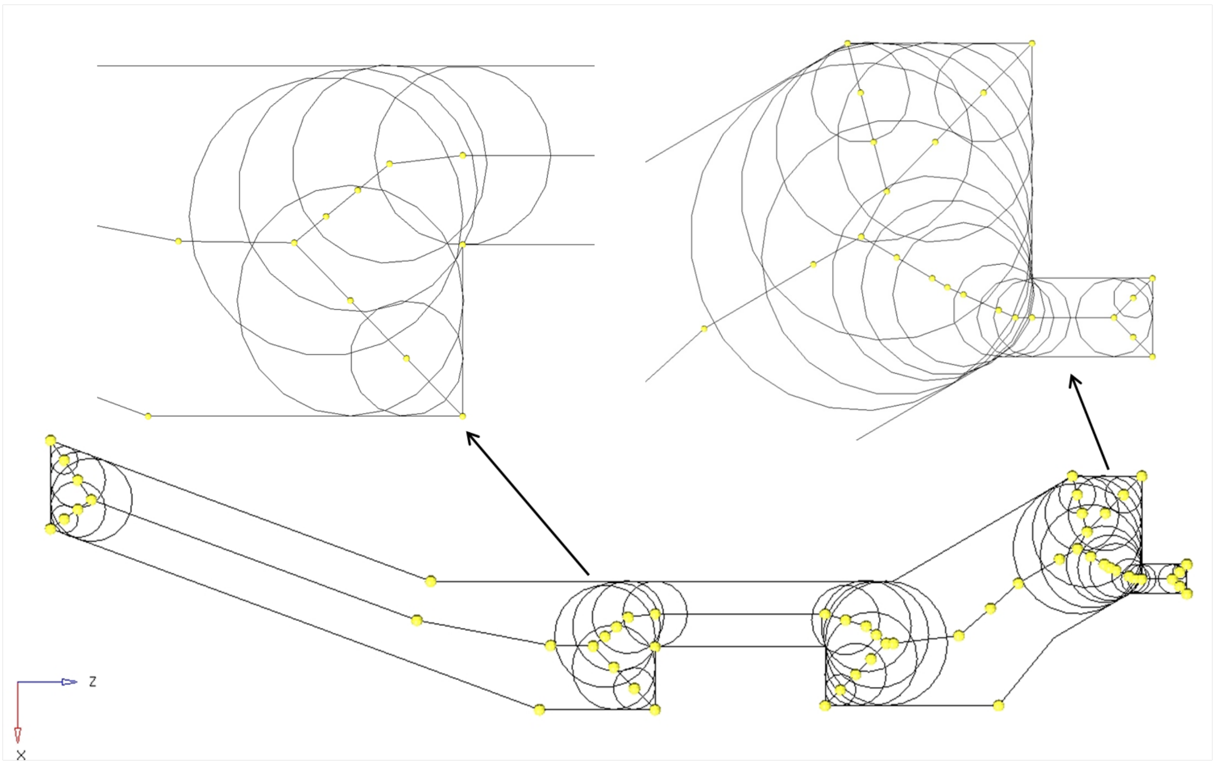


FIGURE 2.7: The medial points (circle center) created using the Matlab toolkit

The use of medial axis and medial surface transform techniques for dimensional reduction and FE generation of complex 3D models for this thesis is still being researched as a feasible option. Unfortunately, medial axis and surface transforms have several characteristics that limit their usefulness for analysis applications as mentioned in [42]. These algorithms create artificial branches at every convex point along the edges of the geometry. Due to these branches, the medial surfaces do not follow the local topology of the geometry exactly. Unlike the face pairing method, the medial axis transform does not recognize the surface form but only considers a surface edge as either being concave or convex. Since each of the branches created are also represented by a thickness, the actual volume and mass of the 3D geometry is not preserved. This results in a medial surface model which is more in volume when compared to the midsurface model extracted using face pairing.

Since a medial surface is defined as the set consisting of the centers of all maximum spheres which fit into that object, branches/spurs are created as soon as the sphere comes into contact with more than two surfaces as shown in Fig. 2.8. These branches must be

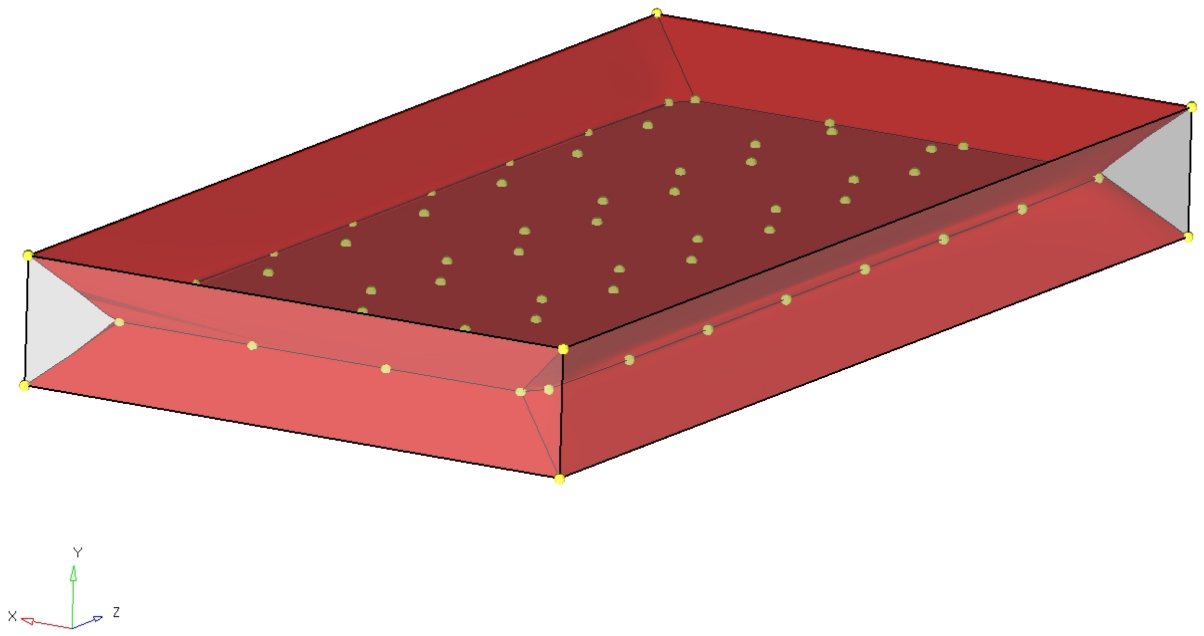


FIGURE 2.8: The medial surface of a plate

dealt with before the medial surface is used for shell meshing. It needs to be seen how the retention or removal of these branches affect the stiffness of a part. The face pairing method as discussed in [42] and the graph based midsurface extraction discussed in [43] generate disconnected midsurfaces which retain the general shape of the component but do not have artificial branches or spurs.

In [50] an algorithm is proposed by Preparata for generating the medial axis of a simple polygon. A more detailed explanation of the maths involved in MAT is discussed in [51]. A related problem is that the topology of the medial surface, in its raw form, does not exactly follow the local topology of the solid. By using the medial surface technique, the user has little control over global suppression of features and also how the medial surface can preserve the volume of the part accurately after spur removal (if this is carried out). But it is noteworthy to mention that the volume of the abstracted model should represent the volume of the solid model as accurately as possible. Sinha *et al.* in [52] discuss how the medial axis along with the spurs can be used to create 2D FE models on medial surfaces. An algorithm coded using Matlab [53] has been used here to generate medial curves and corresponding results are discussed in Chapter 4.

2.5 Chordal Axis Transform

The Chordal Axis Transform (CAT) is a method proposed by Lakshman in [54], [55] and by Quadros in [56] and [57] to extract midsurfaces of thin walled solids. "CAT has been projected as an alternative to the MAT for obtaining skeletons of discreet shapes. Although the definition of CAT appears to be a variation of that of the MAT, there are important differences between the two transforms. First, the CAT, as defined, yields a piece wise smooth disconnected protoskeleton. Second, the CAT can be stably designed for a shape whose boundary is discrete, i.e. specified as sequences of points separated in space. This is done by replacing maximal discs by empty circles that pass through three or more points of the shape's discrete boundary such that no circle contains a boundary point in its interior that is visible to two boundary points lying on the circle. Each empty circle identifies a triangle whose edges lying in the shape's interior replace maximal chords of tangency in the discrete version of the CAT. The triangles so formed are the Delaunay triangles of a Constrained Delaunay Triangulation (CDT) of the shapes interior", as mentioned in [54].

The three steps involved in extracting the midsurface using the CAT of a thin walled solid are, generating a tetrahedral (tet) mesh of the thin walled solid without inserting interior nodes, generating raw midsurface by smart cutting of tets and remeshing the raw midsurface via smart clean up as represented in Fig. 2.9 (from [56] and [57]). One disadvantage of this method is that a tet mesh has to be generated on a 3D model before the chordal axis surface can be created. Creating tet meshes on complicated solid CAD models are difficult in their own terms. This is because, to create tet meshes either a solid model or a well connected set of surfaces that define a closed volume (solid) is required. More often than not, FE packages like ANSYS or HyperMesh will not be able to create tet meshes on those geometries which have poorly modeled complex features. This is due to loss of geometric information as models are moved from CAD to CAE. Though extensive work has been done by commercial finite element and preprocessing packages to reduce this problem, unless a geometric model is prepared for use in an FE package, it is highly probable that the geometry will not be meshable. The tolerances and higher order spline surfaces and curves that form a complex surface creates using Siemens NX or CATIA cannot be handled by FE packages. In such circumstances, substantial amount of manual intervention maybe required to clean the geometry before tet meshes can be created. Therefore by using the CAT method, apart from the geometry as an input, the solid mesh also features as an input which could turn out to be quite a difficult task to handle in optimization work flows. However, chordal axis can

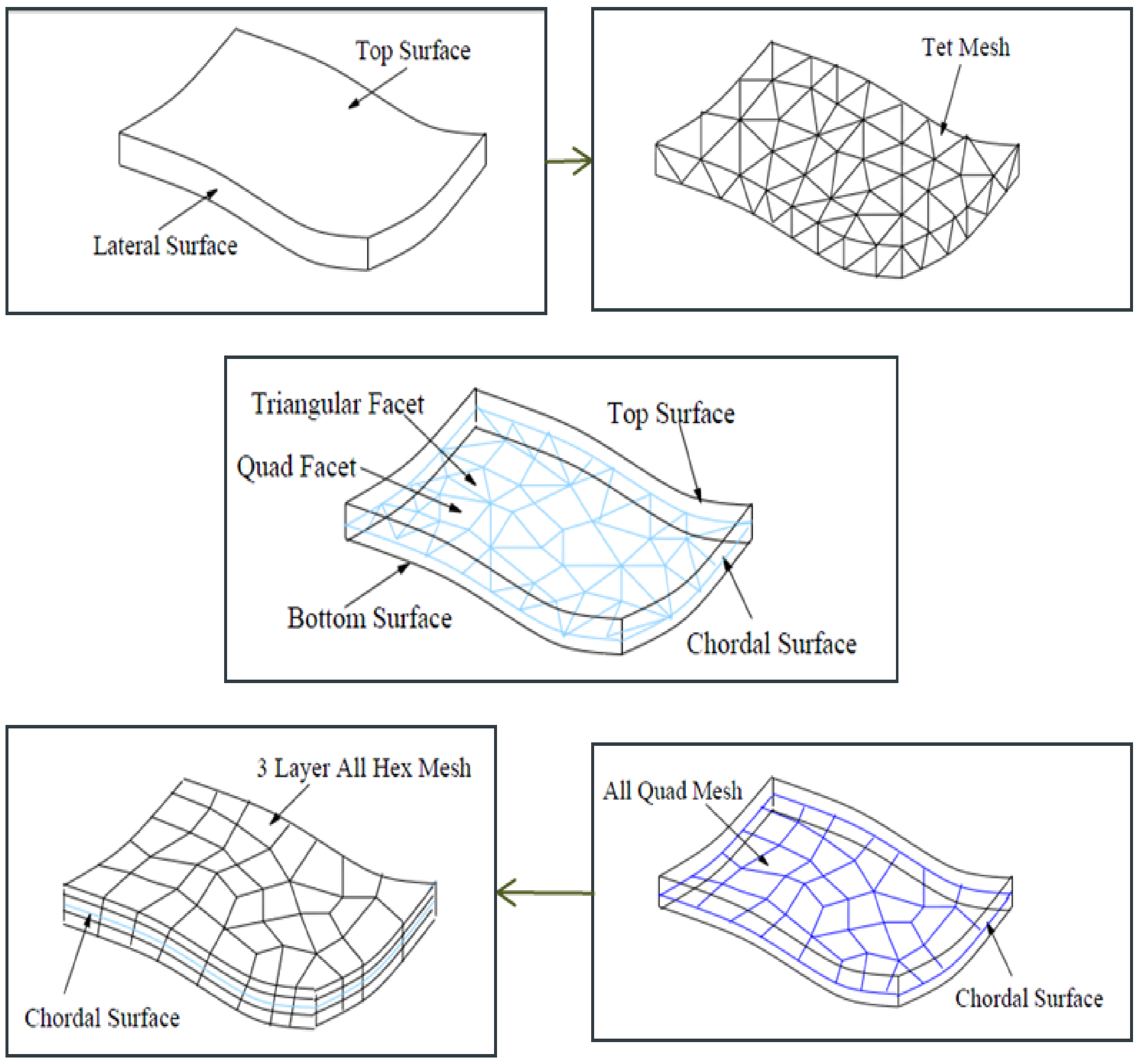


FIGURE 2.9: Midsurface extraction using the chordal axis transform

be used effectively at the 2D level unless the geometry diverges to form thick and complex regions.

2.6 Midlines using Level Sets

The level set methods (LSM) have been introduced as efficient numerical techniques for solving a particular class of hyperbolic partial differential equations known as Hamilton-Jacobi, while satisfying its vanishing viscosity weak solution as discussed in [58] and [59] by Telea *et al.* As mentioned in [60] by Hassouna *et al.* "The LSM can be used to track the motion of interfaces propagating under complex speed laws while handling complex topology changes such as branching and merging. An object centerline point is selected automatically as the point of global maximum Euclidean distance from the boundary, and is considered a point source that transmits a wave front that evolves over time and traverses the object domain as shown in Fig. 2.10 (from [60]). The front propagates at each object point with a speed that is proportional to its Euclidean distance from the boundary. The motion of the front is governed by a nonlinear partial differential equation whose solution is computed efficiently using level set methods. Initially the point source transmits a moderate speed wave to explore the object domain and extract its topological information such as merging an extreme point. Then it transmits a new front that is much faster at centerline points than non central ones. As a consequence, centerlines intersect the propagating fronts at those points of maximum positive curvature. Centerlines are computed by tracking them, starting from each topological point until the point source is reached, by solving an ordinary differential equation using an efficient numeric scheme. This method is computationally inexpensive and computes centerlines that are centered, connected, one point thick and less sensitive to boundary noise". The uses of level set method for this research program poses one problem. Along with finding the centerline, the information of distance of a point on the centerline from the boundary is also very essential to create a good 2D FE model. This distance information is not needed to create the centerlines while using 'level sets'.

2.7 Other codes considered

The CGAL [61] (Computational Geometry Algorithms Library) project from the European community provides a comprehensive library of data structures and algorithms for computational geometry. It is an open source project to provide easy access to efficient and reliable geometric algorithms in the form of C++ libraries.

Mesecina [62], has been developed by Miklos *et al.* [63] as a tool for visualizing and analyzing the medial axis of two dimensional geometry based on their work in [64]. This code can be downloaded as an executable but cannot be edited or modified. However, the code gives a good representation of the many geometric reasoning algorithms available in the academia. The tools that can generate a medial axis should also provide the thickness information for meshing purposes. Mesecina is effective while creating medial axis for a given set of boundary points but there is no obvious way to export the distance of the medial points from the boundary for use externally.

Sphere-Tree [65] construction toolkit developed by Bradshaw *et al.* [66] was downloaded

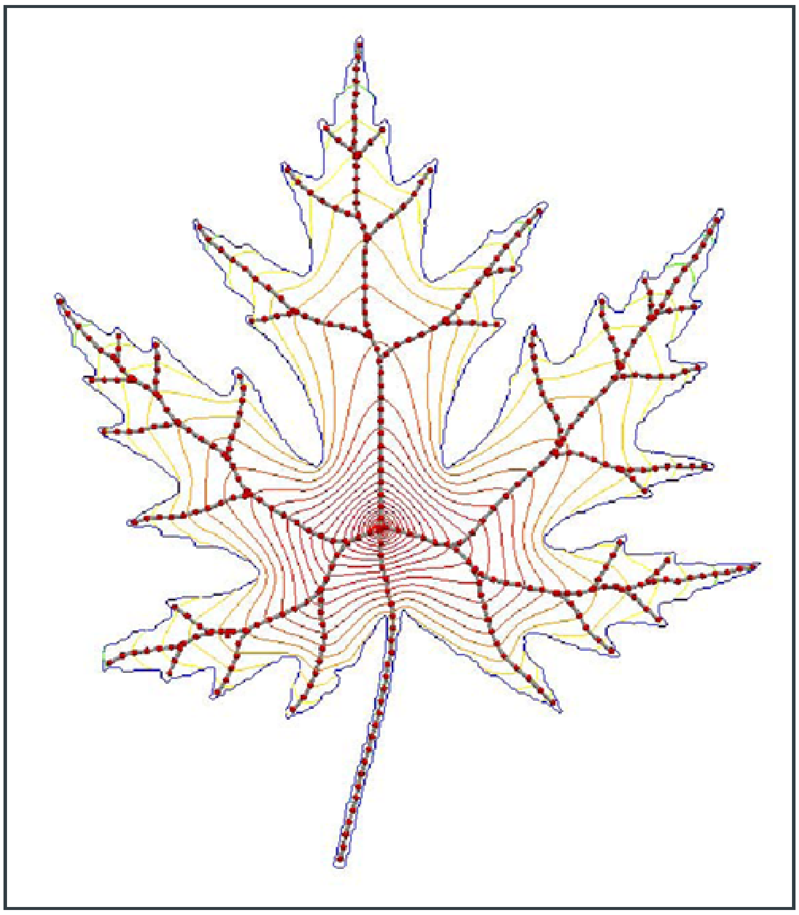


FIGURE 2.10: Midsurface extraction using the level set method

and tested for its use in creating sphere's inside a given volume. Though this code is not designed for creating medial points, the code's ability to fill spheres inside a given volume made it interesting. This toolkit consisted of many sphere tree construction algorithms of which many were based on the Hubbard's medial axis algorithm. The code in this case is available as an executable and no source code is available for customizing.

Yoshizawa *et al.* in [67] have presented a novel approach to obtain a Voronoi based skeletal mesh which is an approximation of the 3D medial axis. The code SM03Skeleton [68], has been created with the intention to morph the geometry by deforming the medial surface and then reconstructing the geometry based on the medial surface. This again is not the purpose of this research program and like the other tools that have been considered, the SM03Skeleton is also an executable which is not customizable.

Another algorithm, 'Power Crust [69]' developed by Amenta *et al.* [70] deals with 3D surface reconstruction based on the medial axis transform. The code creates an approximation to the medial axis of the solid given a set of points from the boundary of a 3D object.

Many other techniques to reduce optimization problem size have been considered. One of them is the CAD/CAE dimensional reduction process where a new concept 'Skeletal Representations' is discussed. While the dimensional reduction process works at the geometry level, model order reduction for large FE models discussed by Rudnyi *et al.* in [71] works at the solver level and the mesh compression techniques mentioned by Alliez *et al.* in [72] works at the FE level. The choice of technique chosen to reduce the problem size is based on the stage at which it is being dealt with; i.e. at the geometry, FE or the solver level. The current research is based on problem size reduction which falls in the geometry category and hence emphasis is laid on midsurface extraction techniques. A few approaches to compute centerlines from 3D objects which had been discussed in the open literature are based on distance transform methods, topological thinning methods and hybrid methods for volumetric data and Voronoi based methods for polygon data.

2.8 Summary

There are various methods to extract midsurfaces as discussed in this chapter. However, the scope of this research is not solely to produce skeletonized 2D or 1D representations of solids, but to obtain a centerline that can be used to create a midsurface which best represents a 3D object and create 2D FE models using these surfaces for optimization purposes. Thakur *et al.* in [73] have surveyed most of the CAD model simplification techniques existing today. The midsurface abstraction using face pairing technique and the medial surface based on MAT are the two techniques that would be looked into more closely during this research.

The face pairing midsurface abstraction preserves the volume of the part, but generates midsurfaces that are disconnected. This technique preserves the general topology of the part which is a desirable parameter while looking at component stiffness. The procedure to connect the disconnected midsurfaces however is not well documented. Test cases have been built to establish a connection procedure for these discontinuities before this technique can be used for this research program and are discussed in detail in Chapter 4. HyperMesh can extract midsurfaces of solid geometry. The extraction fails when the geometry gets complex. However in Chapter 3, a Fancase model has been built to understand the pros and cons of using a black box commercial tool.

The Level set method can be used to extract midlines which are centred, connected and less sensitive to boundary noise. The midlines created using Level sets and its use in the creation of FE models for this research program needs to be studied further. However, its use in structural topology optimization is well documented. This technique is more common in the medical imaging, etching, deposition, and photolithography development. One of the reasons for this is because none of these applications need the thickness information along the centerline as is required while creating a FE model.

The MAT produces continuous centrelines but with spurs and branches. The algorithms that use medial axis also provide the shortest distance to the boundary from each medial point which defines the thickness of the component at that point. This information is vital when defining the shell thickness which has been modeled on the medial surfaces created by revolving the medial axis. The Chordal Axis Transform (CAT) on the other hand produces centerlines very similar to the ones produced by MAT but with more inputs than are required by MAT. The CAT is based on finding the midpoints of all edges of the triangles enclosed within the boundaries of a 2D section by cutting tetrahedral elements along a plane to extract a surface. Once these midpoints are found for all the triangles, they can be joined to create a Chordal Axis. Apart from the geometry which is an input for the MAT and CAT methods, the CAT requires the geometry to be meshed using 2D (3 or 6 noded triangle element) or 3D elements (4 or 10 noded tetrahedral element) to extract the chordal axis or the chordal surface.

Chapter 3

Use of midsurfaces : A case study using HyperMesh

Based on the various methods discussed in Chapter 2, a choice of using Face pairing and MAT has been made out of the many possible methods available to either mesh or dimensionally reduce a complex model. These methods have been chosen based on previous experience and the literature review. Some of the factors that have been considered, while selecting a meshing process for the optimization cycle, have been based on the fact that this research program deals primarily with the whole engine (gas turbine engine) model consisting of all the load bearing structures. The fancase has been chosen for this case study bearing in mind the general shape of gas turbine engine casings. However, the choice of meshing process is made based on the following factors:

- ⋆ The complexity of the model that needs to be dimensionally reduced.
- ⋆ The amount of manual intervention when the work flow is invoked each time (in this case the manual intervention should be nil).
- ⋆ The choice of tools is also seriously restricted by possible deployment needs.
- ⋆ To avoid tetrahedral (tet) elements while creating 3D FE models. Use of higher order tet elements is recommended for good FE simulations and meshing using tet elements would mean a model with very many degrees of freedom. Fewer elements can counter this problem, but fewer tet elements would not capture the correct stiffness of the component being modeled.

⋆ Dimensional reduction should not lead to over simplification of casing geometry that would be eventually modeled. Since the stiffness of the casing structure relates directly to the tip clearances in a gas turbine engine, all the preprocessing that goes into preparing an FE model should be done with utmost care. Too much approximation as would be the case if modeled using shell and beam elements would mean more corrective measures to balance weight, center of gravity etc. Such measures are detrimental to the automated process being envisaged.

A mesh convergence study using higher and lower order hexahedral and tetrahedral elements was conducted for a thin casing like geometry. The findings of this study is presented in Appendix. A

3.1 3D FE modeling using hexahedral dominant mesh

Hexahedral (hexa) dominant meshing is an attempt to break the shackles presented by brick meshing of complicated geometric models. ANSYS Workbench [20] and Harpoon [74] are just two tools of the numerous ones currently available which can create a hexa element dominant mesh on complex geometries. Hexa dominant meshing is not particularly suitable for slender structures like the fancase and often fails to produce enough hexa elements in complicated parts. The area of concern really is the refinement required for the mesher to capture the flanges and build at least one layer of good brick elements across the flange thickness. Harpoon has been used to study how it can handle meshing thin structures with many flanges and ribs, as in a fancase. In the first case, mesh size was set to the default value of '1' and the mesh generated is as shown in Fig. 3.1

It is evident from the figure that even after using almost 350,000 elements, the hexa dominant meshing algorithm is not able to capture the ribs and flanges on the fancase accurately. By using the next level of refinement i.e. mesh size of 3 in Harpoon, a mesh as shown in Fig. 3.2 was created for the fancase geometry. Harpoon has 3 levels of mesh density and is independent of model units. In this case the tool has been able to capture all the ribs and flanges to a major extent but at the expense of an enormous element count. With approximately 2 million elements for a simple structure like the fancase, the element count for the whole engine can turn out to be too large to solve in an optimization problem. This is one of the main reasons for shelving the idea of creating a new 3D FE model every time the geometry is updated by the optimizer.

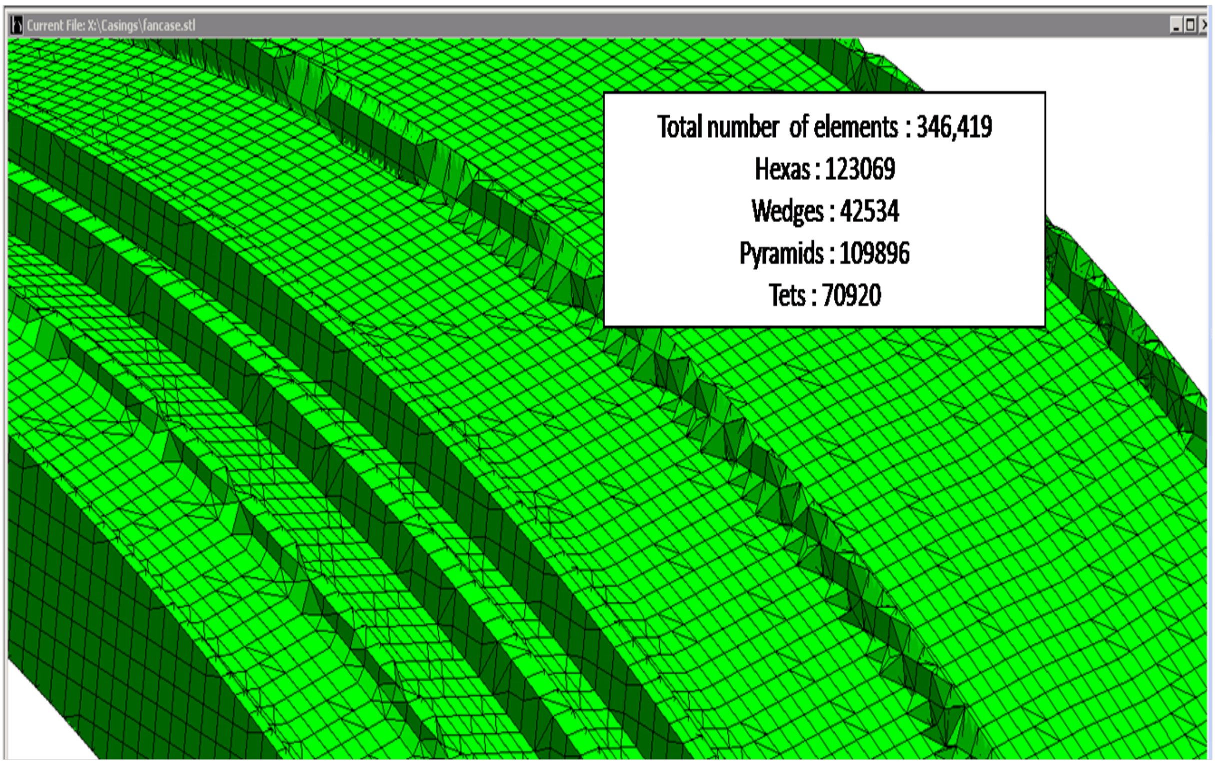

Current File: X:\Casings\fancase.stl
Total number of elements : 346,419
Hexas : 123069
Wedges : 42534
Pyramids : 109896
Tets : 70920

FIGURE 3.1: Hexa dominant mesh of a fancase using default mesh settings in Harpoon

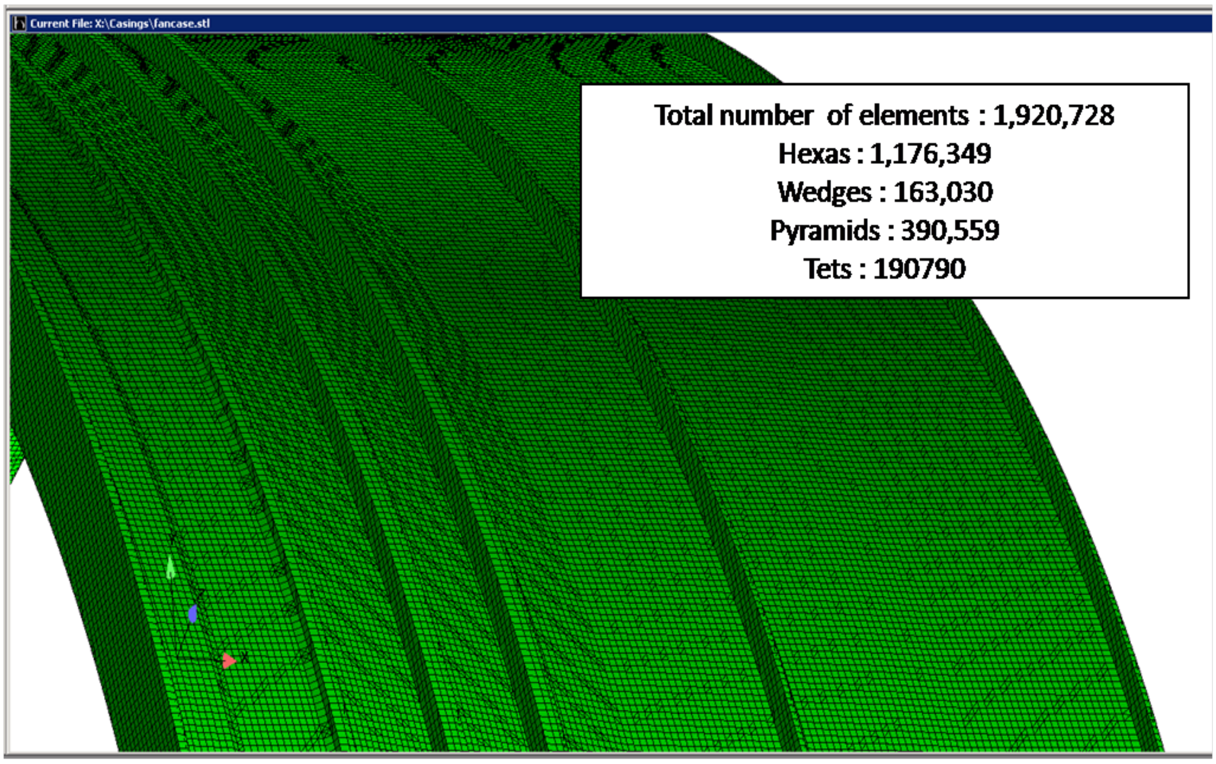

Current File: X:\Casings\fancase.stl
Total number of elements : 1,920,728
Hexas : 1,176,349
Wedges : 163,030
Pyramids : 390,559
Tets : 190790

FIGURE 3.2: Hexa dominant mesh of a fancase using fine mesh settings in Harpoon

The major disadvantage of such a meshing system is the lack of control over the element density at critical regions and most importantly the total element and node count. The lack of local mesh control and 3D mesh compression makes the hexa dominant mesh an infeasible option for this research program. The pros and cons of this along with some other methods is presented in Table 3.2.

3.2 Meshing a cross section and revolving

This is one of the simplest of all meshing techniques for dealing with engine cases. The 2D mesh generated can either be used as axisymmetric elements or revolved along 360 degrees to create 3D brick elements. When creating a 2D mesh on the cross section of an axisymmetric geometry, a good quality mesh with mostly quadrilateral (quad) elements can be easily created as shown in Fig. 3.3.

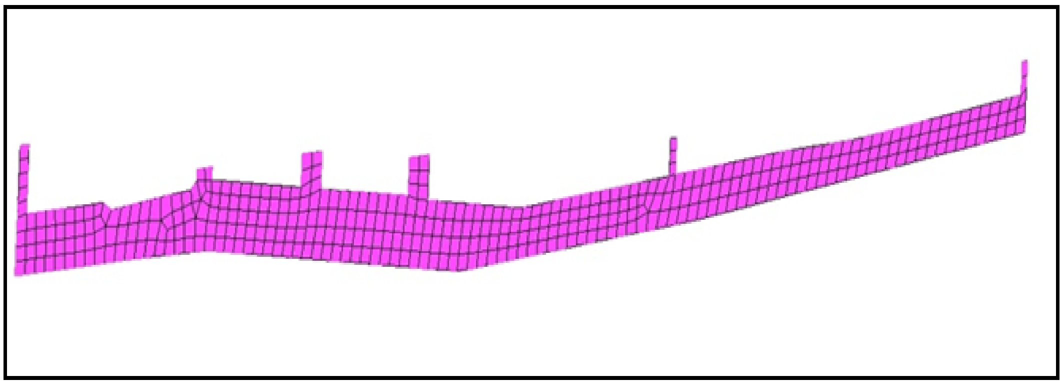


FIGURE 3.3: Cross section of the fan case meshed with 2D elements

Quad elements in the cross section means brick elements in the 3D mesh that would eventually be created by revolving the 2D elements. This is probably the result that would be expected of a good hexa dominant mesh. Good control over the element count can also be maintained by deciding on the number of elements required along the rotational axis. Since the 2D mesh is revolved to create a 3D mesh, any feature that is non-axisymmetric cannot be meshed using this method, see Fig. 3.4.

This procedure effectively results in a model that could be either over stiff or less stiff than the original model. For example, consider all the holes in a casing that have been cut out for various design reasons and all the bosses that are used to connect the bleed valves etc to the casings. By revolving a 2D mesh, these holes or bosses are not modeled and a simple cylinder-like structure is created. In some cases like the bearing housing or the intermediate

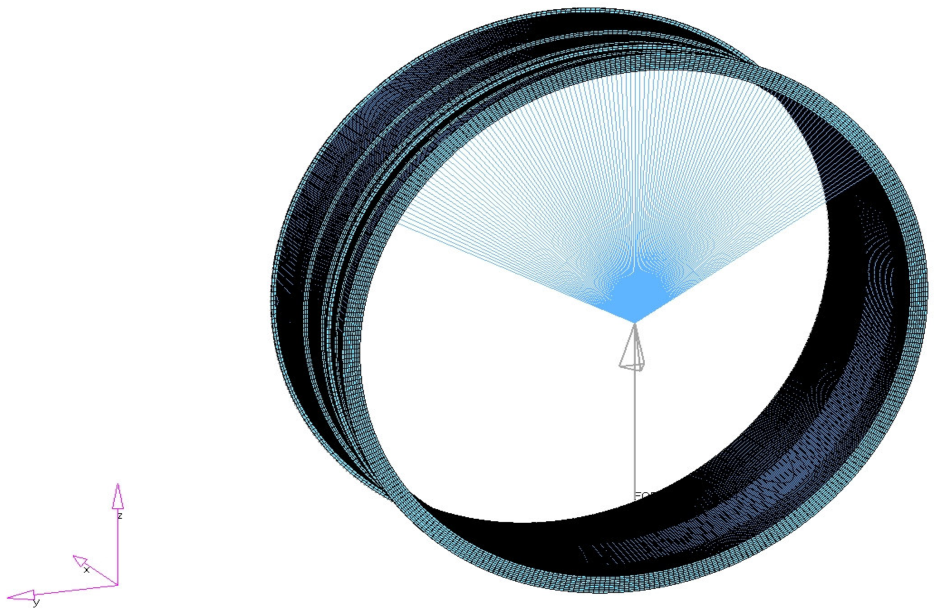


FIGURE 3.4: Cross section mesh revolved to create a 3D mesh

case, correct definition of the stiffness of the structure is of utmost importance. In such cases, revolving a 2D mesh could result in over simplifying the overall geometry of the structure.

3.2.1 Analysis with loads and boundary conditions

The fancase analysis was setup in ANSYS Mechanical as well as LS-Dyna3D. These solvers have been chosen due to their accessibility to this research program. Stiffness analysis has been performed on the 3D mesh created using the 2D mesh revolving option. Stiffness analysis has been chosen for the case study in order to ensure that the stiffness representation of the two methods chosen (2D mesh revolve and midsurface extraction) compare well. The 2D mesh revolve technique is considered as the baseline since it represents a complete 3D model built with hexahedral elements. This technique rules out the need to analyze a model created using the hexa dominant mesh as explained in Section 3.1. Henceforth, no analysis has been performed to build a case using the hexa dominant mesh method.

In the aeroengine assembly, the fancase is connected to the rear inlet case at its rear end, while being left free at the fan end. In the analysis, the end that connects to the rear inlet case is fixed while a unit radial load is applied at the front end as shown in Fig. 3.4. A constraint equation is used to transfer the unit radial load to the fancase structure. RBE3 in ANSYS incorporates constraint equations such that the motion of the “master” is the average of the “slaves” (the master node is where the load is applied and the slave nodes are part of the casing to which the load is transmitted). For the rotations, a least-squares approach is used to define the “average rotation” at the master from the translations of the slaves. If the slave nodes are colinear, then one of the master rotations that is parallel to the colinear direction cannot be determined in terms of the translations of the slave nodes. Care is taken to make sure that only nodes from the top 100° or 120° of the fancase front flange inner radius are used to create the constraint equations. These constraint equations are used to transfer the load applied at the master node to the slave nodes which form the top 100° or 120° of the fancase inner circumference. This method of simulating the radial loads on the fancase for stiffness analysis by transferring the loads only to the top few nodes is considered the best practice since the radial loads are transmitted onto a casing from a bearing housing only through the top 100° arc. A Young’s modulus of 216620 MPa, density of $8\text{x}10^{-6}\text{kg/mm}^3$ and a Poisson’s ratio of 0.27 are used as material properties (steel) for the analysis. Refer [75] for tips and tricks that can be made use of while handling ANSYS.

3.2.2 Results

The solid FE model consists of approximately 145,000 hexahedral elements. The resultant deflection of the fancase for the unit load is as shown in Fig. 3.5 for LS-Dyna3D. The analysis was also performed in ANSYS and the results compared to ensure that the default solid element formulation in LS-Dyna produced comparable results. ANSYS will be used for all future analysis because of previous experience and also since the results produced by ANSYS are comparable to the ones produced by LS-Dyna. The deflection is measured at the master node of the RBE3 element to calculate the stiffness of the part. The difference in the radial deflection value between the LS-Dyna model and the ANSYS model was around 0.32% and is presented in Table. 3.1. Since the percentage difference between the results produced by each of these tools is insignificant, it can be concluded that the EQ18 element formulation in LS-Dyna works well when compared to the Solid-45 element formulation in ANSYS Mechanical and that these results can be used as benchmark for midsurface modeling.

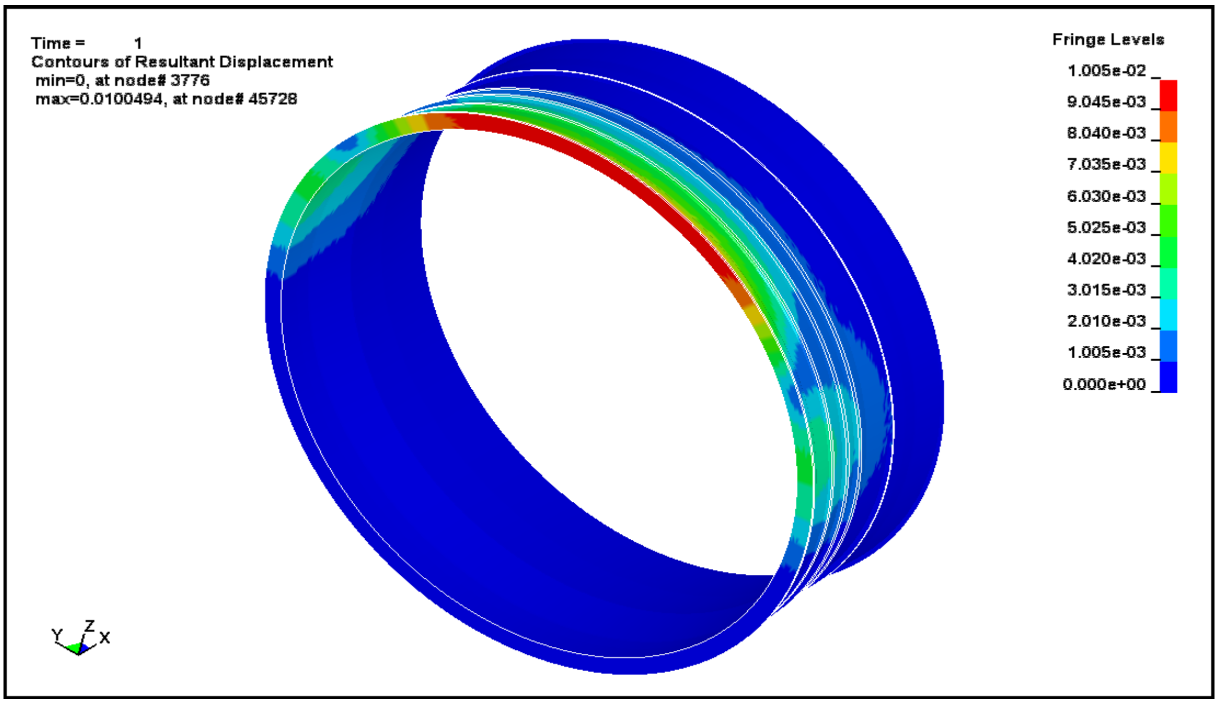


FIGURE 3.5: Resultant deflection of the revolved mesh in LS-Dyna

3.3 Modeling using midsurfaces

The advantages of using midsurfaces to create FE models has been explained in Chapter 2. Consequently, in this section, a fancase model is built using the mesh created on a midsurface extracted from a fancase 3D solid model generated in Siemens NX4. The 3D model is imported in HM V8.0SR1 as a 'part' file and midsurfaces are extracted using this geometry. No geometry cleaning is done in order to evaluate the capabilities of HM to work with geometry in its raw imported form.

3.3.1 Midsurface extraction

There are two ways in which a midsurface can be extracted in HM. One method is similar to the process adopted in generating midsurfaces in NX4. In this method, the pairs of surfaces between which the midsurfaces have to be found are manually picked. This can be a very laborious and cumbersome process while handling geometries with many surfaces and is the reason why the midsurfaces of the 3D fancase model are not extracted using NX4. The second method to extract midsurfaces in HM is to simply pick the whole solid and then let HM extract the midsurfaces. The midsurfaces extracted for an axisymmetric geometry in this casestudy are as shown in Fig. 3.6. The geometry in the image is not rotationally smooth due to graphics rendering issues in HM. The fancase model in this case study is not fully detailed and lacks features such as holes, bosses etc. However, the important step to note while performing the extraction using HM is that, the 'Align Steps' option in HM V8.0SR1 midsurface panel has to be toggled 'off' to avoid any discontinuity in the midsurfaces produced. This option is not a problem in HM V9.0 onwards.

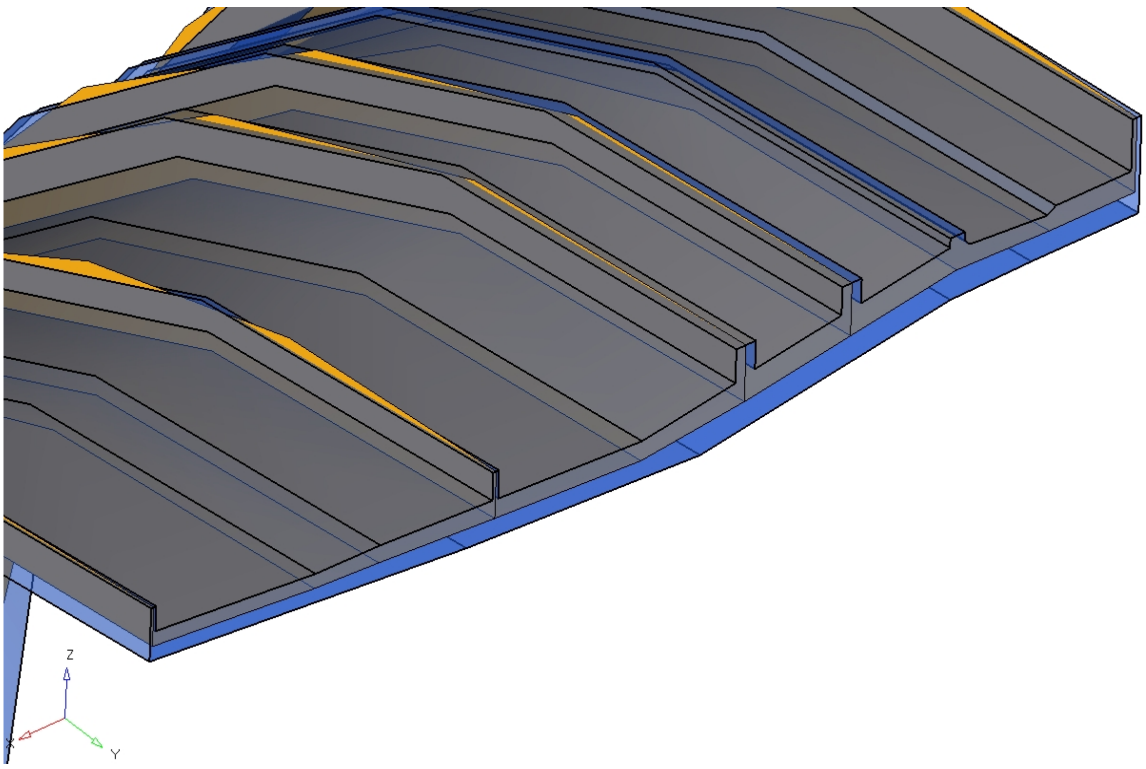


FIGURE 3.6: Midsurfaces as extracted for the fancase model using HM V8.0SR1

3.3.2 Preprocessing

The midsurface extracted from the solid geometry can be used to create a shell mesh using the 'automesh' option as would be done on any normal surface. Automesh is preferred when compared to interactive meshing while operating in batch mode. A fixed element size is set as a meshing parameter. The mesh produced by the HM auto meshing algorithm is as shown in Fig. 3.7. A more complex mesh pattern can be created based on special features or location such as flange and casing intersection etc, but its usefulness at this stage has not been considered. However, the greatest advantage of midsurface method of mesh creation is the total model size that would be analyzed, which in this case study is around 28,000 elements compared to the 145,000 elements in the 3D solid FE model built using hexahedral elements discussed in the Section 3.2. Although HyperMesh has advanced meshing capabilities, they are all available as manual meshing options. But it is evident from the scope of this research program that manual meshing capabilities of any tool are of no relevance when aiming to perform design optimization studies in batch mode.

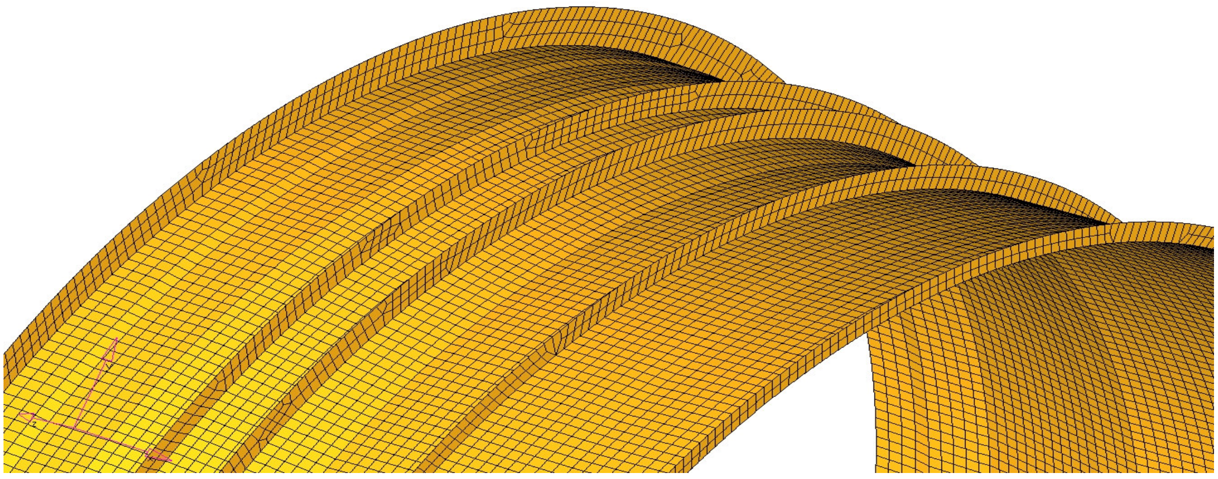


Figure 3.7: Extracted midsurfaces in the wire frame mode

3.3.3 Assigning thickness

The most important step while creating a FE model based on midsurfaces is to ensure that the shell elements have been assigned the correct thickness values. The HyperMesh midsurface algorithm stores the information it generates when calculating the mid distance between the outer and the inner surface to place the midsurface. This information which is used to create the midsurface is also used to assign the thickness of the shell elements present in each particular coordinate in 3D space. HM has a utility tab as part of the Mesh tools in the Geom/Mesh menu called “Midsurf thickness”. When operating using this tab, the elements that have been created on the extracted midsurfaces need to be picked and assigned thicknesses. This assigning of thickness is taken care of internally by HM and no manual intervention is necessary except for selecting the elements on which the thickness have to be assigned. Fig. 3.8 shows a contour plot of the thicknesses assigned by HM for export into LS-Dyna.

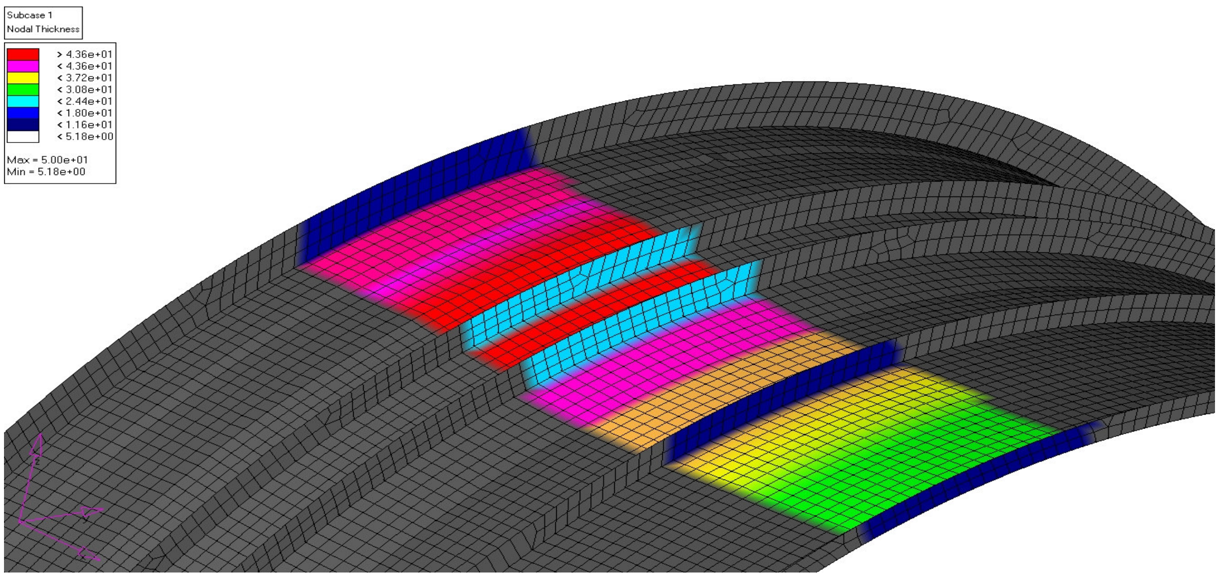


FIGURE 3.8: Thickness assigned on elements shown as a contour

The mesh generated for export into LS-Dyna may also be exported into ANSYS. A comparative study has been made between similar meshes with thickness assigned in HM using both ANSYS and LS-Dyna shell elements. The mesh when imported into ANSYS and viewed in the faceted mode graphically presents the thickness as shown in Fig. 3.9. However, when the thickness is mapped onto the mesh and exported into ANSYS from HyperMesh, one real constant is created for every thickness mapped. If all the elements on the surface have the same thickness, then one real constant is created in ANSYS to represent them. However in some cases as shown in Fig. 3.10, one real constant is created for each element if the thickness at all nodes on one surface are not same for all the elements on that particular surface. In this case, the model in ANSYS has close to 6,056 real constants. Though the 3D geometric model has been created by spinning a 2D cross-section geometry, HM detects very slightly different thickness for each element along one surface. A generic thickness is assigned to all elements that belong to one surface manually using ANSYS preprocessor and the resulting fancase thickness groupings are as shown in Fig. 3.9.

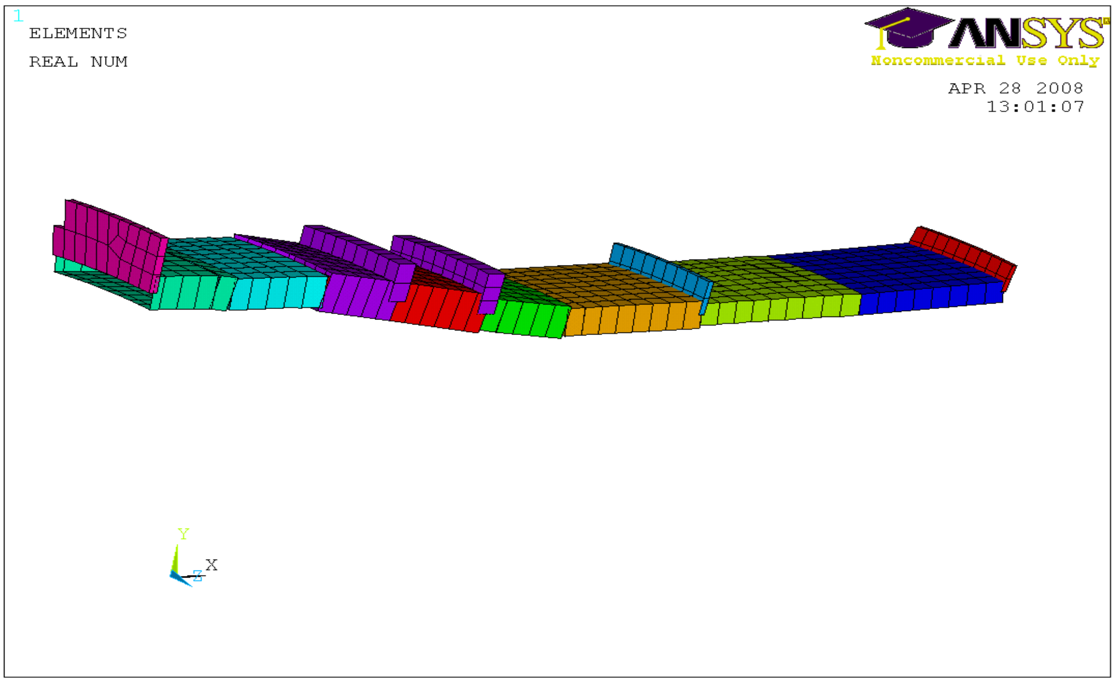


FIGURE 3.9: Thickness assigned on elements represented in 3D in ANSYS

3.3.4 Analysis and Results

The loads and boundary conditions applied on the midsurface mesh model are the same as explained in Section 3.2. The resultant displacement of the shell model is as shown in Fig. 3.11. The main purpose of this study is to compare the stiffness of the fancase when modeled using solid elements and shell elements with thickness defined on the midsurface. The difference in displacement results between the solid and the shell techniques should not be more than 5% as a general practice. The difference in radial deflection between the solid FE model and the midsurface shell model in LS-Dyna 3D is not more than 1.39%. Similarly, the difference between the results obtained from the ANSYS shell model and the LS-Dyna shell model is not more than 1.01%. The results when compared to the LS-Dyna solid model are as shown in Table. 3.1. This comparison between LS-Dyna and ANSYS is made since Rolls-Royce Group PLC, who are sponsors of this research use LS-Dyna and NASTRAN

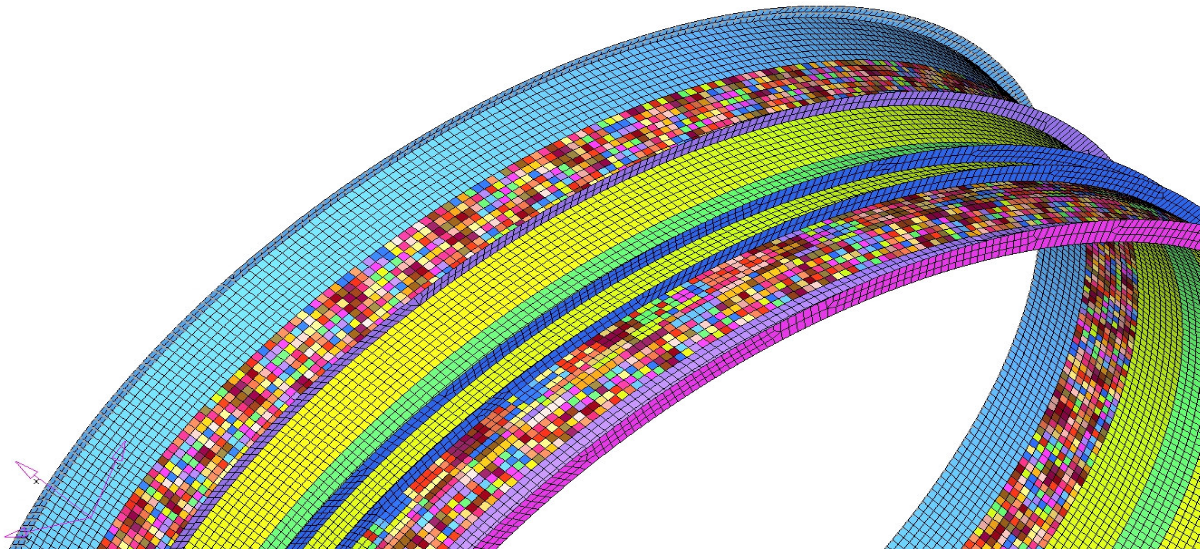


Figure 3.10: Thickness as mapped and exported to ANSYS from HyperMesh

for their FE analysis. However, there will be no access to NASTRAN during this research program and using LS-Dyna for linear static analysis is not recommended. Since the linear static analysis results from ANSYS match with those from LS-Dyna, ANSYS will be used for all FE analysis conducted from here on.

Table 3.1: Summary of fancase analysis results

	Elements	Nodes	Run Time (min)	Radial Disp (mm)	%Diff
Dyna Shell	28143	28623	1.46	0.0099	1.39
Dyna Solid	145081	189721	10.0	0.01004	-
ANSYS Shell	28143	28623	1.4	0.0098	2.39
ANSYS Solid	145081	189721	9.14	0.010007	-0.32

3.4 Discussion

This case study shows that creating shell models of 3D structures based on the midsurfaces extracted is a viable option. The variation in stiffness between a complete 3D FE model and

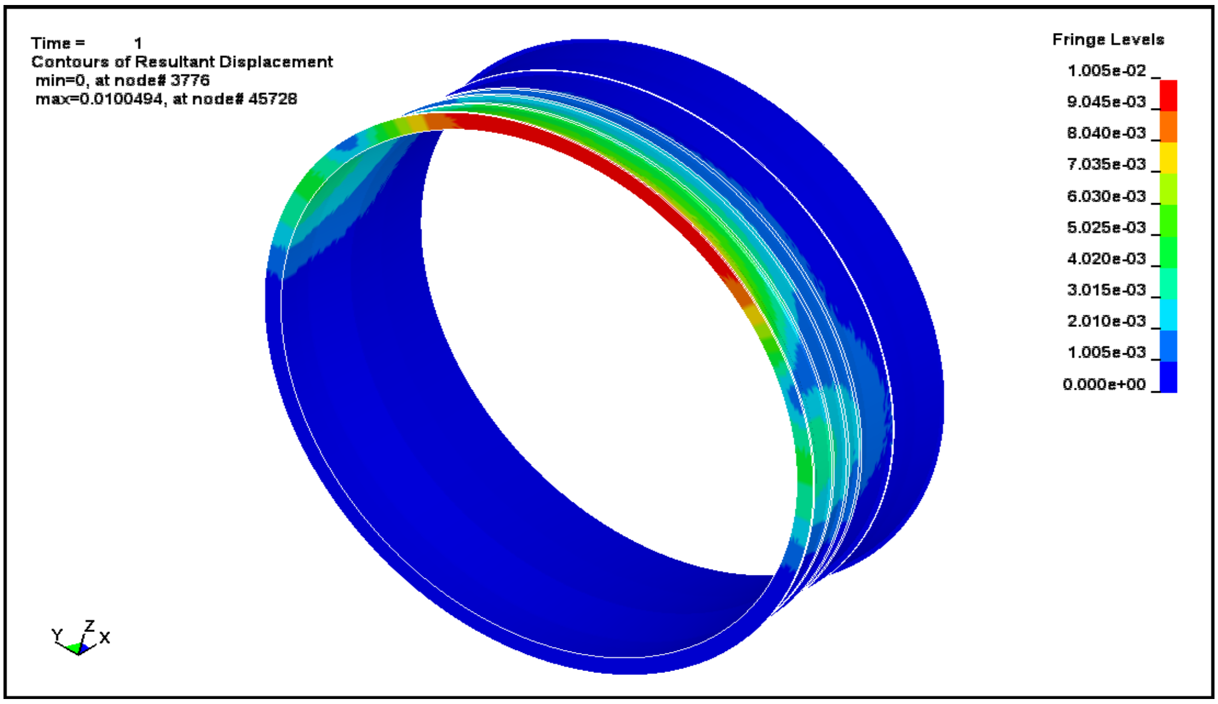
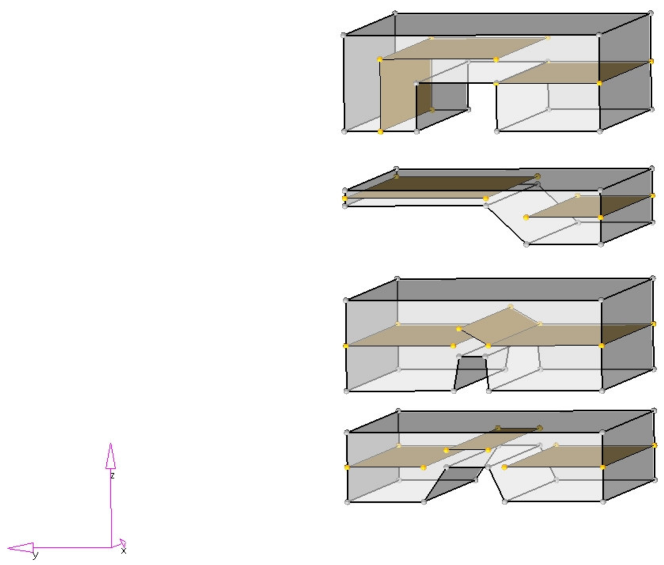


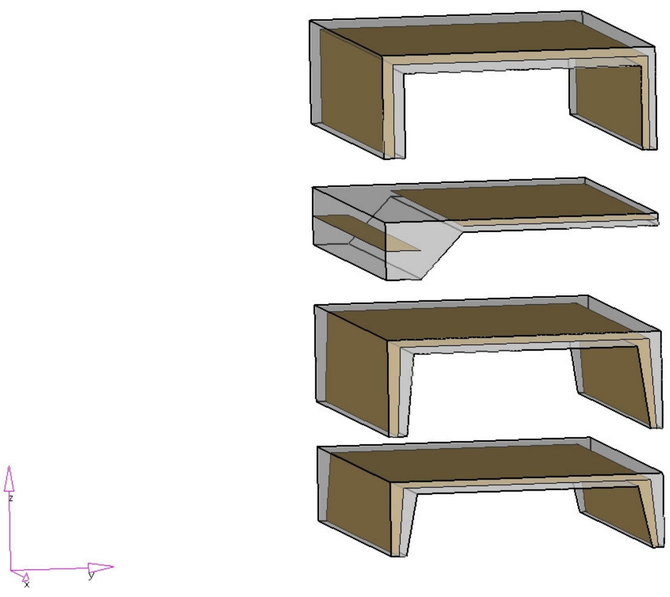
FIGURE 3.11: Resultant displacement of the shell model as seen in LS-Dyna

a midsurface shell model is not more than 5%. This shell model can be used as a low fidelity model in the optimization cycle with small run times. However, the major disadvantage in using the shell model currently is in the way midsurfaces are extracted. The process described revolves mainly around HyperMesh as the FE pre-processing tool. The heavy dependancy on a commerical package like HyperMesh and its black box style of operation is disconserting. Though HyperMesh is able to generate good midsurfaces in majority of the cases, the algorithm it uses to extract midsurfaces is not known for validation purposes. Some cases where HyperMesh fails to extract midsurfaces are shown in Fig. 3.12. It can be observed from this figure that when the thickness change of the solid geometry goes beyond a threshold value, HyperMesh fails to create a midsruface. Even when the solid sections are slender, there is one instance when HyperMesh fails to extract a midsurface.

It is clear from these figures that when the thickness of the geometric model changes beyond some set limits, HyperMesh fails to connect the midsurfaces and ends up with discontinuities. These limits in HyperMesh are not well documented and cannot be modified by the user. Under such circumstances, the user has to build the midsurfaces by extending



(a) Thicker sections - HyperMesh fails to extract midsurfaces in all instances



(b) Slender sections - HyperMesh fails to extract midsurfaces in just one case

FIGURE 3.12: Midsurface extraction capabilities of HyperMesh

and trimming surfaces manually.

As mentioned earlier, two methods of extracting midsurfaces are being considered. Of these, the face pairing method is not robust enough to handle complicated geometries. Since it is known that extracted midsurfaces could be discontinuous, a method needs to be developed which can connect these disconnected midsurfaces for stiffness and stress analysis.

The other option considered here is to create centerlines using the Medial Axis Transform and revolve these lines to create midsurfaces. To make the medial surface a viable option, it needs to be seen how well the mesh created on these medial surfaces represent the 3D model. A way to handle the medial branches, with or without them participating in the component stiffness also need to be studied further. In Chapter 4, a detailed study to connect discontinuous midsurfaces has been made. A model built using medial axis and its comparison with solid and other shell models is also presented.

Table 3.2: Summary of meshing methods discussed in Chapter 3

METHOD	ADVANTAGES	DISADVANTAGES
3D FE modeling	All features can be modeled	Complicated to create a complete brick model
	Any 3D model with an enclosed volume could be meshed	Too many elements
	Easy to create tet mesh	Difficult to maintain 2 elements along the thickness with tet meshes
	Possibility of automation	Tet mesh models are generally stiffer than brick models
Revolving a cross section mesh	Easiest of all methods considered so far	Cannot model holes on casings, bosses etc
	Can capture all axisymmetric features along the rotation axis	
	Possibility of automation	
	Element count and quality can be controlled	
Shell modeling using Midsurfaces	Can capture all holes and fillets if the midsurfaces are extracted from solid geometry	Features such as fillets and chamfers cannot be modeled
	Fewer elements, better simulation	Midsurface construction tools are not readily available for general structures
	Shell elements compare well with solid elements	
	Can handle any geometry, axisymmetric or not when applied to 3D solids	
	Possibility of automation	

Chapter 4

Extracting midlines of complex 2D sections

It is evident from Chapter 3 that shell elements modelled on the midsurface of a solid geometry compare well with solid FE models with respect to their stiffness. However, the fancase model, discussed in Chapter 3, is much simpler in topology when compared to compressor or turbine casings. The exact representation of these geometries using 2D FE models need to be made with great care since tip clearances depend heavily on these components. A test cross section has therefore been defined keeping in mind the most probable regions that might be challenging while extracting midlines. This cross-section has been defined based on the regions where HyperMesh failed to either create a midline or midsurface; refer to Fig. 3.12. In Chapter 3, the midsurface of the fancase solid model was created without any difficulties, however when the geometry gets complicated, commercial packages such as HM fail to generate a midline of the geometry. This leads to the need for manual intervention which is incompatible with automated search work flows.

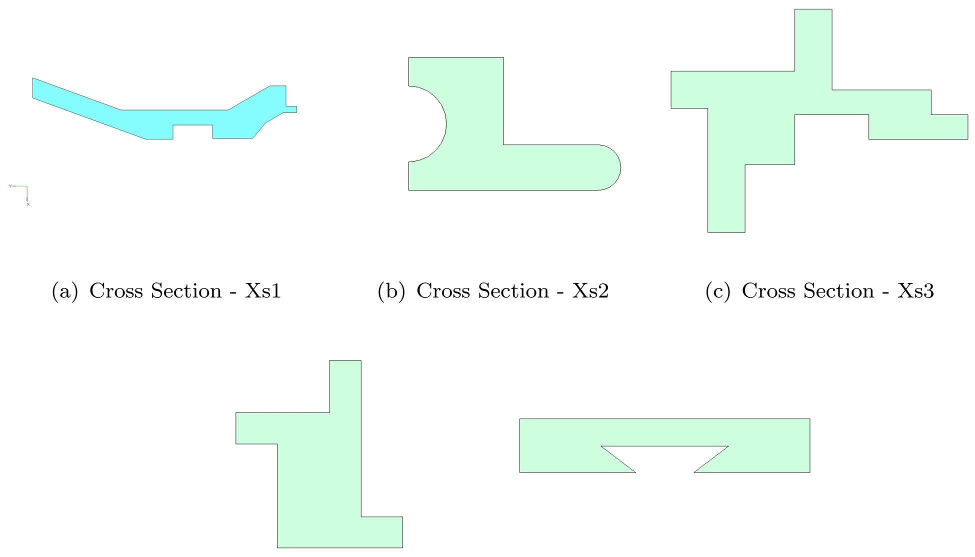
Under such circumstances, an algorithm needs to be coded (here either using Matlab or Pacelab) which works well for geometries that are of interest to this research program i.e. compressor and turbine casings). However, before developing an algorithm to extract the midline of a 2D geometry in batch mode, the midline extraction process has been studied manually. This midline will subsequently be used to create a surface. This surface which represents the midsurface of the solid model can then be used to create the 2D mesh with the thickness to represent the 3D geometry being modeled. The one big problem that needs some understanding is how to connect the shell elements together in the event of discontinuities of the midsurfaces generated. No such discontinuities were encountered in the fancase midsurface model. This made the generation and validation of the fancase shell model much simpler.

Bournival *et al.* [41], Trujillo[76], Pantano *et al.* [77], Puso[78] and Dohrmann *et al.* in [79], [80], [81] present techniques to connect elements of different dimensions with or without the use of constraint equations. A lot of research has been done to handle such problems, but not much work has gone into methods to model disconnected surfaces when they are offset by a large distance. This chapter however is entirely devoted to developing a procedure to model discontinuities if they are present in midsurfaces at the FE modeling stage. The main requirement is to generate midsurfaces without discontinuities. Section 4.7 describes the use of medial axis in working towards this aspect of FE model generation.

Many different cross sections that could pose problems while extracting midlines have been envisaged. Each was named as Xs1, Xs2...XsN. These cross-sections have been modeled based on geometries for which HyperMesh failed to extract midlines/midsurfaces as shown in Fig. 3.12 earlier. However only section Xs1 was chosen for the study because it had almost all the features a complicated aeroengine casing (excluding the Intercase) could pose. The cross-sections initially considered are shown in Fig. 4.1. It is evident from the cross-sections defined in the figure that the complexity posed by sections Xs2, Xs3, Xs4 and Xs5 are all dealt with in one way or the other by section 'Xs1'. This cross-section is studied using a variety of methods to build equivalent shell based FE models. The Xs1 section has an overall length of 200mm, a maximum thickness of 26mm and a minimum thickness of 5mm.

4.1 Xs1D1 : Face pairing

As already noted, cross section Xs1 leads to midsurfaces which are disconnected, see Fig. 4.2. The idea behind Xs1D1 is to link these discontinuities using shell elements as opposed to beams, bars or rigid elements. The resulting midline is as shown in Fig. 4.3. The midline is revolved to create midsurfaces which are in turn meshed with shell elements. However, the main question this approach poses while creating the FE model is the choice of thickness to apply to the connecting elements in Zone1 and Zone2. In this example a thickness of 10mm has been chosen for shell elements in Zones 1 and 2 which is around 38% of the thickest portion of this geometry. These shell elements replicate the effect of a stiff beam when used to connect such discontinuous regions. In case of a beam, a cross-section which defined a very



(a) Cross Section - Xs1 (b) Cross Section - Xs2 (c) Cross Section - Xs3

(d) Cross Section - Xs4 (e) Cross Section - Xs5

FIGURE 4.1: Cross sections considered to study the modeling techniques for midlines and midsurfaces

high stiffness would have been created. The advantage of creating shell elements is the use of just one type of element to define the geometry. The creation of shell elements on a group of surfaces can be done in batch mode using many commercially available FE softwares. A great advantage of this method arises when it comes to transferring the bending loads from one disconnected surface to the other. Shell elements are formulated to transfer these loads along their thickness.

4.1.1 Analysis

The stiffness of this component depends strongly on how the connections in Zone 1 and Zone 2 are made. Here the model is built in HyperMesh V9.0 while the FE Analysis was performed using ANSYS. Sufficient elements as shown in Fig. 4.4 have been used in order to capture the correct response/stiffness of the component. Analysis was performed for two load cases i.e. axial and radial loads. The loads and the boundary conditions are similar to the ones used in the fancase stiffness analysis study carried out in Chapter 3. When performing this analysis in ANSYS, the use of the 'Rigid Body Element' (RBE3) is different for the two load cases i.e. for axial load, the nodes on the entire circumference are used for application of the load, while for the radial load, only nodes on the top 100° are used. This can be done in ANSYS

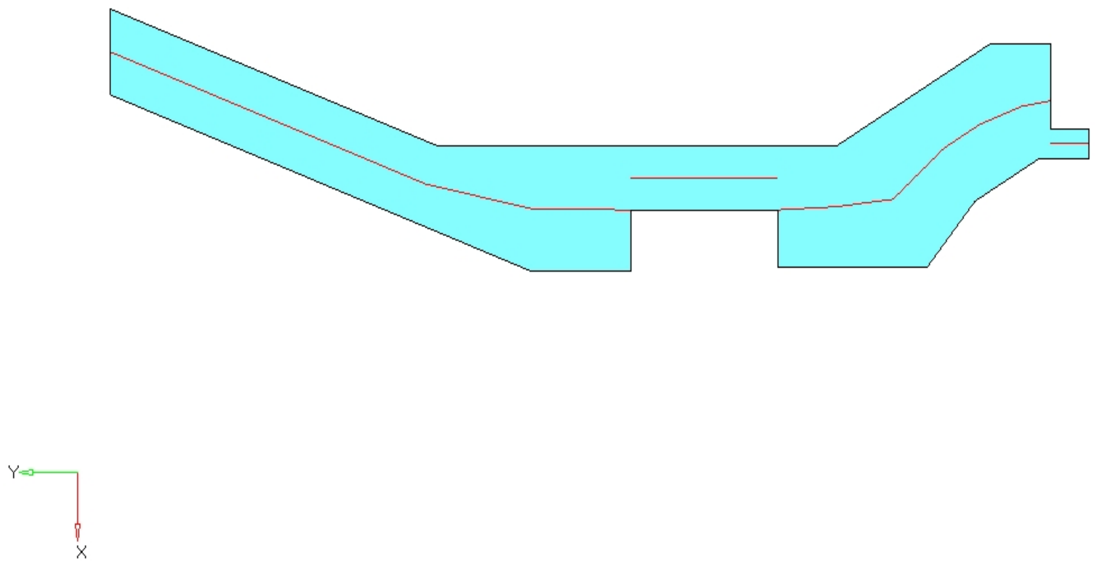


FIGURE 4.2: Midline extracted as per Xs1D1 method

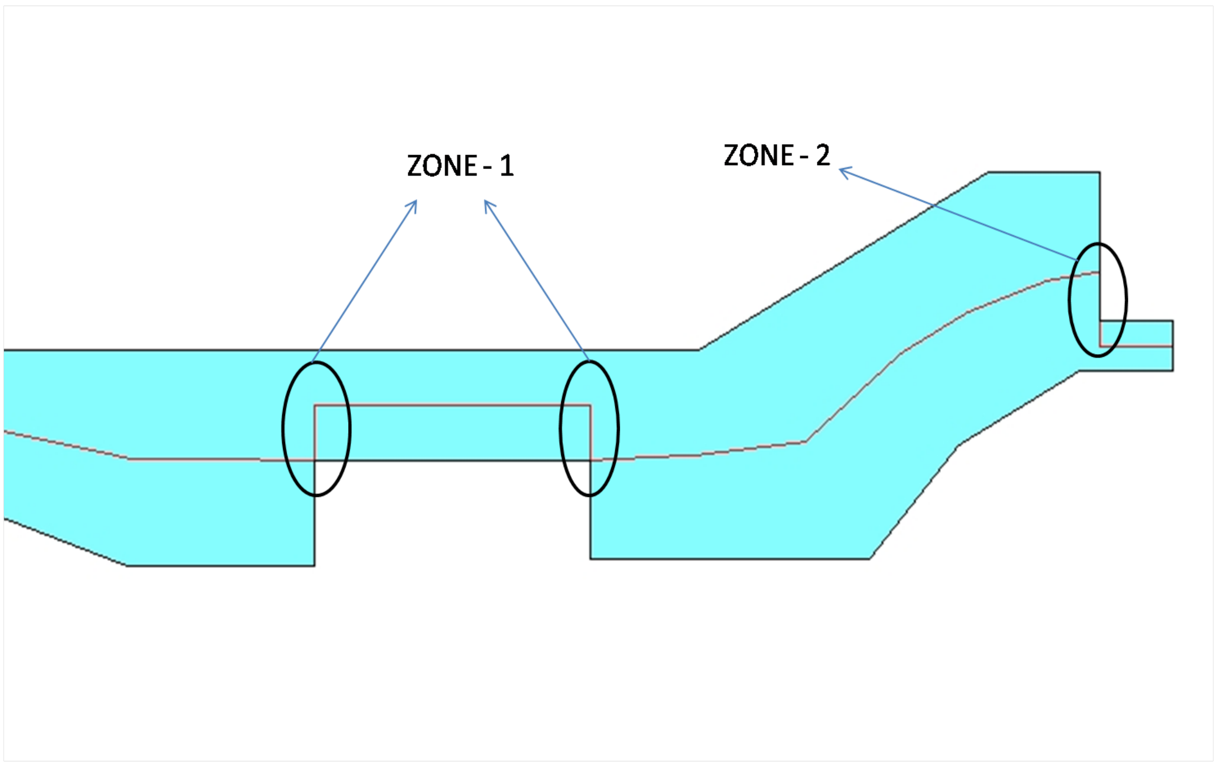


FIGURE 4.3: Xs1D1 midline connected using shell elements at the discontinuity

because the RBE3 is a boundary condition and therefore two different RBE3 configurations allow for two different load cases, refer [75].

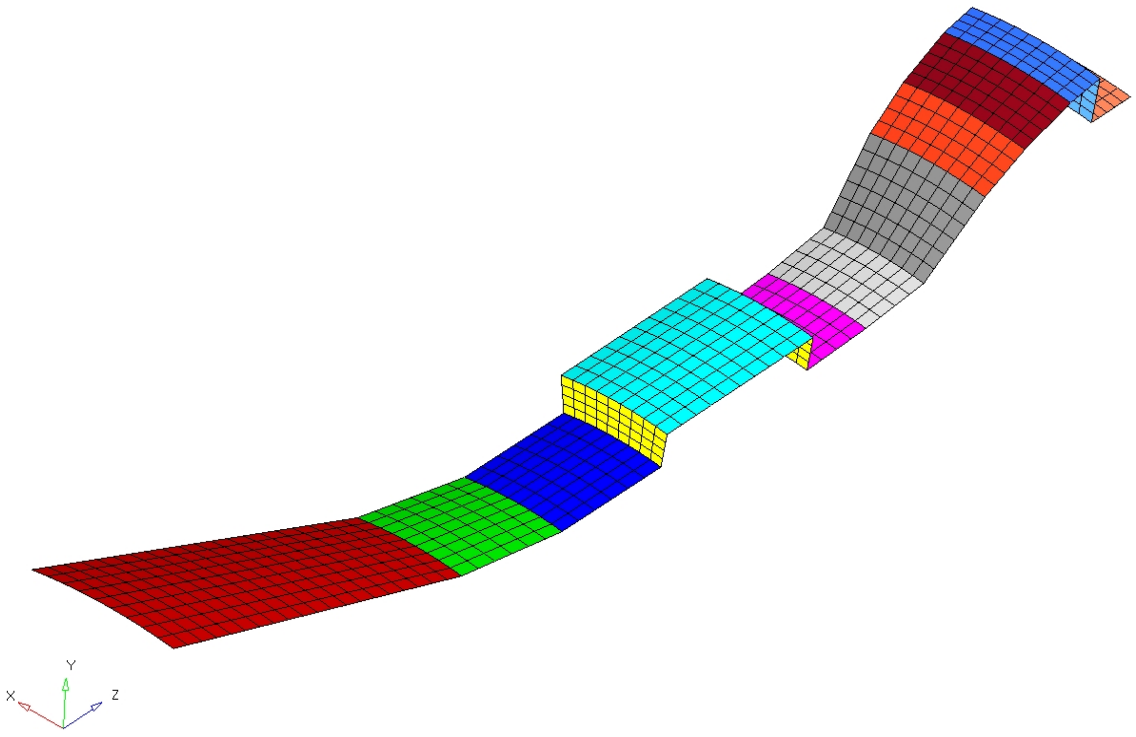


FIGURE 4.4: Shell mesh created on the Xs1D1 midsurfaces

The shell thickness is defined to each set of shell elements using real constants in ANSYS and these are based on the thickness of the solid model in each region. The thickness applied is realized as per the element normals. This generally turns out to be one reason for the disparity in results between shell and solid FE models when shell thickness offsets are defined. An arbitrary thickness of 10mm is applied to the elements connecting the disconnected parts. This model is analyzed with a Young's Modulus of 216,620 MPa, Poisson's ratio of 0.27 and mass density of 8x10^{-6}kg/mm^3 as before. When the whole model is analyzed with this material defined to all the elements, the mass of the FE model is around 13.08kg while the solid model weighs only 11.96kg ,i.e. a mass difference of 9.36%. This difference in mass is due to the thickness assigned to the shell elements connecting the disconnected shell elements in Zone 1 and Zone 2. This additional mass is not desirable when performing analysis using manoeuvre loads derived from gravitational forces.

In order to overcome this problem, another material without density but the same Young's modulus and Poisson's ratio was created. This material is applied to the shell elements used for connecting midsurfaces in Zone 1 and Zone 2. By doing so we make sure that the elements that have been defined with zero or no density still participate in the stiffness matrix but not in the mass matrix. Modal analysis for the first 10 mode shapes was also performed to make sure that the connecting shell elements in Zone 1 and 2 participate in the stiffness matrix, refer to Table. 4.4 . By assigning no density to the elements that are used for such connections, the mass of the model was now 12.36kg with a difference of 3.29%, down from the earlier 9.36%.

4.1.2 Results

The FE Analysis shows that when using such a modeling technique, the appropriate stiffness can be achieved but at the expense of additional mass. The results obtained from Xs1D1 and when compared to the displacements along the solid model's neutral line are as shown in Table. 4.1. When performing such an analysis, it is a general practice to look at only the master node deflection to calculate the radial and axial stiffnesses. However, for better understanding of the shell model behaviour, results at 40 different locations have been compared with the equivalent solid model. 10 critical locations have been chosen, each at 0.0°, 90.0°, 180° and 270° as shown in Fig. 4.5 for the solid model and as shown in Fig. 4.6 for the Shell model. The stiffness is calculated using the following relation:

$Stiffness = \left(\frac{Load}{Deflection}\right) N/mm$

On close examination of the results, it was observed that the results when queried on the external surface of the solid model were different from the results measured along the solid model's neutral line. Since the shell elements are modeled on the midsurface, the results in the solid model should be queried along the neutral line for an apple to an apple comparison. It is known that the results will vary through the thickness of solid FE models since they are based on the number of hexahedral elements modeled through the thickness. In this case, the results on the exterior of the solid model vary by 1.06% and 3.26% for the radial and axial loads when compared to the results from the neutral line. It is noteworthy to mention that the results of the shell model in Fig. 4.7(a) matched better when the results of the solid model were queried along the neutral line as shown in Fig. 4.7(b). Hence, the comparison is made at locations as shown in Fig. 4.5(with the solid model's neutral line) and Fig. 4.6. The results are output by ANSYS as a [10x4] matrix and Table. 4.1 shows the sum of displacements of the shell model compared to the sum of displacements of the solid model. The Xs1D1 shell model sum of displacements for the radial and axial loads are 0.38%

and 5.93% of the corresponding solid model respectively. Modal analysis of this shell model shows that the first 5 frequencies vary by $\leq 2.05\%$ as shown in Table. 4.4 without significant changes to mass and Center of Gravity.

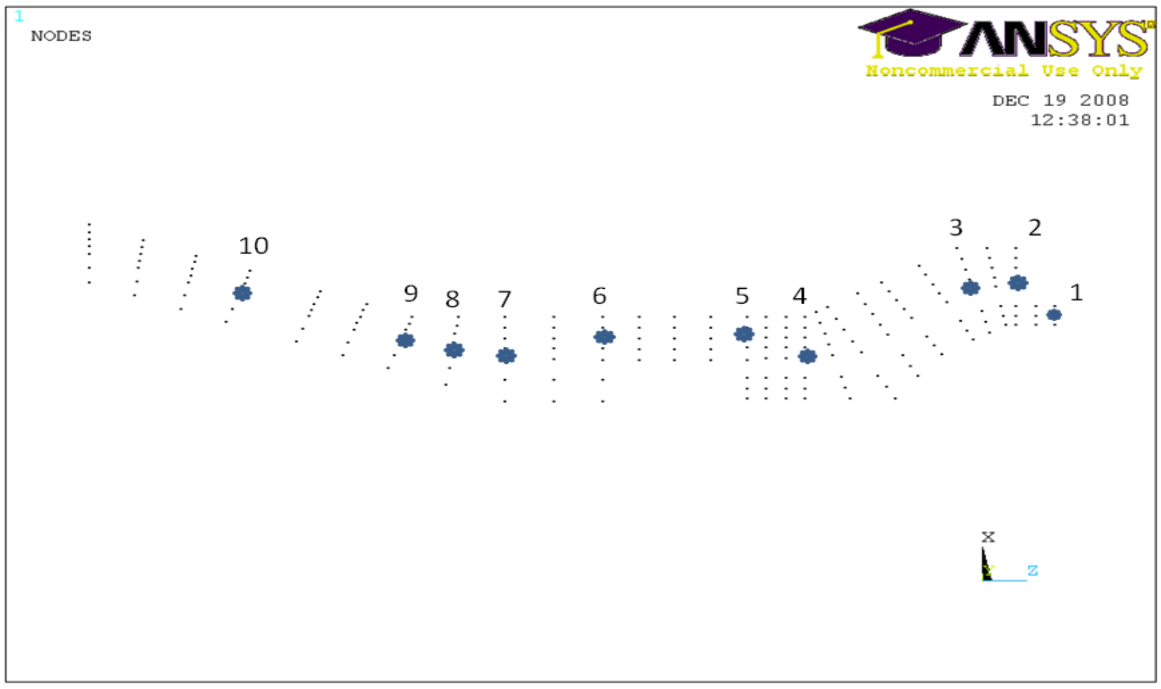


FIGURE 4.5: Node locations from solid model for comparison with other models

It is evident from the results that Xs1D1 captures the radial stiffness of the component with a difference of less than 8% in most cases when compared to the solid model. It is best to have such difference below 5%. A few spikes in % variation can be noticed at locations 1, 8, 9 and 10. The results also match well for the axial loading. There is a good match with the axial results differing by less than 5% in most cases but with spikes at locations 1, 8 and 10. It can be seen from the results shown in Table. 4.1 that Xs1D1 model is sensitive to changes in loads and thickness. Location 1 shows a difference of 12.22% in radial and 9.7% in axial stiffness. But locations 2,3 etc which follow soon after, present a better comparison. This indicates that the shell elements that form the loading region are too close to the forces applied and hence behaving as per the St.Venant's principle. This principle states that the localized effects caused by any load acting on the body will dissipate or smooth out within regions that are sufficiently far away from the location of the load. However, location 9 and 10 are away from the loads but closer to the boundary condition i.e; geometry is constrained

TABLE 4.1: % Difference of radial and axial displacements of the Xs1D1 shell model when compared with the solid model's neutral line

Xs1D1 shell model - Radial displacements (mm)								
Loc	0.0°	90.0°	180°	270°	%Diff 0°	%Diff 90°	%Diff 180°	%Diff 270°
1	0.003418	0.00122074	0.00065986	0.00122074	12.22	-7.14	-7.52	-7.04
2	0.00239884	0.0011709	0.00065383	0.0011709	1.16	-5.40	-5.78	-5.32
3	0.00199528	0.00109369	0.00063077	0.00109369	-1.31	-3.86	-5.85	-3.76
4	0.00114802	0.00082107	0.00052417	0.00082107	-3.62	-2.21	-5.28	-2.01
5	0.00094129	0.00068839	0.00047815	0.00068839	-2.83	-0.03	-2.96	0.16
6	0.00046209	0.00039711	0.00031966	0.00039711	5.08	0.17	9.11	0.30
7	0.00022964	0.00023985	0.0002083	0.00023985	3.51	7.39	15.70	7.43
8	0.0001433	0.00018002	0.00016346	0.00018002	4.72	12.48	20.46	12.48
9	0.00008696	0.00013061	0.00013145	0.00013061	-9.11	14.91	21.99	14.97
10	0.00003295	0.00003069	0.00006651	0.00003069	18.44	6.72	32.41	6.43
Xs1D1 shell model - Axial displacements (mm)								
Loc	0.0°	90.0°	180°	270°	%Diff 0°	%Diff 90°	%Diff 180°	%Diff 270°
1	0.0001934	0.00019341	0.0001934	0.00019336	-9.70	-9.71	-9.70	-9.68
2	0.00016354	0.00016356	0.00016354	0.00016351	-3.90	-3.91	-3.90	-3.88
3	0.00015682	0.00015684	0.00015682	0.00015678	0.24	0.23	0.24	0.27
4	0.00012559	0.00012561	0.00012559	0.00012557	-1.28	-1.30	-1.28	-1.27
5	0.00011088	0.0001109	0.00011088	0.00011085	-3.05	-3.07	-3.05	-3.02
6	0.00008178	0.0000818	0.00008178	0.00008176	5.35	5.32	5.35	5.37
7	0.00006114	0.00006117	0.00006114	0.00006112	12.53	12.49	12.53	12.56
8	0.00005532	0.00005535	0.00005532	0.0000553	9.90	9.85	9.90	9.93
9	0.00004868	0.00004872	0.00004868	0.00004866	3.22	3.14	3.22	3.26
10	0.00002032	0.00002042	0.00002032	0.00002032	-24.66	-25.28	-24.66	-24.66

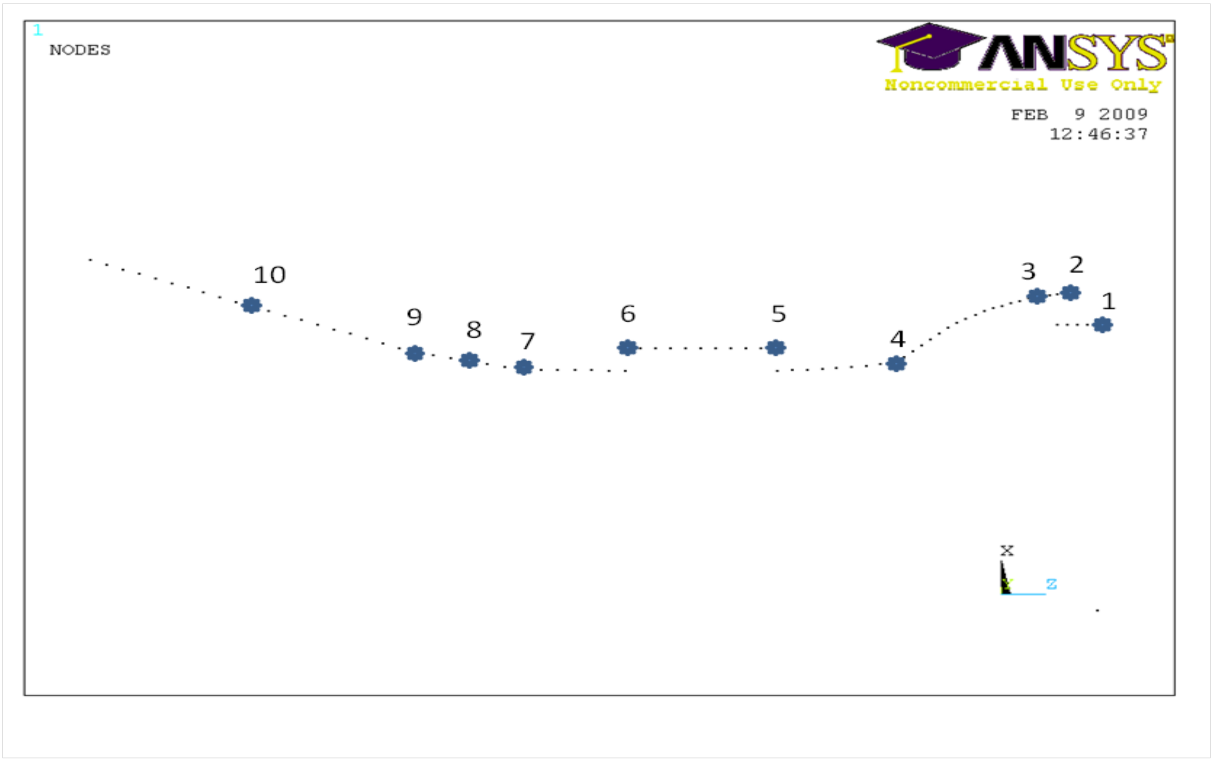


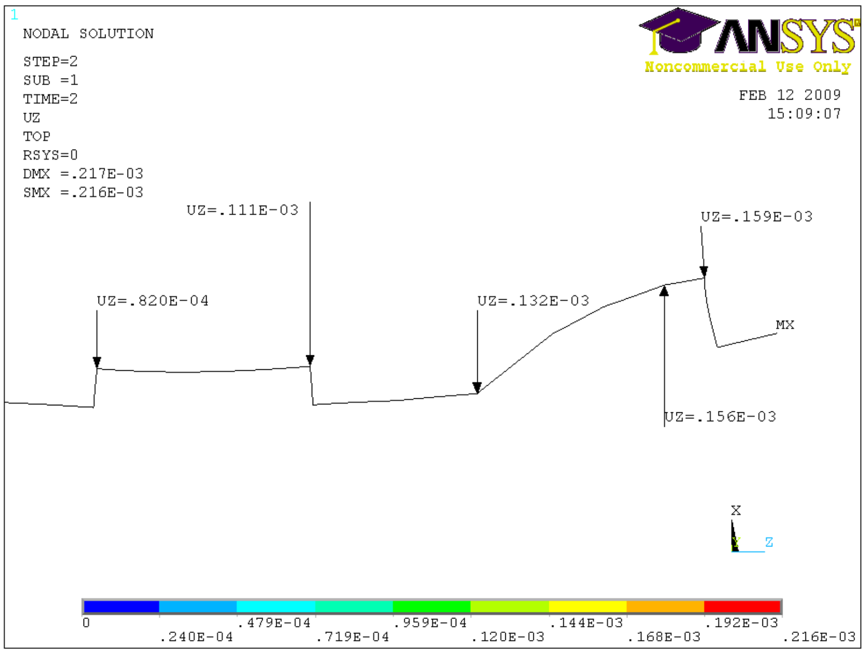
FIGURE 4.6: Node locations from Xs1D1 model used for comparison with solid model

in all directions near this location.

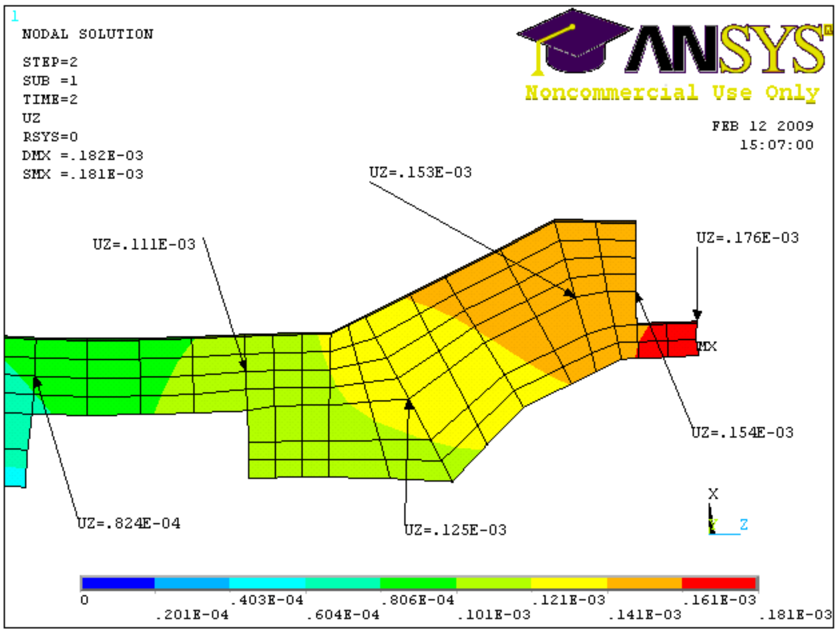
4.2 Xs1D2 : Shell without discontinuity

The idea behind Xs1D2 is based on the results from Xs1D1. One of the problems in Xs1D1 was the thickness that had to be assigned to Zone 1 and Zone 2 to obtain the correct stiffness of the component. It was also noticed in Xs1D1 that Zones 1 and 2 were needed because of the discontinuities in the midsurfaces created. The idea behind Xs1D2 is to avoid the creation of discontinuities while extracting midsurfaces. Though discontinuities cannot be completely avoided, an attempt has been made with a second model, Xs1D2 (all shell) to achieve a continuous midsurface model.

The FE model created for the Xs1D2 study is as shown in Fig. 4.8. In Xs1D2, Zones 1 and 2 have been moved from the actual discontinuity to a region with more material. This thicker portion of the model is where all the load transfer will eventually take place. The problem of what thickness to assign in Zone 1 and 2 of Xs1D1 model is still persistent in Xs1D2. All vertical surfaces are used to transfer loads from one horizontal region to another



(a) Xs1D1 Shell Model



(b) Xs1 Solid Model

FIGURE 4.7: Displacements of the Xs1D1 shell model and along the neutral line of the solid model

as shown in Fig. 4.9.

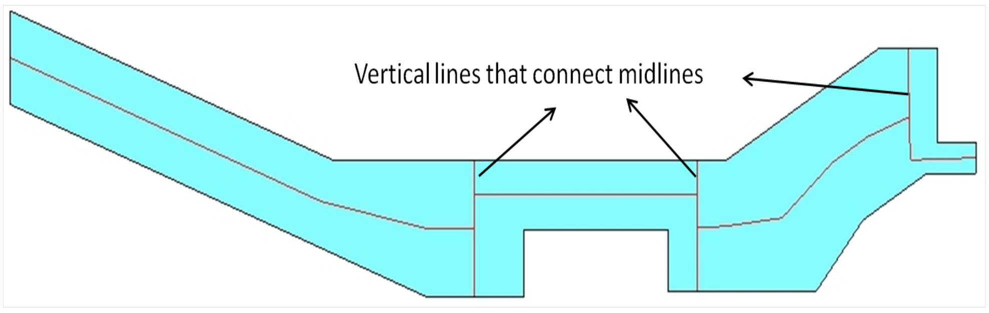


Figure 4.8: Midline extracted as per Xs1D2

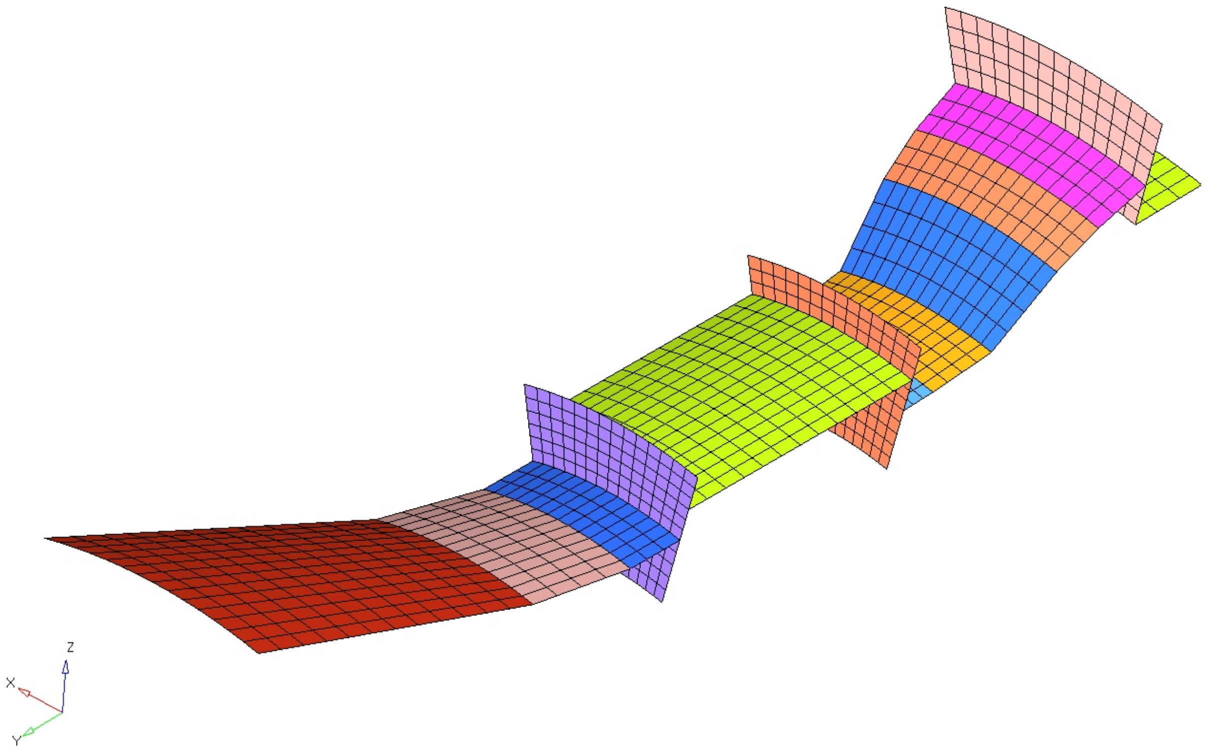


Figure 4.9: Shell mesh on the midsurfaces generated based on Xs1D2 midlines

This model has also been analyzed for both axial and radial loads in order to obtain the axial and radial stiffness of the component. All loads and boundary conditions are as in the Xs1D1 model. In Xs1D1, the modeling technique was able to successfully capture the radial stiffness. But Xs1D2 which was built based on the results of Xs1D1, has resulted in an FE model which has captured the axial stiffness better than the Xs1D1 model. The axial sum of displacements of Xs1D2 differs from the solid model by around 5.11% when compared to the 5.93% of Xs1D1 as shown in Table. A.1. However, the sum of radial displacements has increased from approximately 0.38% in Xs1D1 to 7.11% in Xs1D2 (for detailed results refer to Table. B.1 in the appendix) with a substantial increase in mass. The increase in mass can be countered as explained earlier by using different material properties. It is evident from Xs1D2 that an optimal combination of thickness in Zone 1 and 2 might eventually lead to capturing both axial and radial stiffness of the component with a better accuracy.

It can be concluded that the difference in results between Xs1D1 and Xs1D2 is not much. The major difference between the two models can be highlighted when the process of midsurface generation has to be automated. Xs1D1 uses a very straight forward model in which the midline starts and stops at every intersection and shell elements are used to connect them. But in Xs1D2, the location to place the vertical lines is not generic. A good understanding of the geometry is needed to create such a shell model.

4.3 Xs1D3 : Solid-Shell Model

Xs1D3 model is built based on what is called a "shell- solid sandwich". This is an FE modeling 'trick' to avoid the complexities involved while trying to join 2 elements with different degrees of freedom. As a theoretical practice, these connections between dissimilar elements are made by creating appropriate constraint equations to connect them. By doing so, the displacements i.e. translations and rotations are transferred from one element to the other in a compatible manner. But creation of too many constraint equations can increase the problem size thereby increasing the overall solution time during analysis. Xs1D3 as shown in Fig. 4.10 tries to connect shell and solid elements using the concept of sandwiching at least 2 shell elements inside the solid elements. The thickness of the sandwiched shell element is kept the same as its parent body and material as shown in Fig. 4.10(all elements sharing the same colour have the same thickness).

This sandwich elements modeling procedure is more of an FE trick than a standard procedure while modeling such problems. It is used when connecting compressor and turbine blades (shell elements) to their respective discs (solid elements). But this is done mainly to capture the mass of the structure and not stiffness. Therefore, the load transferring capacity of such a technique has to be investigated and is the reason why Xs1D3 has been built. One of the greatest advantages of this technique is that Zone1 and 2 which caused discontinuities in the midsurface, have been eliminated by modeling those junctions using solid elements as shown in Fig. 4.10. By doing so, it was expected that the stiffness lost at the midsurface

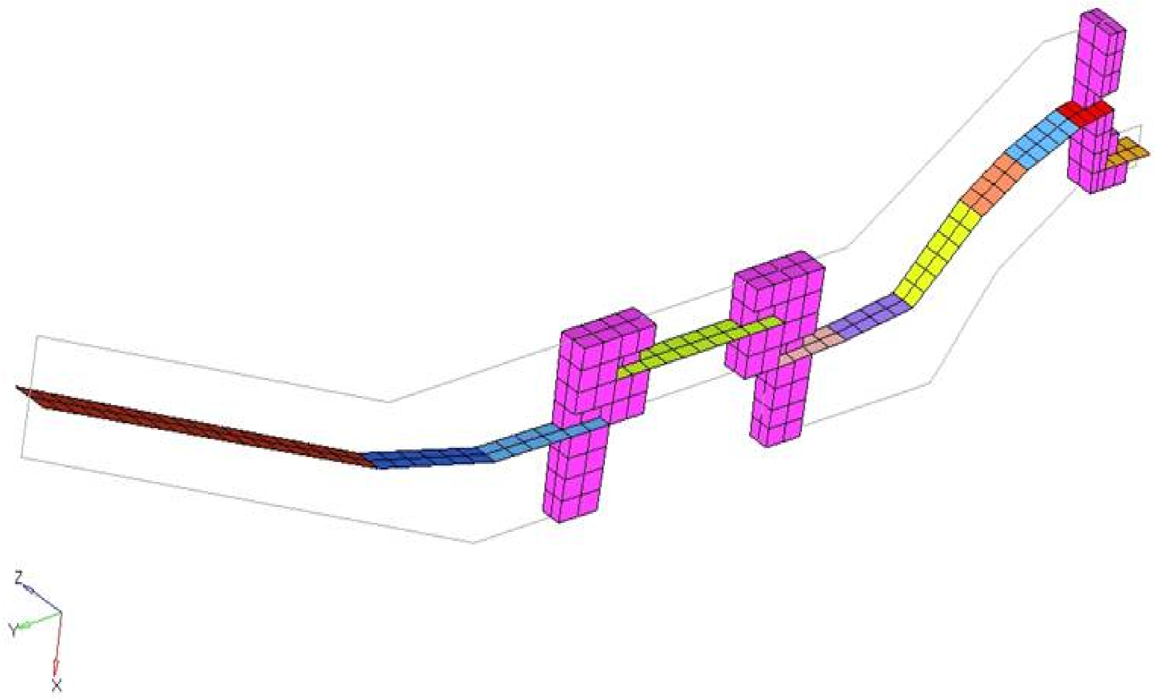


FIGURE 4.10: Midline with shell and solid elements in Xs1D3

discontinuity will now be captured by the solid elements. The loads and boundary conditions are the same as in Xs1D1 and Xs1D2.

The analysis shows that the transfer of loads or displacements from the shell elements to the solid elements and vice-versa is not as expected. This method of modeling does not cause any rigid body motion which means to say that the model built using shells and bricks behaves as one part. However, the difference in radial and axial deformation as shown in Table. A.1 for Xs1D3 model is around 18.81% and 75.11% respectively. The primary reason for this is the difference in the element configurations i.e. shell elements have 6DoF while solid elements have 3DoF. The general practice in such scenarios is to connect the two elements using 'Constraint Equations' or without them as mentioned in [41]. However, constraint equations increase the problem solution time significantly and hence are not recommended when building large models.

4.4 Xs1D4 : Coupling

The “Coupling” of nodes is the method chosen for Xs1D4 to connect the disconnected midsurfaces. Of the various modeling techniques described so far, this technique provides reasonable results. The major problem is that the stiffness at Zones 1 and 2 as shown in Fig. 4.3 cannot be modeled accurately with simple couplings. The coupling of two nodes ensures that these nodes behave in a synchronized manner and are as shown in Fig. 4.11 . They would move together and behave as welded joints. The solid model being dimensionally reduced in this case has different stiffnesses at the disconnected regions and modeling this using coupling is not possible. The radial loading of the component on the nodes comprising of the top 100° of the part means that the component will take an oval shape during deformation. This behaviour of the component during loading needs to be captured by the shell model.

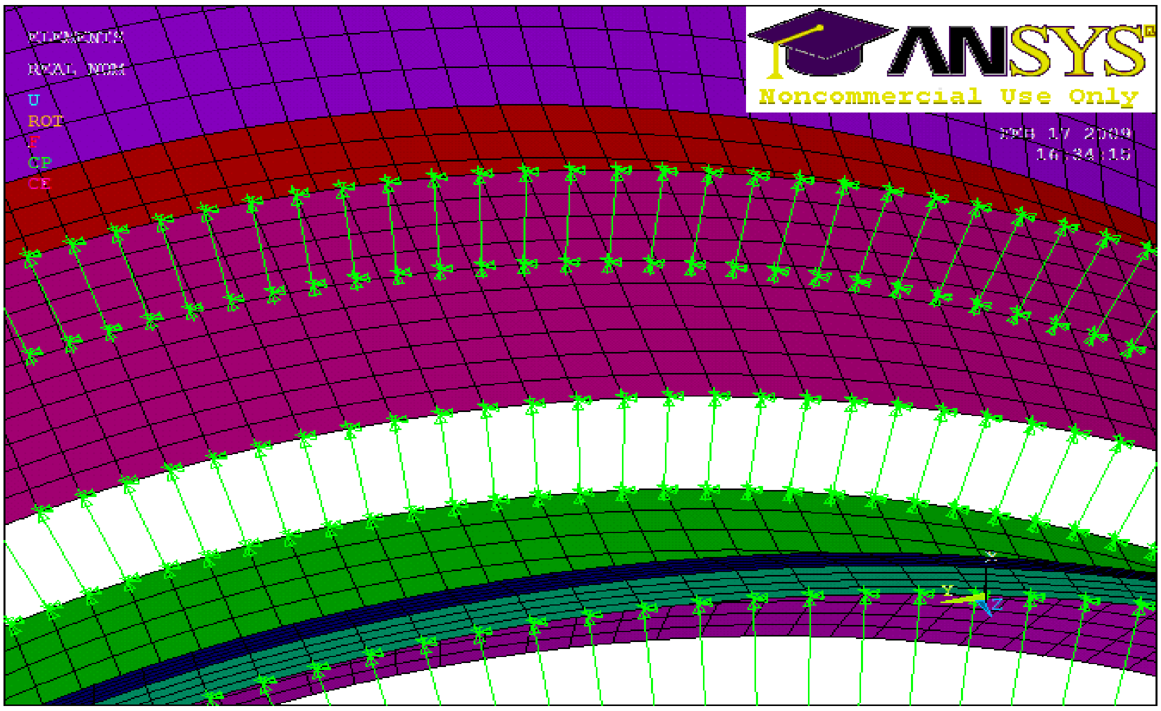


FIGURE 4.11: Xs1 model built by coupling nodes at the discontinuities

This model is built with shell elements on the midsurfaces extracted and no attempt is made to connect them using either shell or solid elements as described in Xs1D1 - Xs1D3 methods described earlier. The disconnected elements are simply connected by coupling their nodes (degrees of freedom) to those nodes that directly lie above or below them offset by some distance. Each node should be coupled in all the degrees of freedom that it exhibits.

In this case study, these nodes have been coupled manually by specifying a tolerance based on the distance by which these nodes are separated. This means that each node that forms a shell element needs to be coupled in 6 DoF, i.e. 3 translations and 3 rotations.

Coupling can be used to model various joint and hinge effects in ANSYS. A more general form of coupling can also be applied with constraint equations. All the nodes that form the coupled set are rotated in the cylindrical coordinate system. As a result of this coupling, these nodes are forced to take the same displacement in the specified nodal coordinate direction. The amount of the displacement is unknown until the analysis is completed. A set of coupled nodes which are not coincident, or which are not along the lines of the coupled displacement direction, may produce an applied moment which will not appear in the reaction forces, refer to [82].

As mentioned earlier, the value by which a coupled node is displaced is decided after the analysis has been performed. This means to say that coupling does not model the stiffness of the component at the discontinued location, but just transfers all displacements from one node to another.It is evident from the results shown in Table. A.1 that the sum of the radial displacements of the component produced by Xs1D4 model varies by 9.84% and the sum of axial displacements varies by 2.63% approximately when compared to the solid model.

On close examination of the results it is evident that the Xs1D4 model with coupling exhibits more stiffness than the solid model and hence a bigger difference in results, refer to Table. B.2 in appendix. It is also noteworthy to mention that in ANSYS a set of coupled nodes which are not coincidental, or which are not along the line of the coupled displacement direction, produce an artificial moment constraint. If the structure rotates, a moment may be produced in the coupled set in the form of a force couple. This moment is in addition to the real reaction forces and make it appear that moment equilibrium is not satisfied by just the applied forces and the reaction forces. This may lead to a nonphysical response in some cases. Detailed results are presented in Appendix. B.

4.5 Xs1D5 : Constant spring stiffness (no coupling)

The XS1D4 model used couplings to link the disconnected nodes to form a continuous structure. However, the stiffness at each of these discontinuities cannot be controlled when couplings are used. In order to counter this problem a model using linear springs to connect the discontinued midsurfaces (elements on midsurfaces) as shown in Fig. 4.12 was built. The problem however is the choice of stiffness values to be assigned to these springs. Many iterations were performed using springs to obtain a shell FE model which represented the same stiffness as the solid FE model. An initial stiffness of 1x10^5N/mm was used to run the model and a final value of 4x10^6N/mm was chosen after various iterations. The model behaviour varied each time the spring stiffness was modified.

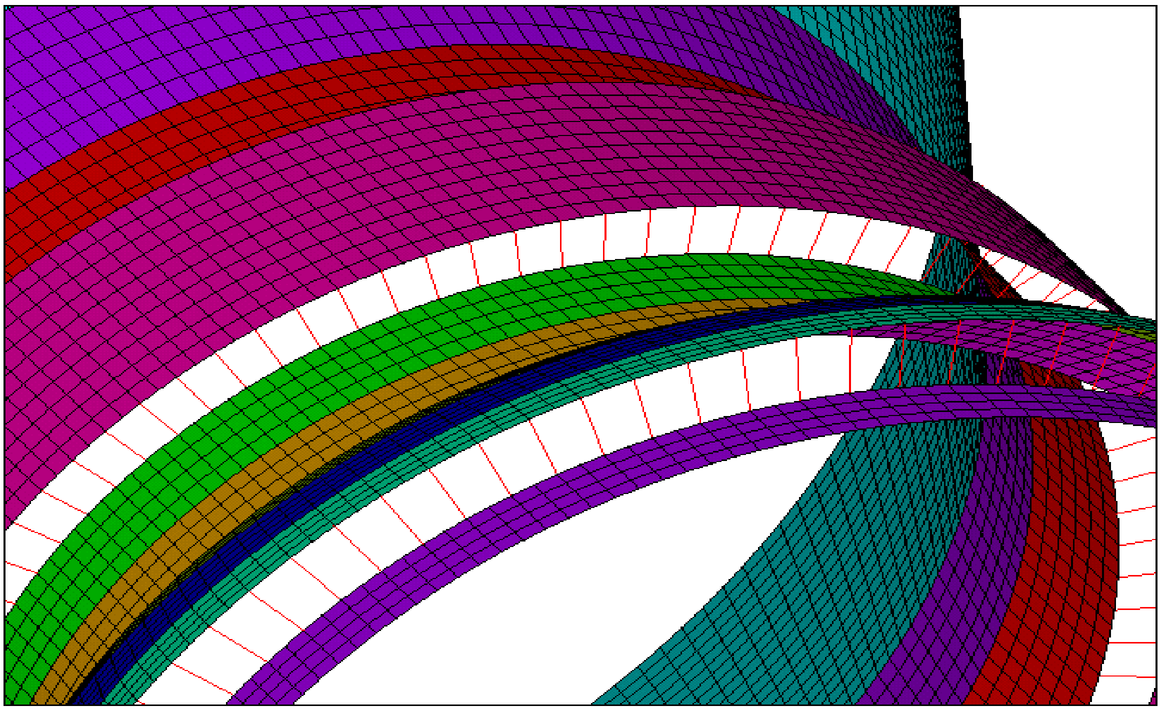


FIGURE 4.12: Xs1 model built by creating springs at the discontinuities

It is known that spring elements in ANSYS act in the local coordinate system. In order to make sure that the load transfer from one element to another takes place correctly, the nodes have to be defined in the cylindrical coordinate system. 6 springs are created between 2 nodes that are at the discontinuous midsurface region , refer Fig. 4.12. In this case, a uniform spring stiffness has been chosen for all 6 sets of springs. Iterations were performed with 1x10^5, 1x10^8, 1x10^{15} and 4x10^6 as real constants (stiffness values) for the spring elements. The spring stiffness of 4x10^6 was chosen based on results obtained from 1x10^5 and 1x10^8 stiffnesses.

Only the model built using the stiffness value of 4x10^6 is discussed in this section. The results shown in Table. A.1 are based on the sum of displacements in the axial and radial directions at 40 locations on both the Xs1D5 and the solid model. A more detailed presentation of results for a spring stiffness of 1x10^{15} N/mm is made in Table. B.3 in the appendix. It is evident from the results that the sum of radial displacements of the component produced by this method varies by 7.37% when compared to the solid model and the axial stiffness varies by 3.48% approximately. A spring stiffness of 4x10^{6} provides a better balance between stiffnesses in axial and radial direction for this model than the spring stiffness of 1x10^{5}, 1x10^{8} and 1x10^{15}. It is also noteworthy to mention that the results obtained from the direct coupling model (Xs1D4) and Xs1D5 with 1x10^{8} and 1x10^{15} as stiffness match with minor variations is as expected. Xs1D5 results pertaining to 1x10^{15} spring stiffness are presented in Appendix. B.

Using springs with constant stiffness in all 6dof, it is possible to produce results that match the solid model either for radial load or axial load but not both. The model behaviour for the axial and radial loads was studied more closely only to find out that the midsurface extracted for a section as in Xs1 makes the model flimsy in the axial direction , refer to Fig. 4.13. It is evident that the deflection at one end of the spring is the same as the other end of the spring, i.e. the spring transfers the displacements seen on one node to the other without any losses. However, in this case it was assumed that the shell element did not capture the stiffness in the region as shown in Fig. 4.13. A couple of changes were made to capture the correct axial stiffness of the component at this zone but without any positive results. In order to counter this problem, a stepped model was envisaged which was thought of as a probable answer to modeling a midline in such complex regions to capture the axial and radial stiffnesses of the component more effectively.

4.6 Xs1D6 : 1 set of optimized springs

At this stage a stepped model as shown in Fig. 4.14 was built in order to capture the complex region, refer to Fig. 4.13, where there is no obvious midline to extract. It was evident from the results obtained from the original spring model described in the previous section that, at one optimum spring stiffness, both the radial and axial stiffness of the component could be captured only moderately well. This was assumed since the results for the radial and axial sum of displacements varied from 78.85% and 119.86% to 4.34% and 7.67% for 1x10^{5} and 4x10^{6} spring stiffnesses respectively as shown in Table. A.1. This meant that there existed a spring stiffness in between 1x10^{5} and 1x10^{15} that would lead to near zero sum of difference for both radial and axial displacements. In order to ascertain this, an optimization

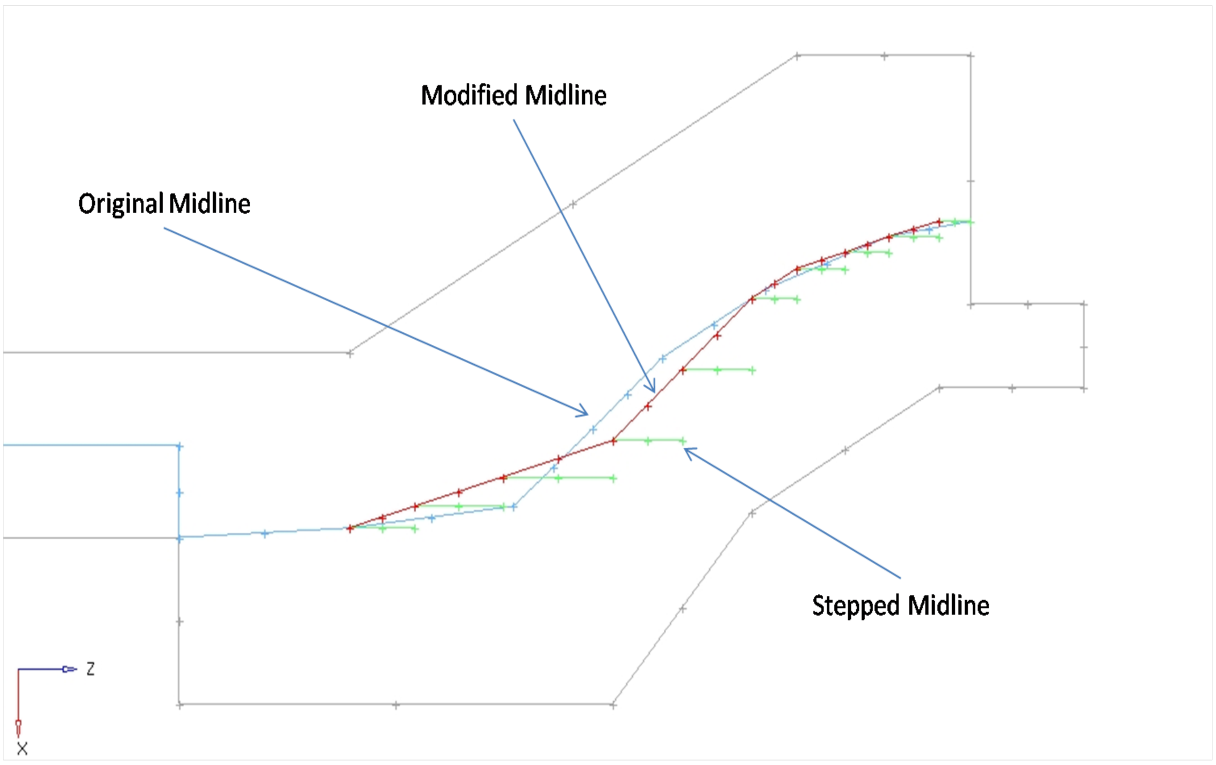


FIGURE 4.13: Evolving into a stepped midsurface model

run was set up to obtain the best spring stiffness (6 spring stiffness to connect the 6 dof of the shell element). The problem was set to achieve 2 objectives, one was to match the axial displacement of the shell model with the solid model and the other was to match the radial displacement of the shell model with the solid model. These objectives needed to be minimized since they were the difference of the results between the solid and shell models (sum of differences).

The spring elements in ANSYS (COMBIN 14) act in the nodal coordinate system and so all the nodes were rotated into a cylindrical coordinate system before performing the analysis. 6 springs each in UX, UY, UZ, ROTX, ROTY and ROTZ directions were created at each step of the stepped model as shown in Fig. 4.15. One real constant is defined in ANSYS for each element that defines a spring in one direction. An optimization run was set up to find an optimum value for all 6 spring settings used between two nodes using OPTIONS based on Voutchkov *et al.* in[83]. The optimization run was made using the NSGA2 (Multi-objective Optimization search method) to minimize the objective functions. Consider (Xr) and (Xa) as the 10x4 matrix of radial and axial displacements of the solid model and (Sr) and (Sa) as the 10x4 matrix of radial and axial displacements of the spring based shell model. The

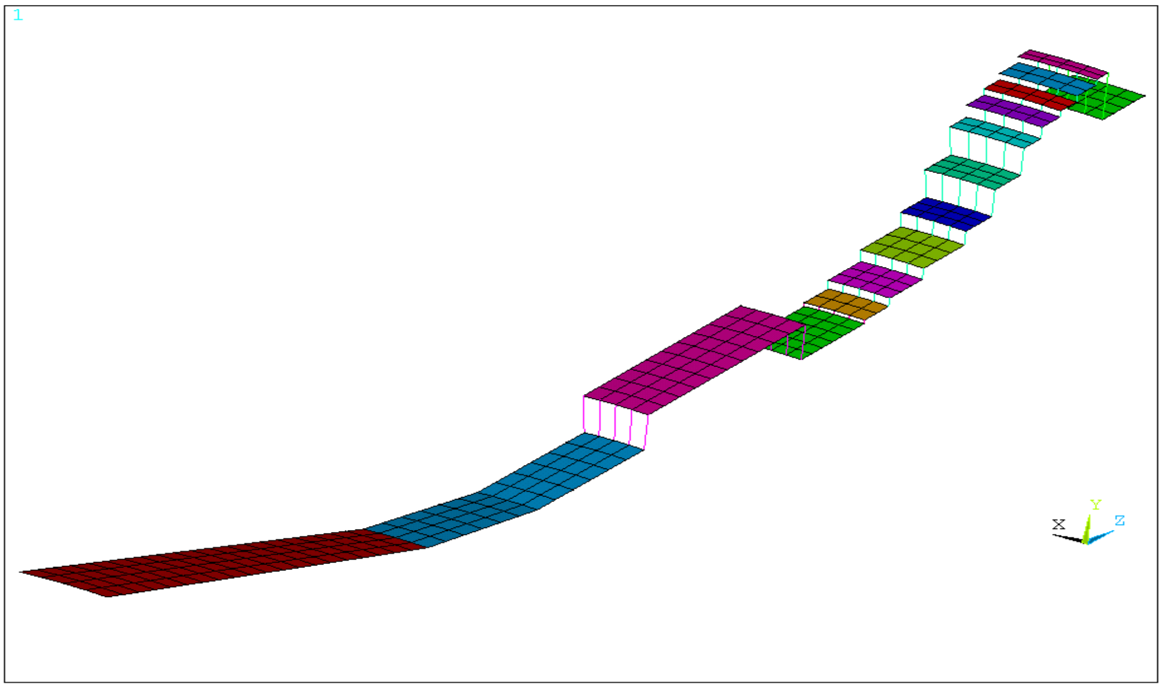


FIGURE 4.14: Step method used to model the front end of cross section Xs1

two objective functions to be minimized are then given by;

$$f(x_1) = 1 * 10^{10} \left(\sum_{i=1}^{10} \sum_{k=1}^{4} \left(Xr_{(i,k)} - Sr_{(i,k)} \right)^2 \right) \tag{4.1}$$

$$f(x_2) = 1 * 10^{10} \left(\sum_{i=1}^{10} \sum_{k=1}^{4} \left(Xa_{(i,k)} - Sa_{(i,k)} \right)^2 \right) \tag{4.2}$$

All loads and boundary conditions are the same as the other analysis runs. An ANSYS stepped model was set up to run in batch mode every time the optimizer came up with new real constants for the 6 springs that were used throughout the model. The optimizer was set up to run 40 generations with a population size of 30 (1200 iterations in all) and the time taken for each analysis run with two load cases (radial and axial load case) was approximately 1.5min.

At the end of the optimization run, a set of optimal spring stiffnesses pertaining to a design point was chosen from the Pareto front. The values extracted for the objective $f(x_1)$ and $f(x_2)$ was 0.000024mm for radial displacements and 0.000007mm for axial displacements

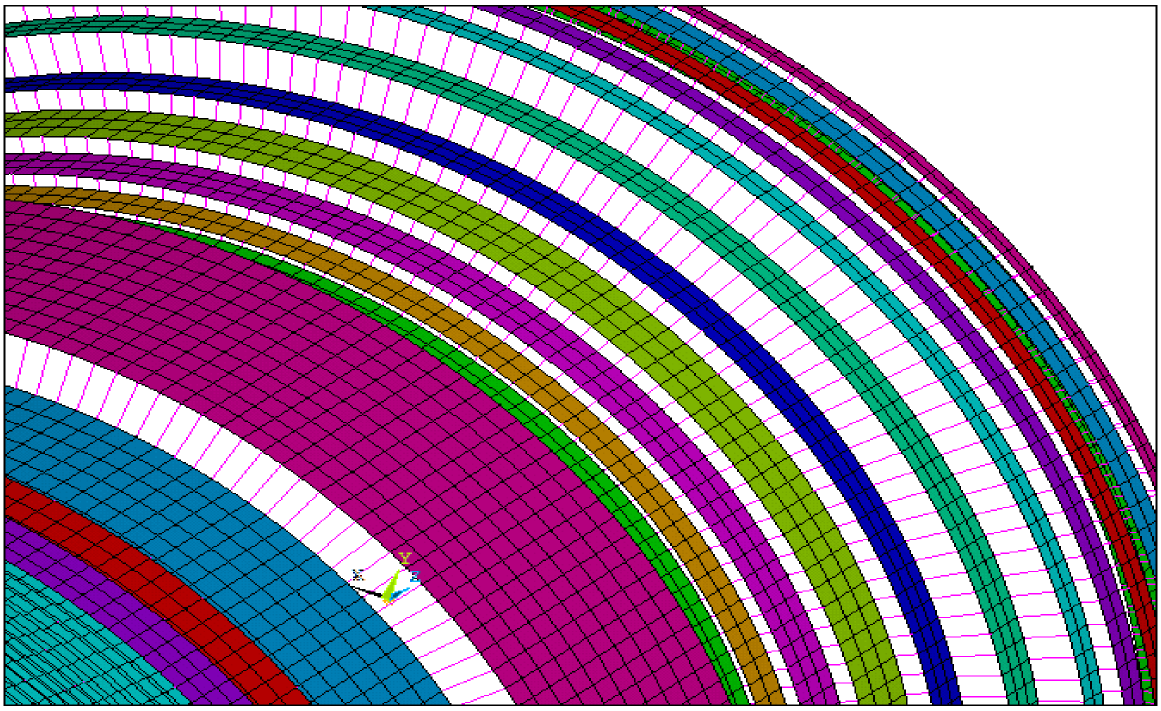


FIGURE 4.15: Springs used to connect each of the steps in the Stepped Shell model

respectively. This was acheieved by using 2.06x10^{4}, 1.11x10^{11}, 4.42x10^{6}, 6.97x10^{9}, 5.06x10^{13} and 7.09x10^{14} as stiffnesses for springs operating in UX, UY, UZ, ROTX, ROTY and ROTZ directions respectively.

The analysis results obtained by using the spring stiffness from the optimization run shows that the model behaviour is sensitive to radial and axial loading. The spring elements are defined in the nodal coordinate systems. This allows for easy directional control, especially in the case of two noded elements with coincident nodes. If UX, UY, or UZ degrees of freedom are being used, the nodes are not coincident, and the load is not acting parallel to the line connecting the 2 nodes, there is no mechanism for the element to transfer the resulting moment load, resulting in loss of moment equilibrium. Due to this reason, a model built based on springs to connect the disconnected surfaces is questionable when the nodes making up the springs are not coincident. The results for this model are as shown in Table. A.1. By considering the model to be made of four parts and separating them with four sets of springs would probably lead to a better approximation of the shell model. An optimization run was set up just as per the previous analysis, but this time the only difference was that the optimizer was set to handle 24 variables instead of 6. Though, it was clear from the beginning that the spring model would not match the axial load results of the solid model (due to the moments that are created), this analysis was performed to learn more about how the optimizer could be used to achieve such objectives. The results of the 24 variable model is as shown in Table. A.1. A model has also been built to study the effect of springs when connecting joints that are offset by a distance for axial loads and is listed in Appendix. C.

4.7 Xs1MAT : Medial Axis Transform

As discussed in Chapter 2, a medial axis inscribed inside a 2D face can be used to create midsurfaces and shell elements. Price *et al.* in [84] describe what operations a medial object toolkit should perform to present data in a readily understandable form such as aspect ratio and minimum thickness values. This work has been implemented as a Medial Object (MO) Toolkit into TranscenData's CADfix package [85]. However, in this section, a medial axis tooklit [53] developed by Suresh Krishnan using Matlab has been used to extract the midlines of the Xs1 cross-section. The exercise in this section is more concerned with the way shell meshes are produced and used when created using the raw medial curves. The tools that are used to create these medial curves and their pros and cons will be discussed later.

4.7.1 Analysis

The MA (Medial Axis) model for this case study is created using the Matlab toolkit as mentioned earlier. The geometric points that form the start and end of each line segment of a cross-section form the input to this code. Medial point data is generated when the code is run. These medial points define the MA of the cross-section and have a thickness associated with them. All convex medial points (mostly vertices) have zero thickness. The preprocessing needed to create a shell model using these medial points is cumbersome but can be automated without much difficulty. The centers of all the circles that are inscribed inside the cross-section form the medial points and are represented in Fig. 2.7. The thickness at each point is defined by the diameter of a circle that is inscribed within the edges of the cross-section. Once the medial points are known, a line segment connecting two medial points is created. This line is meshed with line elements in HyperMesh and then revolved to yield shell elements. It is always a good practice to make sure these lines created and the corresponding shell meshes created are directed into specific collectors within HyperMesh, i.e. one collector per line since each line and its corresponding elements will have a different thickness. Such segregation using collectors helps when applying thickness since element picking can be done based on collectors and not based on surface numbers or element locations. Fig. 4.16 shows the shell mesh created using the MA in their respective collectors. Each of these collectors can be named appropriately and a thickness assigned to them. Any element moved or created into these collectors will have the thickness assigned to them. This allows for ease of automation when used as part of an optimization work flow.

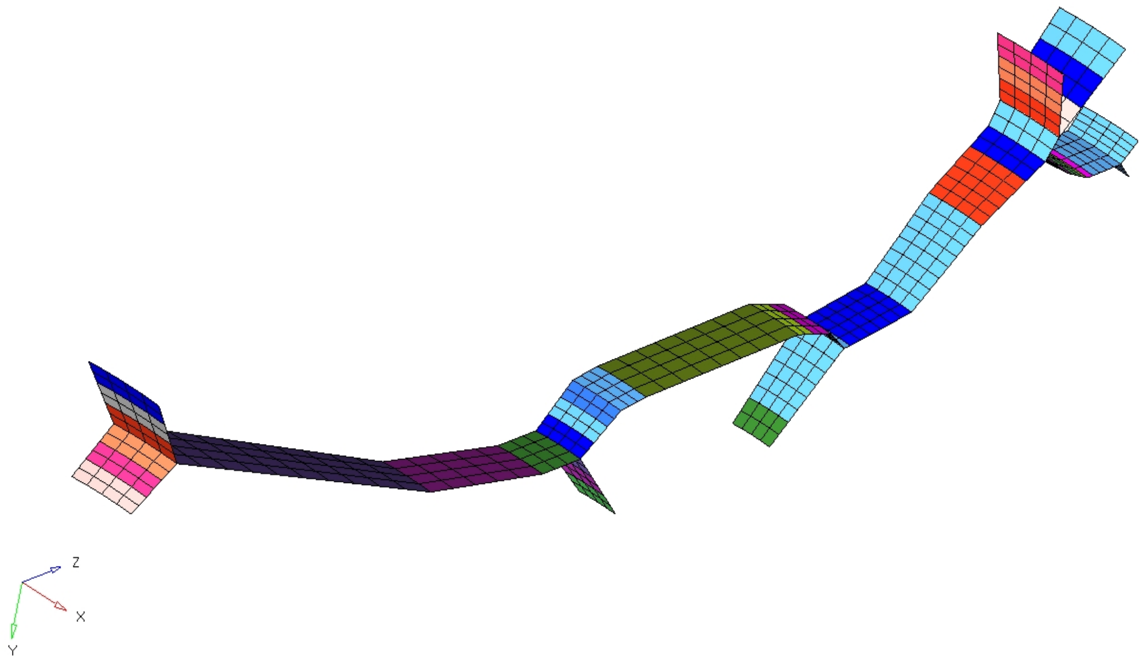


FIGURE 4.16: Mesh created using lines connecting medial points in their respective collectors/groups

Though all vertices have zero thickness, in this case the shell elements that form the end of branches have been assigned a small thickness value to avoid the solver from crashing. The shell elements in ANSYS with their thickness 0.1 times the actual thickness is as shown in Fig. 4.17. The loads and boundary conditions are the same as used for other modeling techniques described in the earlier sections.

4.7.2 Results

The radial and axial sum of displacements for Xs1MAT vary by 7.59% and 9.26% when compared to the solid model as shown in Table. A.1. The results of the MAT model are

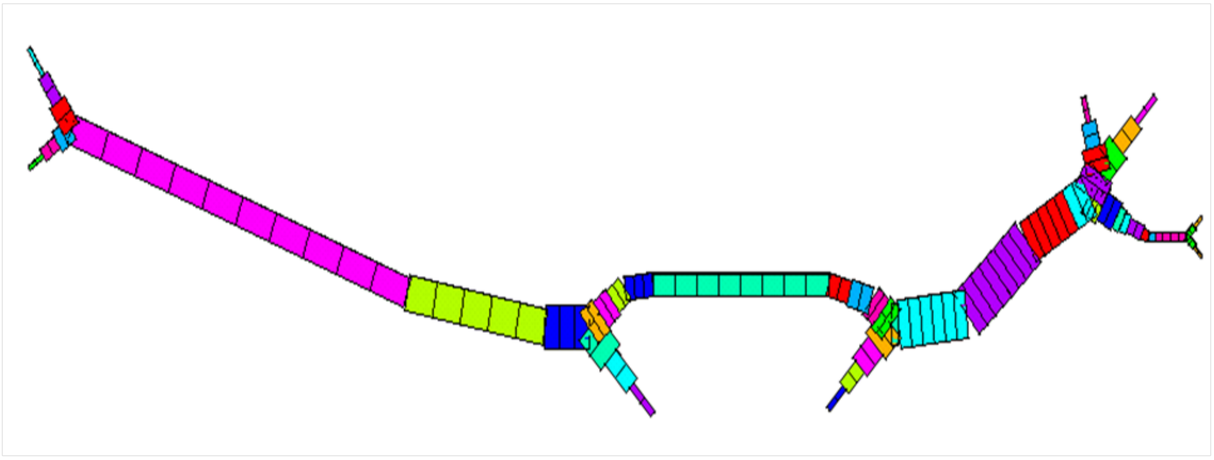


Figure 4.17: 2D mesh on the medial axis with 0.1*Thickness shown in 3D

queried along the nodes as shown in Fig. 4.18 and compared to the exterior nodes of the solid model unlike the neutral line comparison that was done for the other models described earlier. This is because a few of the result query location in the Xs1MAT model lie on the exterior boundary (branch ends). The corresponding results are presented in Table. 4.2. It can be seen from Table. 4.1 and Table. 4.2 that the MAT shell model is less stiff (more displacement) when compared to the Xs1D1 model which compared well with the solid model.

Though the MAT shell model varies by 7.58% and 9.26% when compared to the solid model, what needs to be noted here is the consistency in variation (though the results vary, they always vary in the same ratio). The variations are closer to each other than wide apart like in the Xs1D1 model. It can be seen from Fig. 2.7 that most of the circles overlap each other and thus lead to addition of mass to the model. The variation in stiffness of the MAT model is also due to the fact that the shell mesh does not represent the actual shape of the component as in the Xs1D1 model. The greatest advantage the MAT model has over the Xs1D1 model is the ease with which it can be automated. The Xs1D1 model was built manually while the MAT model can be fully automated. The mass of the original MAT model was around 14kg when a density of 8.0x10^{-6}kg/mm^3 was used. A mass correction was done by using a density of 6.83x10^{-6}kg/mm^3 and the total mass of the component matched within 0.06% and the first 10 frequencies matched within 8% of the solid model as shown in Table. 4.4.

TABLE 4.2: % Difference of radial and axial displacements of the Xs1MAT shell model with the solid model

Xs1MAT shell model - Radial displacements (mm)								
Loc	0.0°	90.0°	180°	270°	%Diff 0°	%Diff 90°	%Diff 180°	%Diff 270°
1	0.0040314	0.00126725	0.00072508	0.00126725	3.53 & 11.22	18.15	11.11	
2	0.00231232	0.00122161	0.00073636	0.00122161	4.73 & 9.97	19.13	9.88	
3	0.00192256	0.00115735	0.0007066	0.00115735	2.38 & 9.91	18.58	9.80	
4	0.00118885	0.00086896	0.00058768	0.00086896	7.31 & 8.17	18.03	7.96	
5	0.0009931	0.00075279	0.00053801	0.00075279	8.49 & 9.39	15.85	9.18	
6	0.00053682	0.0004502	0.00038935	0.0004502	10.28 & 13.17	10.71	13.03	
7	0.00024585	0.00027567	0.00025033	0.00027567	3.30 & 6.44	1.31	6.40	
8	0.00016883	0.00021992	0.00020896	0.00021992	12.25 & 6.91	1.68	6.91	
9	0.00011591	0.00017211	0.00017772	0.00017211	45.43 & 12.12	5.47	12.05	
10	0.00006252	0.0000633	0.00011461	0.0000633	54.75 & 92.40	16.47	92.99	

Xs1MAT shell model - Axial displacements (mm)								
Loc	0.0°	90.0°	180°	270°	%Diff 0°	%Diff 90°	%Diff 180°	%Diff 270°
1	0.00021143	0.00021142	0.00021144	0.00021142	19.93	19.92	19.93	19.92
2	0.00016551	0.00016551	0.00016551	0.00016551	5.15	5.15	5.15	5.15
3	0.00016486	0.00016486	0.00016486	0.00016486	4.87	4.87	4.87	4.87
4	0.00012816	0.00012816	0.00012816	0.00012816	3.35	3.35	3.35	3.35
5	0.00011849	0.00011849	0.00011849	0.00011849	10.12	10.12	10.12	10.12
6	0.00008959	0.00008959	0.00008959	0.00008959	3.69	3.69	3.69	3.69
7	0.0000693	0.0000693	0.0000693	0.0000693	0.86	0.86	0.86	0.86
8	0.00006426	0.00006426	0.00006425	0.00006426	4.66	4.66	4.64	4.66
9	0.00005856	0.00005856	0.00005855	0.00005856	16.42	16.42	16.40	16.42
10	0.00003195	0.00003195	0.00003194	0.00003195	96.01	96.01	95.95	96.01

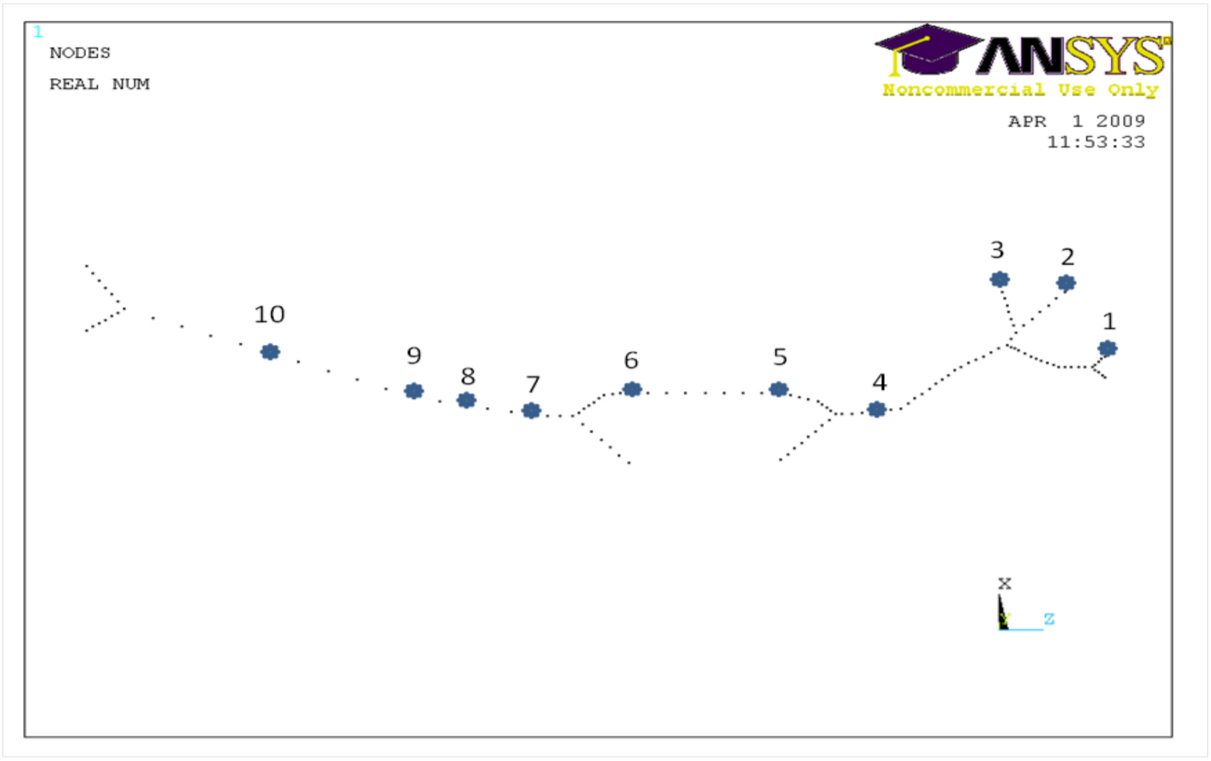


FIGURE 4.18: Node locations on the Xs1MAT model for comparison with the solid model results

4.8 Discussion

The Xs1D1 model, which is based on face pairing has disconnected midsurfaces which need to be connected such that the radial and axial stiffnesses of the shell model are still comparable to the solid model. A range of connection models have been researched and none give wholly satisfactory results. The simplest and first attempted approach of Xs1D1 is probably the best of those tested. The Xs1MAT model when created using the Matlab Medial Axis Transform code adopts a generic process and can be automated to be used in batch mode. The hindrance caused by using this method is the correct representation of the component's stiffness. After density correction, the MAT model has the same mass as the solid model with a small variation in the Center of Gravity (CoG) as shown in Table. 4.4. This is because the mass distribution of the MAT model is not same as in the solid or Xs1D1 model. Density correction can help match the mass when compared to a specific model but will not correct the CoG. This however can be overcome by mass balancing i.e. adding some extra mass along the centerline of the component to move the CoG to the required location. This can be done at a component level or at an assembly level i.e. the mass of the entire assembly

TABLE 4.3: Summary of results as sum of radial and axial displacements - Xs1

		Sum of radial dis	Sum of axial dis	% Diff radial	% Diff axial
1(a)	Solid(neutral Line)	0.026579	0.0039		
1(b)	Solid(Ext. line)	0.02686	0.00402	1.06	3.26
2	Xs1D1 : Face pair	0.026679	0.00413	0.38	5.93
3	Xs1D2	0.02847	0.00409	7.11	5.11
4	Xs1D3 : SolSH	0.03157	0.00682	18.81	75.11
5	Xs1D4 : Coupling	0.02396	0.00379	9.84	2.63
6	Xs1D5 Spr : 1x10^5	0.04753	0.00857	78.85	119.86
7	Xs1D5 Spr : 1x10^8	0.02466	0.00404	7.25	3.70
8	Xs1D5 Spr : 1x10^{15}	0.02462	0.00403	7.37	3.48
9	Xs1D5 Spr : 4x10^6	0.02542	0.00419	4.34	7.67
10	Xs1D6 : 1Set	0.03277	0.003157	23.32	19.03
11	Xs1D6 : 4Sets	0.0263	0.00434	0.83	11.46
12	MA with Branches	0.0289	0.0044	7.59	9.26

is balanced rather than each component in the assembly. What needs to be investigated however is the usage of branches that are created by the medial axis tool in the meshing process. The first cut results discussed in this chapter are promising. Some customization of the medial axis could lead to a more usable midsurface or midline that represents the 3D object more reasonably.

The Xs1D1 model on the other hand has its CoG at the same co-ordinates as the solid model with a mass difference of 3.29%. Of the first 10 natural frequencies, almost all modes differ from the solid model by less than 3% except mode 8 and 9 which differ by 4.98% and 7.32%. This shows that Xs1D1 model is a good representation of the solid model. However, the model with springs, i.e. Xs1D4 has the same mass and CoG as the Xs1D1 model, but varies by 3.38% and 0.395% in mass and CoG respectively and has better axial properties when compared to the solid model. The modeling technique with springs seems to be a good representation of the solid model in terms of mass distribution but not stiffness distribution. This is the reason why all the Xs1D4 modes vary between 7.41% and 23.5% difference when compared to the solid model modes. It is also worth noting at this stage that the method chosen to dimensionally reduce a solid model into a 2D model should satisfy properties such as mass balance, CoG match, stiffness match and natural frequencies matching within 5% of the solid model and ease of automation.

This chapter dealt with a few dimensional reduction techniques of complex cross-sections, each having its own pros and cons. Of these, there are two shell modeling techniques that stand out due to their representation of the shape and stiffness of complex features as in Xs1. Thus the Xs1D1 and the Xs1MAT model seem to be the likely choice for further study and use.

Table 4.4: Summary of modal analysis results - Xs1

	Solid		Xs1MAT	% Diff		Xs1D1	% Diff		Xs1D4	% Diff
Mass	11.97		11.96	0.06		12.36	3.29		12.37	3.38
CoG(Z)	93.77		90.50	3.49		94.66	0.95		94.6	0.95
Mode 1	72.57		67.02	7.65		71.08	2.05		65.83	9.29
Mode 2	72.57		67.02	7.65		71.08	2.05		65.83	9.29
Mode 3	116.31		111.03	4.54		114.10	1.90		129.01	10.92
Mode 4	120.11		117.48	2.19		120.12	0.01		129.01	7.41
Mode 5	120.07		117.48	2.16		120.12	0.04		130.15	8.40
Mode 6	163.90		161.47	1.48		163.91	0.01		194.54	18.69
Mode 7	163.90		161.47	1.48		163.91	0.01		201.41	22.89
Mode 8	164.18		164.56	0.23		172.36	4.98		201.41	22.68
Mode 9	164.18		168.32	2.52		176.20	7.32		202.74	23.49
Mode 10	174.80		168.32	3.71		176.23	0.82		202.74	15.98

Chapter 5

Geometry Parametrization and Optimization - Creating a work flow

In this chapter, a simple stepped beam is used to identify a design search and optimization technique along with a suitable work flow. The ideas from this exercise will then be used as inputs to design an aeroengine casing structure which would reduce tip clearance variation. The various steps involved in designing a stepped beam for optimum deflection and optimum volume are discussed in detail.

The cantilever stepped beam model was generated primarily to study the current practices and technologies available for a seamless integration of the CAD to CAE process. Virtual prototyping through simulation is widely adopted as part of the product design cycle. The analysis process is a key part of the design cycle. The analysis performed on the initial conceptual design provides essential information regarding the component's structural behaviour. Should these results indicate any unfavorable behaviour of the component, appropriate changes need to be made to rectify these flaws. However, in a design optimization cycle the transition from a CAD model to a FE model happens repeatedly. Detailed work has been done in the area of transferring information from CAD to CAE and vice-versa as mentioned in [24],[86] and [87].

CAD and CAE are two different systems, each evolved from different backgrounds to cater to different needs which has resulted in different data representations. As a result, attempts to integrate CAD and CAE processes tends to introduce a chain of problems. For the stepped beam case study a design optimization work flow has been developed as shown in the Fig. 5.1

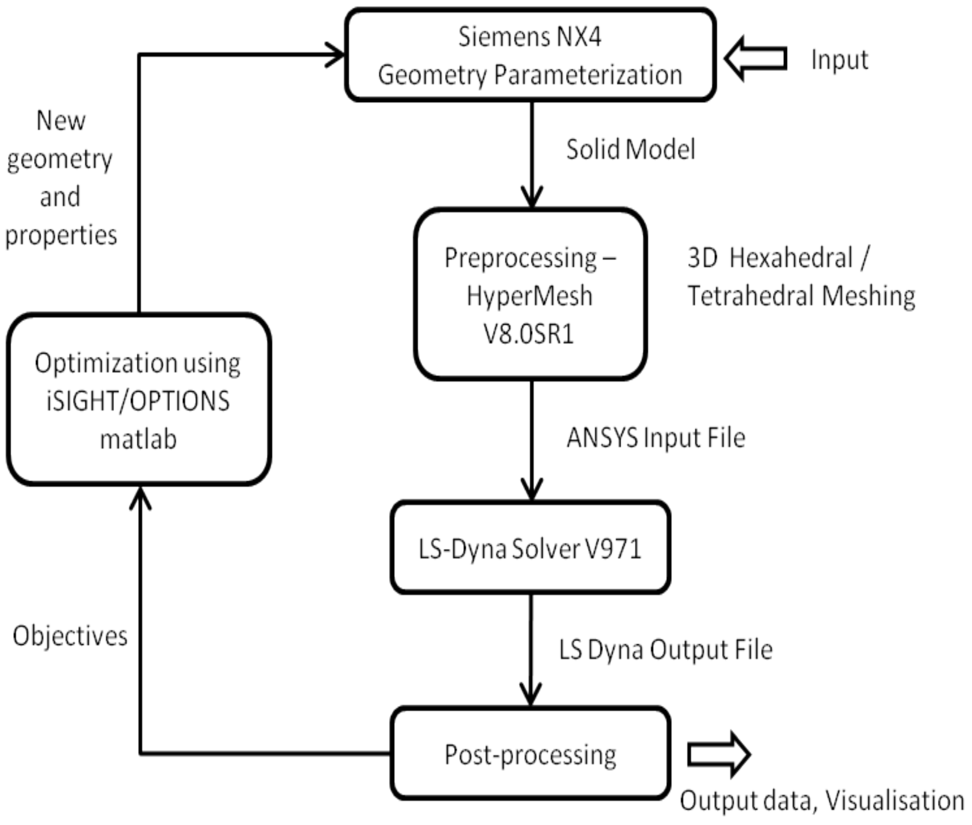

Siemens NX4
Geometry Parameterization
Input
Solid Model
New geometry and properties
Preprocessing – HyperMesh V8.0SR1
3D Hexahedral / Tetrahedral Meshing
Optimization using iSIGHT/OPTIONS matlab
ANSYS Input File
LS-Dyna Solver V971
Objectives
LS Dyna Output File
Post-processing
Output data, Visualisation

FIGURE 5.1: Stepped beam design optimization workflow

5.1 Parameterization

The CAD model is generated using Siemens NX 4.0 [17] as a 2D sketch and then extruded to form the 3D stepped beam. NX has been chosen in this case as the CAD engine because of its flexibility to parameterize solid models and also, more importantly, because the parameterization can be invoked in the batch mode. This is done by using NX GRIP(Graphics Interactive Programming). NX also happens to be the preferred CAD engine within Rolls-Royce Group PLC, Civil Aerospace UK, who are sponsors for this project.

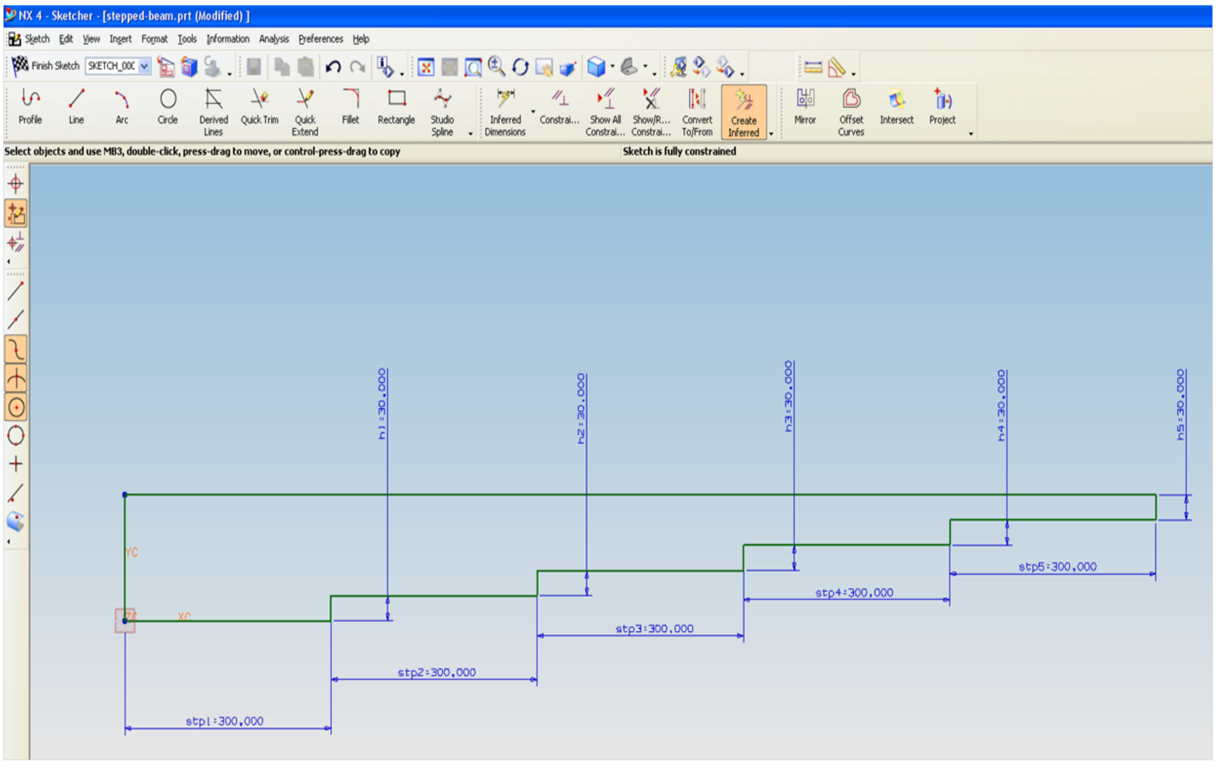
The general procedure while creating a parametric model using NX constitutes a few essential guidelines:

- ⋆ A clear description of the geometry that needs to be sketched has to be prepared well in advance.

- ⋆ Once the cross section has been thought of, the next step involves definition of the parameters or variables that would form the core of the optimization problem, which are shown as dimensions in Fig. 5.2
- ⋆ Relationship between these variables also need to be defined in order to make sure that they change appropriately when changes are being made during the optimization process. This is particularly important when we know that a geometric feature (e.g. fillet radius) has to change depending on a specific thickness of the component. The knowledge gained from previous experience or lessons learnt can be incorporated into the current model this way.
- ⋆ The geometric model (2D sketch in this example) needs to be well constrained. The constraints generally are made up of "Geometric constraints" and "Dimensional constraints". Care has to be taken to make sure that the model is neither over-constrained nor under-constrained. Creating geometric constraints is analogous to a set of rules that a geometry must follow. A combination of dimensions and geometric constraints gives the ability to create very complex geometry sketches. It not always necessary to fully constrain a sketch, but doing it makes a sketch behave more predictably, [88].
- ⋆ The well constrained 2D sketch can then be used to create the 3D stepped beam as shown in Fig. 5.3
- ⋆ However, the 3D stepped beam NX part file has to be compiled as a GRIP file in order to be used for the optimization cycle.
- ⋆ The purpose of compiling a part file is to allow NX to modify the dimensions in either the 2D sketch or the 3D solid model in batch mode. When the optimizer evaluates and outputs new values for the variables, they need to be updated in the NX part file and this is achieved by compiling the part file as a GRIP file.
- ⋆ A batch file, which forms a part of the optimization cycle is responsible for updating the NX geometry based on inputs provided by the optimizer.

5.2 Preprocessing

Once a parameterization scheme capable of generating the required geometry has been developed, the next step is in building an automated analysis scheme. The structural analysis



Select objects and use MB3, double-click, press-drag to move, or control-press-drag to copy

Sketch is fully constrained

FIGURE 5.2: Stepped beam elevation with the heights of each step as a variable

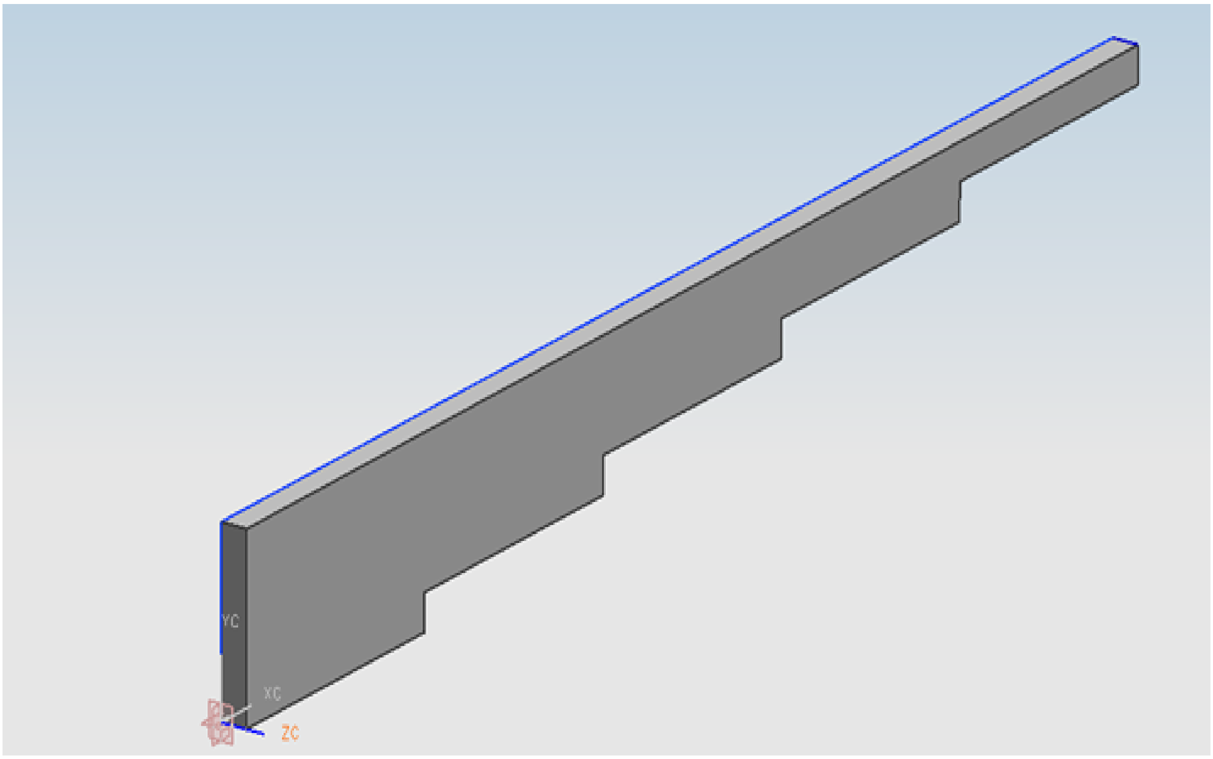


FIGURE 5.3: Stepped beam extruded into a 3D model

of any component involves three basic steps; Meshing, Solving and Post-processing.

In this very simple case, meshing constitutes 60% of the analysis time and is a vital ingredient in acquiring the near exact solutions in a Finite Element Analysis. The Finite Element modelling scheme and the choice of element needs to be decided well before beginning to mesh the component. In this case study, HyperMesh (HM) has been chosen as the preferred tool because of its strength in importing and exporting CAD and FEA models between various CAD/CAE engines. HM can if necessary also be used as a translator between many of the commercially available CAD/CAE packages.

However, there are other commercially available meshing codes which have been looked into before making a decision to use HyperMesh. These include ANSYS Workbench, LS-Prepost, Harpoon and ANSYS Prepost to name a few. ANSYS Workbench and Harpoon were studied to create automated brick meshes for use in the batch mode. Though Workbench and Harpoon are able to create good quality brick meshes on simple geometries, their capabilities are questionable when handling complicated or slender structures. It is also noteworthy to mention that these tools create what is known as a "Hexa dominant mesh". These meshes are made up of hexa, penta, pyramid and tetrahedral elements. It was observed in Chapter 3, that for a mesh density as shown in Fig. 3.1, the number of non-hexahedral elements were about 35.5% of the total mesh. For a finer mesh density as shown in Fig. 3.2, the number of non-hexahedral elements were about 61.1% of the total mesh.

The other disadvantage of these tools are their geometry import capabilities. ANSYS has an inbuilt NX translator which can read NX part files. However, tools like Harpoon and LS-Prepost rely more on STL [89], STEP, Parasolid or IGES files. Whenever geometry is transferred between tools and in different formats there is inherent loss of information. In order to avoid such losses, it is always good practice to use those preprocessors which have built in translators for different CAD engines. HyperMesh V8.0SR1 and V9.0 have a NX model translator which can read an NX part file directly. HM is also a good meshing tool but its meshing capabilities in batch mode are not well documented for 3D geometries. Since the Hexa dominant meshes primarily consist of more tet elements than brick elements, tet elements have been used for meshing in this case using HM.

A command file has been created using the HyperMesh commands that generates an FE model with all the required loads, boundary conditions, property cards, material cards etc., for analysis purposes. The FE model generated can then be exported based on the choice of FE solver. The command file invokes a TCL/ TK routine which changes the force load applied on each node on the cantilever edge of the stepped beam. The max load on the cantilever edge is 1000N and is distributed along a set of nodes, e.g; 1000N/5 nodes = 200N/node. However, when the optimizer changes the width or the thickness of the stepped beam, the number of nodes along the cantilever edge changes appropriately since the mesh size has been set as a constant. Under such circumstances the load on each node would then change accordingly e.g; 1000N/3 nodes = 333.33N/node.

The FE model file generated consists of all the relevant control cards pertaining to the FE solver, in this case, LS-Dyna 3D. The LS-Dyna keyword file generated then constitutes the grid data, element data, loads, boundary conditions, element properties, material properties and output blocks. The FE model created using HM and viewed in LS-Dyna prepost is as shown in Fig. 5.4. A sample HM command file and LS-Dyna keyword file are given in Appendix D.

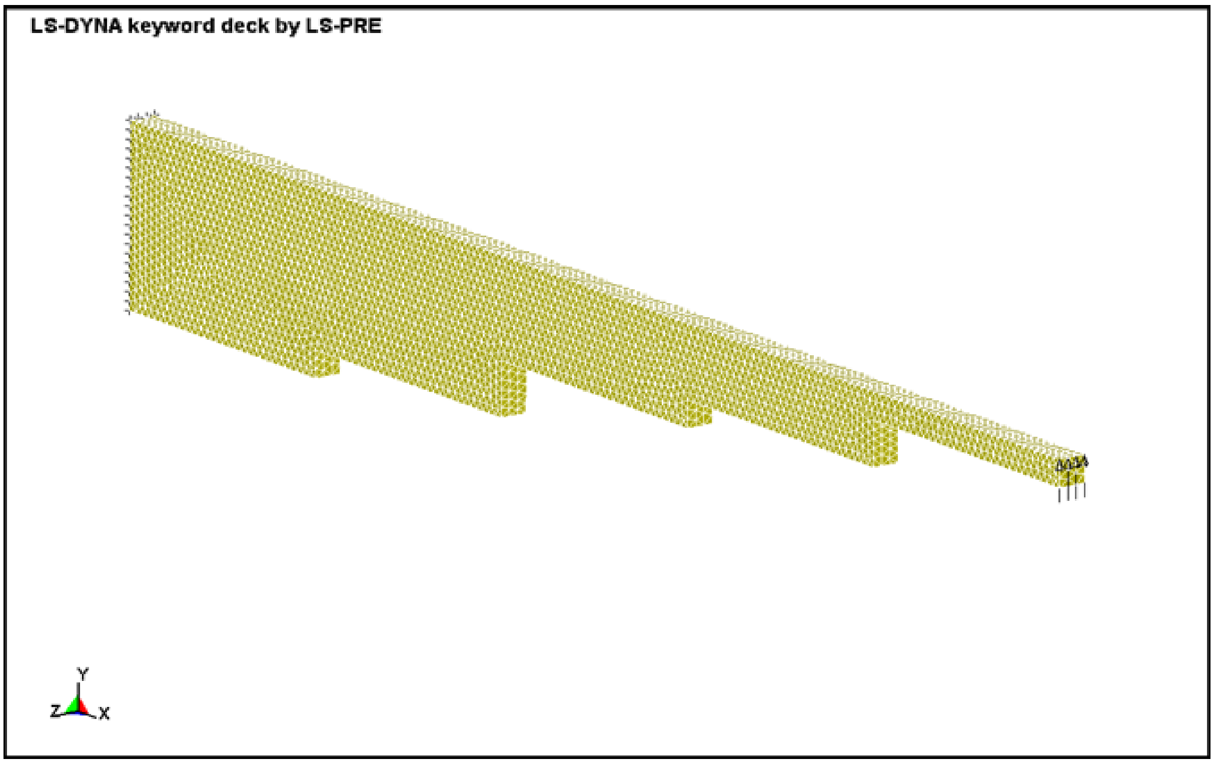


FIGURE 5.4: Stepped beam 3D FE model in LS-Dyna using 4 noded tetrahedral elements

One of the serious drawbacks of using HM is its inability to understand "tag" information that can be created within NX. A name (tag) given to a surface or edge of a surface in Siemens NX is retained when exported as part files into tools like ANSYS. Though HM can import a NX part file, it does not recognize the tags associated with the geometry. A work around to this problem has been used to design the stepped beam, but the method used is not robust enough to handle more complicated geometries. In this case, the areas and edges on which the loads need to be applied are imported as different layers into HM. These layers when imported into HM form different components. These components can then be picked to perform various tasks on them. The inefficiency of this method lies in the fact that these different layers are not part of the main geometry and so if an area that needs to be constrained is imported into HM in a separate layer, then this area is not attached to the main part.

To overcome this issue, all the areas are currently meshed with the same element size and edge length. Since the element edge length is kept constant, the number of nodes on the area/surface (the one stored in a different layer) that needs to be constrained and the corresponding original surface on the main part will have same node and element count. Using this knowledge as a starting point, the loads are applied on the dummy surface stored as a separate layer and then the nodes and elements merged to transfer the loads/boundary conditions to the main part. The duplicate elements and nodes can then be equivalenced with a specific tolerance. This is clearly a cumbersome process lacking in robustness.

5.3 Solver

A choice had to be made between ANSYS mechanical and LS Dyna 3D [22]. A study constituting ANSYS (implicit analysis) and LS-Dyna (explicit analysis) has been condcuted by Rust *et al.* in [90]. The FE model generated using HM can be exported to either of these two solvers without much difficulty. However, in this case, the choice between the solvers have been made keeping in mind the final implementation of the work flow into the Rolls-Royce business process. However, during this research, ANSYS and LS-Dyna will be retained to ensure flexibility of approach. A full engine FE model generated for use in LS Dyna for structural analysis can also be used for fan blade-off and containment studies which form the strengths of LS Dyna and bring in more value addition for the FE model created. LS Dyna by default produces a “d3plot file”, which is used for graphical post-processing using LS-Prepost. A typical result when viewed using LS-Prepost is as shown in Fig. 5.5. But since the work flow operates in batch mode, no graphical post-processing can be performed. In order to facilitate post-processing of the analysis results in batch mode, necessary results have to be output as text files.

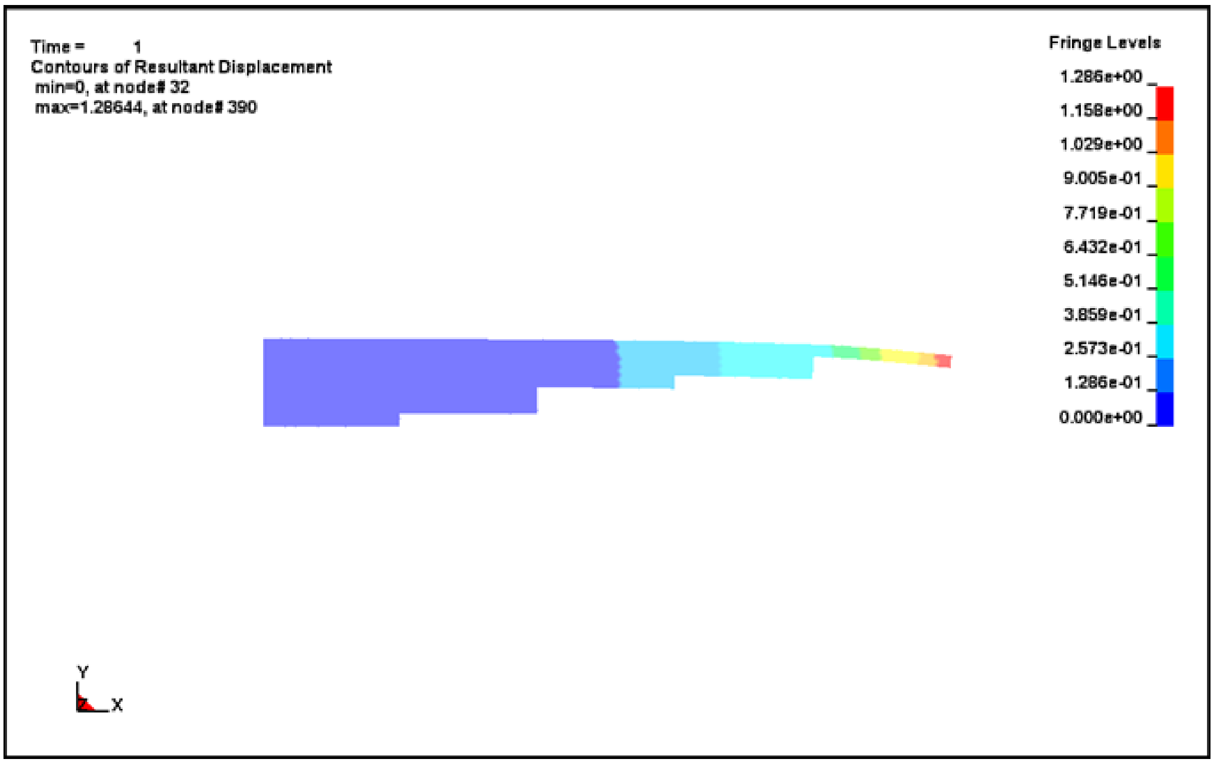


FIGURE 5.5: Maximum deflection of the cantilever stepped beam

5.4 Post-processing

The result text files generated by the LS Dyna solver are used to identify the values the objectives have taken based on the design that has just been analyzed. The objectives either need to be minimized or maximized based on the problem goals. Values for these comparisons are obtained from the FE Analysis. The results can be parsed either using Matlab or iSight. When the node number which has the maximum deformation keeps changing, iSight cannot be easily used to parse such results. This is because iSight looks for outputs or inputs based on templates. But in this case, since the mesh is regenerated all the time, the node numbers and the element numbers change during each iteration. And this change in node and element numbers would need a more generic parsing tool.

A small script has been written using Matlab which opens the nodal results output by LS-Dyna, parses the file for the maximum deflection along the specified direction/coordinates (in this case Y-Coordinate) and then reports it printed in another text file.

5.5 Optimization

iSight FD is the preferred choice to perform the design search and optimization of the stepped beam since this is also a Rolls-Royce adopted tool. The OPTIONS NSGA2 plugins to iSight FD were used along with the native iSight FD NSGA Optimization Algorithms. For a simple problem such as the stepped beam, the difference in results produced by the many available optimization algorithms was not very significant. The definition of the optimization problem was set such that there were five variables and two objectives. The variables have been set as the heights of each of the five steps and the two objectives were minimum deflection and minimum volume, refer Fig. 5.6

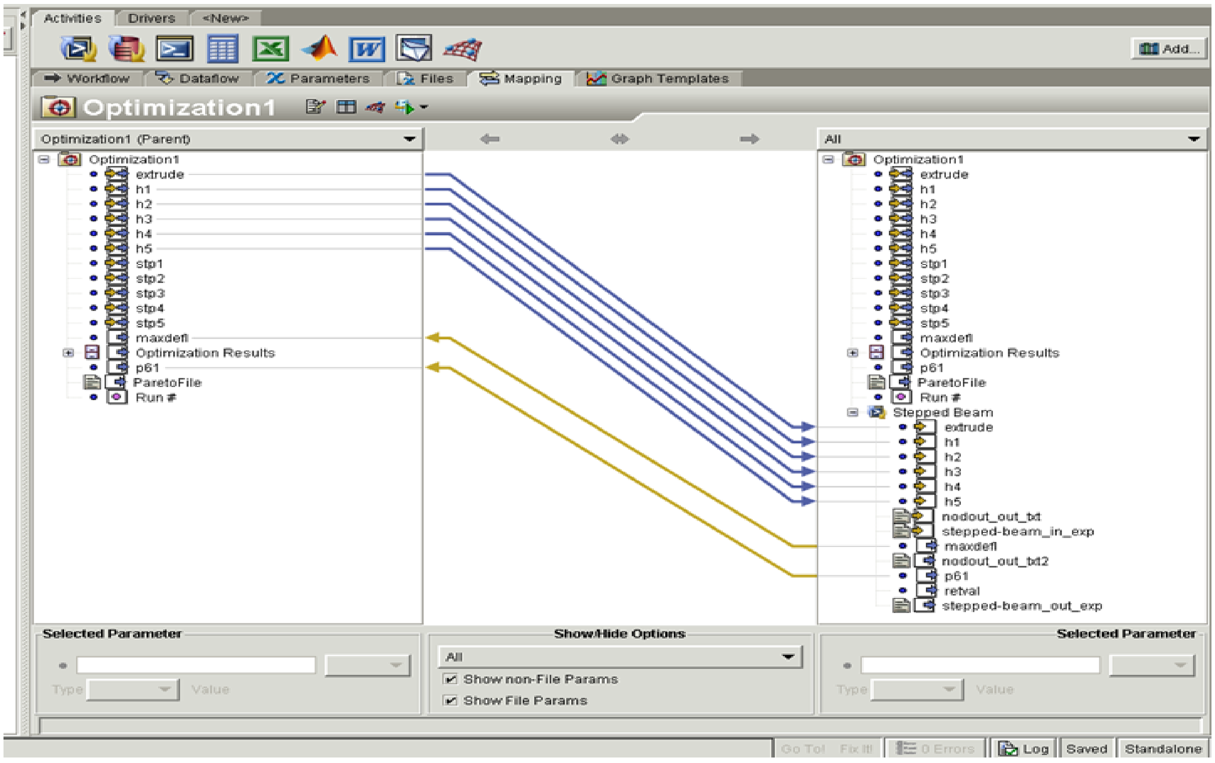


FIGURE 5.6: Mapping of variables, objectives and constraints in iSIGHT-FD

The best results produced by the optimization algorithm can be plotted as a Pareto Front, see for example Fig. 5.7. It is clear from the problem setting that the two objectives are competing with each other. This is because, to achieve minimum deflection of the stepped cantilever beam, the steps have to be made thicker. But having thicker steps would result in more volume. And this would probably be the scenario when a real gas turbine engine problem is being optimized because, to reduce tip clearances, we would need stiffer casings. Stiffer casings would mean more material and hence more weight. However, the principle objectives for such components as aeroengine parts would be reducing stresses for minimum/no addition of weight. The deflection and its relation to volume of the stepped beam for various combinations of the variables is as shown in Fig. 5.8. The OPTIMAT-NSGA2 multiobjective optimization algorithm was chosen to obtain the best combination of variables for the stepped beam model. A total of 5 generations with a population size of 100 was chosen. The Pareto front of this optimization run is as shown in Fig. 5.9. It can be seen from this Pareto front that there is a set of possible light - high deflection designs as compared to heavy - low deflection designs. As an engineer, any design from this set could be chosen as the best design based on manufacturing and other requirements. The heaviest model has a volume of 1,113,120.0mm^3 for a deflection of 0.36mm when compared to the lightest model which has a volume of 528,696.00mm^3 for a deflection of 8.07mm. However, since by problem definition a maximum deflection of upto 5.0mm is acceptable, the design which results in a volume of 572,979.00mm^3 for a deflection of 4.94mm can be considered as the optimal.

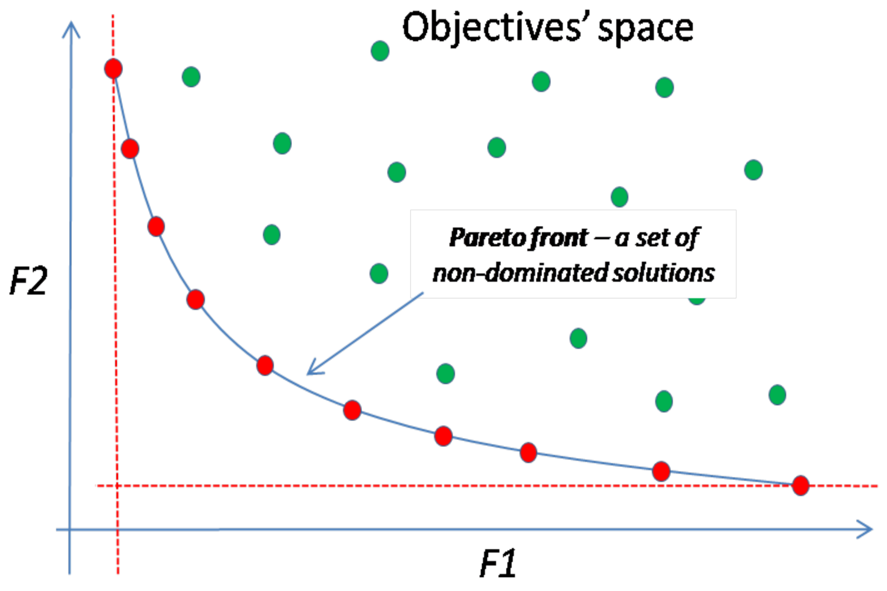


FIGURE 5.7: Pareto front of two competing objectives

5.6 Summary

In this chapter, we have seen how the stepped beam case study has been used to understand the hurdles that need to be crossed in order to perform optimization of a structural component. The tasks performed to achieve a volume and deflection optimized stepped beam would

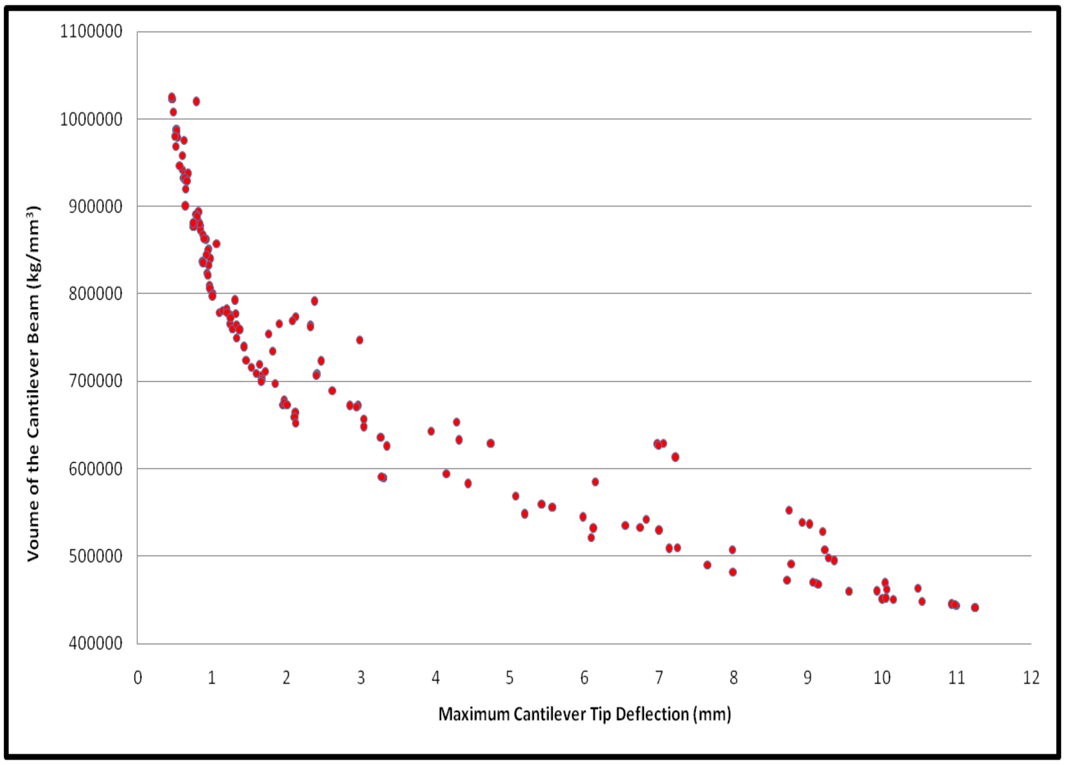


FIGURE 5.8: Results of the two competing objectives of the stepped beam

be broadly the same when optimizing a gas turbine engine part. The optimization work flow defined in this chapter can thus be readily used for any geometry with minor changes to the code. The use of HM to define boundary conditions still poses some problems since HM does not recognize “tags” created by NX. A work around for this has been used while designing the stepped beam work flow, but its use when handling more complicated geometries needs to be reworked.

A work flow needs to be developed on the same lines as shown in Fig. 5.1 to optimize the whole engine model. The preprocessing stage will be split into two sub-processes, one using Matlab to generate the Medial Axis and output information for HyperMesh to create a 2D mesh while the other half would be to run a TCL code in batch to create the FE model in HM for use in either ANSYS, NASTRAN or LS-Dyna. The other steps would essentially remain the same. The optimization of the stepped beam has been instrumental in gaining understanding of the requirements of all tools participating in the work flow when operating in batch mode.

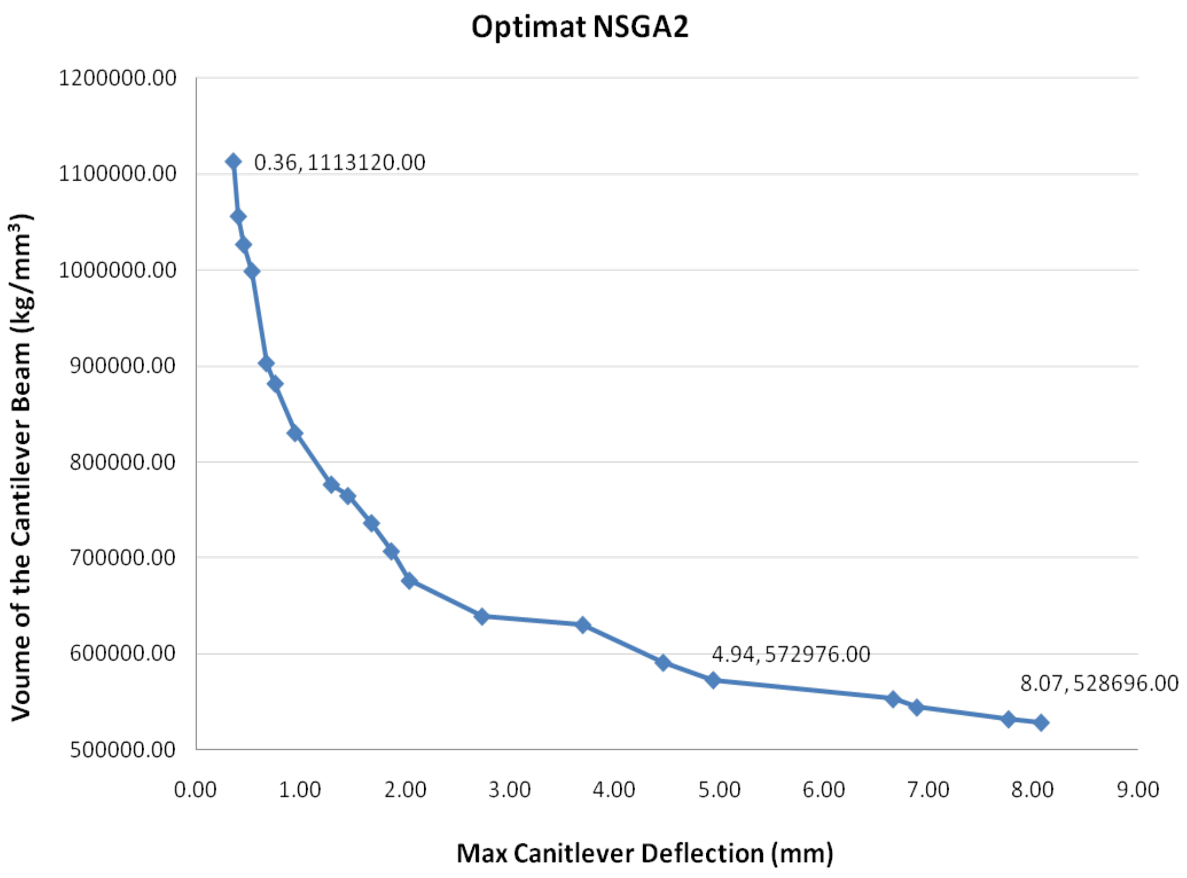


FIGURE 5.9: Pareto front of the stepped beam multiobjective optimization

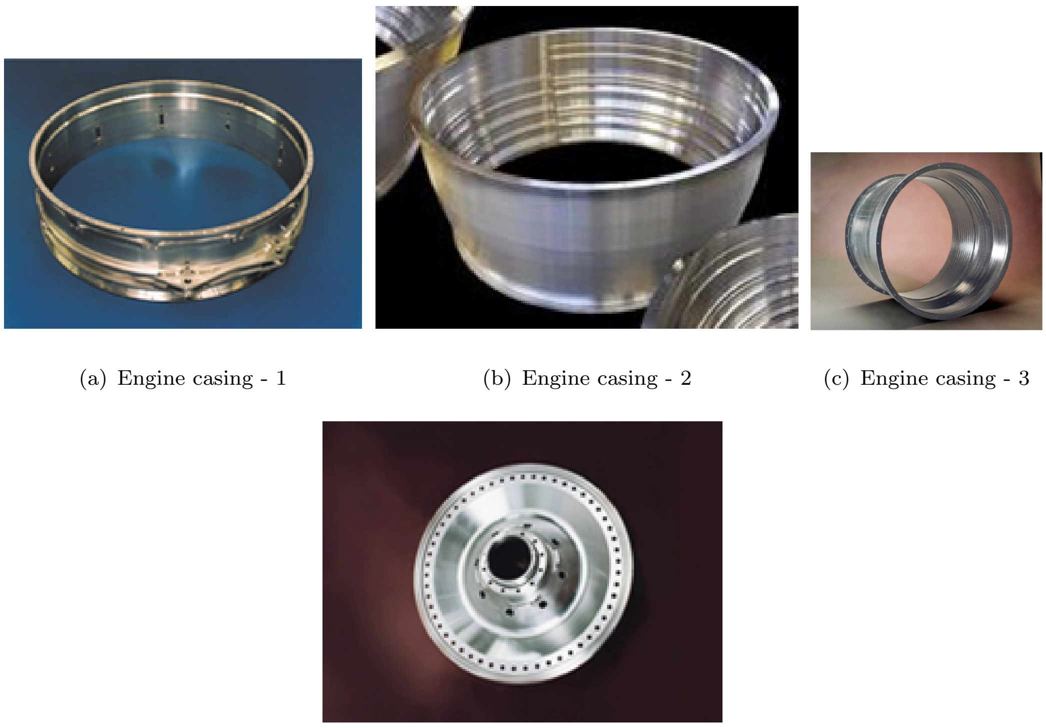
Chapter 6

Multi-objective optimization of axisymmetric 3D geometries

Based on work done in Chapter 4 and 5, a definite need for a dimensional reduction process that could be operated in an automated fashion was noticed. As mentioned earlier, the majority of the load bearing casings in a gas turbine engine can be considered as axisymmetric. Commercial tools are not customizable and to do so, substantial help is needed from the software's customer support system. Hence a bespoke code has been scripted as part of this research program to demonstrate the use of midsurfaces in a design optimization process and is called 'MANTLE-2D' (Modeling and Analysis of finite elements in Neutral Plane - 2D). In order to achieve this, work done by Krishnan in [47] and [48] and his 2D medial axis Matlab code [91] has been extremely helpful.

Extensive pre-processing and post-processing was essential to use the 2D medial axis Matlab code in batch mode. Krishnan in the online Matlab file exchange forum [91] has mentioned about a few bugs in the code cautioning the users about the same. The inefficiencies of this code had to be overcome in order to use it for this research program. The code needed substantial manual understanding of the inputs required for its effective usage. But manual intervention in a design optimization process is detrimental except while setting up the process the first time.

Another script that has been used from the public domain and plays a major role in the functioning of Mantle-2D is 'IGES Toolbox' created by Per Bergström [92]. This code converts the geometric information present in an IGES (Initial Graphic Exchange Specification) file into parameters that can be used in Matlab for further processing. The algorithm developed as part of Mantle-2D and each of its modules are explained in detail in the following sections. A few typical gas turbine engine casing geometries that can be considered axisymmetric and represented using midsurfaces are as shown in Fig. 6.1. When the bolt holes, cutouts on the casing, small bosses etc are ignored, the midline of the cross-section of these geometries can be used to create midsurfaces thus dimensionally reducing these 3D models into 2D models.


(a) Engine casing - 1 (b) Engine casing - 2 (c) Engine casing - 3

(d) Engine casing - 4

Figure 6.1: Engine casings that can be considered axisymmetric

6.1 Challenges in creating medial mesh of complex 3D axisymmetric geometries

The medial axis generated is based on the geometry provided as input. If a geometry is considered axisymmetric, then a 2D cross-section is sufficient to define it and a sector model is sufficient if it is cyclic-symmetric. Every convex and concave vertex on the boundary of the geometry results in branching of the medial axis. These branches alter the stiffness and mass of the geometry being represented as mid/medial surfaces. If the boundary of the geometry is noisy, numerous branches are created. Methods and techniques to identify and remove branches that are not significant and those that do not represent the geometry have been proposed by Tam *et al.* in [93], Shaked *et al.* in [94], Sud *et al.* in [95], Sakai *et al.* in [96], Ward *et al.* in [97] and Ogniewicz [98].

It is noteworthy to mention that the work done by these authors are in most cases pertaining to computer imaging and computer graphics. Branches created during the extraction of medial axis can be removed based on a few rules and guidelines when they are used in such areas of study. However, when the medial axis is used to represent a complex geometry for use in engineering design studies, the reasoning required to remove branches is non trivial. The branches extracted at all concave vertices are useful in capturing the stiffness of the geometry being represented. Hence, a pruning technique that can identify the useful and not so useful branches from an engineering analysis perspective needs to be developed. During the many studies carried out in Chapter 4, a study was also conducted of the Xs1MAT model without any branches. However, since the scope of this research program is not just to create a medial axis with/without branches for geometries with/without boundary noise, but to use a medial axis for engineering design purposes, a process to identify and remove spurious branches has been put aside.

6.2 The medial axis algorithm : MANTLE-2D (Axisymmetric geometries)

Mantle-2D, developed as part of this research program forms part of an optimization workflow which demonstrates the benefits of using medial or midsurface based dimensionally reduced finite element meshes during design studies constituting an individual or assembly of components. The main criteria during the creation of Mantle-2D was to ensure that it performed stably in a design optimization setting when invoked hundreds of times. A good understanding of requirements set by finite element solvers with regards to element quality checks was needed to ensure that the dimensionally reduced FE models output by Mantle-2D were usable. The MAT based shell mesh generation algorithm is as shown in Fig. 6.2. The medial surface based shell mesh for a few sections of a complex test case is as shown in Fig. 6.3.

As mentioned earlier, many modules of Matlab scripts were developed and integrated to build Mantle-2D. The proper functioning of each of these modules is essential for the code to work without errors. A flowchart describing the sequence of execution of various modules of

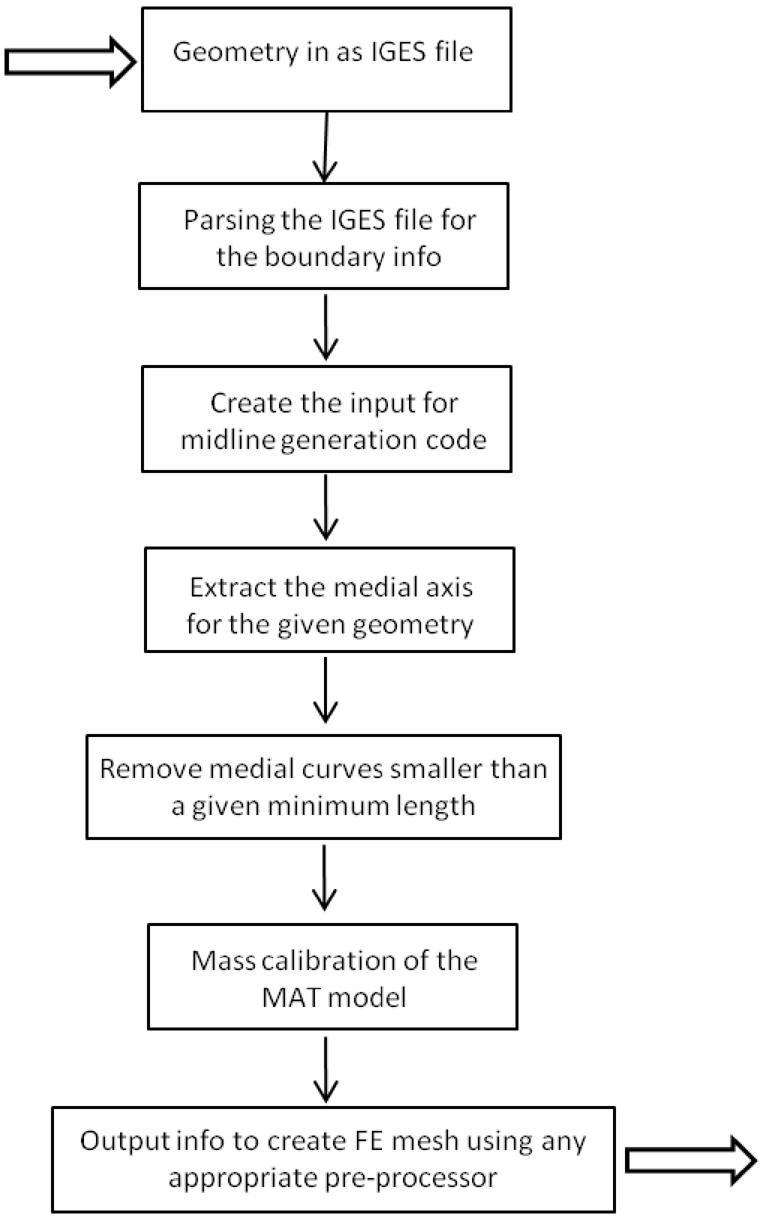

Geometry in as IGES file
Parsing the IGES file for the boundary info
Create the input for midline generation code
Extract the medial axis for the given geometry
Remove medial curves smaller than a given minimum length
Mass calibration of the MAT model
Output info to create FE mesh using any appropriate pre-processor

FIGURE 6.2: Medial axis based mesh generation algorithm

Mantle-2D is as shown in Fig. 6.4. The functionality of each of these modules is as explained below;

- ⋆ IGES to Matlab Generic: This module is responsible for the parsing of the input IGES geometry and converting it into parameters for use in Matlab. Ideally each of the entities present in the IGES format should be represented as parameters. However, complex IGES geometries can contain curve entities such as 100(Circular arc), 102(Composite curve), 104(Conic arc), 110(Line), 112(Parametric spline curve), 124(Transformation

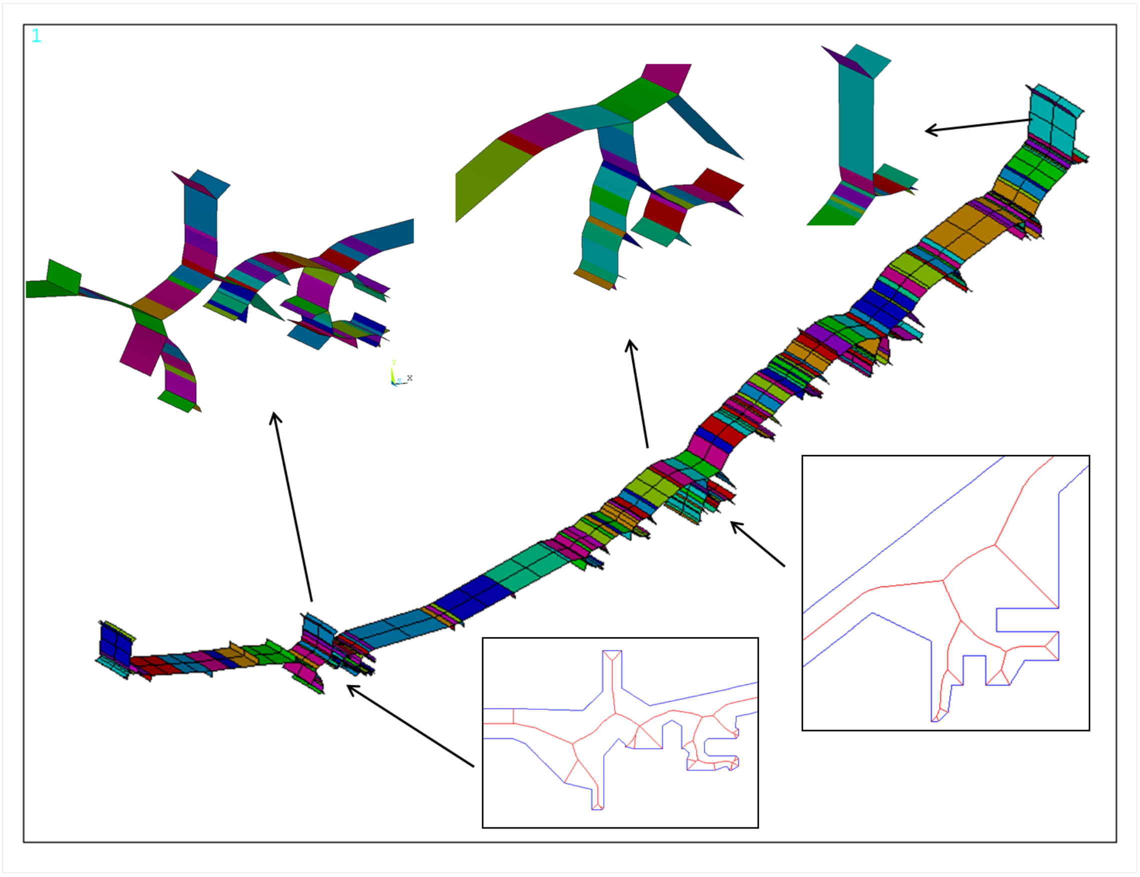


FIGURE 6.3: Medial axis and the corresponding shell mesh at a few complex regions of the high pressure turbine case

matrix), 126(Rational B-spline curve) etc. The code developed by Per Bergström as mentioned earlier does not handle all entity types. Therefore, either the geometry should be built in such a way that it does not contain any entity types that cannot be handled or modify the code to handle all IGES curve entity types. Due to time constraints, the former option has been chosen i.e, the geometry was prepared in such a way that it consisted of parameter types 110, 112 and 126 only. By doing so, the IGES to Matlab code could be used effectively within its range of operation without having to customize it heavily.

⋆ MAT input Rearrange: This module of Mantle-2D performed two tasks, to identify discontinuities in the IGES geometry and to arrange the geometry text input in a manner that the 2D medial axis code does not fail. The 2D medial axis toolkit fails when a discontinuity in the geometry is noticed within a tolerance of 1x10^{-6}mm. In order to prevent a discontinuous geometry being input to the medial axis generation

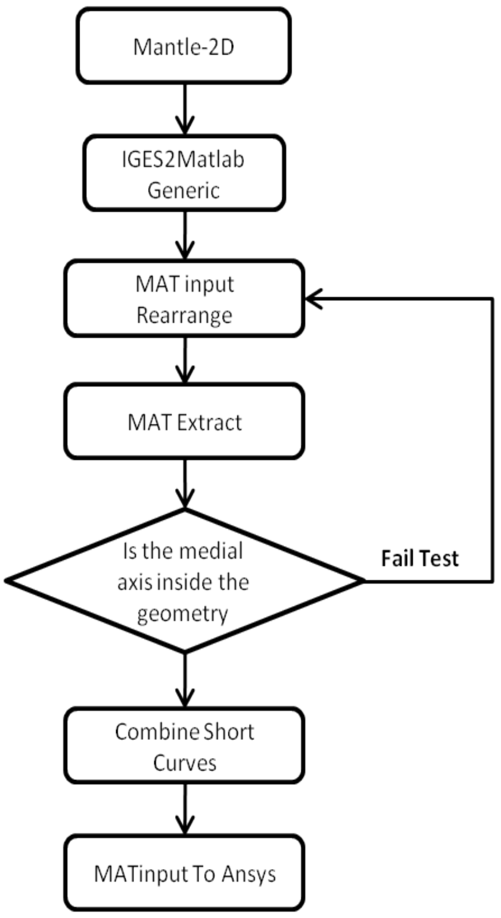


FIGURE 6.4: Execution sequence of various modules of Mantle-2D

code, this module of Mantle-2D parses the file for such discrepancies in the geometries and an error message is displayed for easy debugging.

If the geometry has no discontinuities, the elements(lines and curves) of the input file are rearranged. This is done in order to ensure that the medial axis of the inside volume of the geometry is extracted and not the outside. If a line is randomly picked from the IGES file, the 2D space to the right of the line is considered as inside by the medial axis toolkit. But the right side of the line depends on the direction in which the line is represented i.e; moving from point A in the left to point B in the right (clockwise) or anti-clockwise. By defining the set of lines in a clockwise direction, if the space lying to the right of a set of closed loop of lines does not form the inner volume of this geometry, the line input data has to be switched to anti-clockwise. Once this is done, the medial axis extracted would be that of the inner volume of the geometry.

⋆ MAT extract: This module makes use of the 'pdeGeom' toolkit which is part of the

Matlab optimization toolbox. The general method of extraction of medial points is based on the 2D Voronoi decomposition of the boundary points. All the Voronoi vertices of a given set of boundary points form the medial points along the medial axis. The steps involved within this module for creation of the medial axis are as follows:

– Rearrange the order of edges such that they form a continuous sequence

– Identify all convex, flat and concave points on the boundary as shown in Fig. 6.5

– Assign all convex points as medial points

– Find all the other medial points

– Construct the medial branches as 'Bezier curves'

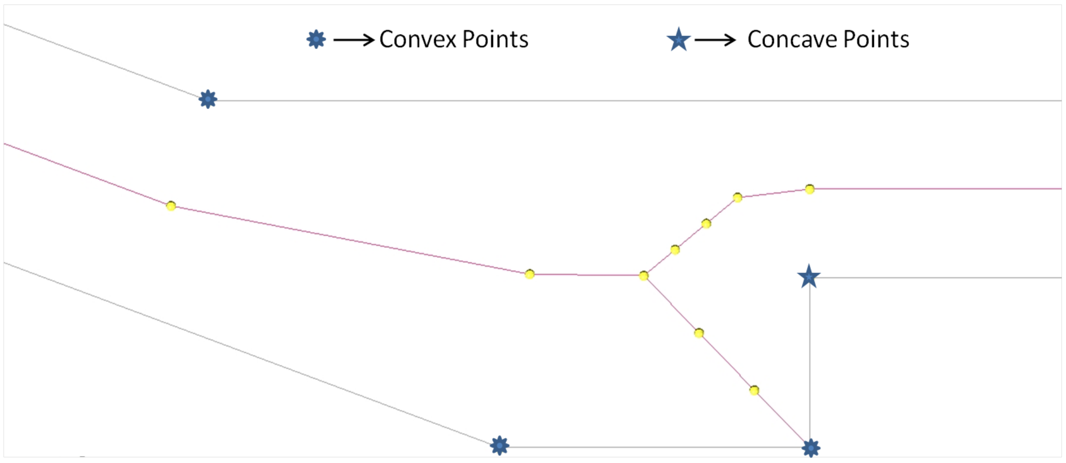


Figure 6.5: Concave and convex points along the boundary of a given 2D geometry

⋆ Combine Short Curves: Since the medial axis toolkit used in Mantle-2D was not developed for use in an engineering process, coincident medial points exist along the medial axis extracted. These coincident medial points cannot be used to create either a medial line or a medial surface (created by revolving a medial line). This is because, when a mesh is generated on this surface, there will be elements with zero area since the surface is degenerate. However, these points cannot be just removed from the medial point matrix since the link they have with other medial points would then be lost. In order to maintain this link and also ensure that the coincident points are removed, the coincident points are joined to their nearest neighbours until they are of the desired length. The smallest line that can be accommodated by the user can be set as a tolerance and all points that lie within this tolerance are then joined to form longer lines.

A tolerance of 0.25mm was set for the high pressure turbine test case discussed later in this chapter.

⋆ MAT input to ANSYS: Based on the medial axis extracted and the medial points information along this axis, an input deck to an FE solver has to be created and is done using this module of Mantle-2D. A line is created connecting two medial points and revolved to form a surface. The diameter of the maximal circle at one end of the line and the diameter at the other end are averaged to form a uniform thickness along the length of the line. This surface created by revolving the line along the medial axis is in effect the medial surface of the given axisymmetric geometry. Each of these lines are revolved to create many such surfaces. Each of these surfaces are then meshed with a user defined element density based on the length of the line which was used to create the surface.

The other most important function of this module is to define the corrected density based on the solid model mass and volume. As discussed earlier, when branches are created, the mass and stiffness of the medial mesh model is altered when compared to the actual solid model which it represents. The change in mass can also result in a CoG (Center of Gravity) shift. This behaviour would be undesirable and to counter this, the density is corrected. The volume and mass of the medial shell model created are calculated. These numbers are then compared to the respective solid model's mass and volume. The density of the medial shell model is then corrected to reflect the mass of the solid model. A similar exercise needs to be carried out for CoG shift as well. However, since the test case chosen for this study did not have its CoG shift by a significant amount when compared to the solid model, it has not been implemented.

As mentioned earlier, the MAT based algorithm generates branches at every convex vertex and this leads to additional mass being added to the FE mesh that is eventually generated. In order to tackle this problem, the density of the shell model is corrected based on the mass of the actual solid model. This results in a shell mesh mass that is comparable to the solid model by less than 2%. This difference in mass after the correction is due to the fact that the mass of the solid model is measured in ANSYS while for the MAT based shell model it is measured in Siemens NX. However, mass matching (density correction) does not mean that the shell model will have the same center of gravity (CoG) as the solid model. In order to ensure that the CoG of the shell model and solid model match, further mass balancing sometimes needs to be done on the shell model. The shell model discussed in this chapter has its CoG differ by less than 1.5% when compared to the solid model as mentioned in Table. 6.1.

6.3 The optimization workflow

Here comparison is made between design optimization carried out using a full 3D FE model and a model based on MAT. The optimization workflow is slightly different for the solid and the shell models. The 3D model workflow shown in Fig. 6.6 uses hexahedral elements to build the FE model, while the 2D model workflow uses shell elements to build its FE model. The 2D cross-section exported from Siemens NX4.0 is revolved using HyperMesh V9.0 to create the 3D FE mesh. The geometric cross-section is first meshed using HyperMesh and these elements are revolved to form a 3D FE model (assuming the geometry is axisymmetric). The element quality can be controlled by deciding the number of elements that is required along the circumference of the casing. The FE model thus built is exported and solved in ANSYS V11.0. The results are extracted from ANSYS and input to iSIGHT 3.5 for optimization studies.

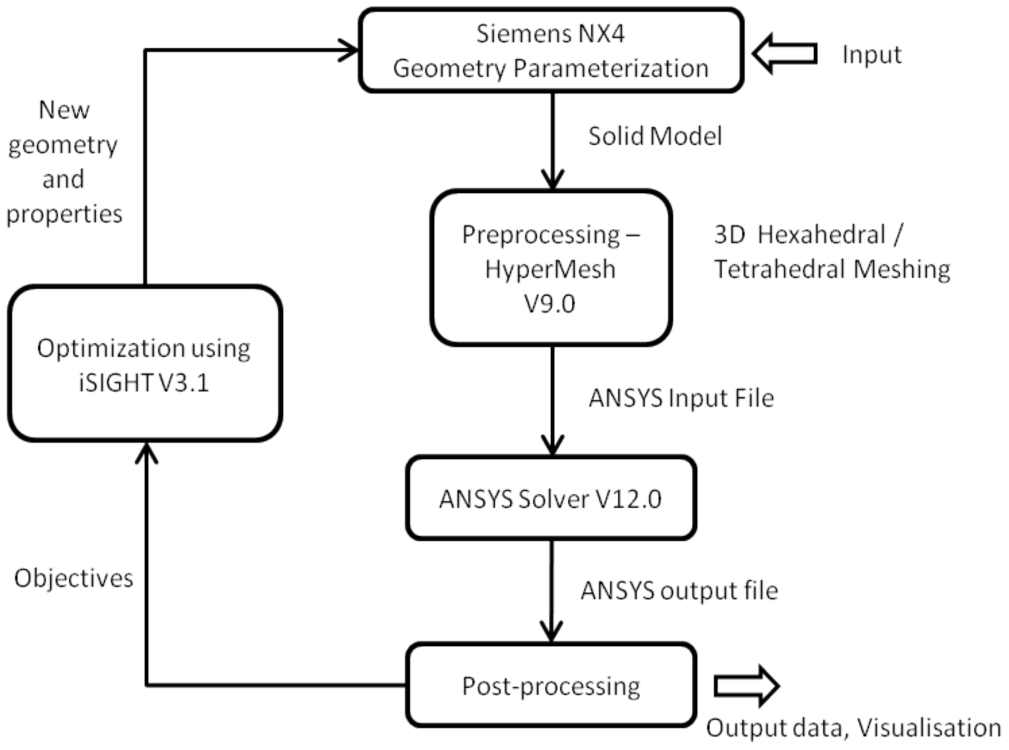

Siemens NX4
Geometry Parameterization
Input
Solid Model
New geometry and properties
Preprocessing – HyperMesh V9.0
3D Hexahedral / Tetrahedral Meshing
Optimization using iSIGHT V3.1
ANSYS Input File
ANSYS Solver V12.0
Objectives
ANSYS output file
Post-processing
Output data, Visualisation

FIGURE 6.6: Optimization workflow involving the 3D FE model

On the other hand, the 2D model workflow shown in Fig. 6.7 uses Mantle-2D based on the algorithm introduced in Fig. 6.2 to build a dimensionally reduced FE model in batch mode. Both the 3D and the 2D optimization workflow's use the same geometric model built using NX. The IGES file exported from NX is used in Mantle-2D to extract the midline of the geometry and use it to create the shell based FE model in HyperMesh/ANSYS. This shell model is then analyzed in ANSYS for the same loads and boundary conditions as the solid model.

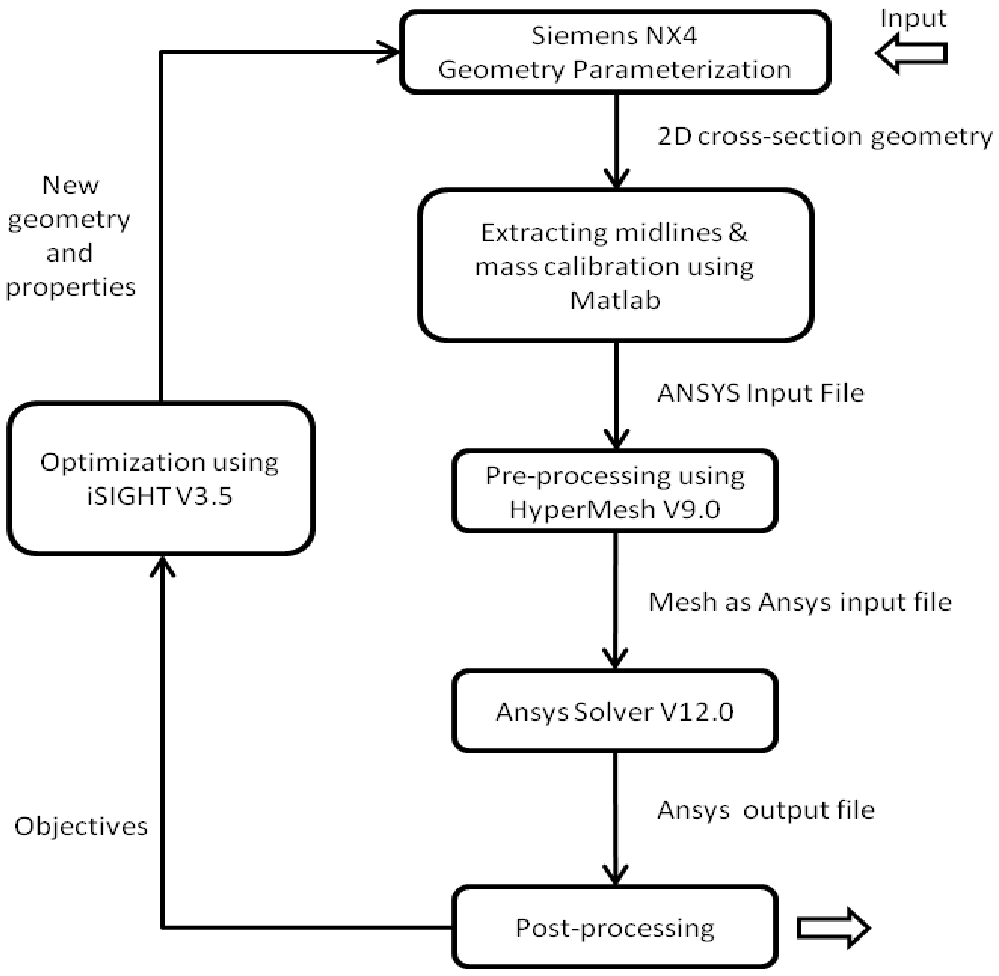

Input
Siemens NX4
Geometry Parameterization
2D cross-section geometry
New geometry and properties
Extracting midlines & mass calibration using Matlab
ANSYS Input File
Optimization using iSIGHT V3.5
Pre-processing using HyperMesh V9.0
Mesh as Ansys input file
Ansys Solver V12.0
Ansys output file
Objectives
Post-processing

Figure 6.7: Optimization workflow involving the MAT based 2D FE model

6.4 Results

A detailed study has been conducted in order to compare the shell FE model's properties with the solid FE model. The solid model mass, center of gravity, stiffness in various directions

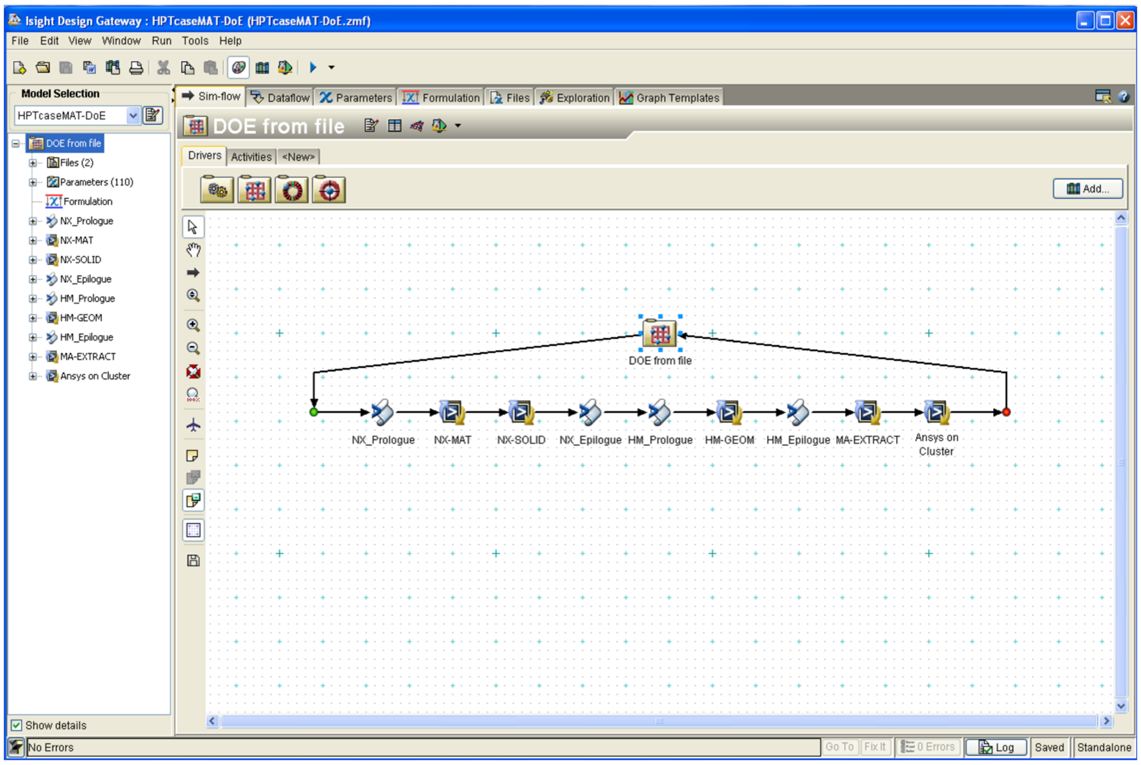


Figure 6.8: Optimization workflow as implemented using iSIGHT-FD V3.1

etc, are properties that the shell model should possess without deviating too much from the solid model. The shell model can replace the solid model in design studies only if it has the same characteristics as the solid model. In order to ascertain this, modal analysis for the first 10 natural frequencies, structural analysis for 196 designs and multi-objective optimization was performed for both the solid and shell FE models and their results compared.

6.4.1 Modal Analysis

Modal analysis of the solid and MAT based shell model has been performed and the results are as presented in Table. 6.1. It may be observed that the MAT shell model natural frequencies differ from the solid model by less than 6% in most cases for the first 10 modes.

6.4.2 Design of Experiments (DoE)

A DoE study using 196 points calculated using 'LPtau' was conducted in order to understand the MAT shell model behaviour when compared to the solid model for the design study. The DoE consists of seven variables and two objectives. The seven variables are six different thicknesses of the engine case being studied and a fillet radius. The two objectives are the

Table 6.1: Summary of modal analysis results

	Solid	MATShell	% Diff
Elements	73400	76200	-
Nodes	107400	76400	-
Mass(kg)	168.89	166.6	1.12
CoG(Z)	267.11	270.49	1.25
Mode 1	420.57	430.49	2.36
Mode 2	420.57	430.49	2.36
Mode 3	461.71	461.73	0.00
Mode 4	461.71	461.73	0.00
Mode 5	481.96	507.89	5.38
Mode 6	481.96	507.89	5.38
Mode 7	590.19	594.03	0.65
Mode 8	590.19	594.03	0.65
Mode 9	616.28	653.38	6.02
Mode 10	616.28	653.38	6.02

total mass of the component and the total deflection of the part at four different locations along the bottom dead center (BDC) of the part. The model was analyzed for a given set of loads and boundary conditions. One end of the model was constrained in all degrees of freedom while the other end had an axial load as shown in Fig. 6.9. In order to account for the mass during the structural analysis, a 1g downward gravity load was also included.

The total analysis time for one run of the datum shell model is about five minutes (including the medial axis extraction process and with 76200 4 noded shell elements) while the datum solid model requires approximately 45 minutes (for 73400 20 noded hexahedral elements) when run on a dual processor 2GB RAM machine. This shows the reduction in solution time that can be achieved by dimensional reduction of a complex solid model into simpler shell model when performing design optimization. The distribution of the DoE results from the 3D model and the 2D model are as shown in Fig. 6.10 and Fig. 6.11 respectively. 'LNTL' and 'UNTL' are the lower and upper natural limits of the data being plot. As can be seen, the typical mass and deflection error is around 1% and 5.0% respectively.

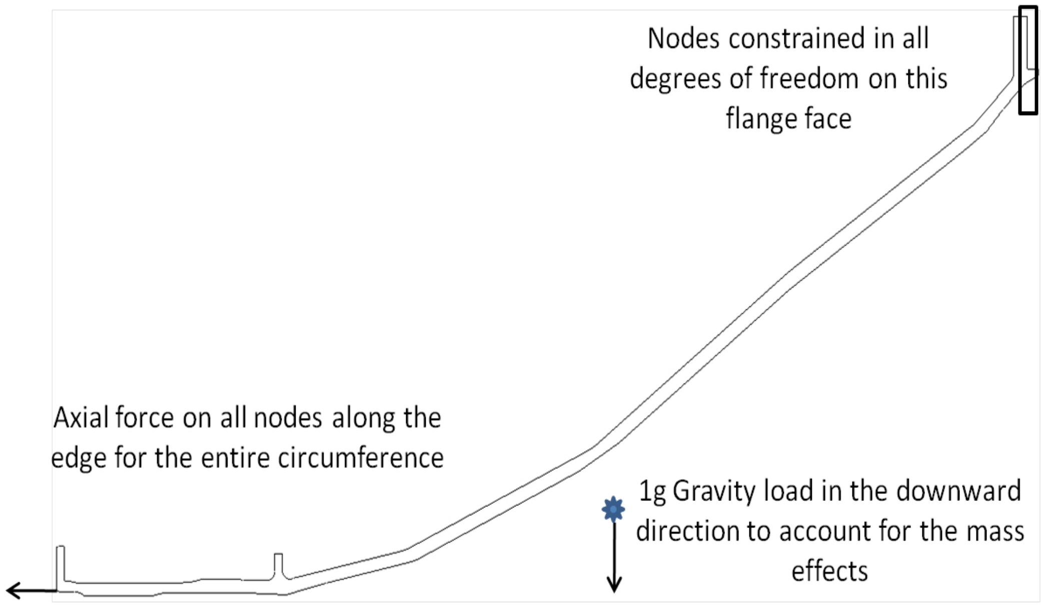

Nodes constrained in all degrees of freedom on this flange face
Axial force on all nodes along the edge for the entire circumference
1g Gravity load in the downward direction to account for the mass effects

FIGURE 6.9: A simplified schematic of the loads and boundary condition used for the high pressure turbine case design study

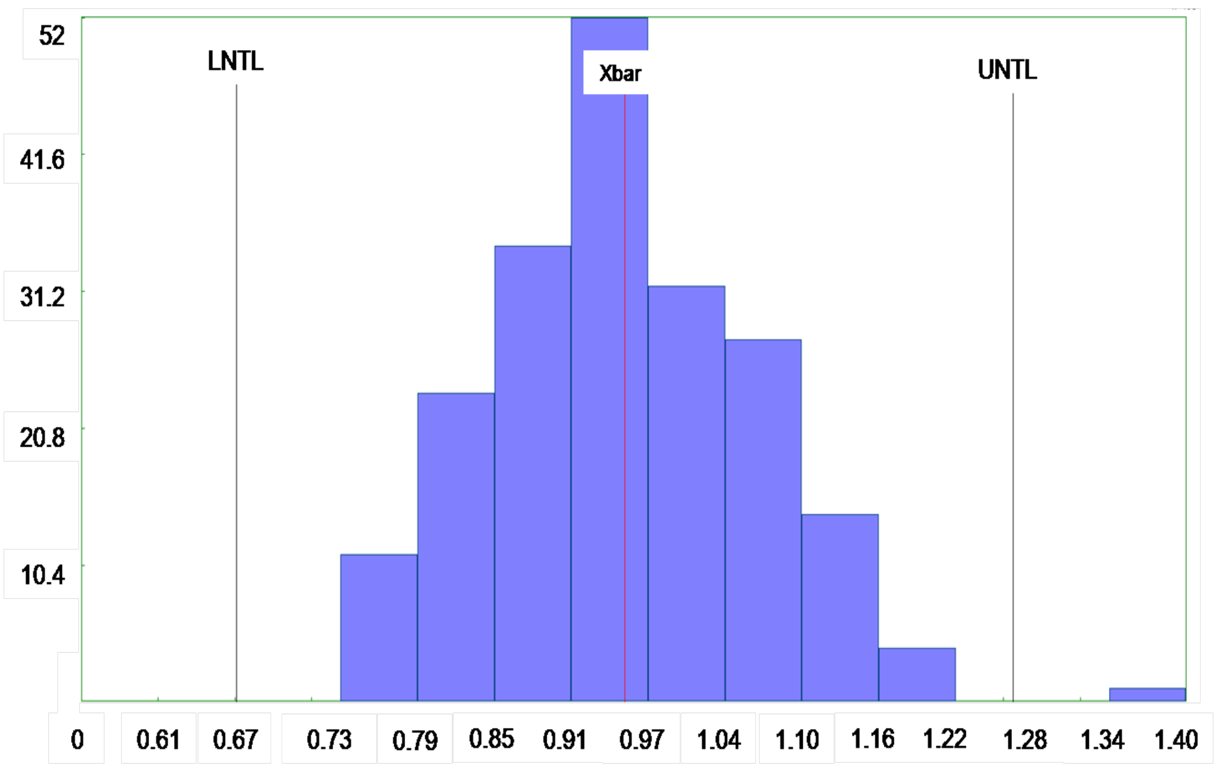


FIGURE 6.10: Distribution of %diff in mass between the shell and the solid model

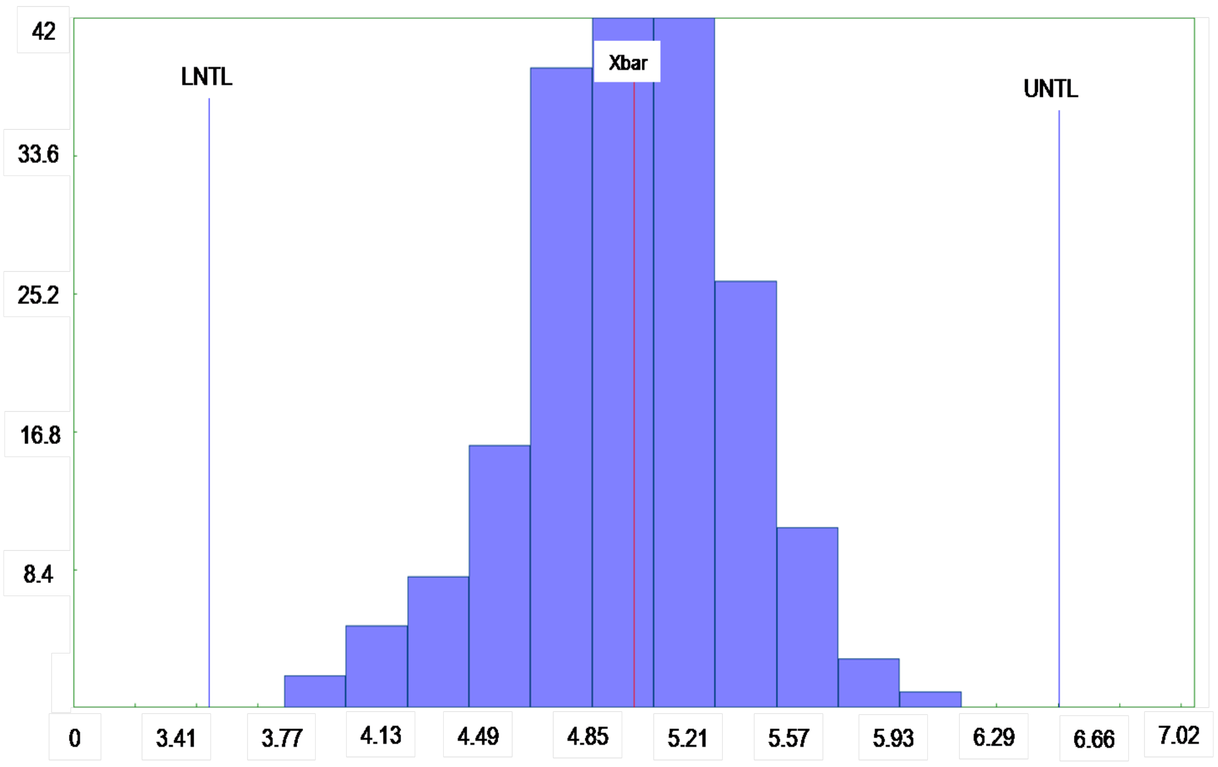


FIGURE 6.11: Distribution of %diff in deflection of the shell and the solid model

6.4.3 Multi-objective Optimization

The two objectives that need to be minimized in most casing design study are the mass and deflection of the component. i.e., increase in stiffness with no or little addition to mass. The current base design has a mass of 168.49kg and a maximum total resultant deflection of 5.54mm. The design optimization study was conducted here using the direct NSGA2 (Non-dominated Sorted Genetic Algorithm, refer [99], [100]) that is well suited for multi-objective cases. A population size of 100 for 12 generations was run making it a 1200 point design space search for the solid model. Since the MAT based approach is so much faster, a longer search of 20 generations using 2000 design points was possible using shell elements. The solid element based FE model and the MAT based shell model were both run using iSIGHT V3.5 and the results are as shown in Fig. 6.12 and Fig. 6.13 The large circular point in the middle of the solid model plot and large square point in the shell model plot represent the mass and deflection of the datum design (current base design).

It can be observed from the optimization results of the solid and shell models that for the given set of loads and boundary conditions, better designs than the current base design can be found. A low mass design with the 117.38kg as mass, results in a total deflection of

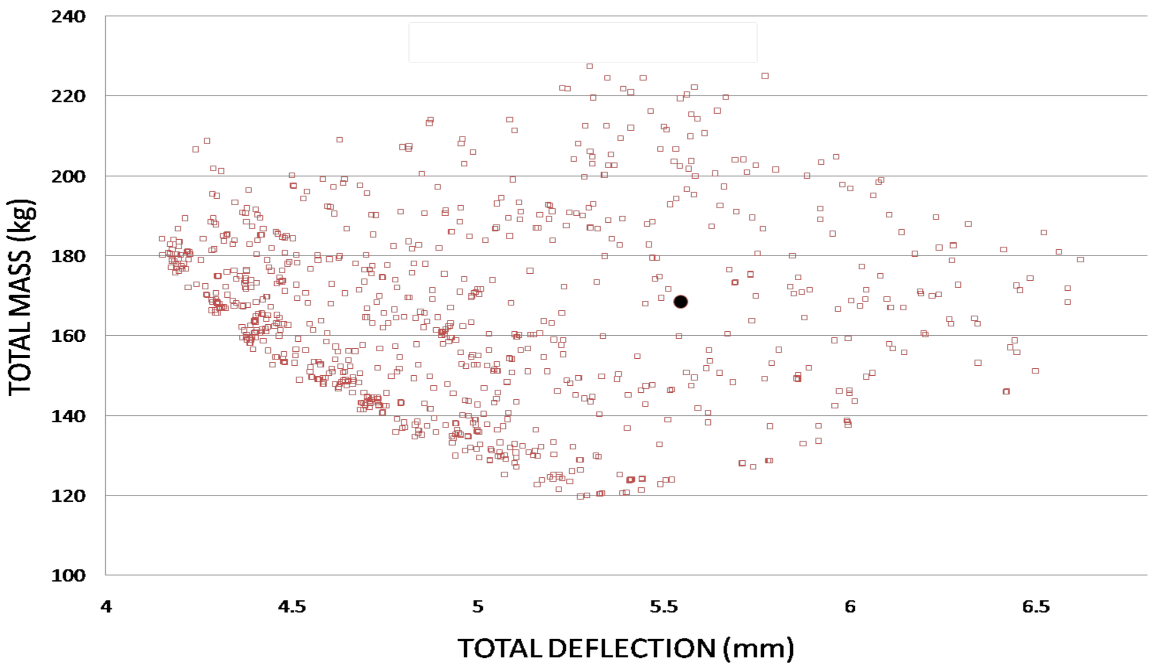


FIGURE 6.12: Solid model deflection Vs Mass plot for 1200 runs

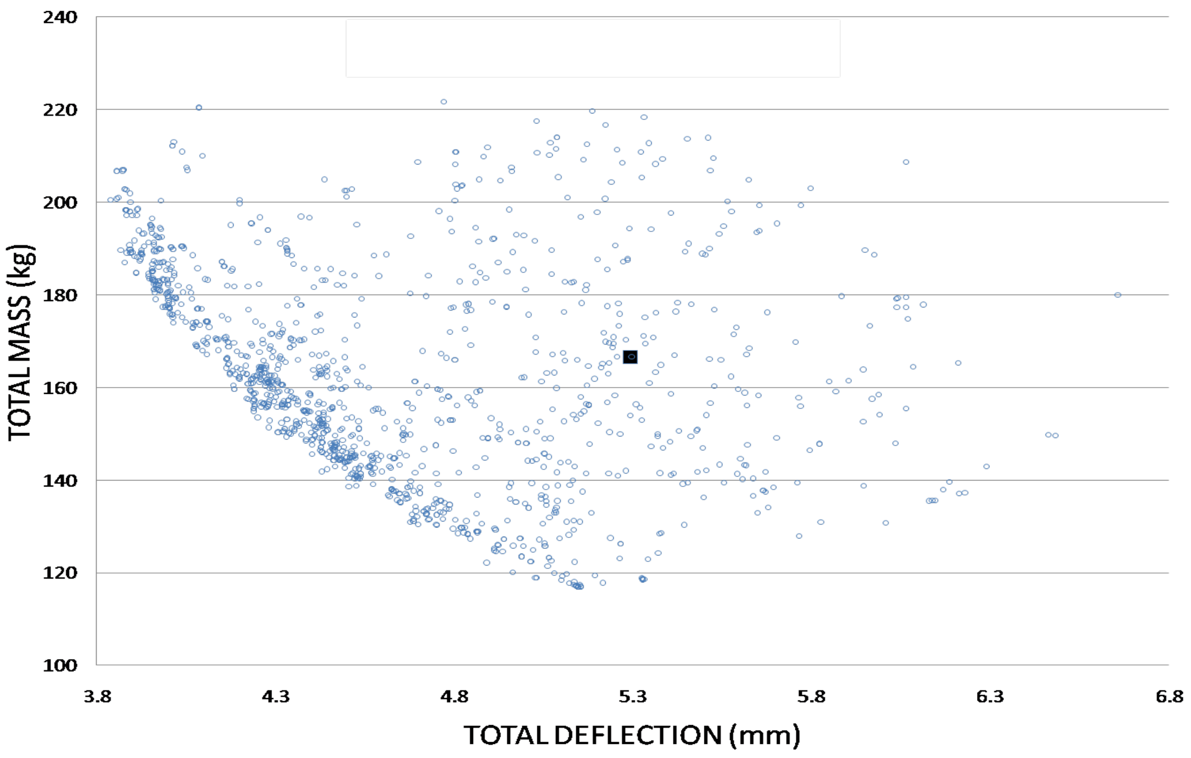


FIGURE 6.13: MAT based shell model deflection Vs Mass plot for 2000 runs

5.15mm which results in the reduction of mass by 51.11kg and deflection by 0.395mm when compared to the base design. Similarly, a low deflection design with 3.83mm total deflection results in a mass of 200.56kg. This is a reduction in deflection by 1.71mm but with an increased mass of 32.07kg. These two points are from the two ends of the Pareto front. The manufacturability of the best design with current manufacturing techniques would need to be accounted for when choosing the optimum geometry as the final design. However, the designer has the option to choose any design in the explored design space such that the optimum design suits the design requirements of a part that goes into an assembly.

The objective of performing the optimization study using both the solid and the MAT shell models was to ascertain if the shell model resulted in the same final design points as the solid model. If the MAT based shell model is used to replace the solid model for optimization purposes, the results from the best shell model designs (when converted to solid model form) should lie on the Pareto front of the solid model. In order to confirm this, the best designs from the shell model were converted back into solid models. Results from these converted models were mapped onto Fig. 6.12 and the resulting plot is as shown in Fig. 6.14. All the small solid circular points are from the Pareto front of the shell model mapped onto the solid model results. It is evident from this plot that the MAT shell model can be used as a dimensionally reduced cheaper to run FE model that can replace the more expensive solid model for optimization studies.

6.5 Conclusions

In the casing model studied here, the savings in terms of FE modeling time and analysis time are significant when Medial Axis Transform (MAT) based shell models are used for axisymmetric aeroengine components. The shell model requires approximately 3-4 minutes to build and solve while the equivalent solid model needs approximately an hour when run using a dual processor 2GB RAM machine. The study presented here shows that multi-objective design optimization carried out using MAT based models are able to improve designs in a fashion consistent with searches carried out using full 3D FE models. It can thus be concluded that the MAT based shell model could satisfactorily replace solid based FE models during design optimization studies.

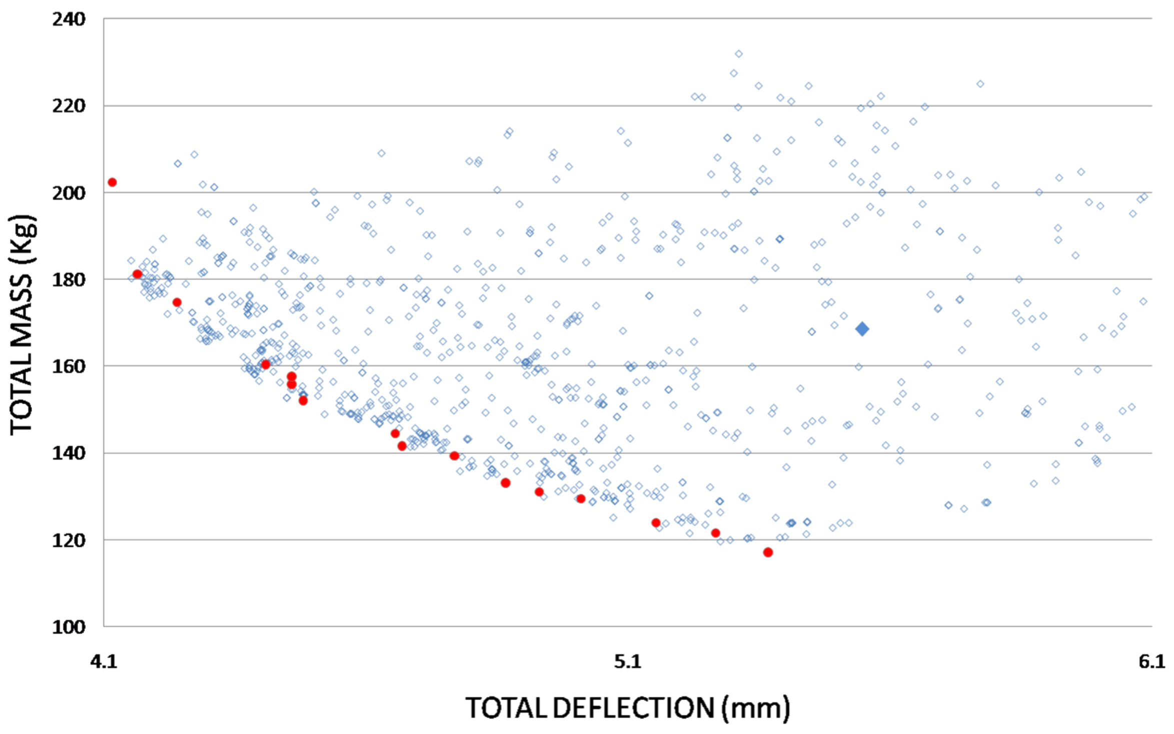


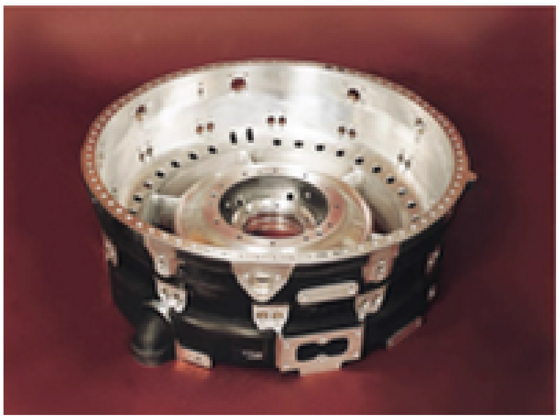
FIGURE 6.14: MAT based shell model Pareto front superimposed onto the solid model results

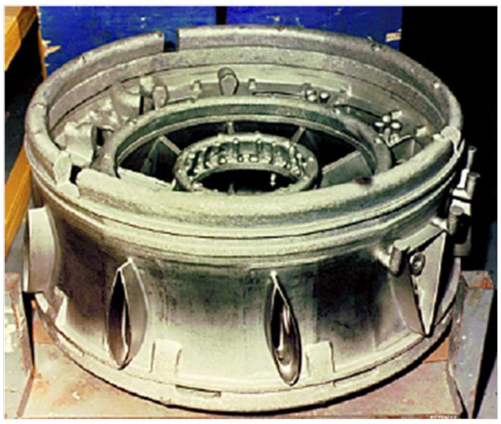
Chapter 7

Medial surface mesh of non-axisymmetric 3D geometry

In Chapter 6, simpler FE models that can replace complex axisymmetric 3D FE models for optimization studies have been demonstrated. However, the optimization workflow presented in Chapter 6 is applicable only to axisymmetric components. The load bearing casings in an aeroengine are not always axisymmetric. The rear fan case, the intermediate case, bearing housings etc are few components that are either cyclic-symmetric or non-axisymmetric. Dimensionally reducing such complex 3D models to simpler 2D surface models is a non-trivial problem. The work presented in this chapter is about achieving this dimensional reduction and its use for optimization studies. Wang *et al.* [101], Xie *et al.* [102], Sherbrooke *et al.* [103], Li *et al.* [104], Yong *et al.* [105] and Du *et al.* [106] discuss how spheres could be used to extract medial surfaces of three dimensional objects. The creation of Voronoi diagrams in 3D are essential when dealing with medial surfaces of 3D objects. Ledoux[107] and Boada *et al.* [108] discuss methods to compute Voronoi of 3D objects robustly. However, in this chapter, since the 3D mesh is created using a commercial software as input while extracting medial surfaces, emphasis is not laid on creation of Voronoi diagrams.

A few typical complex non-axisymmetric 3D components in a gas turbine engine are as shown in Fig. 7.1. Such components are generally meshed using tetrahedral elements for use in finite element analysis. Depending on the analysis requirement, the number of elements could range from a few thousands to a few million. Creating hexahedral mesh for a complex geometry is a very cumbersome task which is seldom done. By ensuring that the mesh satisfies the best practice requirements at all critical regions, the engineer would eventually end up with a large FE model. As mentioned in earlier chapters, the time taken to solve such large FE models is detrimental when used in an optimization framework constituting an assembly of models. However, the biggest advantage 3D meshing has is that it can be achieved in batch mode. Though this depends on the geometry and the FE package's ability to import the geometry without any loss of information. For the optimization process being put together in this research program no such losses between CAD and FE packages can be accommodated.


(a) Engine casing - 1


(b) Engine casing - 2

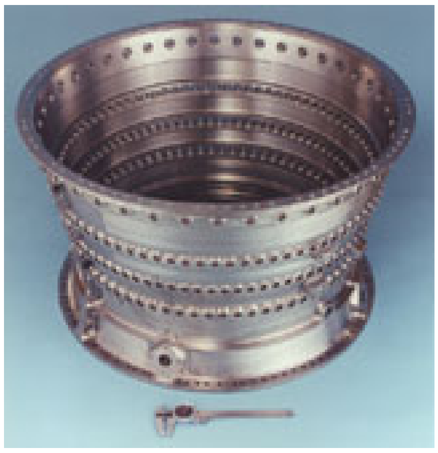

(c) Engine casing - 3

FIGURE 7.1: Non-axisymmetric geometries in a gas turbine engine

7.1 Dimensional reduction of complex 3D models

It is evident by now that complex 3D models are ideal for detailed design studies but not for the preliminary stages when various designs are looked into. A method to break these complex models into simpler models needs to be created. Based on the literature review done, no existing package, commercial or academic, is able to handle real industrial problems effectively. This research program is about optimization methods for use in a real industry setting. A process to achieve this has been demonstrated in the next few sections.

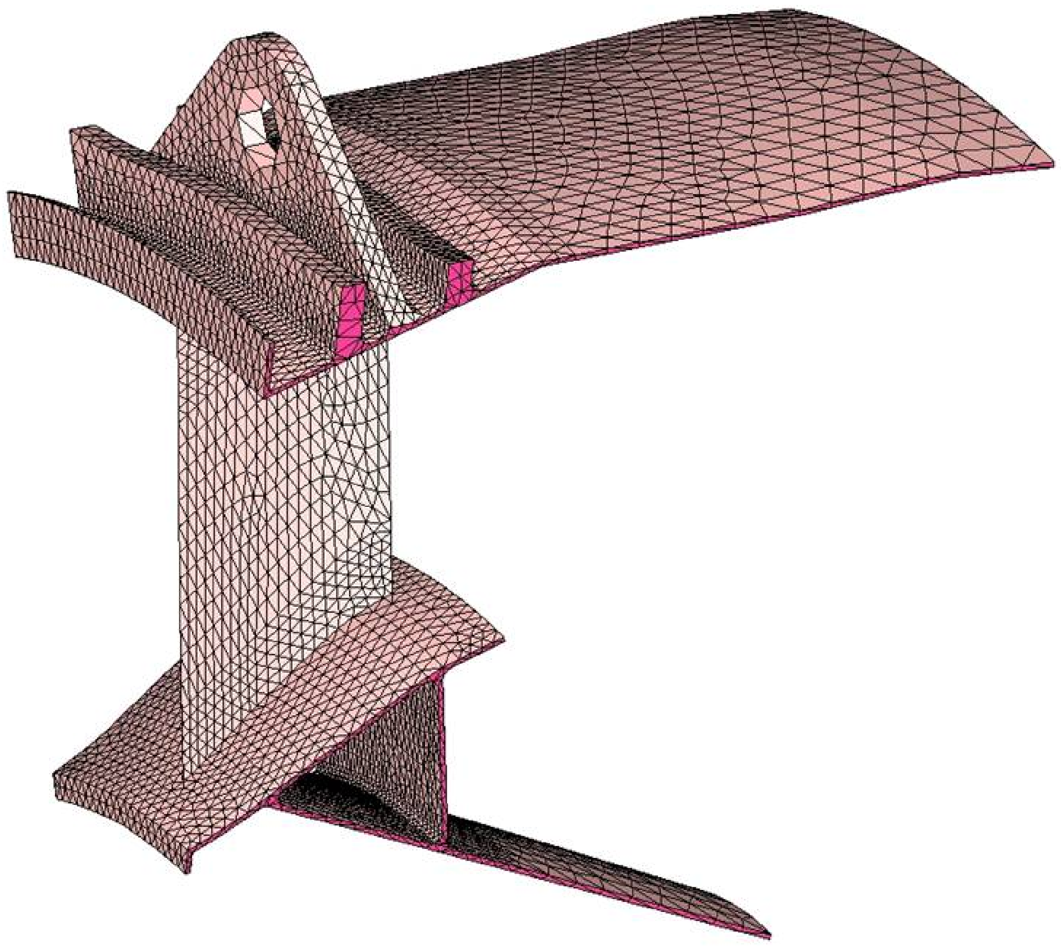


FIGURE 7.2: The solid mesh used an input to extract midsurface using Mantle-3D

An ideal scenario would be to represent the rear mount case FE mesh shown in Fig. 7.2 as a midsurface shown in Fig. 7.3 retaining the stiffness properties of the solid model. These shell based FE models could then be used to replace all 3D non-axisymmetric models when simulating the whole engine for tip clearance studies. Lower order 2D finite elements result in fewer degrees of freedom when compared to higher order 3D FE models created using 10 noded tetrahedral or 20 noded hexahedral elements.

A 3D FE mesh of a typical gas turbine rear mount link (1/13 sector model) as shown in Fig. 7.10 consists of 141,226 elements, 171,569 nodes and approximately 514,707 degrees of freedom. When this sector model is used to create a full 360° model, the FE model size would increase to approx 6.7 million degrees of freedom. On the contrary, a very fine 2D FE mesh of the same rear mount link shown in Fig. 7.11 would result in 85196 elements, 42994 nodes and 257964 degrees of freedom. When this sector model is used to create a full 360° model, the degrees of freedom of the FE model would have reduced from 6.7 million to 3.35 million. However, the number of elements in the 2D FE mesh can be further reduced by more than half and still retain the same properties as the fine mesh. This will reduce the problem size to less than a million degrees of freedom.

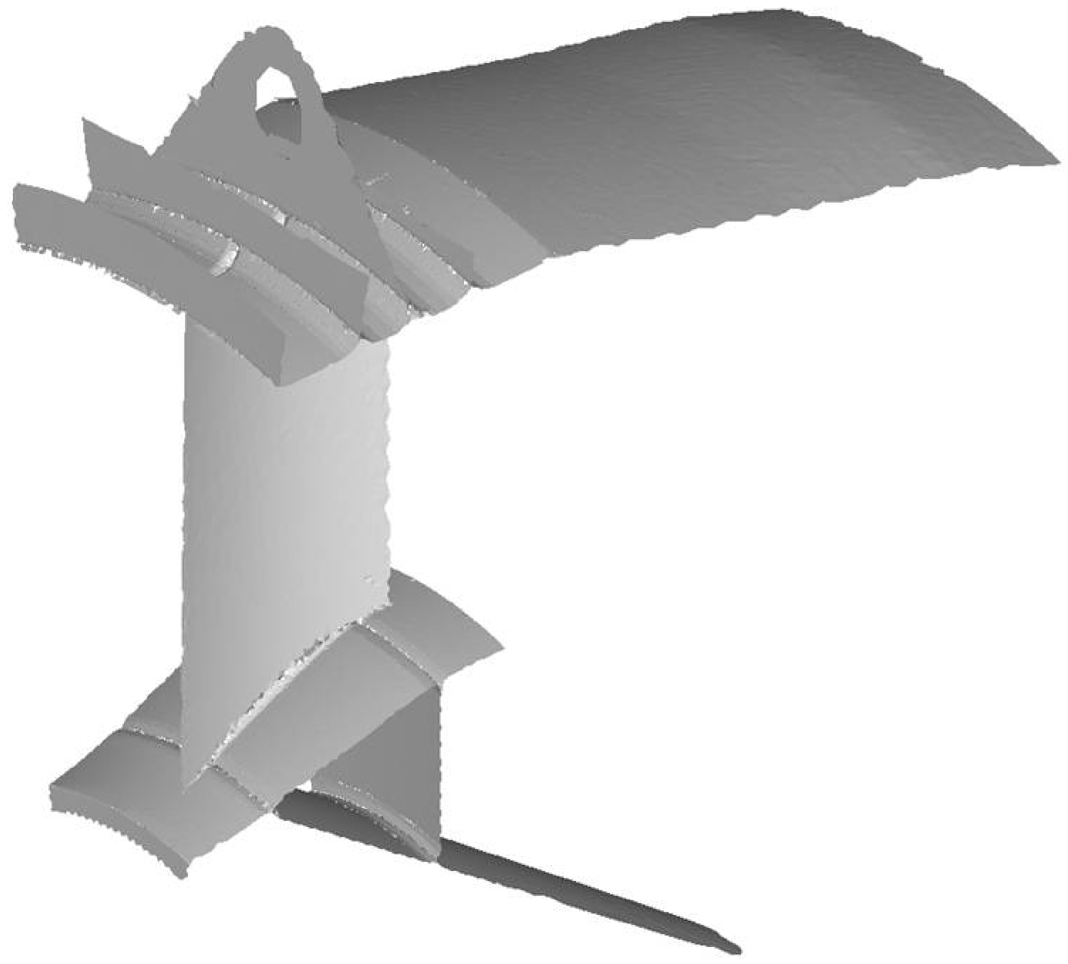


FIGURE 7.3: The medial surface of a the rear mount as extracted by Mantle-3D

7.2 Challenges in creating medial mesh of complex 3D non-axisymmetric geometries

The medial surface algorithm in general is based on 3D Voronoi diagrams which is a dual of Delaunay tetrahedralization. To create the 3D Voronoi diagram an enclosed geometry is mandatory. A similar requirement is also set by meshing tools while creating tetrahedral elements for a 3D geometry. Creating a complex 3D geometry which can be successfully meshed the first time it is imported from a CAD engine is very unlikely. This is due to the difference in geometric tolerances between the CAD and CAE software as mentioned in [7] by Beall *et al.* Quite often a solid geometry from the CAD world when imported into the FE realm would only consist of points, lines and surfaces, but no solids. This is due to the sliver surfaces created in the CAD software which are not imported into the FE software. When these sliver surfaces fail to be represented in an FE software as geometric entities, the solid imported is not an enclosed volume anymore. Most of the FE packages that tetrahedralize solid geometry need an enclosed volume as the most basic and vital input. If the input to a medial surface algorithm based on 3D Voronoi's is not an enclosed volume, the code will fail to generate the medial surface since the geometry is not an enclosed volume.

However, there are geometry fixing tools like CADfix which identify these sliver surfaces and repair them before exporting into a FE package. These tools are still not very generic since the complexities involved in identifying these problem areas in a detailed 3D geometry require manual intervention. For an optimization process such manual intervention is detrimental. The CADfix medial object tool kit as demonstrated in the VIVACE project is built around its geometry cleaning capabilities and is based on the work done by Butlin *et al.* in [84] and [109]. The principle behind this modeling method is to retain all complex features as 3D and replace all other regions (constant thickness regions) with either 2D or 1D elements. Identifying these regions in simple geometries might be trivial but an uphill task when handling geometries as shown in Fig. 7.1. To use such a process without any manual intervention in an optimization workflow is still not feasible.

This is one of the primary reasons why the medial surface algorithm proposed in this thesis has been developed with the idea of using either the solid mesh or the surface mesh of the geometry as an input. If the geometry is meshable with tetrahedral elements, then the FE analysts can use this geometry for creating the medial mesh for use in their optimization studies directly. However, if the solid geometry is not meshable, the surface mesh of this geometry can be used to create the medial mesh. This feature of the proposed code when implemented could be used to create the medial mesh of both enclosed volumes or otherwise. This enables the algorithm to be used even on those geometries which are not created for use in FE packages.

7.3 The medial surface algorithm : MANTLE-3D (Non axisymmetric geometries)

This medial surface code has been developed during the course of this research program to demonstrate how shell meshes can be used to replace complex geometries as seen in gas turbine engines during optimization studies. An attempt has been made to make the code as generic as possible, but still needs manual intervention for its working. Unlike Mantle-2D, the 3D code is still not ready for use in an optimization cycle. The medial surface generation algorithm is as shown in Fig. 7.4. In Appendix E, the Matlab script that forms Mantle-3D is presented while each module as executed in Fig. 7.5 of Mantle-3D is described in the sections below.

A few simple geometries were built and their medial surfaces meshed to test the use of Mantle-3D. The geometries in Fig. 7.6(b), 7.6(c) and 7.6(d) can be considered as axisymmetric and handled by the Mantle-2D algorithm. However, emphasis has been laid on handling geometries as shown in Fig. 7.6(e). This geometry represents a casing with 3 flanges on the top and a solid strut running perpendicular to the flanges at the bottom. The volume where the 3 flanges interact with the perpendicular strut and the medial surface at these junctions makes this a complex test case.

7.3.1 Inputs to Mantle-3D : 'Mantle3D' module

This module of Mantle-3D uses the finite element data defined as a NASTRAN input file and not the geometry. Most medial surface codes would require closed volumes in order to ascertain Voronoi vertices in 3 dimensions. However, if the geometry does not form a closed volume, manual intervention is necessary to create a closed volume. Hence, only meshable geometries can be used at this stage as the solid elements which define the volume of the model are vital. The quality of the input mesh defines the quality of the output medial mesh. It is recommended at this stage to de-feature the model during the parametrization process in order to reduce the elements that are required to capture these intricate features. The

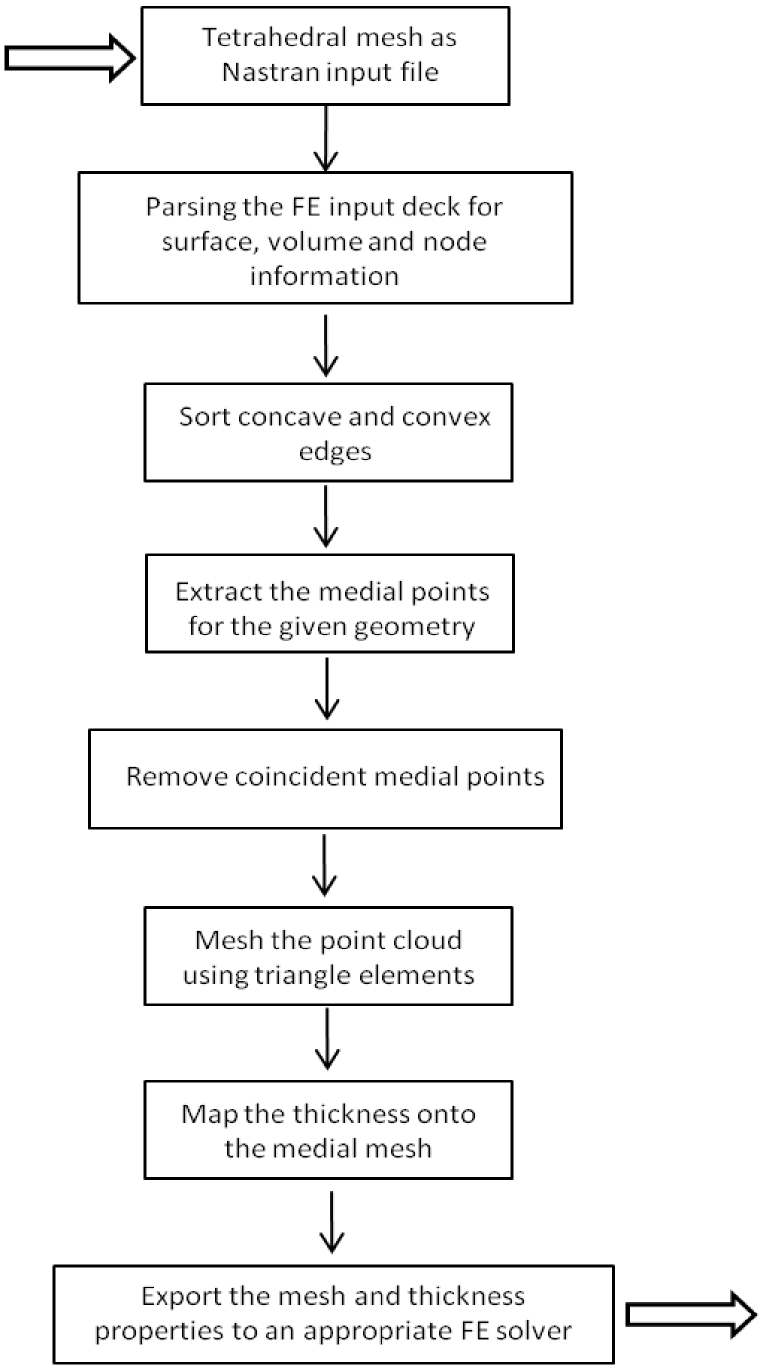

Tetrahedral mesh as Nastran input file
Parsing the FE input deck for surface, volume and node information
Sort concave and convex edges
Extract the medial points for the given geometry
Remove coincident medial points
Mesh the point cloud using triangle elements
Map the thickness onto the medial mesh
Export the mesh and thickness properties to an appropriate FE solver

FIGURE 7.4: The Mantle-3D based medial surface extraction algorithm

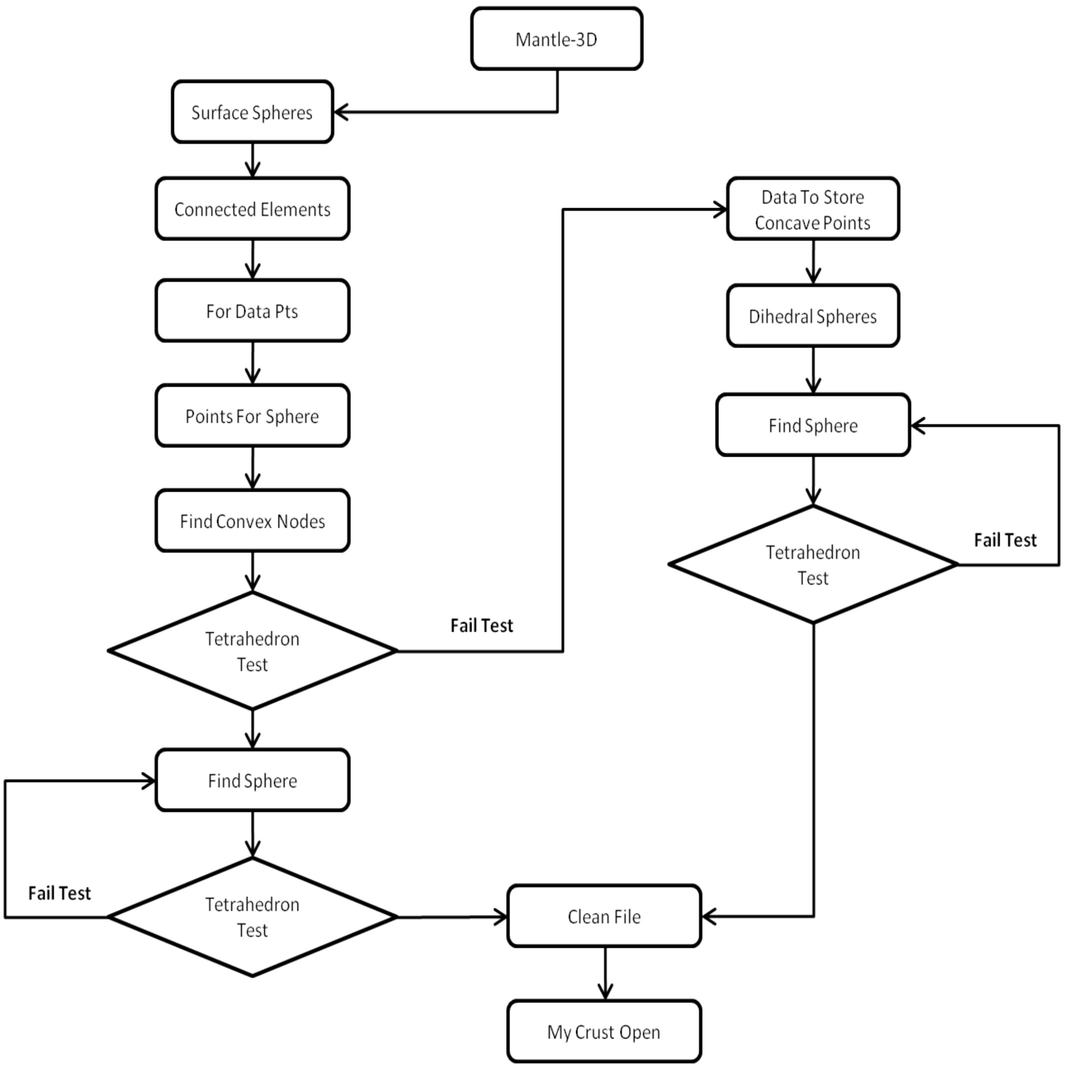
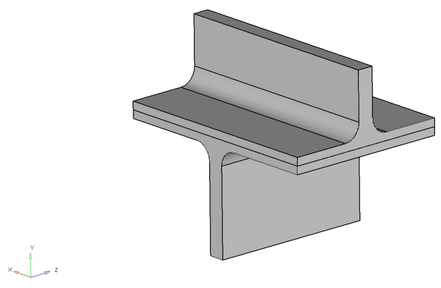
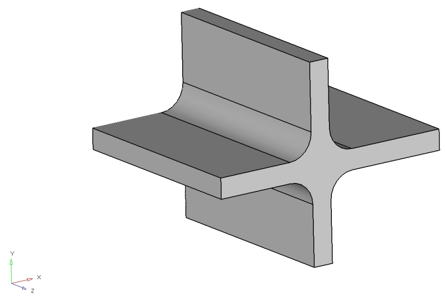


FIGURE 7.5: Execution sequence of various modules of Mantle-3D

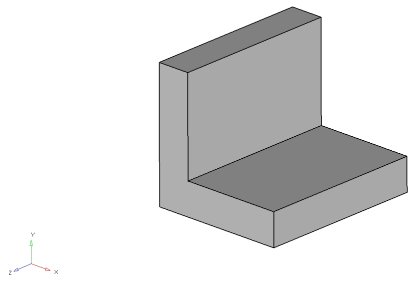
number of tetrahedral elements in the solid model also defines the time taken by Mantle-3D to create the medial mesh. Fewer solid elements would result in fewer elements in the medial mesh and vice-versa. However, a finer medial mesh can be obtained by setting the right values for the input parameters mentioned below. The NASTRAN deck has been chosen instead of ANSYS since the FE input deck for ANSYS consists of duplicate columns which need to be parsed before use. When a shell element is exported in ANSYS format, the number of node columns are 8 while it should ideally be 4 (3 for a triangle element). These 8 columns have to be parsed for a unique set of 3 node numbers before discarding the rest. The NASTRAN input deck on the other hand consists of only 3 columns while defining a CTRIA card. A few of the input parameters for Mantle-3D described below are based on the requirements of the user:



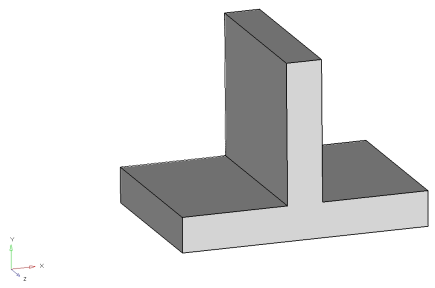
(a) Test case - 1



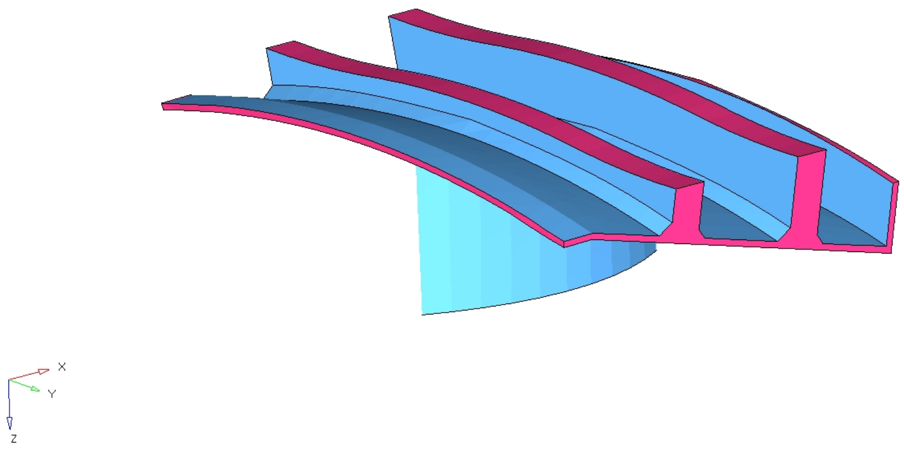
(b) Test case - 2



(c) Test case - 3



(d) Test case - 4



(e) Test case - 5

FIGURE 7.6: Test case geometries used with Mantle-3D

- ⋆ "elemsAcrossThickness = 3": If this parameter which represents the number of tetrahedrons along the thickest section of the geometry is set to less than the number of tetrahedron's in the thickest region of the input geometry, the code might not find the maximal spheres at all thick regions. If a number greater than the max number of elements along the thickness is chosen, it would take longer for the code to find medial points for the whole geometry. This number should be changed based on mesh density definition set during the mesh generation process.

- ⋆ "NumberOfNodes = 11": This parameter defines the number of parts the distance between two chosen points will be split into for use in 'ForDataPts' module (never to be set less than 1). It defines the number of points on the boundary required while finding the maximal sphere for a given input point on the boundary. If this number is set to 1, the diameter of the sphere found could be more than the thickness of the geometry depending on the boundary point set defined. If the input boundary point set is dense and 'NumberOfNodes' set to 1, chances are the sphere found is optimal, but if the boundary point set is coarse, 'NumberOfNodes' should be set to more than 1 depending on the accuracy desired by the user.

- ⋆ "numnodes = 1": Defines the number of spheres to be found between a set of points (never to be set less than 1). This parameter should be set to atleast 2 for coarse boundary point sets. The parameter 'NumberOfNodes' defines the number of points on the boundary, while 'numnodes' defines the number of spheres that need to be found for each set of points on the boundary of the geometry. When set to 1, only the centroid of three points is used for calculating the sphere. When the boundary points are dense, it is advisable to set this parameter to 1 but needs to be set to atleast 2 when boundary points are coarse.

- ⋆ "AngleTolerance = 15": This parameter is set for use in 'FindConvexNodes' module of Mantle-3D. The angle tolerance is input to obtain the convex and concave vertices of the geometry. Based on this segregation, the concave points are used in the 'Dihedral-Spheres' script to find medial points at all concave vertices while the convex vertices are used in the final creation of the medial surface without further processing.

- ⋆ "VectorsAlongDihedralAngle = 5": This parameter should be set to 3, 5 or 7 for use in 'DataToStoreConcavePoints' module of the code. This parameter defines the number of spheres required at all concave vertices and it can be set to 3 if the input boundary point set has captured fillets at critical locations else can be set to 5 or 7 depending on user requirement.

7.3.2 Extract medial points: ‘SurfaceSpheres’ module

This module is responsible for extracting medial points based on user requirement. The loop that is run inside this module is as follows:

⋆ Find the normal vector of each point on the boundary. Let $(p1)$, $(p2)$ and $(p3)$ be the three points on the boundary. The normal vector $(\vec{n})$ is then given by:

$$\begin{aligned}\vec{a} &= p2 - p1,\\ \vec{b} &= p3 - p1,\\ \vec{n} &= \vec{a} \times \vec{b}\end{aligned} \tag{7.1}$$

⋆ Define the boundary nodes between which to fit the sphere. If the boundary points are coarse and the input parameter ‘NumberOfNodes’ is set to more than 1, the ‘ForDataPts’ Matlab module is used to define more points on the boundary of the geometry. The accuracy of the maximal sphere for this boundary point being processed depends on the density of points on the boundary based on ‘NumberOfNodes’ parameter.

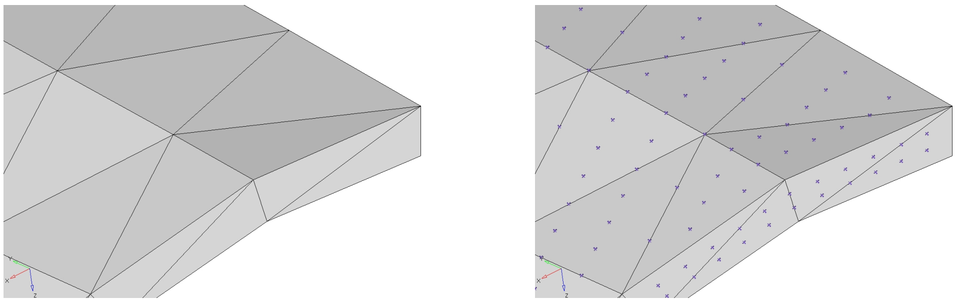
⋆ Check if these are concave or convex points using ‘FindConvexNodes’ module. If the points are concave, then store them for use in ‘DihedralSpheres’ else go ahead.

⋆ Find the maximal spheres using a line search algorithm.

7.3.3 Data points while searching for the maximal sphere : ‘ForDatapts’ module

This script as mentioned earlier populates the outer boundary region where the maximal spheres are being found with a dense cloud of points as shown in Fig. 7.7. The steps to define a fine set of boundary points from the given coarse boundary point set is as follows:

⋆ Find all the boundary points that are around the target point. Pick the boundary edges formed by these points one at a time and divide each based on the input parameter ‘NumberOfNodes’. If a line is defined by 2 points (A) and (B) in 3 dimensions, defined by coordinates (x_a, y_a, z_a) and (x_b, y_b, z_b), the internal division (P) of a line segment and its coordinates (x_p, y_p, z_p) are given by;



(a) Coarse mesh without data points (b) Coarse mesh with data points

FIGURE 7.7: Data points defined on the boundary of the geometry for a coarse surface mesh

$$x_p = \frac{k_1 x_b + k_2 x_a}{k_1 + k_2}, \tag{7.2}$$

$$y_p = \frac{k_1 y_b + k_2 y_a}{k_1 + k_2}, \tag{7.3}$$

$$z_p = \frac{k_1 z_b + k_2 z_a}{k_1 + k_2} \tag{7.4}$$

where (k_1) and (k_2) are the ratio in which the line needs to be split.

⋆ Store the coordinates of (P) in an array.

7.3.4 Number of spheres per boundary shell element: 'PointsForSphere' module

This Matlab module stores a set of points in an array for the number of spheres to be found per loop based on the user input parameter 'numnodes'. Only the centroid of three points is chosen to create a sphere when 'numnodes' is set to 1. This is a valid choice when the input boundary points density is fine. A recommended number for this parameter is not more than 4 for a coarse boundary point set. If a number greater than 2 is chosen, the distance between any two chosen points on the boundary is split based on Equation 7.2 - Equation 7.4 and stored in an array. The point coordinates from this array and the point coordinates form the centroid of the three points are then used to create the spheres. The bigger this number, longer is the time taken to extract all the medial points. The centroid with three vertices is given by;

$$Centroid = \left(\frac{\sum_{i=1}^{3} X_i, \sum_{i=1}^{3} Y_i, \sum_{i=1}^{3} Z_i}{3} \right) \tag{7.5}$$

7.3.5 To find the convex and concave points: 'FindConvexNodes' module

Concave and convex edges of a geometry provide valuable information while extracting the medial data. All the edges of the geometry have to be identified as either concave or convex and dealt with accordingly. All convex edges will be used to identify medial points and also used to create the medial surface/mesh. This is because, all the branching medial surfaces share one edge with the boundary and the other with the branch junction. The information of these edges along the boundary is stored for later use in an array. The concave edges on the other hand are used only to create the medial points and the number of medial points created for every point on the concave edge can be set to either 3, 5 or 7. The choice is left to the user to decide on what number to choose based on the complexity of the 3D model.

Let (L_1) and (L_2) be the two lines that define the two normals, (P_1) and (P_2) be the point vectors and $(\vec{N_1})$ and $(\vec{N_2})$ be the normal vectors of these two lines. The vector equation of these lines can then be given by;

$$L_1 = P_1 + a\vec{N_1} \tag{7.6}$$

$$L_2 = P_2 + b\vec{N_2} \tag{7.7}$$

If we assume that the vectors intersect, we can look for a point on (L_1) that satisfies Equation 7.7, i.e;

$$P_1 + a\vec{N_1} = P_2 + b\vec{N_2}$$

rewriting;

$$a\vec{N_1} = (P_2 - P_1) + b\vec{N_2}$$

Now taking the cross product of each side with $(\vec{N_2})$;

$$a\left(\vec{N_1} \times \vec{N_2}\right) = (P_2 - P_1) \times \vec{N_2} \tag{7.8}$$

substituting Equation 7.8 in Equation 7.6 would yield the intersection point of the two vectors.

This intersection point can either lie inside or outside the geometry. As mentioned earlier, if this intersection point lies outside, the point being considered for finding spheres is from a concave edge, else is from a convex edge. All the tetrahedrons stored during the execution of 'ForDataPts' module discussed in section 7.3.3 are run in a loop to check if the intersection point found lies inside it or outside. If at the end of the loop, none of the tetrahedrons have this point inside them, the point lies outside the geometry.

Let the tetrahedron's 4 vertices be (V_1), (V_2), (V_3), (V_4) and the test point be (P) where;

$$\begin{aligned} V_1 &= (x_1, y_1, z_1) \\ V_2 &= (x_2, y_2, z_2) \\ V_3 &= (x_3, y_3, z_3) \\ V_4 &= (x_4, y_4, z_4) \\ P &= (x, y, z) \end{aligned}$$

Then the point (P) is in the tetrahedron if following five determinants all have the same sign.

$$D0 = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{bmatrix}$$

$$D1 = \begin{bmatrix} x & y & z & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{bmatrix}$$

$$D2 = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x & y & z & 1 \\ x_3 & y_3 & z_3 & 1 \\ x_4 & y_4 & z_4 & 1 \end{bmatrix}$$

$$D3 = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x & y & z & 1 \\ x_4 & y_4 & z_4 & 1 \end{bmatrix}$$

$$D4 = \begin{bmatrix} x_1 & y_1 & z_1 & 1 \\ x_2 & y_2 & z_2 & 1 \\ x_3 & y_3 & z_3 & 1 \\ x & y & z & 1 \end{bmatrix}$$

The other additional checks that can be made with the results of these determinants are:

- ⋆ If the determinant of $(D0 = 0)$, then the tetrahedron being considered for the check is degenerate i.e; the tetrahedrons vertices are coplanar and hence have no volume. This case should never arise since the mesh created as input for use in Mantle-3D is from a commercial FE processor and there are element quality checks to avoid degenerate elements from being exported.
- ⋆ If any other $(Di = 0)$, then (P) lies on the boundary of this tetrahedron.
- ⋆ If the sign of any (Di) differs from that of $(D0)$, then (P) is outside the boundary of this tetrahedron.

- ⋆ If the sign of any (Di) equals that of $(D0)$, then (P) is inside the boundary of this tetrahedron.
- ⋆ If P is inside all 4 boundaries, then it is inside the tetrahedron.
- ⋆ As a check, it must be that $(D0 = D1 + D2 + D3 + D4)$.

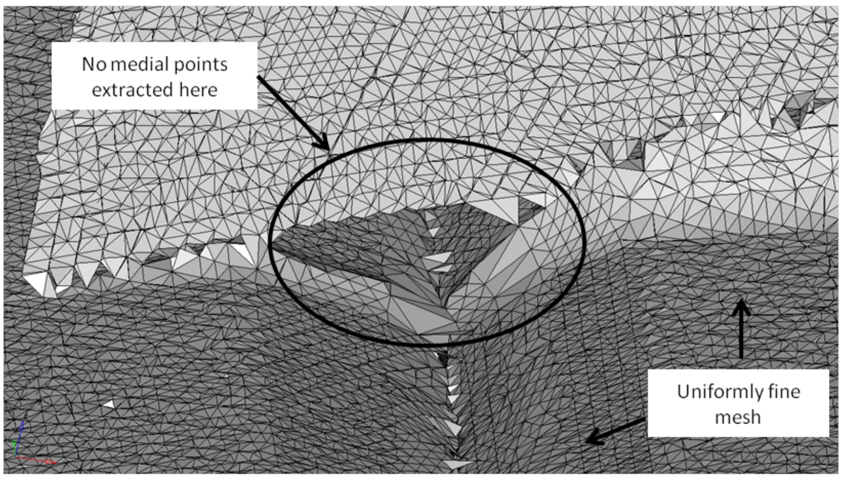
7.3.6 To find spheres at all concave points: 'DihedralSpheres' module

Before implementation of the 'DihedralSpheres' module, all boundary points input to Mantle-3D were treated similar. When complicated geometries without fillets were considered for medial surface mesh extraction, medial points for large regions were not being found as shown in Fig. 7.8(a). As can be seen in Fig. 7.8(a), even for a very fine medial mesh setting and increase in the element count, there were regions where no medial points were extracted. This was a cause of concern since these empty regions constitute the load path and have to represent the right stiffness when compared to the solid model.

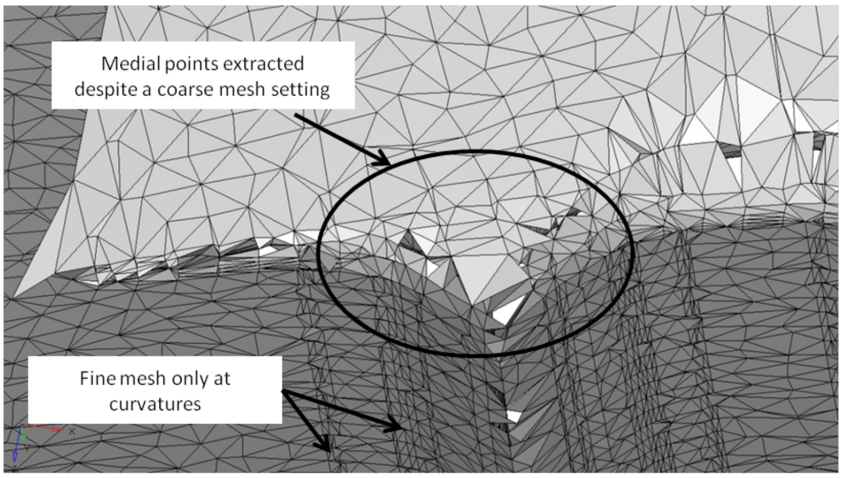
However, with the implementation of 'DihedralSpheres', it was guaranteed that atleast 3, 5 or 7 medial points would be extracted for every concave boundary point as can be seen in Fig. 7.8(b). The user input parameter 'VectorsAlongDihedralAngle' to be set in the 'mantle3D' module described in section 7.3.1 defines whether the number of medial points extracted at these curvatures are 3, 5 or 7. The other big advantage of implementing this module has been reduction in the medial mesh element count. The implementation of this code allowed the creation of a coarse medial mesh for most of the geometry and refining the mesh at all critical locations. Concave edges are generally found when the the topology of the geometry changes. Therefore, with the implementation of this module, instead of refining the medial mesh for the entire geometry, the refinement of the mesh could be concentrated to the regions of the geometry where the topology changed. All concave points are dealt with by this module while the remaining boundary points are processed by 'SurfaceSpheres'.

7.3.7 Remove all duplicate medial points: 'cleanfile' module

Mantle-3D in its current state generates medial points for all boundary points provided as input. No attempt is made to restrict the script from extracting medial points depending on their distance from each other. If there is a boundary point as shown in Fig. 7.9(a) on the top surface and a point exactly opposite on the bottom surface, two medial points for each of these boundary points would be generated. The medial points extracted for such boundary



(a) Before implementation of 'DihedralSpheres' module

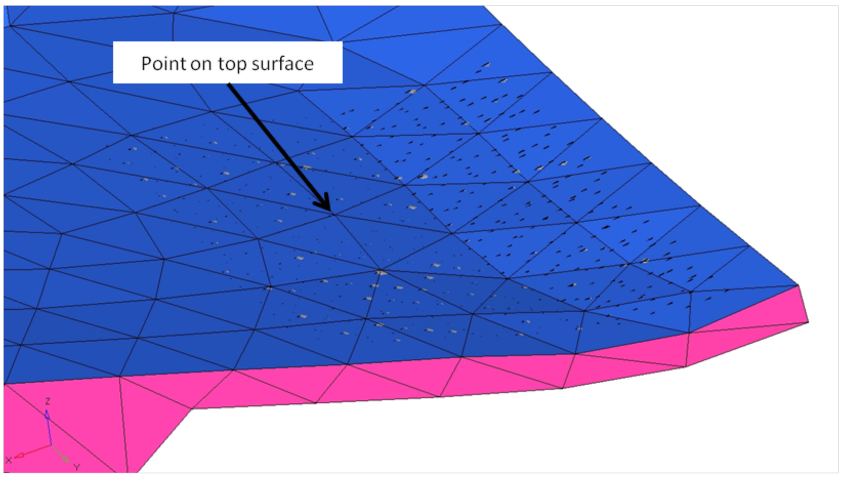

Medial points extracted despite a coarse mesh setting
Fine mesh only at curvatures

(b) After implementation of 'DihedralSpheres' module

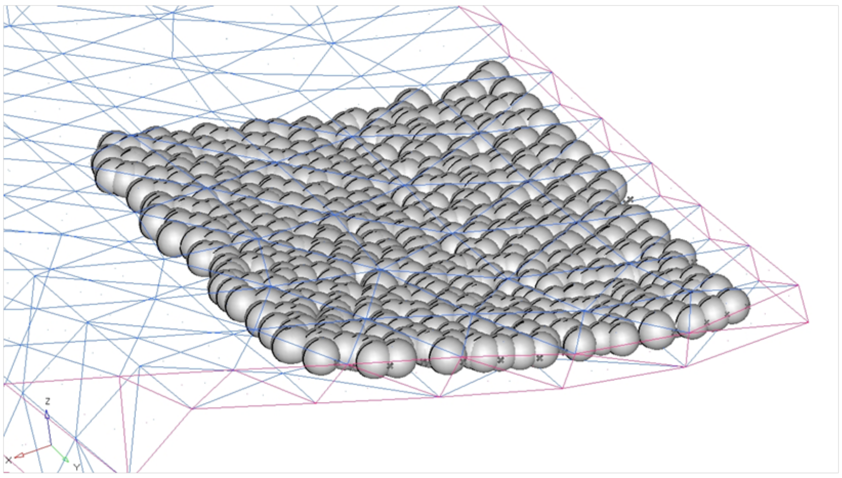
FIGURE 7.8: Medial mesh quality after implementation of 'DihedralSpheres' module

points where the thickness is uniform would be separated by a very small distance between them. Such medial points cannot be used to triangulate since these elements would have a very small area. If the three points/nodes that create a triangle element are almost colinear, this shell element will fail quality checks due to jacobian errors.

The problem these duplicate medial points pose is related to the overall time required to create the medial mesh for a given geometry. The time taken to parse the array of medial points and remove them based on the distance between them is rather large. A medial surface model as shown in Fig. 7.3 which constituted approximately 65,000 medial points before cleaning was reduced to about 43,000 after cleaning. The tolerance can be set to any reasonable number based on user requirement and the model units (dimension units). In this study, for the rear mount case, the number set was 0.25mm which meant that any medial point within a distance of 0.25 with each other would be removed. Time taken for such an activity is around 2-3 hours for a model as complex as the rear mount case. However, it is thought that the creation of duplicate medial points could be avoided as they are being generated by comparing them to an existing medial point matrix.



(a) Reason for duplicate medial points



(b) Sphere packing for a given input mesh

Figure 7.9: Spheres found for a given input mesh density

7.3.8 To mesh the medial point cloud in 3D space: 'MyCrustopen' module

An algorithm to generate a surface or a mesh with a cloud of medial points in 3D space was essential to complete the dimensional reduction process using Mantle-3D. Lee *et al.* [110] and Fabio [111] discuss about shape reconstruction from a point cloud in 3D space. When plans were laid down for a code to generate medial mesh of a given 3D geometry, no commercial codes accessible during this research program could achieve triangulation of a point cloud in 3D space. However, a commercial code 'SYCODE' [112] has a plugin for 'Rhinocerous' [113]. This plugin is not free but can be downloaded and used on a trial basis for 10 days. Since the creation and testing of Mantle-3D would take longer than this, codes to perform such a task and those that could be edited were looked for in the the public domain. One such program developed by Luigi Giaccari called 'MyRobustCrust' [114] could triangulate point clouds which formed closed volumes. Since 'MyRobustCrust' could only mesh closed volume point clouds, it was of little use for this research program. However, another code called 'MyCrustOpen' [115] by the same author could mesh open point clouds. This meant that the medial points from a geometry which has holes, cutouts etc could be used to create the medial mesh.

Once the duplicate points are removed, the unique medial point array is input to the triangulation module of Mantle-3D. This module creates triangle elements between 3 nodes based on Delaunay triangulation such that no triangle overlaps or intersects the other. The mesh connectivity information is output by the triangulation module. Each triangle has 3 nodes and the coordinate of each of these three nodes is known. The thickness at each of these 3 locations is picked from the medial point array and averaged. The average thickness is then stored for later use. The mesh created using the medial points and the new averaged thickness mapped onto each element is then written out as an input deck for use in ANSYS. The medial mesh can be exported for use in NASTRAN or any other FE solver by making appropriate changes to this module of Mantle-3D.

7.4 Mantle-3D : Features

7.4.1 Advantages of using the proposed algorithm

- ⋆ Converts a computationally expensive tetrahedral mesh into a simpler shell mesh.
- ⋆ After development mentioned in section 7.4.2, even geometries that do not form a closed volume can be used to create medial meshes.
- ⋆ The code can be used in batch mode for use in optimization purposes. In the current code, the thickness of the medial shell model can be modified to represent a new design. However, after development when geometric features are modified or moved based on

the level of geometry parametrization used, the medial mesh can be created each time the geometry is changed.

- ⋆ Can be used to create medial meshes with or without branches. Medial meshes with branches would need mass balancing like was done using Mantle-2D. However, medial meshes without branches can be used without mass and CoG balancing since non-structural mass resulting from branches are avoided.
- ⋆ FE model with reduced degrees of freedom.
- ⋆ In conjunction with the Mantle-2D, a whole engine finite element model consisting of shell elements created using both axisymmetric and non-axisymmetric geometries can be created in batch mode.

7.4.2 Development required for the proposed algorithm

- ⋆ The current code needs a meshable solid model to extract the medial mesh. However, with further development, it can be made to work with just the surface mesh of the geometry as an input and no volume information. This improvement when implemented will enable Mantle-3D to be used without geometry cleaning or with those geometries that do not form an enclose volume.
- ⋆ Since Mantle-3D has been scripted using Matlab, the time taken to process large FE models and extract their medial mesh is time consuming. The final code when scripted in a high level language such as C or Fortran would result in larger matrices being handled faster and more efficiently.
- ⋆ The medial mesh created using the current code has one thickness property per element. Though the number of property cards is not a cause of worry at this stage, when assembling larger models it could prove to be detrimental. In order to solve this issue, all elements that have thickness within a tolerance and are in the same plane should be grouped together with just one thickness representing them.
- ⋆ In order to be deployed in an industry setting, the code needs to be more generic with more user defined options with a grahical user interface rather than a text based input.

7.5 Results - Rear mount case

The rear mount case is a load carrying member in an aeroengine. Emphasis is laid on the mount location on this casing and its stiffness in order to ensure the overall movement of the engine for a given set of loads. The rear mount case represented using 3D and 2D finite element models and respective results are discussed in this section. The 3D FE model built using 10 noded tetrahedral elements is as shown in Fig. 7.10 and an equivalent representation of this geometry using 2D shell elements is as shown in Fig. 7.11. A density of 1.31x10^{-5}kg/mm^3, Young's modulus of 199972kg/mm^2 and Poisson's ratio of 0.33 is used to perform this FE analysis.

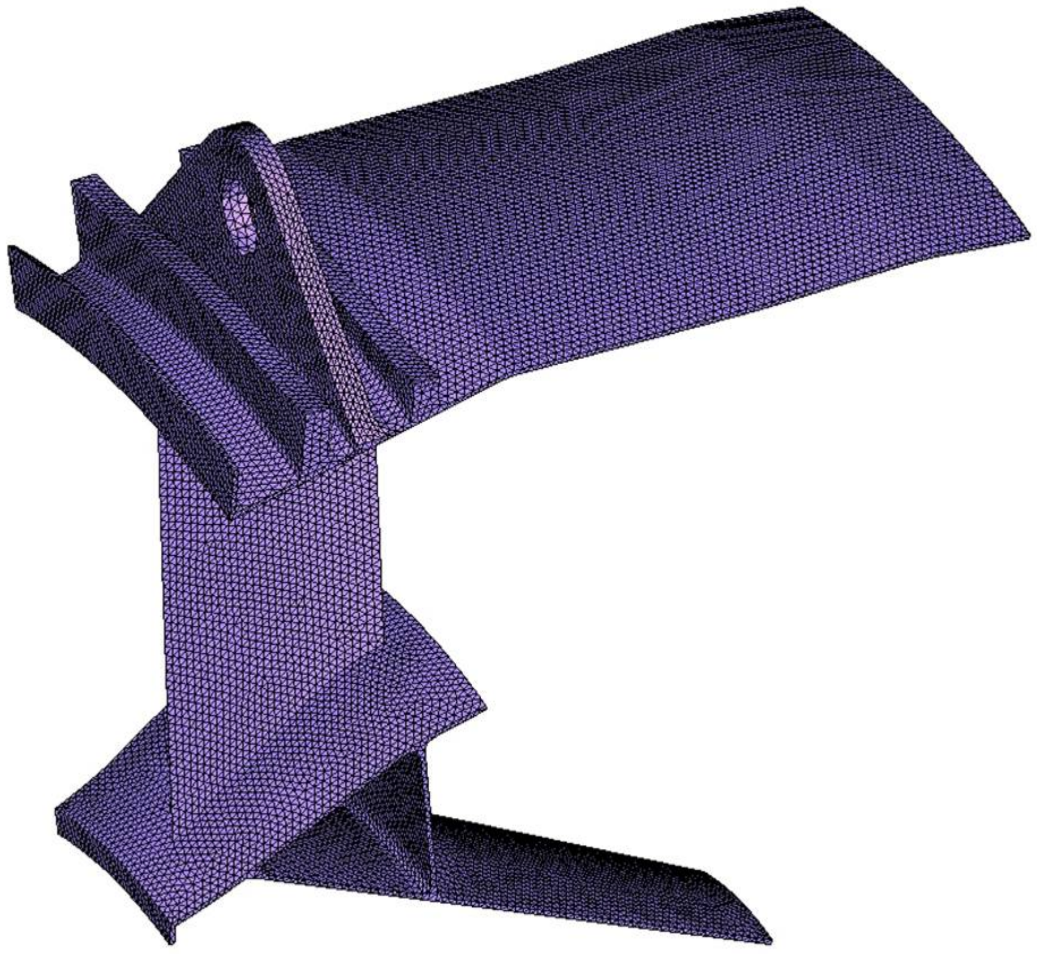


FIGURE 7.10: The solid mesh of the rear mount used to perform static analysis

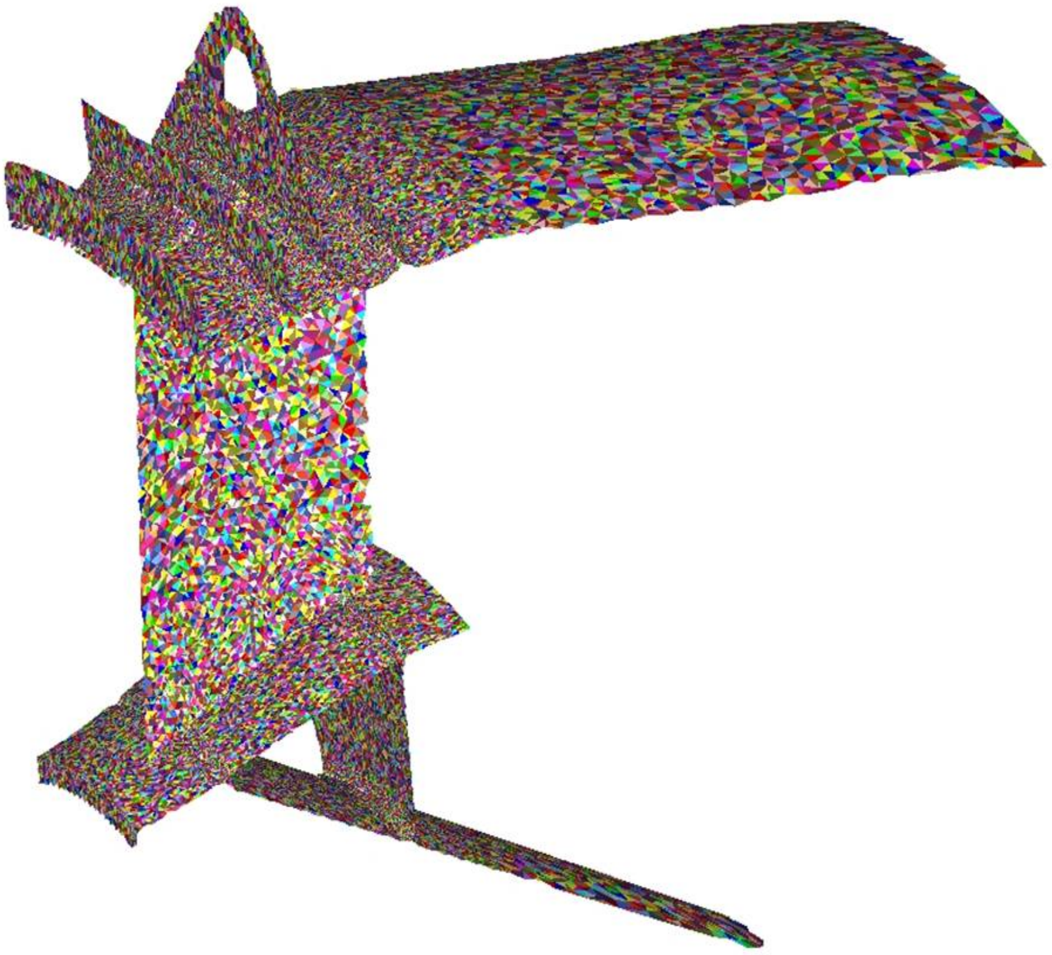


FIGURE 7.11: The thickness of each element as output by Mantle-3D

7.5.1 Static Analysis

The rear mount casing was analyzed for two separate loading conditions, one using a force load of 1000kgf as shown in Fig. 7.12 and the other using 1G loads in the X, Y and Z directions. The results were compared at the max deflection end for the first loading condition while 2 locations where chosen for the gravity loads analysis for both the 2D and 3D models. The test case with only gravity loads is important since this loading condition accounts for the mass of the finite element model and its stiffness in 3 directions. Results pertaining to the force load condition and G-loads are presented in Table. 7.1 and Table. 7.2.

As can be observed from the results table, the runtime of the solid model is about 80% more than the shell model for both the static analysis performed. The difference in mass between the shell model and the tetrahedral model is 3.22% while the center of gravity deviates by less than 2.5%. The results for gravity loads are also comparable for resultant

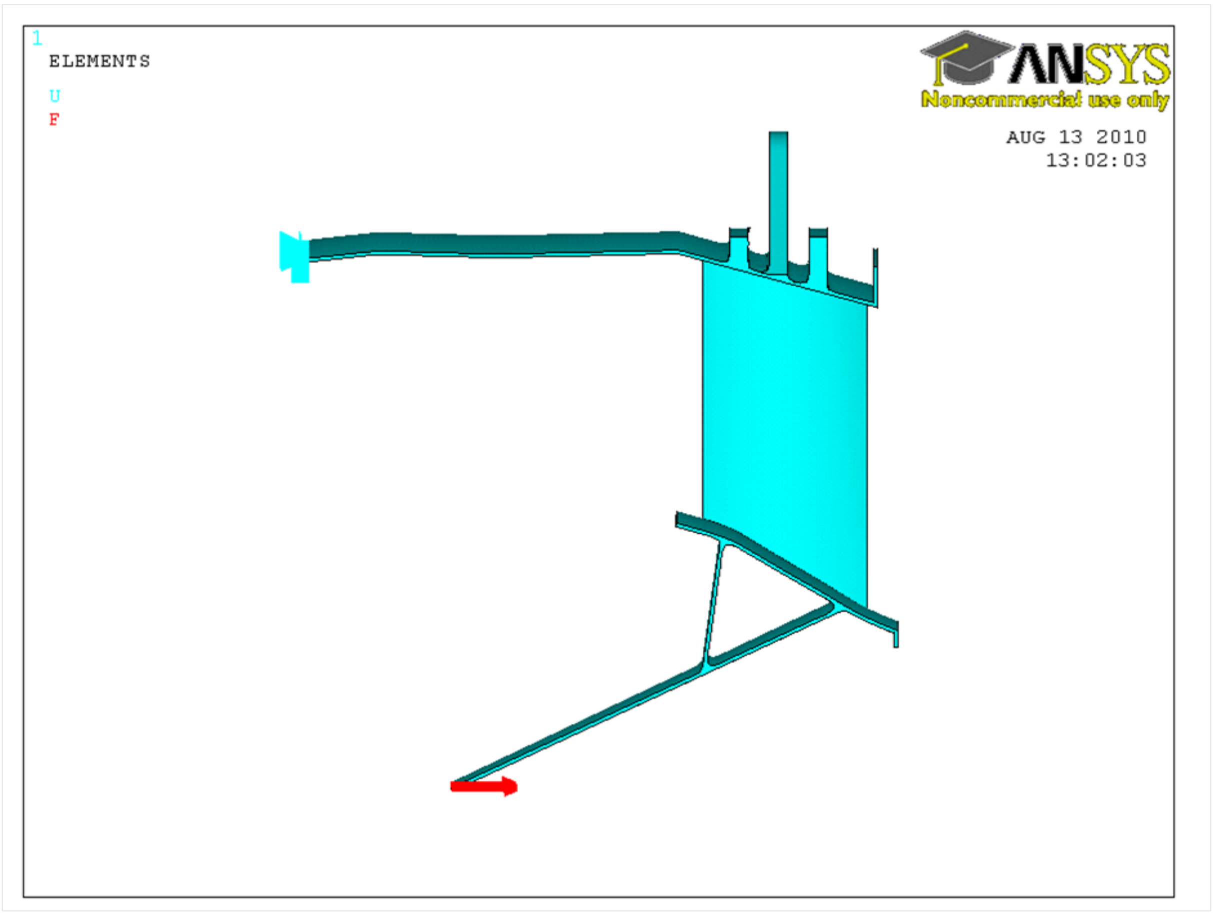


FIGURE 7.12: The unit load and boundary condition for a the first case study

deformation and deflections in X and Y axes for both analysis. The deflection along the z-axis deviates by approximately 15% leading to the conclusion that the z-axis stiffness of the shell model is more than that of the solid model. It is noteworthy to mention at this stage that the shell model used to perform the static analysis is used as output by the Mantle-3D code and no cleaning of any sort has been undertaken. As can be seen from Fig. 7.11, this shell model requires preprocessing before use in any FE analysis. Since Mantle-3D is just a proof of concept, the mesh output from this code cannot be used in its original form. However, in order to emphasise on the fact that such shell models can represent the solid model in stiffness, the midsurface shell model is used in its original form and the results presented here are all pertaining to this uncleaned shell model. With more work, the shell model produced would result in stiffness matching in all 3 axes when compared to the solid model.

As can be seen in Fig. 7.13 and Fig. 7.14, the maximum stress locations in the solid and the shell model are same for the force only loading condition. Since the solid model does

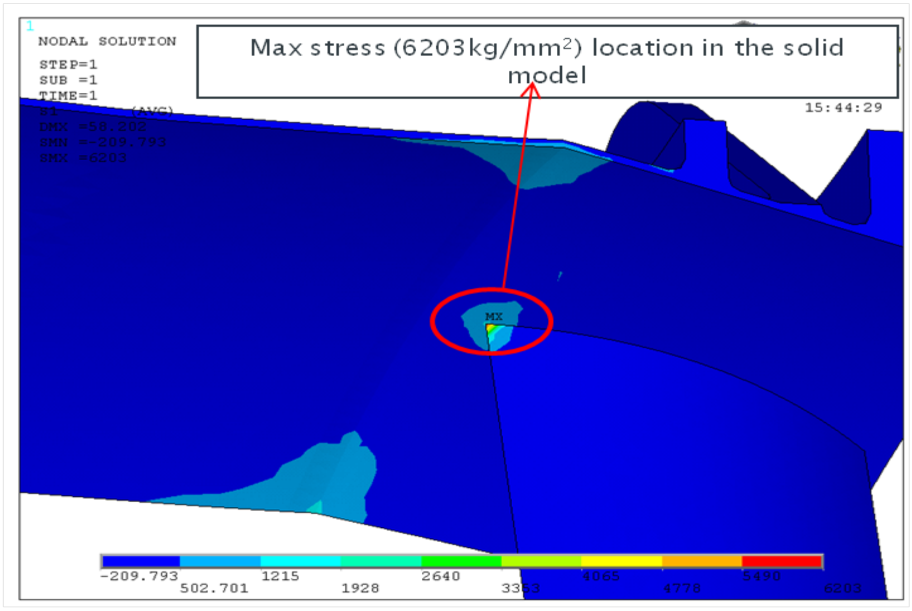


FIGURE 7.13: Max stress location in the solid model for the force load

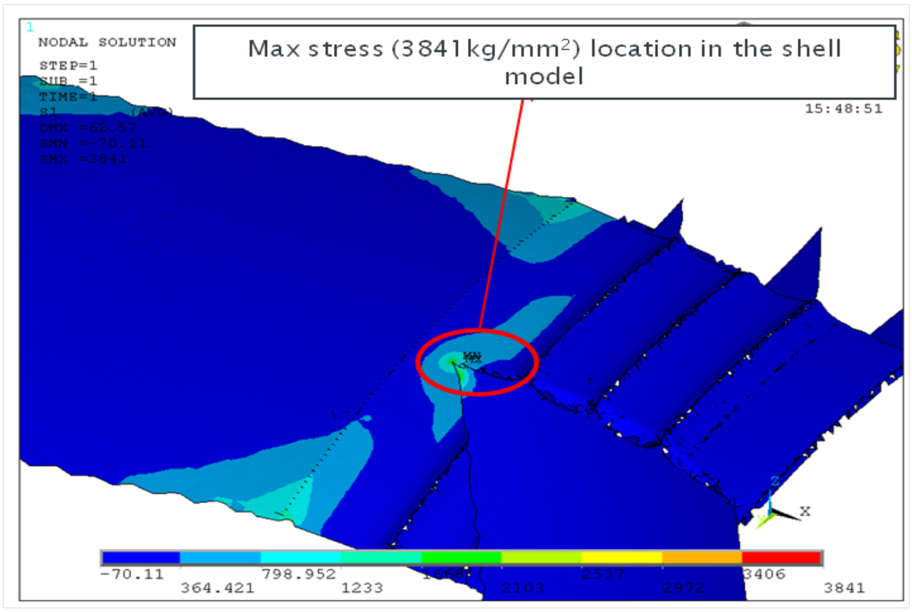


FIGURE 7.14: Max stress location in the shell model for the force load

Table 7.1: Static analysis results for force load only

	Solid	Midsurface Shell	% Diff
Elements	88842	85567	-
Nodes	171569	43506	-
Mass(kg)	19.055	19.67	3.22
CoG(X)	6036.4	6037.4	0.017
CoG(Y)	1.58	1.616	2.33
CoG(Z)	542.9	543.74	0.155
Runtime(sec)	170	23	86.5
Max Deflection	64.0	62.57	2.24

Table 7.2: Static analysis results for gravity loads in X,Y,Z axes only

	Solid	Midsurface Shell	% Diff
Runtime(sec)	165	28	83.03
Resultant Deflection	1.023	1.018	0.489
X Deflection	0.1136	0.1124	0.994
Y Deflection	0.1424	0.1386	2.635
Z Deflection	0.1207	0.1386	15.129

not contain any features such as fillets or chamfers, stress concentration is noticed while stresses in the shell model are spread over a larger area. The max stress in the solid model is 6203kg/mm^2 while is 3841kg/mm^2 in the shell model.

7.5.2 Modal analysis

After retaining the boundary condition used during the static analysis, the solid and the shell model were used to conduct modal analysis. The mode shapes of the solid and shell models are shown in Fig. 7.15 - Fig. 7.20. The first 10 natural frequencies of the solid and the shell model are tabulated in Table. 7.3. It can be noted from the modal analysis that the mode shapes of both the solid and the shell model are comparable with a deviation of less than 3.8% in most cases. This is an indication that the stiffness of the shell model is representative of the solid model with a 3.22% difference in mass.

7.5.3 Design of Experiments

A DoE study consisting of 101 design points was conducted using both the solid and shell models. In order to demonstrate the use of the medial mesh created using the Mantle-3D

Table 7.3: Summary of modal analysis results

	Solid	MATShell	% Diff
Elements	88842	85567	-
Nodes	171569	43506	-
Mass(kg)	19.055	19.67	3.22
CoG(X)	6036.4	6037.4	0.017
Mode 1	0.218	0.217	0.244
Mode 2	0.285	0.283	0.624
Mode 3	0.844	0.848	0.514
Mode 4	2.407	2.365	1.733
Mode 5	2.645	2.590	2.076
Mode 6	4.633	4.637	0.086
Mode 7	5.187	5.133	1.033
Mode 8	6.225	6.283	0.940
Mode 9	6.694	6.950	3.832
Mode 10	8.545	8.567	0.255

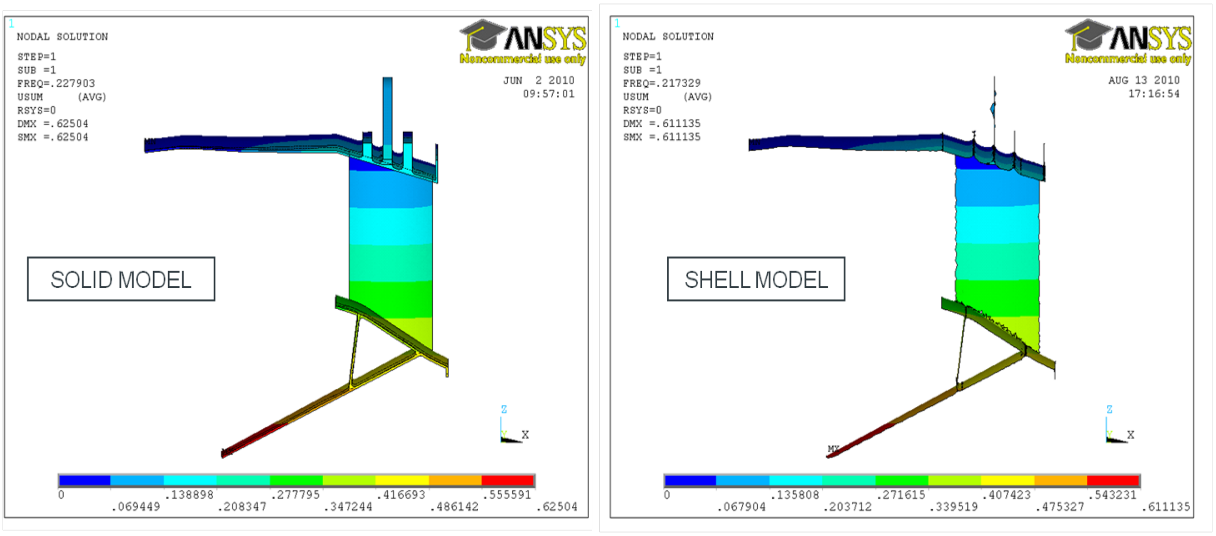

```
NODAL SOLUTION
STEP=1
SUB =1
FREQ=.227903
USUM      (AVG)
RSYS=0
DMX =.62504
SMX =.62504
```
```
NODAL SOLUTION
STEP=1
SUB =1
FREQ=.217329
USUM      (AVG)
RSYS=0
DMX =.611135
SMX =.611135
```

Figure 7.15: Mode shape of the first natural frequency

code for optimization studies, elements were grouped into sections based on regions of the geometry where the thickness is constant. The medial shell model shown in Fig. 7.11 consists of 85567 elements and real constants (thicknesses), i.e each element has its own thickness and are not grouped. The clear definition of variables is an essential requirement while setting up an optimization study. The elements in the shell model shown in Fig. 7.11 were grouped into

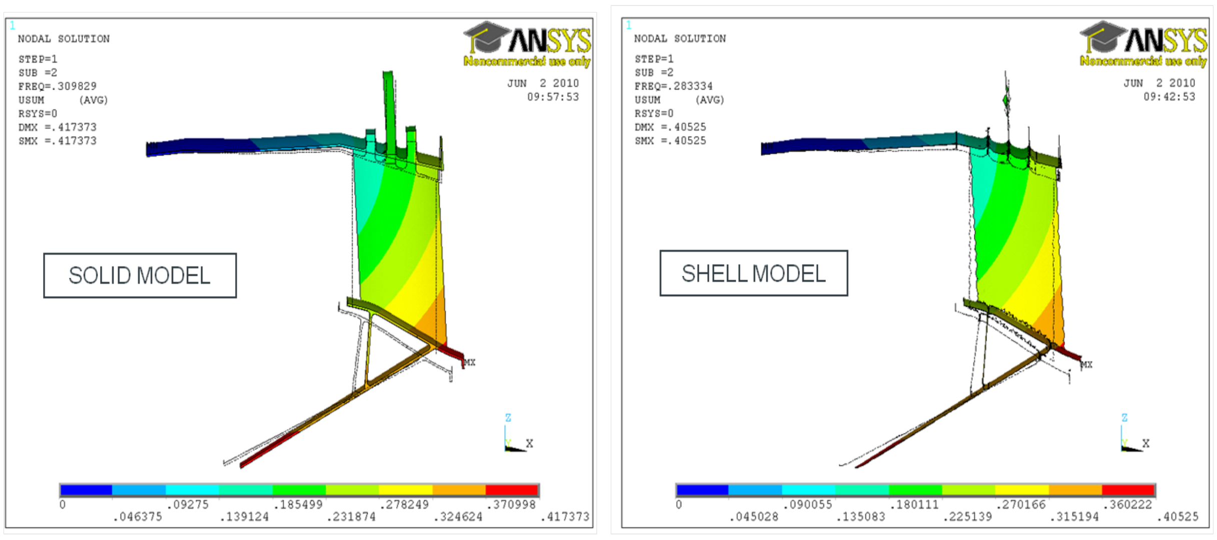


FIGURE 7.16: Mode shape of the second natural frequency

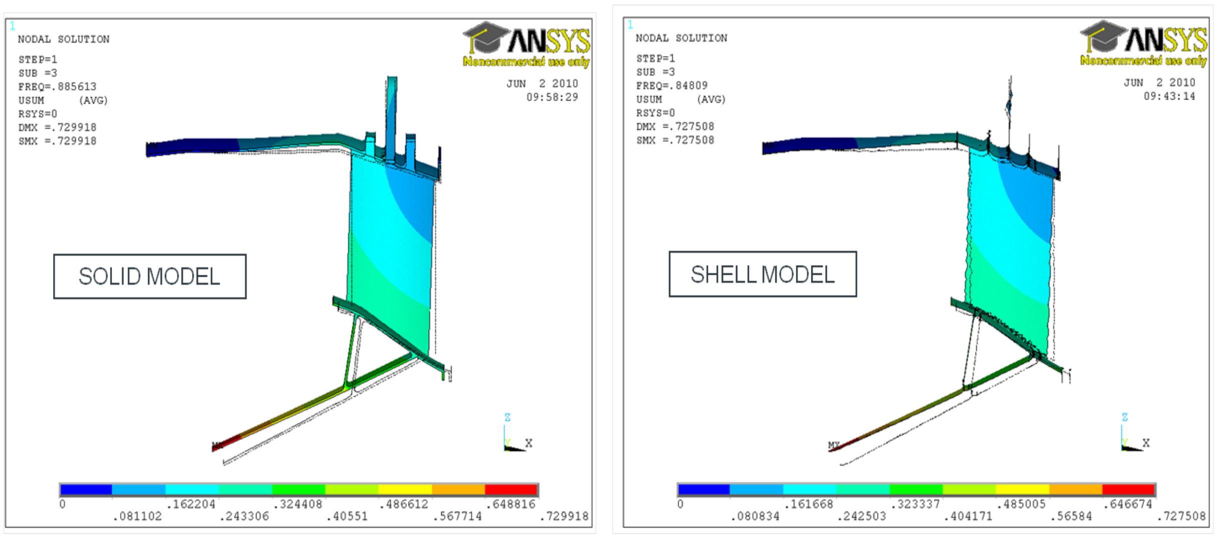


FIGURE 7.17: Mode shape of the third natural frequency

sections as shown in Fig. 7.21. Each of these 7 groups are now represented by 7 thickness sets and considered as variables which can then be modified to reflect change in geometry. Morphing the solid mesh is the approach adopted in this DoE study to ensure that the solid mesh matches the thickness changes made to the shell mesh. When the shell thickness is changed for any of the 7 variables, the nodes that form the top and bottom surface of the 3D model at that region are morphed to reflect the change in the shell model.

The Mantle-3D code needs development before it can be used in batch mode for use in optimization workflows. The time taken by the existing code to generate the medial mesh

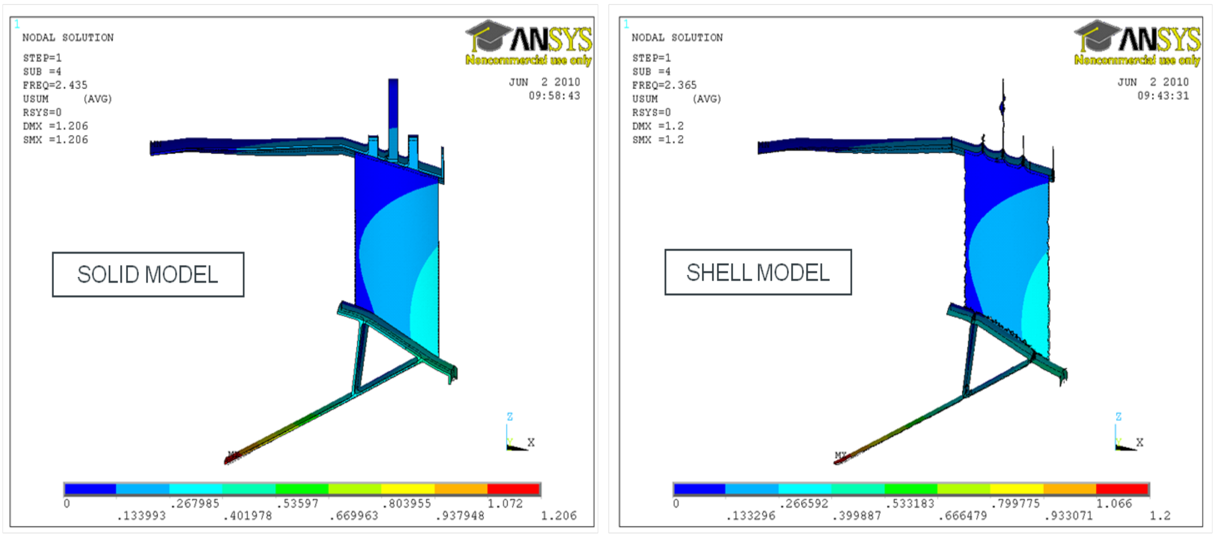


FIGURE 7.18: Mode shape of the fourth natural frequency

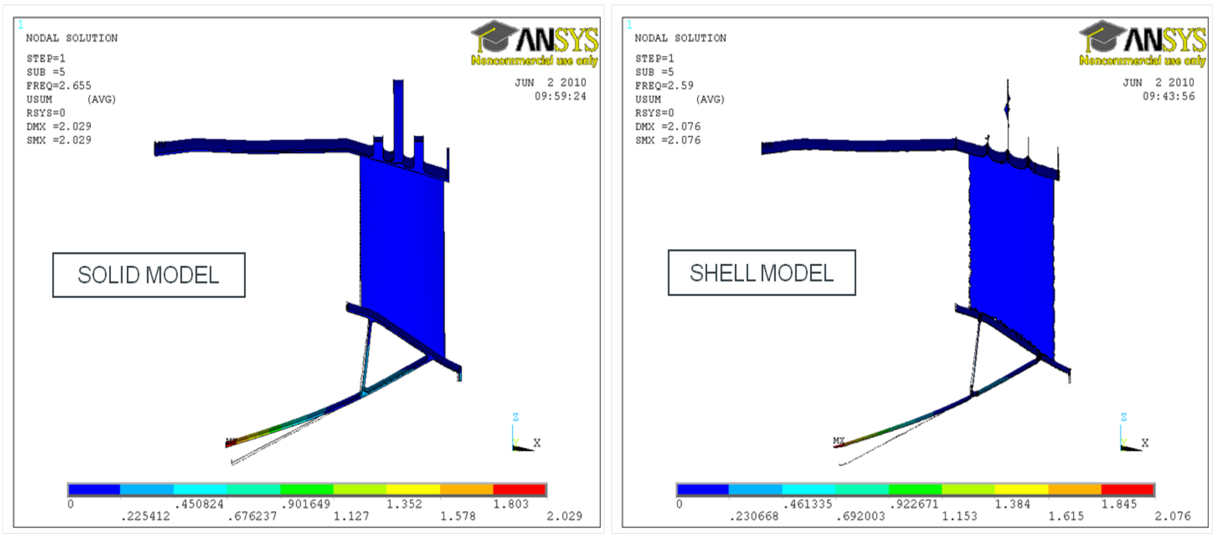


FIGURE 7.19: Mode shape of the fifth natural frequency

of the rear mount case sector model is around 6-9 hours. But once generated, the shell model requires 80% lesser CPU time when compared to a 3D FE model. If the medial mesh has to be created everytime the geometry changed, the time taken for DoE study would run into >900hrs for 100 design points. This is the primary reason for adopting the morphing technique for this DoE study when compared to the geometry parametrization used in Chapter 6. The time taken to create the medial mesh could be reduced by scripting in C or C++ programming language. Until this is done, the medial mesh needs to be created once and then thickness changed to represent different geometry as set up by Voutchkov *et*

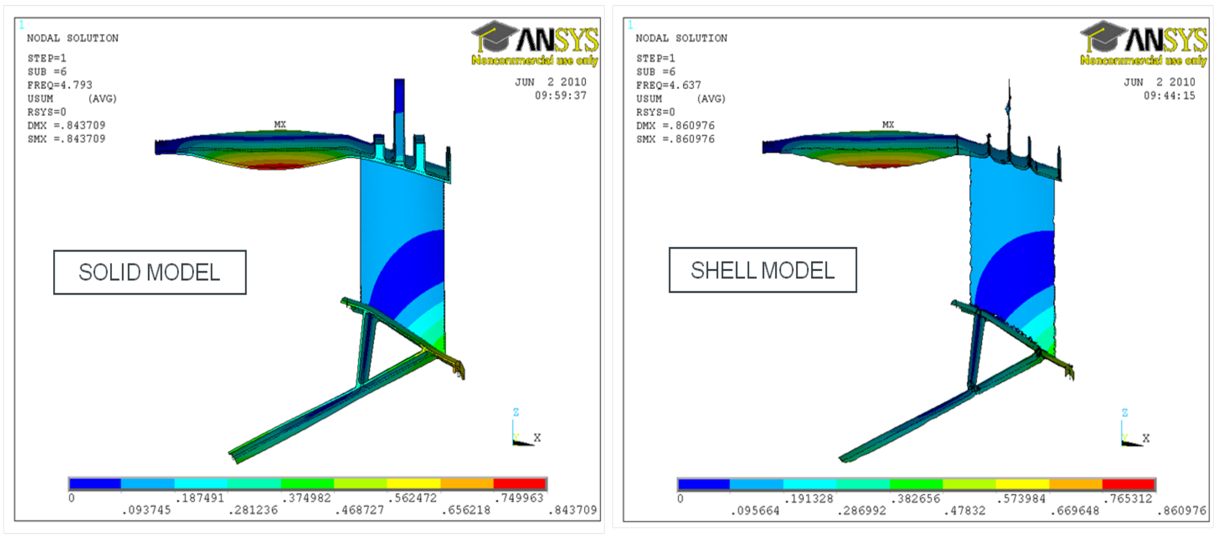


FIGURE 7.20: Mode shape of the sixth natural frequency

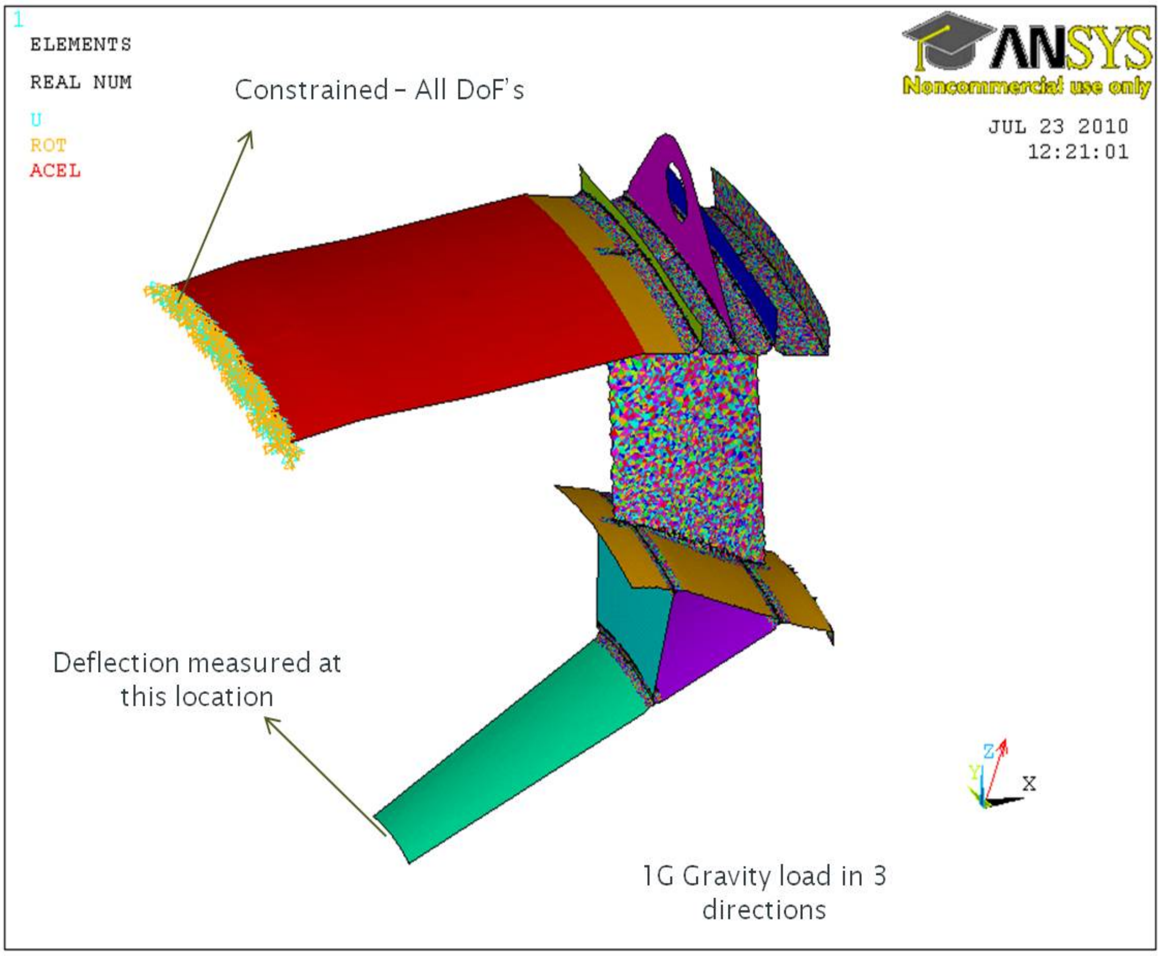


FIGURE 7.21: Elements grouped into 7 major sections for use as variables during the DoE study

al. in [9]. Only one shell mesh has been used for all 101 design points during this DoE study. However, the 2D mesh used in [9] had to be created manually. The primary objective of this case study is to prove the concept of using auto generated shell meshes for DoE or optimization studies and not entirely about creating shell meshes in a generic and robust manner.

The distribution shown in Fig. 7.22 of the % difference in deflection shows that the shell model is representative of the solid model in stiffness for varied designs (100 changes to the variables with 1 design being the datum/current design). A similar representation of mass is as shown in Fig. 7.23 for the solid and shell models. The results are slighlty skewed when compared to similar plots in Chapter 6. The reason for this is the way in which the solid and shell models are modified to represent new designs. The various designs were studied by creating them from scratch in Chapter 6, while the mesh is morphed for the solid FE model and thickness changed for the shell model for all designs in this chapter. However, the run time for each shell model was approx 28 secs when compared to 165 secs for the solid model. The design study was conducted using a dual processor 2GB RAM local machine. This shows the reduction in time that can be achieved by dimensional reduction of compex 3D non-axisymmetric models into simpler shell models when performing design optimization. As can be seen, the typical mass error and deflection error for the solid and shell FE models is <3.5% and <9% respectively. The % difference in mass and deflection for the datum design is 3.316% and 6.11% respectively.

When the results of the solid model for the 101 design points are plot as shown in Fig. 7.24, it can be observed that the heaviest design of the solid model (top left of the plot) weighs around 24.565kg with a casing deflection of 0.338mm. The baseline design on the other hand (large triangle in the plot) weighs 19.06kg and deflects around 0.97mm. It is evident from here that as the weight and stiffness increases, deflection drops. A similar plot for the medial surface based shell model in Fig. 7.25 shows that the heaviest shell model design (top left of the plot) weighs around 25.003kg with a casing deflection of 0.341mm while the baseline design (large dot in the plot) weighs 19.692kg with a deflection of 1.029mm. It is also noteworthy to mention that the heaviest solid and shell model's have been observed for the same set of design variables during the DoE study. The difference in mass and deflection for heaviest design of the solid and shell model is about 1.78% and 0.88% respectively.

Similarly the lightest solid model weighs around 18.435kg with a casing deflection of 1.248mm. When compared to the datum solid model, it can be observed that the lightest

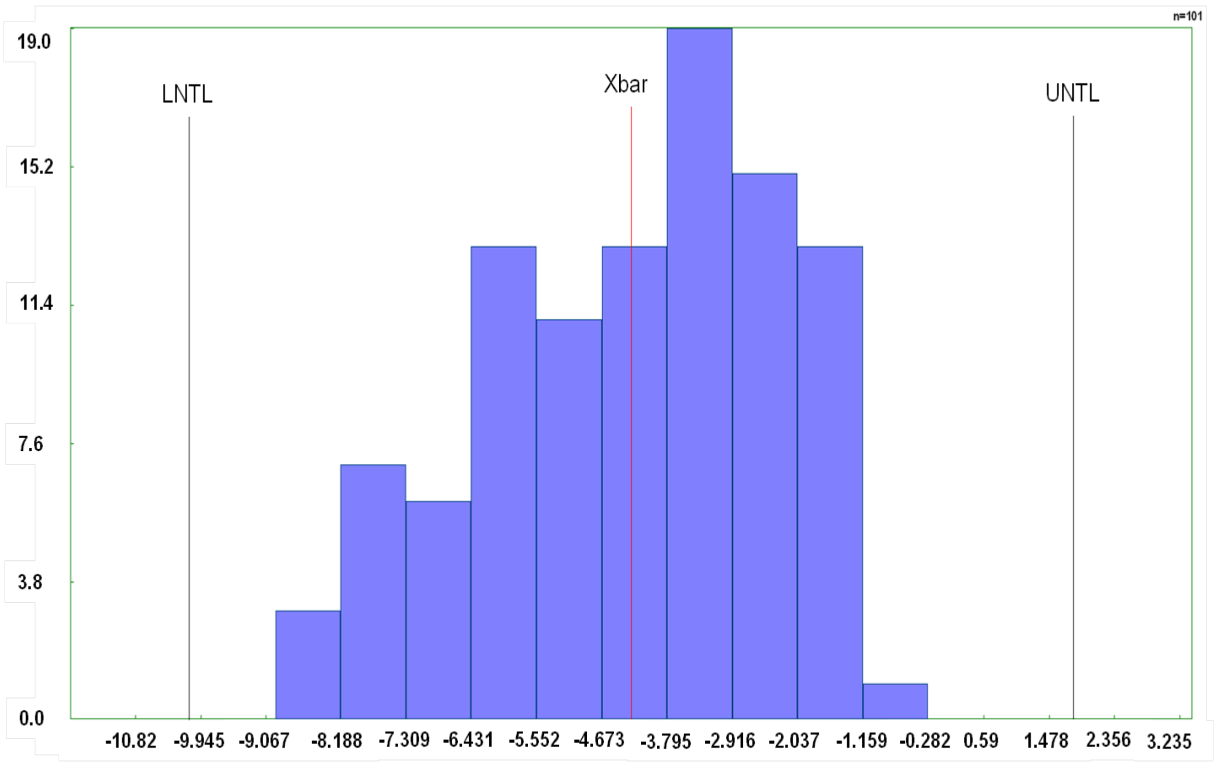


FIGURE 7.22: Distribution of %diff in deflection of the shell and the solid model

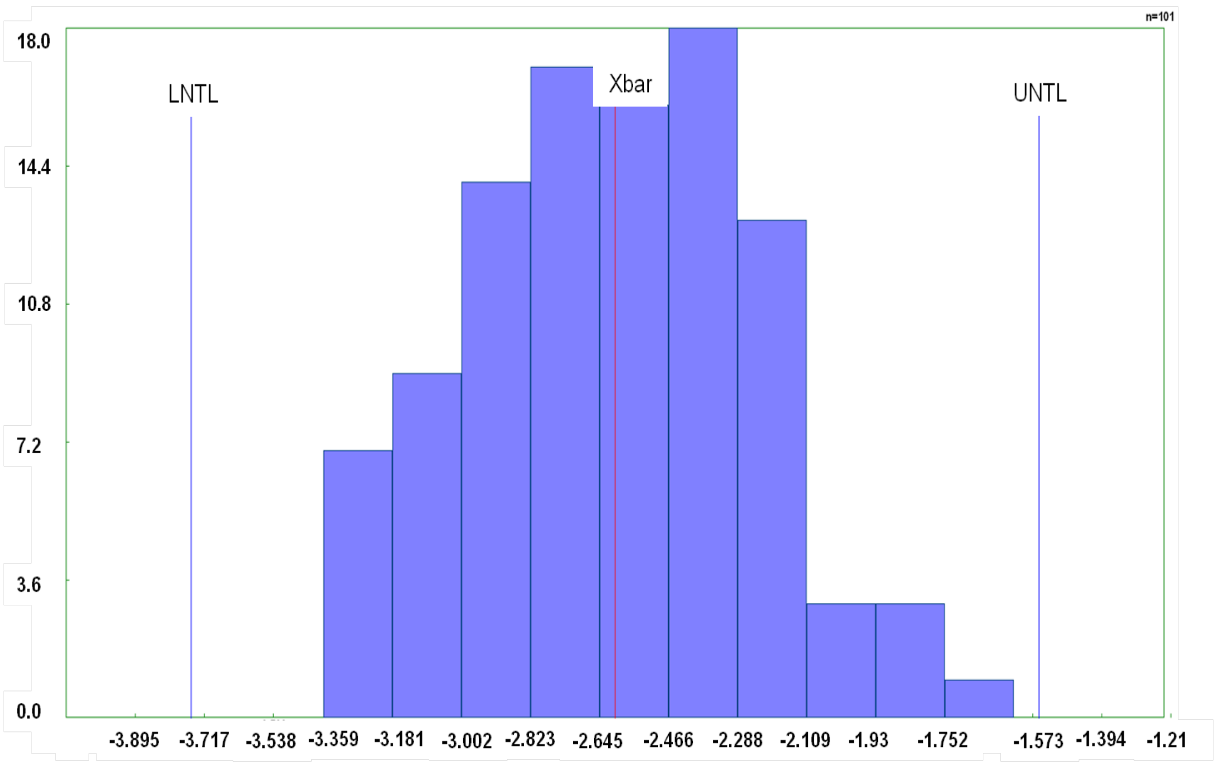


FIGURE 7.23: Distribution of %diff in mass of the shell and the solid model

solid model results in a larger deflection than the datum design due to loss of material and hence loss of stiffness. However, the lightest shell model weighs around 19.063kg with a casing deflection of 1.338mm. Again, when compared to the baseline shell model, the increased deflection is due to loss of material and stiffness. As mentioned earlier, the lightest solid and shell model have been produced for the same set of design variables during the DoE study. The difference in mass and deflection for the lightest design of the solid and shell model is about 7.2% and 3.26% respectively.

When the medial surface shell model's Pareto front is superimposed onto the solid model results as shown in Fig. 7.26, it can be observed that the best designs from the shell model follow the same pattern as the solid model designs. The circular dots which represent the shell model Pareto front have shifted slighlty upwards and to the right due to the fact that the medial surface based shell mesh is less stiffer but heavier than the solid model in its representation for all designs. This deviation in mass and deflection can be attributed to the dimensional reduction process and the complexity of the model being studied.

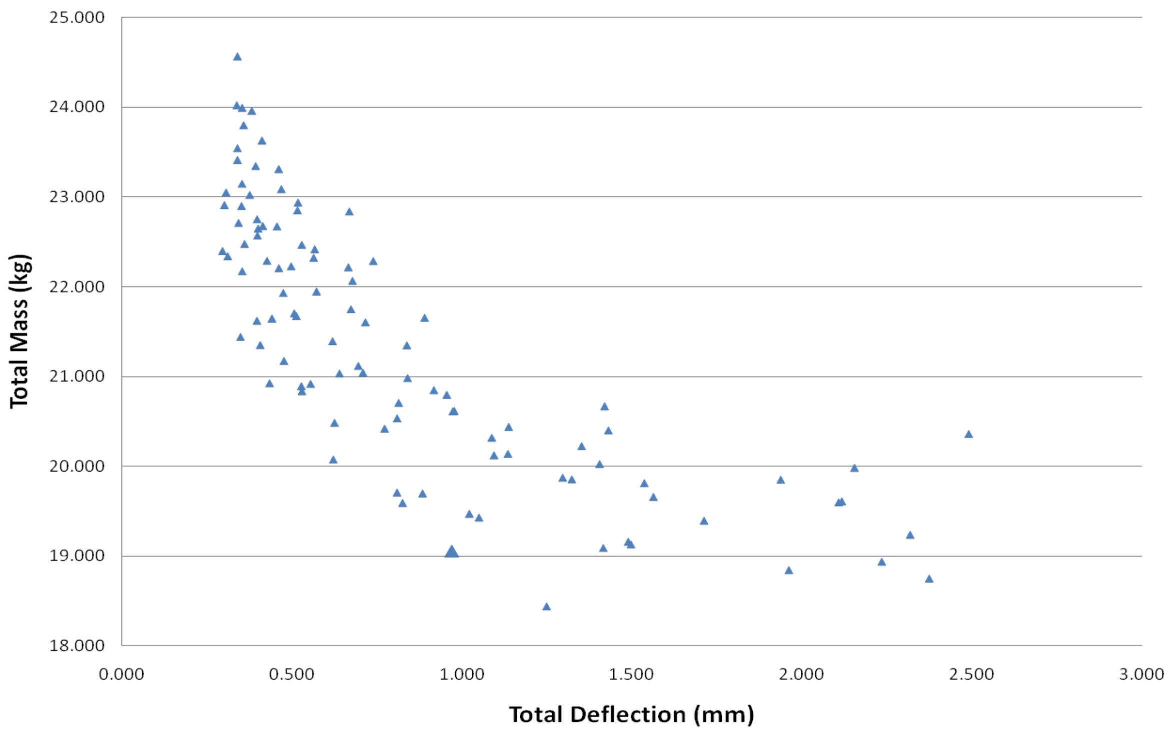


FIGURE 7.24: Solid model deflection Vs mass plot for the DoE study

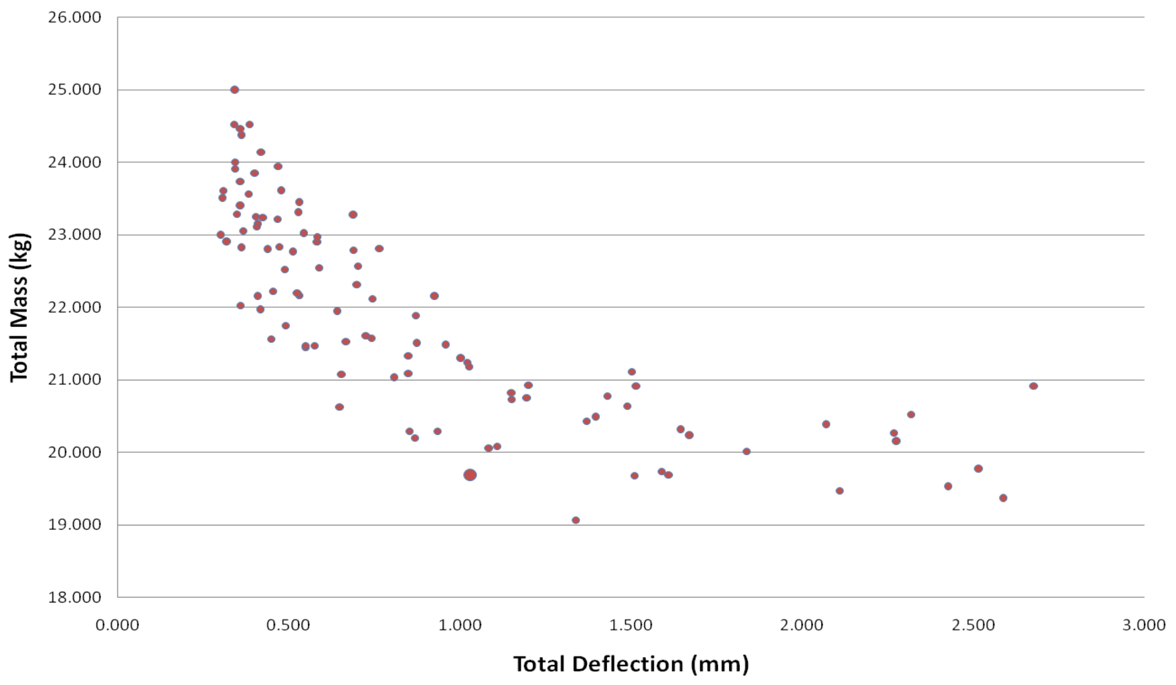


FIGURE 7.25: Medial surface shell model deflection Vs mass plot for the DoE study

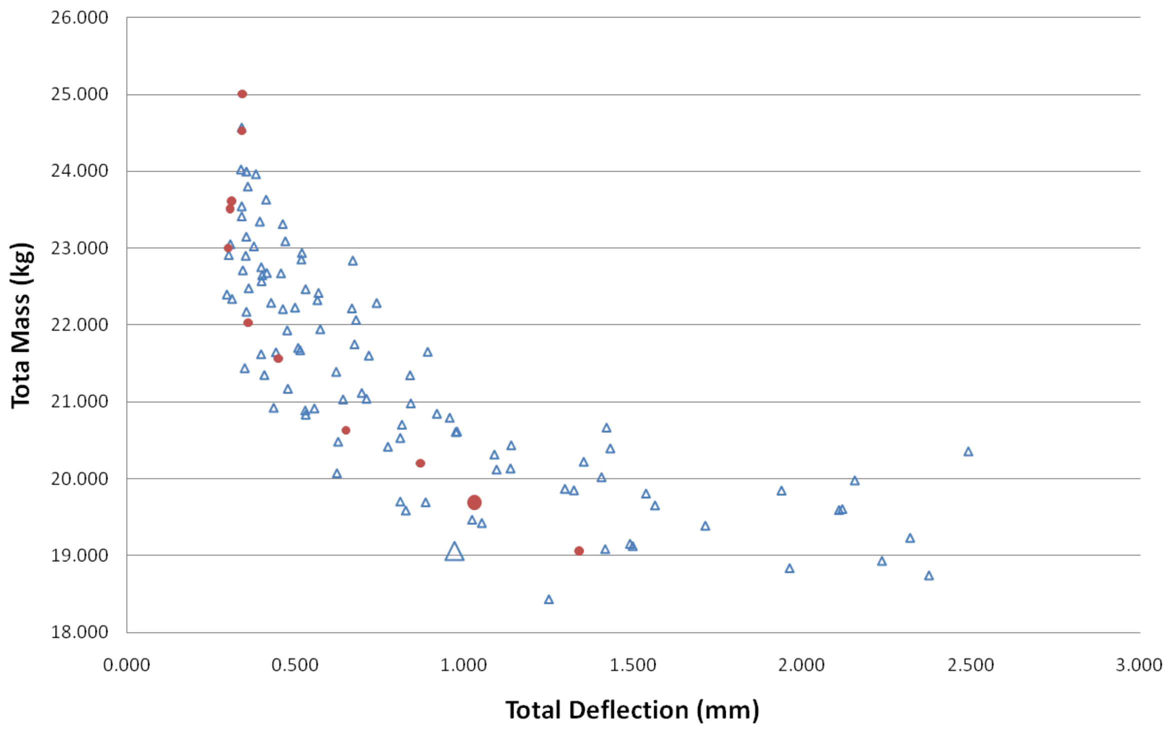


FIGURE 7.26: The medial surface shell model's Pareto front superimposed onto solid model's results

7.6 Conclusion

Based on the results obtained from the study conducted in Chapter 6, a similar approach of using medial axis transform applied to 3 dimensions had to be demonstrated. Without a process to simplify complex 3D models to simpler FE models, a design optimization study constituting an assembly would not be achievable. Since no available code either in the academia or the industry could be used to perform such a dimensional reduction process pertaining to finite element analysis and design optimization, a bespoke code 'Mantle-3D' has been created. The code has been tested on a geometry as complex as the rear mount case of a gas turbine engine and the resulting shell mesh compared with the respective solid model. The medial surface based shell models created for non-axisymmetric 3D casings have lesser degrees of freedom than an equivalent solid model. Through the dimensional reduction process, an overall gain of 80% or more can be made during static analysis without substantial decrease either in mass or stiffness terms. It can be observed from the results that the dimensionally reduced shell model has a mass and stiffness difference that is <3.5% and <9% respectively. Mass correction had to be made for the shell models created using Mantle-2D. But since no branches have been generated during the creation of the medial surface mesh using Mantle-3D, the non-structural mass added onto the shell model is negligible. This would mean that shell models created using a more refined Mantle-3D code would result in a better representation of the solid model. The current shell model which consists of 85567 elements can be reduced by half with further work and run times of this leaner shell model would reduce from the current 28-23 secs to about 10-15 secs. As seen in Chapter 6, even if the geometry is non-axisymmetric, reducing the complexity by extracting the midsurface and using it for FE analysis would result in improving designs in a fashion consistent with the searches carried out using 3D FE models. It can thus be concluded that medial surface based shell mesh models could satisfactorily replace solid based FE models during design optimization studies.

Chapter 8

Concluding remarks and future work

8.1 Concluding remarks

In this report, we have reviewed a few of the many existing methods available for meshing, preprocessing, solving and optimizing an engineering component. In Chapter 5, we saw how a simple stepped cantilever beam provided enough challenges while creating an optimization design cycle. Based on the optimization workflow developed for the stepped beam problem, the design cycle for an assembly constituting the load bearing outer casings of a gas turbine engine can be developed. But when the components that form the outer casings of the engine are assembled, a large FE model could result. Such large models could lead to long FE analysis run times and maximum use of resources. In order to create a simple yet effective representation of the complex geometric model, a process has been developed to represent the 3D model. In Chapter 2, we looked at possible methods to create a low cost FE model which is a good representation of the 3D geometric model and has lower optimization cycle times. Of the many techniques discussed in Chapter 2, the medial axis transform based midsurface extraction method has been chosen as the possible solution.

The fancase midsurface shell model of a gas turbine engine was chosen for comparison with the solid FE in Chapter 3. Using midsurfaces proves to be a very effective way to dimensionally reduce a 3D model for optimization purposes. However, the hindrance in using midsurfaces for creating 2D FE models is while extracting them from their native 2D/3D geometries. Commercial tools like Siemens NX or HyperMesh fail to extract all the midsurfaces of complex geometries. No “off the shelf” CAD or CAE softwares today boast of having capabilities to dimensionally reduce 3D models to 2D or mixed dimensional FE models in a completely automated fashion. In Chapter 4, shell models created using face pairing techniques and the medial axis transform have been studied in detail. These two methods have their pros and cons.

The basic face paired model has disconnected midsurfaces which need to be connected such that the radial and axial stiffnesses of the shell model are still comparable to the solid model. This method of shelling retains the general shape of the component and by using zero density for connecting elements, the mass and stiffness can be matched with the solid model. However, the hindrance to use this technique is when it comes to extracting the midlines in batch mode. There is no code in open source or commercial that can be used off the shelf for automatically creating midlines that are completely connected.

A MAT based model adopts a generic process and can be automated for use in batch mode. The hindrance to using this method is the correct representation of the component’s stiffness. After density correction, the MAT model has the same mass as the solid model with a small variation in the Center of Gravity (CoG).This shift in CoG can be corrected with mass balancing. Two codes, Mantle-2D and Mantle-3D developed using Matlab, have demonstrated how the midsurface of 3D axisymmetric and non-axisymmetric geometries can be represented as shell based FE models for use in finite element analysis and design optimization. However, it is estimated that 18 more months of work would be required to generalize the two codes for use in real world problems.

The most important benefit of adopting the proposed process into a real design environment is the insight it provides into the structural integrity of the whole engine model for a given gain in SFC and/or weight. However, the advantages of adopting the MANTLE based design process is as follows:

- ⋆ Improved SFC through reduced mechanical tip clearances and improved operability.
- ⋆ Lead time reduction virtually no manual intervention.
- ⋆ Capturing existing knowledge - as rules and variables within NX and iSIGHT.
- ⋆ Product optimization - exploring large design space at the expense of CPU time.
- ⋆ Dependency tracking - how the variables are related to each other.
- ⋆ Demand driven can be used to optimize an assembly of casings or just one part.

⋆ Can be coded using any programming language.

⋆ Can be applied to simulate complex 3D geometries (axisymmetric and non-axisymmetric) as simpler 2D FE models.

⋆ Can be used to study new geometries and not just the effect of change in thickness as was done in VIVACE project.

⋆ The knowledge base available in geometric design, manufacturability, material properties, cost model etc can be incorporated as part of the basic design.

8.2 Future work

8.2.1 Generalization of Mantle-2D

Mantle-2D as used in Chapter 6 has its limitations. In order to use this code in a more generic industrial setting, the following changes need to be made:

⋆ Include all parameter types constituting the IGES format of geometry representation.

⋆ Adopt a new approach which works based on the principles of Mantle-3D.

⋆ 2D mesh on the cross-section as input will allow for holes and cut-outs to be included.

⋆ A bespoke code to calculate the volume and mass of a given cross-section geometry reduction in lead time for the HPT case (10sec*2000iter = 333.33min or 5.5hours for 2000 design iterations).

⋆ An option to create medial axis with or without branches as demonstrated in MANTLE-3D.

⋆ Include a plug-in to correct the CoG of the MAT model whenever necessary.

⋆ Identify discontinuities in the input geometry and proceed accordingly.

⋆ Process many geometries in parallel and assemble them to run tip clearance studies.

⋆ Create plug-ins to apply loads and boundary conditions for use in NASTRAN / LS-DYNA.

⋆ Information required to simulate flanged connections (welded or bolted) to be stored.

⋆ Code using C or C++ if necessary.

8.2.2 Generalization of Mantle-3D

- ⋆ Good medial mesh is currently dependent on the input mesh density. A generic code, should be able to produce good medial mesh irrespective of the input mesh quality. In order to achieve this, at all convex points the sphere has to be driven along the resultant vector until the maximum diameter is reached and also retain all the incremental spheres along its path.
- ⋆ The current process needs a meshable geometry as an input. This could be a problem when handling large complex models. To handle such cases, the code should be modified to work with just the surface mesh as an input. This would result in creating medial meshes of even those geometries that do not form a closed volume.
- ⋆ Medial meshes without convex branches are ideal for FE analysis. It has been demonstrated here, but a lot of work still needs to be done to make it robust. As mentioned in the 2D code features, the elimination of branches will remove the problem of mass and CoG balancing. But the concave edge branches representing the stiffness of a region should not be dealt with.
- ⋆ The user should be given the option of creating medial meshes with or without branches by toggling a switch within the code.
- ⋆ The current code can pick a coarse mesh and refine it to any degree. This is done by adding sampling points inside the existing element. If there were two elements along the fillet, the existing refinement will not convert this into 4 elements along the fillet, but will add lots of points along the area and edges of existing elements.
- ⋆ To be scripted in C or C++ in order to speed up the code.

8.2.3 Multi-fidelity optimization

Multi-fidelity optimization was initially defined as being within the scope of this research program. However, based on the results obtained from the MAT shell models in Chapter 6 and 7, the need for multi-fidelity optimization was not observed. Parameters from the low fidelity model are updated based on how it compares with the high fidelity model. The low fidelity model (MAT based shell model) has its properties i.e, mass, CoG and stiffness consistent with the high fidelity model. But when many such cheaper to run FE models are put together to form an assembly during tip clearance studies, the difference between the shell and the solid i.e, low fidelity and the high fidelity models would be more prominent. Under such circumstances, multi-fidelity optimization as mentioned by Voutchkov *et al.* in [83] needs to be adopted. A study has been conducted by them on the use of high and low fidelity models for use in multi-objective optimization. They discuss how the number of function evaluations can be reduced with the use of response surface (surrogate) modeling and its use in real life engineering problems.

Appendix A

Mesh sensitivity study of a thin casing

This study was conducted in order to make the right element type and the mesh density required when handling thin gas turbine engine casing like components. A downward gravity load of '1g' was used and the maximum deflection corresponds to this condition for all models in the table. It can be noticed from the table below that a coarse 20 noded hexahedral mesh compares well with a very fine 8 noded hexahedral. A fine and coarse 10 noded tetrahedral mesh was also built, but the results show that they do not compare well with hexahedral meshes. Based on this study, coarse higher order or very fine lower order hexahedral meshes are the right choice when comparison studies between solid and shell meshes are to be conducted.

Table A.1: Result comparison for various 3D meshes

	Hex8 - Fine	Hex8 - Coarse	Hex20 - Coarse	Tet10 - Coarse	Tet10 - Fine
Element Type	Solid 45	Solid 45	Solid 95	Solid 92	Solid 92
Elements	722020	74000	73000	159820	564131
Nodes	876460	115800	417200	318416	1104286
Time (min)	54	5	14.5	6.5	51
Processors	4	2	2	2	4
Mass	64.452	64.45	64.325	74.494	67.3665
Max Defl	0.604	0.56995	0.602	0.4983	0.566
% Error		5.64	0.33	17.5	6.29

Appendix B

Results of the other methods described in Chapter 4

Only results pertaining to Xs1D1 and Xs1MAT were presented in Chapter 4. The behavior of Xs1D4 (Coupling) and Xs1D5 (Constant spring stiffness) models for unit axial and radial loads are comparable. Results pertaining to Xs1D2 model can be used to compare with the Xs1D1 model.

TABLE B.1: % Difference of Radial and Axial displacements of the Xs1D2 Shell model with the Solid model

Xs1D2 Shell Model - Radial Displacements (mm)								
Loc	0.0°	90.0°	180°	270°	%Diff 0°	%Diff 90°	%Diff 180°	%Diff 270°
1	0.00514344	0.00120128	0.00061158	0.00120128	-31.14	-6.23	0.51	-37.11
2	0.00278921	0.00114752	0.00063486	0.00114752	-13.64	-4.99	-3.16	-4.88
3	0.00196869	0.00107754	0.00062414	0.00107754	6.18	-2.82	-4.25	-2.69
4	0.0011422	0.00082108	0.00052667	0.00082108	6.72	-4.08	-6.11	-3.81
5	0.00092599	0.00067265	0.00047764	0.00067265	-1.96	-9.91	-3.35	2.29
6	0.0004448	0.0003818	0.00033381	0.0003818	7.22	3.81	4.42	3.95
7	0.00022238	0.0002422	0.00022767	0.0002422	-0.29	6.68	4.76	6.79
8	0.00013054	0.00018184	0.00017782	0.00018184	-5.44	8.77	7.44	8.83
9	0.00007252	0.00013201	0.00014221	0.00013201	-27.45	8.66	8.49	8.67
10	0.00002581	0.00003093	0.00007041	0.00003093	28.62	8.90	21.36	8.71

Xs1D2 Shell Model - Axial Displacements (mm)								
Loc	0.0°	90.0°	180°	270°	%Diff 0°	%Diff 90°	%Diff 180°	%Diff 270°
1	0.00019362	0.00019362	0.00019362	0.00019362	-11.47	-11.47	-11.47	-52.91
2	0.00017562	0.00017562	0.00017562	0.00017562	-14.03	-14.03	-14.03	-14.03
3	0.00015557	0.00015557	0.00015557	0.00015557	-1.94	-1.94	-1.94	-1.94
4	0.00012347	0.00012347	0.00012347	0.00012347	-1.04	-3.05	-3.05	-3.05
5	0.00011028	0.00011028	0.00011028	0.00011028	0.48	-6.05	0.48	0.48
6	0.00008047	0.00008047	0.00008047	0.00008047	2.38	2.38	2.38	2.38
7	0.00006071	0.00006071	0.00006071	0.00006071	4.51	4.51	4.51	4.51
8	0.00005541	0.00005541	0.00005543	0.00005541	5.83	5.83	5.80	5.83
9	0.00004909	0.00004909	0.00004909	0.00004909	6.19	6.19	6.19	6.19
10	0.00002059	0.00002059	0.00002062	0.00002059	-3.68	-3.68	-3.83	-3.68

TABLE B.2: % Difference of Radial and Axial displacements of the Xs1D4 (Coupling) Shell model with the Solid model

Xs1D4 Shell Model - Radial Displacements (mm)								
Loc	0.0°	90.0°	180°	270°	%Diff 0°	%Diff 90°	%Diff 180°	%Diff 270°
1	0.00312658	0.00107786	0.00056666	0.00107786	20.28	4.69	7.81	-23.02
2	0.00229221	0.00102783	0.00055858	0.00102783	6.61	5.96	9.24	6.06
3	0.00187359	0.00096135	0.00053816	0.00096135	10.72	8.27	10.11	8.38
4	0.00099199	0.00070837	0.00044262	0.00070837	18.99	10.20	10.82	10.44
5	0.00076262	0.00061469	0.0003976	0.00061469	16.03	-0.44	13.97	10.71
6	0.00038314	0.00034674	0.00029769	0.00034674	20.08	12.65	14.76	12.77
7	0.00021864	0.00023225	0.00022028	0.00023225	1.39	10.52	7.85	10.62
8	0.00014608	0.00017628	0.00018224	0.00017628	-17.99	11.56	5.14	11.62
9	0.00009418	0.00012946	0.00015088	0.00012946	-65.52	10.42	2.91	10.43
10	0.0000344	0.00003152	0.0000735	0.00003152	4.87	7.16	17.91	6.97

Xs1D4 Shell Model - Axial Displacements (mm)								
Loc	0.0°	90.0°	180°	270°	%Diff 0°	%Diff 90°	%Diff 180°	%Diff 270°
1	0.00018153	0.00018153	0.00018153	0.00018153	-4.51	-4.51	-4.51	-43.37
2	0.0001643	0.0001643	0.0001643	0.0001643	-6.68	-6.68	-6.68	-6.68
3	0.00015094	0.00015094	0.00015095	0.00015094	1.09	1.09	1.09	1.09
4	0.000103	0.000103	0.000103	0.000103	15.71	14.03	14.03	14.03
5	0.00009691	0.00009691	0.00009691	0.00009691	12.54	6.81	12.54	12.54
6	0.00007101	0.00007101	0.00007101	0.00007101	13.85	13.85	13.85	13.85
7	0.00006103	0.00006103	0.00006103	0.00006103	4.01	4.01	4.01	4.01
8	0.00005408	0.00005408	0.00005408	0.00005408	8.09	8.09	8.09	8.09
9	0.00004693	0.00004693	0.00004693	0.00004693	10.32	10.32	10.32	10.32
10	0.00001964	0.00001964	0.00001964	0.00001964	1.11	1.11	1.11	1.11

TABLE B.3: % Difference of Radial and Axial displacements of the Xs1D5 (Constant Spring Stiffness of 1e15) Shell model with the Solid model

Xs1D5 Shell Model - Radial Displacements (mm)								
Loc	0.0°	90.0°	180°	270°	%Diff 0°	%Diff 90°	%Diff 180°	%Diff 270°
1	0.00332701	0.0011144	0.00058236	0.00111439	15.17	1.46	5.26	27.19
2	0.00233951	0.00106024	0.00057473	0.00106022	4.68	2.99	6.61	3.10
3	0.00191471	0.00098959	0.00055305	0.00098958	8.76	5.57	7.63	5.69
4	0.00101297	0.00072492	0.00045216	0.00072491	17.27	8.11	8.90	8.35
5	0.00077551	0.00062602	0.00040448	0.00062601	14.61	2.29	12.48	9.07
6	0.00038513	0.00035267	0.00030054	0.00035267	19.67	11.15	13.95	11.28
7	0.0002203	0.00023228	0.00022053	0.00023228	0.64	10.51	7.75	10.61
8	0.00014779	0.00017625	0.00018154	0.00017625	19.37	11.58	5.50	11.64
9	0.00009591	0.00012937	0.00014963	0.00012937	68.56	10.48	3.71	10.50
10	0.00003536	0.00003145	0.00007264	0.00003145	2.21	7.36	18.87	7.17

Xs1D5 Shell Model - Axial Displacements (mm)								
Loc	0.0°	90.0°	180°	270°	%Diff 0°	%Diff 90°	%Diff 180°	%Diff 270°
1	0.00019848	0.00019847	0.00019848	0.00019847	14.27	14.27	14.27	56.74
2	0.00017488	0.00017488	0.00017488	0.00017488	13.55	13.55	13.55	13.55
3	0.00016	0.00016	0.00016	0.00015999	4.84	4.84	4.84	4.84
4	0.00011166	0.00011166	0.00011166	0.00011166	8.63	6.80	6.80	6.80
5	0.00010408	0.00010408	0.00010408	0.00010408	6.07	0.09	6.07	6.07
6	0.00007817	0.00007817	0.00007817	0.00007817	5.17	5.17	5.17	5.17
7	0.00006099	0.00006099	0.000061	0.00006099	4.07	4.07	4.06	4.07
8	0.00005406	0.00005406	0.00005407	0.00005406	8.12	8.12	8.11	8.12
9	0.00004693	0.00004693	0.00004693	0.00004693	10.32	10.32	10.32	10.32
10	0.00001965	0.00001965	0.00001965	0.00001965	1.06	1.06	1.06	1.06

Appendix C

Offset nodes connected using Spring elements

In Section. 4.5 and Section. 4.6 it was argued that there existed a spring stiffness between a range of 10^5N/mm and 10^8N/mm that could be used to connect discontinous sections to capture the actual stiffness of the component being modeled. But it was evident by the end of Chapter. 4 that discontinuities with an offset connected using springs will not match the solid model that it is compared against. A simple model is built in this appendix to prove the difference in results when a spring is used to connect coincident and offset discontinuities.

C.1 Single component - No discontinuities

A simple cylinder is modelled with shell elements and axially loaded using a force of 1,000N as shown in Fig. C.1. One end of the cylinder is fixed in all degrees of freedom and the other end is loaded in the axial direction. The 1,000N load is equally distributed between all the nodes along the circumference of the cylinder at the loading end. A linear static analysis is performed using ANSYS and it is observed that the cylinder elongates with a maximum axial deformation of 5.84^{-4}mm as shown in Fig. C.2.

C.2 Two halves - Coincident Joint

In this case, the simple cylinder is split into two exact halves along its length. The split in the cylinder's length creates a discontinuity and this has to be connected for the two halves to behave as one single component. In order to do this, 6 springs (one for each

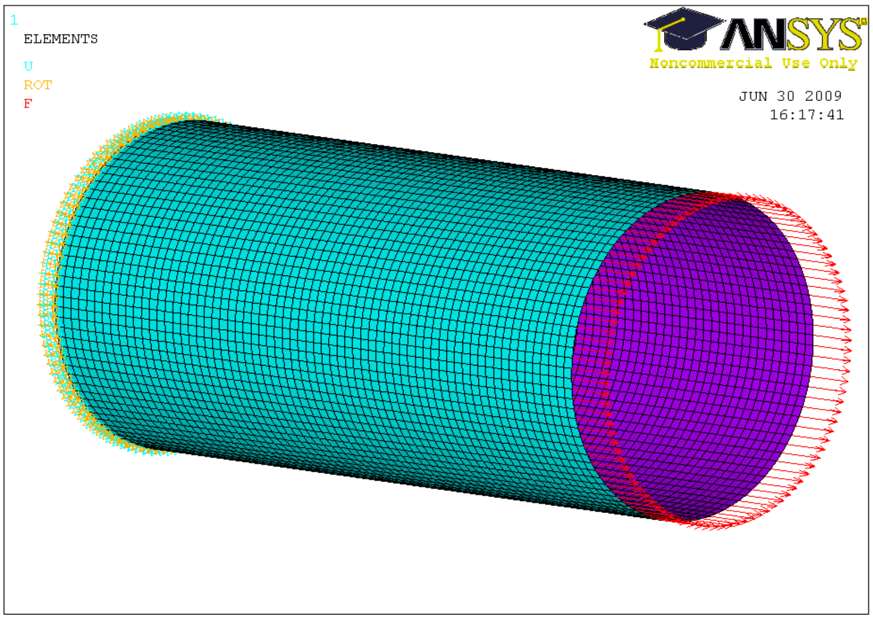


Figure C.1: Simple cylinder with axial load - One component, no discontinuities

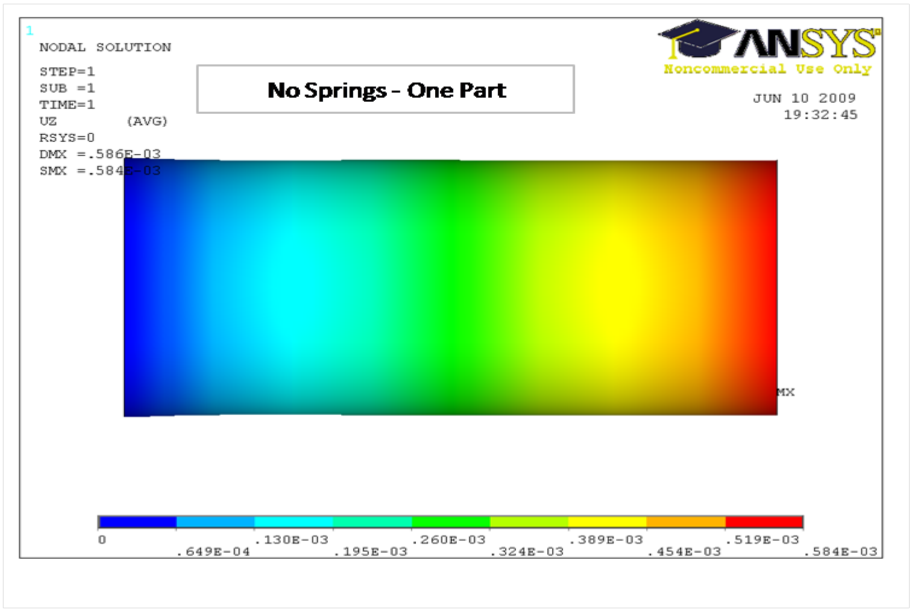


Figure C.2: Max axial deformation of the simple cylinder

degree of freedom) are created with 10^8N/mm as the spring stiffness between the coincident nodes at the discontinuous joint. The two connected cylinders are analyzed with the same boundary condition and load as was done with the single cylinder. The results are as shown in Fig. C.3. It is evident from the figure that when two discontinuous surfaces are connected at their edges using coincident springs, the model behaves like a single component as is expected. The maximum axial deformation of the split model is 5.84^{-4}mm and is same as the max deformation of the single model described in Section. C.1.

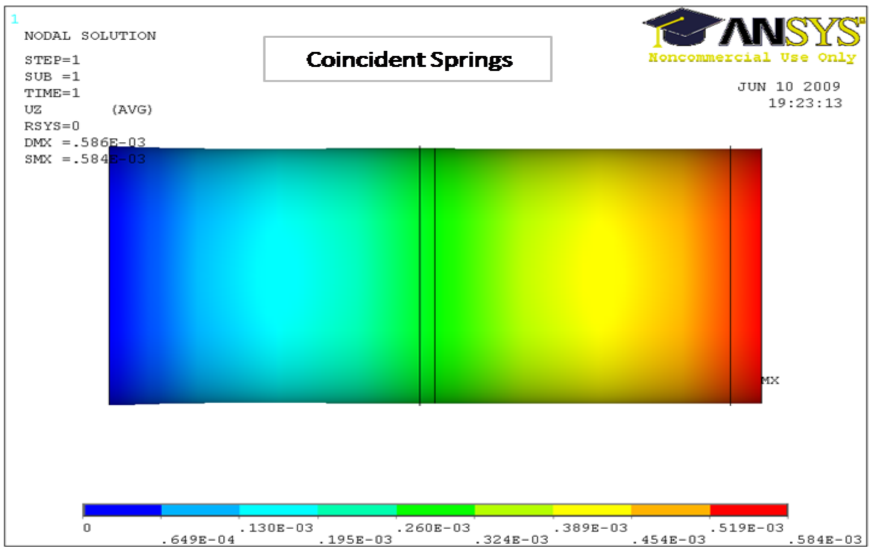


FIGURE C.3: Max axial deformation of the split cylinder

C.3 Two halves - Offset Joint

To demonstrate the effect of non-coincident springs on the results for a typical axial load, the front half of the split model used in Section. C.2 is remodelled with a reduced diameter as shown in Fig. C.4. The model is analyzed exactly the same way as the other two models in this appendix. The only difference however is that the springs that have been used to connect the two surfaces are not coincident anymore but separated by a distance which is given by the relation:

$Offset = \left(\frac{MaxDiameter - MinDiameter}{2}\right) mm$

The maximum axial deformation of this model is 6.17^{-4}mm as shown in Fig. C.5. It is clearly evident that when a spring is offset by a distance, its axial deformation is not comparable to either the single cylinder or the split cylinder with coincident nodes at the discontinuity. The axial deformation of the offset model when compared to the other two model varies by 5.65%. This inherent modelling error of offset springs and the approximation made while extracting the midline will add to the total error of the reduced model stiffness.

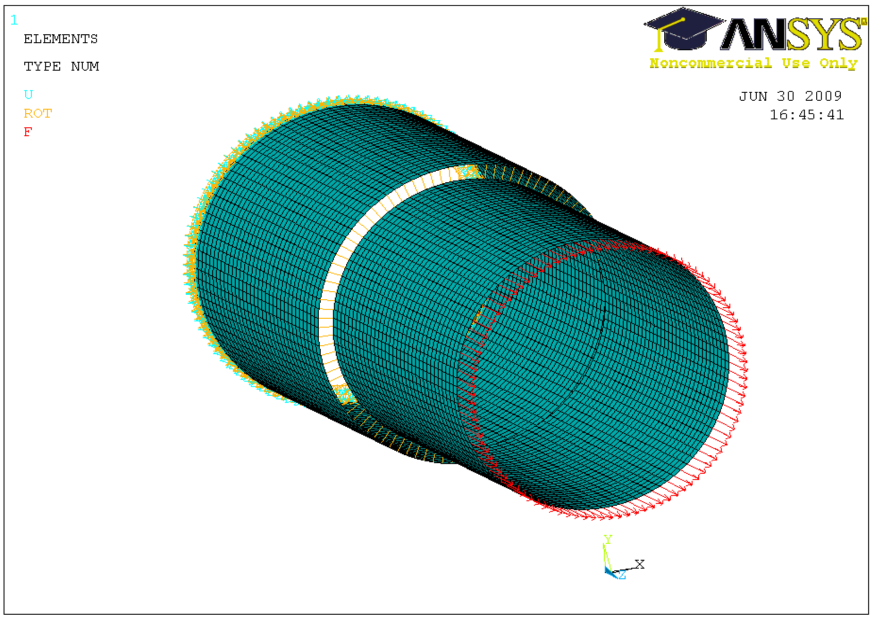


FIGURE C.4: Front half of the split cylinder reduced in diameter to create an offset at the discontinuity

C.4 Conclusion

It can thus be concluded that when discontinuous surfaces have to be connected, springs can be used if the edges of the surfaces that form the discontinuous joint are coincident, i.e. coincident nodes exist at the discontinuity to create springs elements having zero length. It can be seen from this study that when there is an offset and springs are used to connect shell elements, the load transfer between the two halves is not consistent. The sum of reaction forces at all the nodes at the constrained end for all three models shows 1,000N. This is what is expected of any FE model, i.e. total load applied and total reaction forces should match. However, the spring element creates an artificial moment when the load defined does not act along the element length. It is therefore not a good practice to use springs to model discontinuous offset joints for transferring loads which do not act along the line of action of

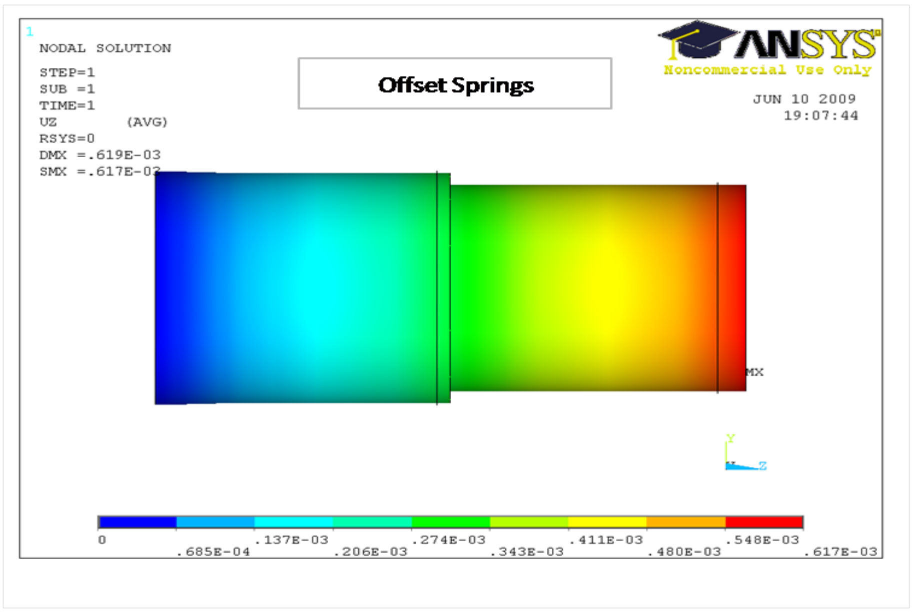


FIGURE C.5: Max axial deformation of the split cylinder with an offset at the discontinuity

the springs created.

Appendix D

Sample Hypermesh script for use in batch mode

D.1 Hypermesh command file to create an LS-Dyna keyword file

The hypermesh command file was created for use in setting up the optimization of the stepped beam model. As discussed in Chapter 3 the geometric model is parameterized in Siemens NX4. Once the geometry has been changed, it needs to go through an FE preprocessing software for meshing and application of loads and boundary conditions. This finite element model file can then be output for use in an appropriate FE solver. In this case, when the hypermesh command file is run within hypermesh in batch mode, it generates an LS-Dyna keyword file as input for the LS-Dyna static analysis solver.

```
############################################################################

!!!!!!!!!!!!! Importing the UG part file with the different layers as
!             components into HM V8.0SR1

*createstringarray(12) "SELECTIONS 0 2 0 0 0 1 0 0 0" "" "BEGIN_PARTS"
"END_PARTS" "" "BEGIN_LAYERINFO" "DISABLE  " "ENABLE  *"
"END_LAYERINFO" "" "COMPONENT_NAME custom <Layer>" ""
```

```
*feinputwithdata("#ug\ug","modifiedpart\stepped-beam.prt",1,0,0.01,1,1,1,12)

!#################### Step-1 ######################################

!!!!!!!!   Set the template file to LS-Dyna user profile

*templatefileset("C:/Altair-HW8.0SR1/templates/feoutput/ls-dyna/dyna.key")

*setmacrofile("C:/Altair-HW8.0SR1/hm/scripts/UserProfiles/..
.../dynakey/dyna_key.mac")

*enablemacromenu(1)

!#################### Step-2 ######################################

! Create Shell mesh on all components based on element size !!!!!!

*setedgedensitylink(0)
*elementorder(1)
*elementtype(103,1) !!!!!! (tria3, ctria3)
*createmark(surfaces,1) "all"

!! ( SHELL MESH SIZE IS 20 )

*defaultremeshsurf(1,10,0,0,2,1,1,1,1,0,0,0,0)

! #################### Step-3 ######################################

! To apply constraints in the fixed end !!!!!!!!!!!!!!

!! NOTE : CHECK THE COLLECTOR NUMBER TO MAKE SURE IT IS THE SURFACE THAT
   NEEDS TO BE FIXED THAT IS IN COLLECTOR 3 !!!!!!
```

```
*createmark(nodes,1) "by collector " 3
*loadcreateonentity_curve(nodes,1,3,1,0,0,0,0,0,0,0,0,0,0,0)

! #################### Step-4 #####################################

!Delete shell elements created in Component 3 !!!!!!!!!

*createmark(nodes,1) "by collector " 3
*loadcreateonentity_curve(nodes,1,3,1,0,0,0,0,0,0,0,0,0,0,0)

*createmark(elements,1) "all"
*equivalence(elements,1,0.001,1,0,0)

*createmark(elements,1) "by collector " 3
*deletemark(elements,1)


!#################### Step-5 #####################################

! Create 1D elements on the extracted EDGE in component 2 !!!!!!!!

!! NOTE : CHECK THE COLLECTOR NUMBER TO MAKE SURE IT IS THE EDGE THAT
   NEEDS TO BE LOADED THAT IS IN COLLECTOR 2 !!!!!!

!! CHECK IF THE ELEMENT SIZING IS SAME AS THAT USED WHILE SHELL MESHING !!!

*elementtype(5,1) !!!!!!! (spring, celas1)

*createmark(lines,1) "by collector " 2
*linemesh_preparedata(lines,1,30)
*elementsizeset(10)
*linemesh_savedata(1,21,0)
```

```
!#################### Step-6 ####################################


!Create Force load on the 1D elements !!!!!!!!!


*createmark(elements,1) "by config" spring

*createmark(elements,1) "reverse"

*maskentitymark(elements,1,0)

*displaycollectorwithfilter(loadcols,"none","",1,0)


!!!!!! Create a nodal component of the nodes

!!!!!! where the forces have been applied


*createmark(nodes,1) "displayed"

*entitysetcreate("ndload",nodes,1)

*createmark(sets,2) "ndload"


*dictionaryload(sets,2,"C:/Altair-HW8.0SR1/templates...

.../feoutput/ls-dyna971/dyna.key","Node")


!!!!! Apply force on the selected nodes and

!!!!! merge them with the shell mesh nodes !!!


!!!! If we equivalence nodes without applying forces on it,

!!!! then the identity of nodes in "ndloads" is lost


!!! This is why I apply a dummy load in this step,

!!! equivalance the nodes and then delete force and

!!! the force load collector


*collectorcreate(loadcols,"force","",7)

*createmark(loadcols,2) "force"


*dictionaryload(loadcols,2,"C:/Altair-HW8.0SR1/templates...
.../feoutput/ls-dyna/dyna.key","LoadNodeSet")

*loadcreateonentity_curve(nodes,1,1,1,0,-1000,0,0,0,10,0,0,0,0,0)
*displaycollectorwithfilter(loadcols,"on","auto1",1,0)

*createmark(elements,1) "all"
*equivalence(elements,1,0.001,1,0,0)

*createmark(loads,1) "by collector" force
*deletemark(loads,1)
*window(0,0,0,0,0)

*createmark(loadcols,2) "force"
*deletemark(loadcols,2)

!#################### Step-7 #####################################

!Delete spring elements !!!!!!!

*createmark(elements,1) "by config" spring
*deletemark(elements,1)

!#################### Step-8 #####################################

!!!!!!!!!!!!!!Create 3D solid elements - CTETRA !!!!!!!!!

*elementtype(106,1) !!!!! (tetra4, ctetra)

*createmark(elements,1) "all"
*createmark(elements,2) "all"
*tetramesh(elements,1,elements,2,1.2,0.75,97)

!#################### Step-9 #####################################
```

```
!!!!! Delete the shell elements elements !!!!!!!

*createmark(elements, 1) "by config type" tria3
*deletemark(elements, 1)
*displaycollectorwithfilter(components,"all","",1,0)

!#################### Step-10 ####################################

!!!!!!!LS Dyna Control Cards setup for export to LS-Dyna 971

!!!!!!      Create materials - ELASTIC

*collectorcreateonly(materials,"Steel","",1)
*createmark(materials,2) "Steel"

*dictionaryload(materials,2,"C:/Altair-HW8.0SR1/templates...
.../feoutput/ls-dyna971/dyna.key","MATL1")

*attributeupdateintmark(materials,2,999,9,2,0,0)
*attributeupdateintmark(materials,2,3220,9,2,0,1)
*attributeupdateintmark(materials,2,90,9,2,0,0)
*attributeupdatedouble(materials,1,118,9,1,0,8e-006)
*attributeupdatedouble(materials,1,119,9,1,0,216620)
*attributeupdatedouble(materials,1,120,9,1,0,0.27)
*attributeupdatedouble(materials,1,122,9,0,0,0)
*attributeupdatedouble(materials,1,123,9,0,0,0)
*attributeupdatedouble(materials,1,124,9,0,0,0)

!#################### Step-11 ####################################

!!!!!!! Create Load Curve as in "DEFINE"
```

```
!!! NOTE : THE LOAD TO BE ENTERED HERE IS BASED ON THE NUMBER OF
    NODES IN THE "NDLOAD" NODAL COMPONENT
!!! i.e; : if 3 nodes then 5000N/3 is the force, so on so forth

*plot()
*createmark(curves,1)
*clearmark(curves,1)
*xyplotcreatecurve("{0}","","","",1,"{0}","","","",1)
*renamecollector(curves,"curve1","1")
*xyplotcurvemodify("1","title","1",0,1)
*curvedeletepoint(1,1)
*createmark(curves,1) "1"
*attributeupdateintmark(curves,1,3016,9,0,0,0)
*createmark(curves,1) "1"
*attributeupdateintmark(curves,1,5068,9,0,0,1)
*createmark(curves,1) "1"
*clearmark(curves,1)

*curveaddpoint(1,0,0,0)
*curveaddpoint(1,1,1,-1)
*xyplotmodifycurve("1","{ 0.0, 1.0 }","","","",1,"{ 0.0, -1 }","","","",1)
*plot()

!################### Step-12 #####################################

!!!!!!!  Now start creating the DATABASE cards

!!!!!!!!!! ASCII-OPTION Card

*cardcreate("DBOpt")
*attributeupdateint(cards,1,4435,9,2,0,3)
*attributeupdatedouble(cards,1,2198,9,1,0,0.2)
*attributeupdatedouble(cards,1,2192,9,1,0,0.2)
```

```
*attributeupdatedouble(cards,1,2203,9,1,0,0.2)
*attributeupdatedouble(cards,1,2191,9,1,0,0.2)
*attributeupdatedouble(cards,1,2199,9,1,0,0.2)

!!!!!!!!! The HISTORY_NODE_SET and HISTORY_ELEMENT_SET

*createmark(nodes,1) "all"
*outputblockscreate("1",nodes,1)

*createmark(elements,1) "all"
*outputblockscreate("2",elements,1)

!#################### Step-13 #####################################

!!!!!!!!!!!  Now start creating the CONTROL cards

!!!!!!! IMPLICIT_AUTO

*createmark(cards,1)
*clearmark(cards,1)
*cardcreate("ImpAuto")
*attributeupdateint(cards,2,3143,9,1,0,1)
*attributeupdateint(cards,2,3144,9,1,0,11)
*attributeupdateint(cards,2,3145,9,1,0,5)
*attributeupdatedouble(cards,2,3173,9,1,0,0)
*attributeupdateint(cards,2,5091,9,2,0,0)
*attributeupdatedouble(cards,2,3029,9,1,0,0)
*attributeupdatedouble(cards,2,4468,9,1,0,0)

!!!!!!! IMPLICIT_GENERAL

*createmark(cards,1) 2
*clearmark(cards,1)
```

```
*cardcreate("ImpGen")
*attributeupdateint(cards,3,5091,9,2,0,0)
*attributeupdateint(cards,3,5010,9,1,0,1)
*attributeupdatedouble(cards,3,5011,9,1,0,0.2)
*attributeupdateint(cards,3,3148,9,1,0,2)
*attributeupdateint(cards,3,3149,9,1,0,1)
*attributeupdateint(cards,3,3150,9,1,0,2)
*attributeupdateint(cards,3,3151,9,1,0,0)
*attributeupdateint(cards,3,3152,9,1,0,0)

!!!!!! TERMINATION

*createmark(cards,1) 3
*clearmark(cards,1)
*cardcreate("Termin")
*attributeupdatedouble(cards,4,2059,9,1,0,1)
*attributeupdateint(cards,4,2060,9,0,0,0)
*attributeupdatedouble(cards,4,2061,9,0,0,0)
*attributeupdatedouble(cards,4,2062,9,0,0,0)
*attributeupdatedouble(cards,4,2063,9,0,0,0)

!################### Step-14 #####################################

###########  Creating the LOAD cards
*collectorcreate(loadcols,"force","",7)
*createmark(loadcols,2) "force"

*dictionaryload(loadcols,2,"C:/Altair-HW8.0SR1/templates...
.../feoutput/ls-dyna/dyna.key","LoadNodeSet")

*loadtype(1,1)

*evaltclstring("source C:/Altair-HW8.0SR1/hm/scripts/VARYLOAD.tcl",0)
```

```
!!!!!!!!!!!!! Moving all the loads to loadcollector "auto"

*createmark(loads,1) "all"
*movemark(loads,1,"auto1")

!!!!!!!!!!! Delete the "force" load collector since there
!!!!!!!!!!! are no nodes in it. It was created primarily
!!!!!!!!!!! for updating the force created with the Load CASE ID.

*createmark(loadcols,2) "force"
*deletemark(loadcols,2)

!#################### Step-15 #####################################

!!!!!!!!!!!  Creating the SECTION_SOLID  cards

*collectorcreateonly(properties,"tet","",7)
*createmark(properties,2) "tet"

*dictionaryload(properties,2,"C:/Altair-HW8.0SR1/templates...
.../feoutput/ls-dyna971/dyna.key","SectSld")

*attributeupdateintmark(properties,2,4540,9,2,0,1)
*attributeupdateintmark(properties,2,90,9,2,0,0)
*attributeupdateintmark(properties,2,399,9,1,0,18)
*attributeupdateintmark(properties,2,1650,9,0,0,1)

!#################### Step-16 #####################################

!!!!!!!!!!!  Creating the LS-Dyna PART

*collectorcreate(components,"steppedbeam","Steel",7)
```

```
*createmark(components,2) "steppedbeam"

*dictionaryload(components,2,"C:/Altair-HW8.0SR1/templates...
.../feoutput/ls-dyna/dyna.key","Part")

*attributeupdateintmark(components,2,459,9,2,0,1)
*attributeupdateentitymark(components,2,460,9,1,0,properties,1)

*createmark(elements,1) "all"
*movemark(elements,1,"steppedbeam")

!#################### Step-17 #####################################

!!!!!! Deleting the nodal component containing
!!!!!! the nodes that carry the force loads

!!!!!  Deleting the geometric components

*createmark(sets,1) "ndload"
*deletemark(sets,1)

*createmark(components,1) beam 1 2 3
*deletemark(components,1)

!#################### Step-18 #####################################

!!!!!! Renumbering all entities

*renumbersolveridall(1,1,0,0,0,0,0)

!#################### Step-19 #####################################

!!!!!  Export the file set-up in HyperMesh to LS-Dyna Version 971
```

```
*feoutput("C:/Altair-HW8.0SR1/templates/feoutput/ls-dyna...
.../dyna.key","stepped-beam.k",1,0,1)
```

D.2 LS-Dyna keyword file as created using HyperMesh in batch mode

```
$$ HM_OUTPUT_DECK created 11:52:34 03-31-2008 by HyperMesh Version 8.0SR1
$$ Ls-dyna Input Deck Generated by HyperMesh Version  : 8.0SR1
$$ Generated using HyperMesh-Ls-dyna 970 Template Version : 8.0sr1
*KEYWORD
*CONTROL_TERMINATION
$$  ENDTIM    ENDCYC     DTMIN    ENDENG    ENDMAS
       1.0
*CONTROL_IMPLICIT_AUTO
$$   IAUTO    ITEOPT    ITEWIN     DTMIN     DTMAX
         1        11         5       0.0       0.0       0.0
*CONTROL_IMPLICIT_GENERAL
$$  IMFLAG       DT0    IMFLAG      NSBS       IGS     CNSTN      FORM
         1       0.2         2         1         2         0         0
$$DATABASE_OPTION -- Control Cards for ASCII output
*DATABASE_DEFGEO
       0.2         3
*DATABASE_ELOUT
       0.2         3
*DATABASE_NODFOR
       0.2         3
*DATABASE_NODOUT
       0.2         3
*DATABASE_SPCFORC
       0.2         3
```

```
*NODE
       1             0.0           200.0             0.0
       2             0.0           190.0             0.0
       3             0.0           180.0             0.0
                         |
                         |
                         |
       7463 348.42278259762 158.04332865505 21.957916209794
       7464 429.90085078268   71.0352154556 9.1419099231255
       7465 306.09937858011 165.45755422654 20.213868334868
       7466 109.20361451059  24.94804366794 20.406449575438
       7467  1263.211661477 179.30014786422 11.267572588996

*MAT_ELASTIC
$HMNAME MATS       1Steel
         18.0000E-06  216620.0      0.27
*PART
$HMNAME COMPS       4steppedbeam
$HWCOLOR COMPS       4       7


         4         1         1
*SECTION_SOLID
$HMNAME PROPS       1tet
         1        18
*ELEMENT_SOLID
       1 4 7055 6299 1322 3813 3813 3813 3813 3813
       2 4 5497 5940 5601 6033 6033 6033 6033 6033
       3 4 1202 1501 1489 6145 6145 6145 6145 6145
       4 4 1128 5357 1124 1127 1127 1127 1127 1127
       5 4 5295 2950 2952 2949 2949 2949 2949 2949
       6 4 6280 6938 5976 5486 5486 5486 5486 5486
       7 4 5979 2369 5951 5242 5242 5242 5242 5242
       8 4 32   1577 31   62   62   62   62   62
```

```
                           |
                           |
                           |
       28826 4 5527 2882 3146 3144 3144 3144 3144 3144
       28827 4 1633 1374 1311 1632 1632 1632 1632 1632
       28828 4 1464 1125 1124 6846 6846 6846 6846 6846
       28829 4 984  3467 1086 3466 3466 3466 3466 3466
       28830 4 1640 1096 1031 1652 1652 1652 1652 1652

*LOAD_NODE_POINT
$HMNAME LOADCOLS       1auto1
$HWCOLOR LOADCOLS       1      11
       392         2         1    1250.0         0
       391         2         1    1250.0         0
       390         2         1    1250.0         0
        87         2         1    1250.0         0
*BOUNDARY_SPC_NODE
$HMNAME LOADCOLS       1auto1
$HWCOLOR LOADCOLS       1      11
         1   0   1  1  1  1  1  1
         2   0   1  1  1  1  1  1
         3   0   1  1  1  1  1  1
         4   0   1  1  1  1  1  1
         5   0   1  1  1  1  1  1
         6   0   1  1  1  1  1  1



*DEFINE_CURVE
$HMNAME CURVES       11
$HWCOLOR CURVES       1      11
$HMCURVE      1    0 1
         1   0     1.0  1.0   0.0   0.0      0
             0.0       0.0
             1.0       -1.0
*END
$HMUSEDSYST
$HMASSEM        1       11 stepped-beam.prt
```

Appendix E

Mantle-3D sample Matlab script

The matlab code scripted to extract the medial mesh of a complex 3D non-axisymmetric geometry based on the workflow shown in Fig. 7.5 has been listed in this appendix for each module.

E.1 Mantle3D.m

```
%% Matlab script for Mantle-3D written by Felix Stanley : Oct 2009 - July 2010
clear all; close all; fclose('all');
if exist('ConvexEdgePoints.cmf','file'), delete('ConvexEdgePoints.cmf');end;
if exist('ConvexEdgePoints.txt','file'), delete('ConvexEdgePoints.txt');end;
if exist('ConcaveEdgePoints.cmf','file'), delete('ConcaveEdgePoints.cmf');end;
if exist('ConcaveEdgePoints.txt','file'), delete('ConcaveEdgePoints.txt');end;
if exist('CornerVertexPoints.cmf','file'), delete('CornerVertexPoints.cmf');end;
if exist('ConvexVertexPoints.txt','file'), delete('ConvexVertexPoints.txt');end;
if exist('ConcaveNodeInfo.txt','file'), delete('ConcaveNodeInfo.txt');end;
if exist('ConcaveNodeInfo-Cleaned.txt','file'), delete('ConcaveNodeInfo-Cleaned.
    txt');end;
if exist('ConcaveEdgePointsCleaned.txt','file');delete('ConcaveEdgePointsCleaned.
    txt');end;
if exist('DataPts2.cmf','file'), delete('DataPts2.cmf');end;
if exist('OptimumSpheres-SurfaceSpheres.cmf','file'),delete('OptimumSpheres-
    SurfaceSpheres.cmf');end;
if exist('PointsForTriangulation-SurfaceSpheres.txt','file'),delete('
    PointsForTriangulation-SurfaceSpheres.txt');end;
```

```matlab
if exist('PointsForTriangulation-SurfaceSpheres.cmf','file'),delete('PointsForTriangulation-SurfaceSpheres.cmf');end;
if exist('OptimumSpheres-DihedralSpheres.cmf','file'),delete('OptimumSpheres-DihedralSpheres.cmf');end;
if exist('PointsForTriangulation-DihedralSpheres.txt','file'),delete('PointsForTriangulation-DihedralSpheres.txt');end;
if exist('PointsForTriangulation-DihedralSpheres.cmf','file'),delete('PointsForTriangulation-DihedralSpheres.cmf');end;
if exist('Triangulation-TsecNew.cmf','file'),delete('Triangulation-TsecNew.cmf');end;

fidcvx = fopen('ConvexEdgePoints.txt','a'); fidcve = fopen('ConcaveEdgePoints.txt','a');
fidcnr = fopen('CornerVertexPoints.cmf','a'); fiddtp2 = fopen('DataPts2.cmf','a'); fidcvetxt = fopen('ConcaveNodeInfo.txt','a');
fidcenters = fopen('OptimumSpheres-SurfaceSpheres.cmf','a');fidpft = fopen('PointsForTriangulation-SurfaceSpheres.txt','a');
fidpftcmf = fopen('PointsForTriangulation-SurfaceSpheres.cmf','a');

nd = dlmread('TsectionM2-nodeData.txt'); tria = dlmread('TsectionM2-surfData.txt'); tetra = dlmread('TsectionM2-tetraData.txt');
tria2=dlmread('RunOnlyTheseTria.txt');

MaxRadius = 5.0;
elemsAcrossThickness = 3; % Number of tetra along the thickest section of the geometry
NumberOfNodes=11; % Number of parts each edge of the triangle element will be split into for DataPts (never to be set less than 1)
numnodes=1; % Number of elements inside a single triangle element for processing spheres (never to be set less than 2) and should be set to atleast 2 for coarse meshes
AngleTolerance = 15; % For use in FindConvexNodes.m
VectorsAlongDihedralAngle = 5; % Should be set to 7, 5 or 3 - For use in DataToStoreConcavePointsMod3.m - '7, 5 & 3' includes the master and slave vectors

t4=clock;

SurfaceSpheres; % Find all spheres touching flat surface points

eti4=etime(clock,t4);
```

```
disp(['***** Total time for processing all Surface nodes is : ',num2str(eti4),'
    sec *****']);


fclose(fidcvx);
fclose(fidcve);
fclose(fidcnr);
fclose(fiddtp2);
fclose(fidcenters);
fclose(fidcvetxt);


ConcaveEdgeVertex1= dlmread('ConcaveEdgePoints.txt');
[ConcaveEdgeVertex1Tmp]=remove_duplicates_allcolumns(ConcaveEdgeVertex1,1e-5);
ConcaveEdgeVertex2=ConcaveEdgeVertex1(ConcaveEdgeVertex1Tmp',:);

ConvexEdgeVertex1= dlmread('ConvexEdgePoints.txt');
[ConvexEdgeVertexTmp]=remove_duplicates_allcolumns(ConvexEdgeVertex1,1e-5);
ConvexEdgeVertex2=ConvexEdgeVertex1(ConvexEdgeVertexTmp',:); % Use data from here
     for Triangulation

ConcaveEdgeVertex3=[];
cvechktol=1e-4;
for cvechk = 1:size(ConcaveEdgeVertex2,1),
    chktmp = find(ConcaveEdgeVertex2(cvechk,1)==ConvexEdgeVertex2(:,2) &
        ConcaveEdgeVertex2(cvechk,2)==ConvexEdgeVertex2(:,3) & ConcaveEdgeVertex2
        (cvechk,3)==ConvexEdgeVertex2(:,4));
    if isempty(chktmp)
        ConcaveEdgeVertex3=[ConcaveEdgeVertex3;ConcaveEdgeVertex2(cvechk,:)]; %
            Use data from here for spheres at concave points
    end
end

fidcveCHK = fopen('ConcaveEdgePointsCleaned.txt','a');
fidcvechk = fopen('ConcaveEdgePoints.cmf','a');
for cvechk2 = 1:size(ConcaveEdgeVertex3,1)
    fprintf(fidcvechk,'*createnode(%8.3f,%8.3f,%8.3f,0,0,0)\n',ConcaveEdgeVertex3
        (cvechk2,1), ConcaveEdgeVertex3(cvechk2,2),ConcaveEdgeVertex3(cvechk2,3))
        ;
    fprintf(fidcveCHK,'%8.3f %8.3f %8.3f\n',ConcaveEdgeVertex3(cvechk2,1),
        ConcaveEdgeVertex3(cvechk2,2),ConcaveEdgeVertex3(cvechk2,3));
```

```
end
fclose(fidcvechk);
fclose(fidcveCHK);

fidcvxchk = fopen('ConvexEdgePoints.cmf','a');
for cvechk3 = 1:size(ConvexEdgeVertex2,1)
    fprintf(fidcvxchk,'*createnode(%8.3f,%8.3f,%8.3f,0,0,0)\n',ConvexEdgeVertex2(
        cvechk3,2), ConvexEdgeVertex2(cvechk3,3),ConvexEdgeVertex2(cvechk3,4));
end
fclose(fidcvxchk);

for pft = 1:size(OptimumSphere,1),
    fprintf(fidpft,'%8.3f %8.3f %8.3f\n',OptimumSphere(pft,1),OptimumSphere(pft
        ,2),OptimumSphere(pft,3));
    fprintf(fidpftcmf,'*createnode(%8.3f,%8.3f,%8.3f,0,0,0)\n',OptimumSphere(pft
        ,1),OptimumSphere(pft,2),OptimumSphere(pft,3));
end
for convxnodes = 1:size(ConvexEdgeVertex2,1),
    fprintf(fidpft,'%8.3f %8.3f %8.3f\n',ConvexEdgeVertex2(convxnodes,2),
        ConvexEdgeVertex2(convxnodes,3),ConvexEdgeVertex2(convxnodes,4));
    fprintf(fidpftcmf,'*createnode(%8.3f,%8.3f,%8.3f,0,0,0)\n',ConvexEdgeVertex2(
        convxnodes,2),ConvexEdgeVertex2(convxnodes,3),ConvexEdgeVertex2(
        convxnodes,4));
end
fclose(fidpft);
fclose(fidpftcmf);

% clean the ConcaveNodeInfo.txt file for all duplicate nodes
clear all

t5 = clock;
DihedralSpheres; % Find spheres at all concave points
eti5=etime(clock,t5);
disp(['***** Total time for processing all Concave nodes is : ',num2str(eti5),'
    sec *****']);

for pft2 = 1:size(OptimumSphere2,1),
    fprintf(fidpft2,'%8.3f %8.3f %8.3f\n',OptimumSphere2(pft2,1),OptimumSphere2(
        pft2,2),OptimumSphere2(pft2,3));
    fprintf(fidpftcmf2,'*createnode(%8.3f,%8.3f,%8.3f,0,0,0)\n',OptimumSphere2(
        pft2,1),OptimumSphere2(pft2,2),OptimumSphere2(pft2,3));
```

```
end

fclose(fidpft2);
fclose(fidpftcmf2);
fclose(fidcenters2);
fclose('all');

clear all
t6 = clock;
TriInput11 = dlmread('PointsForTriangulation-SurfaceSpheres.txt');
TriInput12 = dlmread('PointsForTriangulation-DihedralSpheres.txt');

TriInput1 = [TriInput11;TriInput12];
TriInput2 = remove_duplicates_allcolumns(TriInput1,0.25);
p = TriInput1(TriInput2',1:3);

TestMyCrustOpen;
eti6=etime(clock,t6);
disp(['***** Total time for triangulation of points is : ',num2str(eti6),'sec
    *****']);

close all
fclose('all');
```

E.2 ConnectedElements.m

```
clear NodeInfo econTetra econTria econTriaTetra
%p4 = Find4thNodeMAsurf(p1,p2,p3,tetra);
econTetra1=[]; econTetra3=[];
% find all tetras that are attached to tetra that belongs to this tria element
%elemsAcrossThickness = 5;
if elemsAcrossThickness <2;
    econTetra = find(p1==tetra(:,1) | p1==tetra(:,2) | p1==tetra(:,3) | p1==tetra
        (:,4)...
        | p2==tetra(:,1) | p2==tetra(:,2) | p2==tetra(:,3) | p2==tetra(:,4)...
        | p3==tetra(:,1) | p3==tetra(:,2) | p3==tetra(:,3) | p3==tetra(:,4)...
        | p4==tetra(:,1) | p4==tetra(:,2) | p4==tetra(:,3) | p4==tetra(:,4));
else
```

```
    econTetra1 = find(p1==tetra(:,1) | p1==tetra(:,2) | p1==tetra(:,3) | p1==tetra(:,4)...
        | p2==tetra(:,1) | p2==tetra(:,2) | p2==tetra(:,3) | p2==tetra(:,4)...
        | p3==tetra(:,1) | p3==tetra(:,2) | p3==tetra(:,3) | p3==tetra(:,4)...
        | p4==tetra(:,1) | p4==tetra(:,2) | p4==tetra(:,3) | p4==tetra(:,4));

    for attachelem = 1:elemsAcrossThickness-1,
        for ectitr = 1:size(econTetra1,1)
            econTetra2 = find(tetra(econTetra1(ectitr),1)==tetra(:,1) | tetra(econTetra1(ectitr),1)==tetra(:,2) | tetra(econTetra1(ectitr),1)==tetra(:,3) | tetra(econTetra1(ectitr),1)==tetra(:,4)...
                | tetra(econTetra1(ectitr),2)==tetra(:,1) | tetra(econTetra1(ectitr),2)==tetra(:,2) | tetra(econTetra1(ectitr),2)==tetra(:,3) | tetra(econTetra1(ectitr),2)==tetra(:,4)...
                | tetra(econTetra1(ectitr),3)==tetra(:,1) | tetra(econTetra1(ectitr),3)==tetra(:,2) | tetra(econTetra1(ectitr),3)==tetra(:,3) | tetra(econTetra1(ectitr),3)==tetra(:,4)...
                | tetra(econTetra1(ectitr),4)==tetra(:,1) | tetra(econTetra1(ectitr),4)==tetra(:,2) | tetra(econTetra1(ectitr),4)==tetra(:,3) | tetra(econTetra1(ectitr),4)==tetra(:,4));
        econTetra3 = [econTetra3;econTetra2];
        end
        econTetra1 =unique(econTetra3);
    end
    econTetra = unique(econTetra1);
end

%find all trias attached to p1, p2 and p3 - For use in ElementOnside.m
econTria = find(p1==tria(:,3) | p1==tria(:,4) | p1==tria(:,5)...
    | p2==tria(:,3) | p2==tria(:,4) | p2==tria(:,5)...
    | p3==tria(:,3) | p3==tria(:,4) | p3==tria(:,5));

econTriaTetra=[]; % to find all the trias connected to the tetras
for itr = 1:length(econTetra);
    id=econTetra(itr);
    econTriangle = find((tetra(id,1)==tria(:,3) & tetra(id,2)==tria(:,4) & tetra(id,3)==tria(:,5))...
        | (tetra(id,1)==tria(:,3) & tetra(id,2)==tria(:,4) & tetra(id,4)==tria(:,5))...
        | (tetra(id,1)==tria(:,3) & tetra(id,3)==tria(:,4) & tetra(id,2)==tria(:,5))...
```

```
            | (tetra(id,1)==tria(:,3) & tetra(id,3)==tria(:,4) & tetra(id,4)==tria
               (:,5))...
            | (tetra(id,1)==tria(:,3) & tetra(id,4)==tria(:,4) & tetra(id,2)==tria
               (:,5))...
            | (tetra(id,1)==tria(:,3) & tetra(id,4)==tria(:,4) & tetra(id,3)==tria
               (:,5))...
            | (tetra(id,2)==tria(:,3) & tetra(id,1)==tria(:,4) & tetra(id,3)==tria
               (:,5))...
            | (tetra(id,2)==tria(:,3) & tetra(id,1)==tria(:,4) & tetra(id,4)==tria
               (:,5))...
            | (tetra(id,2)==tria(:,3) & tetra(id,3)==tria(:,4) & tetra(id,1)==tria
               (:,5))...
            | (tetra(id,2)==tria(:,3) & tetra(id,3)==tria(:,4) & tetra(id,4)==tria
               (:,5))...
            | (tetra(id,2)==tria(:,3) & tetra(id,4)==tria(:,4) & tetra(id,1)==tria
               (:,5))...
            | (tetra(id,2)==tria(:,3) & tetra(id,4)==tria(:,4) & tetra(id,3)==tria
               (:,5))...
            | (tetra(id,3)==tria(:,3) & tetra(id,1)==tria(:,4) & tetra(id,2)==tria
               (:,5))...
            | (tetra(id,3)==tria(:,3) & tetra(id,1)==tria(:,4) & tetra(id,4)==tria
               (:,5))...
            | (tetra(id,3)==tria(:,3) & tetra(id,2)==tria(:,4) & tetra(id,1)==tria
               (:,5))...
            | (tetra(id,3)==tria(:,3) & tetra(id,2)==tria(:,4) & tetra(id,4)==tria
               (:,5))...
            | (tetra(id,3)==tria(:,3) & tetra(id,4)==tria(:,4) & tetra(id,1)==tria
               (:,5))...
            | (tetra(id,3)==tria(:,3) & tetra(id,4)==tria(:,4) & tetra(id,2)==tria
               (:,5))...
            | (tetra(id,4)==tria(:,3) & tetra(id,1)==tria(:,4) & tetra(id,2)==tria
               (:,5))...
            | (tetra(id,4)==tria(:,3) & tetra(id,1)==tria(:,4) & tetra(id,3)==tria
               (:,5))...
            | (tetra(id,4)==tria(:,3) & tetra(id,2)==tria(:,4) & tetra(id,1)==tria
               (:,5))...
            | (tetra(id,4)==tria(:,3) & tetra(id,2)==tria(:,4) & tetra(id,3)==tria
               (:,5))...
            | (tetra(id,4)==tria(:,3) & tetra(id,3)==tria(:,4) & tetra(id,1)==tria
               (:,5))...
```

```
        | (tetra(id,4)==tria(:,3) & tetra(id,3)==tria(:,4) & tetra(id,2)==tria
            (:,5)));
    econTriaTetra=[econTriaTetra,econTriangle'];
end
econTriaTetra = unique(econTriaTetra);
```

E.3 ForDataPts.m

```
% To fill equidistant points inside a triangle element
clear Allpoints DataPts PtsOnLine usednodes
DataPts = [];
%NumberOfNodes=8; % divides the edge of the triangle into these many parts and
    should never be set to zero
maxdist = 0;

for ectr = 1:size(econTriaTetra,2);
    if NumberOfNodes≥2,
        node1 = [nd(tria(econTriaTetra(1,ectr),3),2) nd(tria(econTriaTetra(1,ectr
            ),3),3) nd(tria(econTriaTetra(1,ectr),3),4)];
        node2 = [nd(tria(econTriaTetra(1,ectr),4),2) nd(tria(econTriaTetra(1,ectr
            ),4),3) nd(tria(econTriaTetra(1,ectr),4),4)];
        node3 = [nd(tria(econTriaTetra(1,ectr),5),2) nd(tria(econTriaTetra(1,ectr
            ),5),3) nd(tria(econTriaTetra(1,ectr),5),4)];

        edge = [1 sqrt((node2(1)-node1(1))^2 + (node2(2)-node1(2))^2 + (node2(3)-
            node1(3))^2); % edge1 b/n node1 & 2
            2 sqrt((node3(1)-node2(1))^2 + (node3(2)-node2(2))^2 + (node3(3)-
                node2(3))^2); % edge2 b/n node2 & 3
            3 sqrt((node3(1)-node1(1))^2 + (node3(2)-node1(2))^2 + (node3(3)-
                node1(3))^2)];% edge3 b/n node3 & 1

        maxdist = max(maxdist,max(edge(:,2)./NumberOfNodes));
        srtedge = sortrows(edge,2);

        %     le = find(max(edge)==edge(:,1));
        %     se = find(min(edge)==edge(:,1));
        if srtedge(3,1)==1 && srtedge(1,1)==2, %le ==1 && se == 2,
            sle = 3; lend = [node1;node2]; slend = [node1;node3]; send = [node3;
                node2]; dend = [node3;node1];
        elseif srtedge(3,1)==2 && srtedge(1,1)==3, %le ==2 && se == 3,
            sle = 1; lend = [node2;node3]; slend = [node2;node1]; send = [node3;node1]; dend = [node3;node2];
        elseif srtedge(3,1)==3 && srtedge(1,1)==2, %le ==3 && se == 2,
            sle = 1; lend = [node1;node3]; slend = [node1;node2]; send = [node2;node3]; dend = [node2;node1];
        elseif srtedge(3,1)==2 && srtedge(1,1)==1, %le ==2 && se == 1,
            sle = 3; lend = [node3;node2]; slend = [node3;node1]; send = [node1;node2]; dend = [node1;node3];
        elseif srtedge(3,1)==1 && srtedge(1,1)==3, %le ==1 && se == 3,
            sle = 2; lend = [node2;node1]; slend = [node2;node3]; send = [node3;node1]; dend = [node3;node2];
        elseif srtedge(3,1)==3 && srtedge(1,1)==1, %le ==3 && se == 1,
            sle = 2; lend = [node3;node1]; slend = [node3;node2]; send = [node1;node2]; dend = [node1;node3];
        end

        for div1 = 1:NumberOfNodes-1,
            ka = div1; kb = NumberOfNodes-ka;
            LEsplit(div1,:) = [(ka*lend(2,1)+kb*lend(1,1))/(ka+kb) (ka*lend(2,2)+kb*lend(1,2))/(ka+kb) (ka*lend(2,3)+kb*lend(1,3))/(ka+kb)];
            SLEsplit(div1,:) = [(ka*slend(2,1)+kb*slend(1,1))/(ka+kb) (ka*slend(2,2)+kb*slend(1,2))/(ka+kb) (ka*slend(2,3)+kb*slend(1,3))/(ka+kb)];
            SEsplit(div1,:) = [(ka*send(2,1)+kb*send(1,1))/(ka+kb) (ka*send(2,2)+kb*send(1,2))/(ka+kb) (ka*send(2,3)+kb*send(1,3))/(ka+kb)];
            DEsplit(div1,:) = [(ka*dend(2,1)+kb*dend(1,1))/(ka+kb) (ka*dend(2,2)+kb*dend(1,2))/(ka+kb) (ka*dend(2,3)+kb*dend(1,3))/(ka+kb)];
        end

        inrncount = 0;
        if size(LEsplit,1)≥2,
            for div2 = 2:size(LEsplit,1),
                for div3 = 1:(div2-1),
                    inrncount = inrncount+1;
                    kaa = div3; kbb = div2-div3;
                    inrnodes(inrncount,:) = [(kaa*LEsplit(div2,1)+kbb*SLEsplit(div2,1))/(kaa+kbb) (kaa*LEsplit(div2,2)+kbb*SLEsplit(div2,2))/(kaa+kbb) (kaa*LEsplit(div2,3)+kbb*SLEsplit(div2,3))/(kaa+kbb)];
                end
            end
        elseif size(LEsplit,1)==1,
            tripoints = [node1; node2 ; node3];
            tricentroid= [sum(tripoints(:,1))/3 sum(tripoints(:,2))/3 sum(
                tripoints(:,3))/3];
            inrnodes = tricentroid;
        else
            inrnodes =[];
        end

        inrncount2 = 0;
        if size(LEsplit,1)≥2,
            for div4 = 1:size(LEsplit),
                for div5 = 1:(div4*2)-1,
                    kaa1 = div5; kbb1 = (div4*2)-1-div5+1;
                    inrncount2 = inrncount2+1;
                    inrnodes1(inrncount2,:) = [(kaa1*SEsplit(div4,1)+kbb1*DEsplit
                        (div4,1))/(kaa1+kbb1) (kaa1*SEsplit(div4,2)+kbb1*DEsplit(
                        div4,2))/(kaa1+kbb1) (kaa1*SEsplit(div4,3)+kbb1*DEsplit(
                        div4,3))/(kaa1+kbb1)];
                end
            end
        elseif size(LEsplit,1)==1,
            inrnodes1=[];
        end
        DataPts = [DataPts;inrnodes;inrnodes1];
    else
        node1 = [nd(tria(econTriaTetra(1,ectr),3),2) nd(tria(econTriaTetra(1,ectr
            ),3),3) nd(tria(econTriaTetra(1,ectr),3),4)];
        node2 = [nd(tria(econTriaTetra(1,ectr),4),2) nd(tria(econTriaTetra(1,ectr
            ),4),3) nd(tria(econTriaTetra(1,ectr),4),4)];
        node3 = [nd(tria(econTriaTetra(1,ectr),5),2) nd(tria(econTriaTetra(1,ectr
            ),5),3) nd(tria(econTriaTetra(1,ectr),5),4)];
        DataPts = [DataPts;node1;node2;node3];
    end
end

% Now create points along each edge of the tria elements stored in
% econTriaTetra

if NumberOfNodes≥2,
```

```matlab
    usednodes=[]; PtsOnLine=[]; DataPtsEdges=[];
    for jmj = 1:size(econTriaTetra,2),
        lnxyz = [tria((econTriaTetra(1,jmj)),3)  tria((econTriaTetra(1,jmj)),4)  tria((econTriaTetra(1,jmj)),5);
            tria((econTriaTetra(1,jmj)),4)  tria((econTriaTetra(1,jmj)),5)  tria((econTriaTetra(1,jmj)),3)];
        for jyj = 1:size(usednodes,1),
            if lnxyz(1,1) == usednodes(jyj,1) && lnxyz(2,1) == usednodes(jyj,2) || lnxyz(2,1) == usednodes(jyj,1) && lnxyz(1,1) == usednodes(jyj,2)
                lnxyz(1,1)=0 ; lnxyz(2,1)=0;
            elseif lnxyz(1,2) == usednodes(jyj,1) && lnxyz(2,2) == usednodes(jyj,2) || lnxyz(2,2) == usednodes(jyj,1) && lnxyz(1,2) == usednodes(jyj,2)
                lnxyz(1,2)=0 ; lnxyz(2,2)=0;
            elseif lnxyz(1,3) == usednodes(jyj,1) && lnxyz(2,3) == usednodes(jyj,2) || lnxyz(2,3) == usednodes(jyj,1) && lnxyz(1,3) == usednodes(jyj,2)
                lnxyz(1,3)=0 ; lnxyz(2,3)=0;
            end
        end
        [rrow,ccol]=find(lnxyz==0);
        lnxyz(:,ccol)=[];
        lnxy=lnxyz;
        PtsOnLine=[];
        for jxj = 1:size(lnxy,2),
            lseg = [nd(lnxy(1,jxj),2) nd(lnxy(1,jxj),3) nd(lnxy(1,jxj),4);
                nd(lnxy(2,jxj),2) nd(lnxy(2,jxj),3) nd(lnxy(2,jxj),4)];
            for div1 = 1:NumberOfNodes-1,
                ka = div1; kb = NumberOfNodes-ka;
                LINEsplit(div1,:) = [(ka*lseg(2,1)+kb*lseg(1,1))/(ka+kb) (ka*lseg(2,2)+kb*lseg(1,2))/(ka+kb) (ka*lseg(2,3)+kb*lseg(1,3))/(ka+kb)];
            end
            PtsOnLine= [PtsOnLine; LINEsplit];
            usednodes = [usednodes; lnxy(1,jxj) lnxy(2,jxj)];
        end
        DataPts = [DataPts;PtsOnLine];
    end
end

```

```
% Now add the coordinates of all the nodes that are not common to the
% elements in econTriaTetra
if NumberOfNodes≥2;
    ndlist=[usednodes(:,1);usednodes(:,2)];
    ndlist=unique(ndlist);
    for jfj = 1:size(ndlist,1)
        DataPts=[DataPts; nd(ndlist(jfj),2) nd(ndlist(jfj),3) nd(ndlist(jfj),4)];
    end
end
```

E.4 PointsForSphere.m

```
% To fill equidistant points inside a triangle element
% clear DataPts2 INRnodes INRnodes1
clear INRnodes INRnodes1
DataPts2=[];

lnode1 = [tripts(1,1) tripts(1,2) tripts(1,3)];
lnode2 = [tripts(2,1) tripts(2,2) tripts(2,3)];
lnode3 = [tripts(3,1) tripts(3,2) tripts(3,3)];

if numnodes≥2,

    edge2 = [1 sqrt((lnode2(1)-lnode1(1))^2 + (lnode2(2)-lnode1(2))^2 + (lnode2
        (3)-lnode1(3))^2); % edge1 b/n node1 & 2
        2 sqrt((lnode3(1)-lnode2(1))^2 + (lnode3(2)-lnode2(2))^2 + (lnode3(3)-
            lnode2(3))^2); % edge2 b/n node2 & 3
        3 sqrt((lnode3(1)-lnode1(1))^2 + (lnode3(2)-lnode1(2))^2 + (lnode3(3)-
            lnode1(3))^2)];% edge3 b/n node3 & 1

    SRTedge = sortrows(edge2,2);

    if SRTedge(3,1)==1 && SRTedge(1,1)==2, %le ==1 && se == 2,
        sle2 = 3; lend2 = [lnode1;lnode2]; slend2 = [lnode1;lnode3]; send2 = [
            lnode3;lnode2]; dend2 = [lnode3;lnode1];
    elseif SRTedge(3,1)==2 && SRTedge(1,1)==3, %le ==2 && se == 3,
        sle2 = 1; lend2 = [lnode2;lnode3]; slend2 = [lnode2;lnode1]; send2 = [
            lnode3;lnode1]; dend2 = [lnode3;lnode2];
    elseif SRTedge(3,1)==3 && SRTedge(1,1)==2, %le ==3 && se == 2,
        sle2 = 1; lend2 = [lnode1;lnode3]; slend2 = [lnode1;lnode2]; send2 = [
            lnode2;lnode3]; dend2 = [lnode2;lnode1];
    elseif SRTedge(3,1)==2 && SRTedge(1,1)==1, %le ==2 && se == 1,
        sle2 = 3; lend2 = [lnode3;lnode2]; slend2 = [lnode3;lnode1]; send2 = [
            lnode1;lnode2]; dend2 = [lnode1;lnode3];
    elseif SRTedge(3,1)==1 && SRTedge(1,1)==3, %le ==1 && se == 3,
        sle2 = 2; lend2 = [lnode2;lnode1]; slend2 = [lnode2;lnode3]; send2 = [
            lnode3;lnode1]; dend2 = [lnode3;lnode2];
    elseif SRTedge(3,1)==3 && SRTedge(1,1)==1, %le ==3 && se == 1,
        sle2 = 2; lend2 = [lnode3;lnode1]; slend2 = [lnode3;lnode2]; send2 = [
            lnode1;lnode2]; dend2 = [lnode1;lnode3];
    end

    for div1a = 1:numnodes-1,
        ka2 = div1a; kb2 = numnodes-ka2;
        LEsplit2(div1a,:) = [(ka2*lend2(2,1)+kb2*lend2(1,1))/(ka2+kb2) (ka2*lend2
            (2,2)+kb2*lend2(1,2))/(ka2+kb2) (ka2*lend2(2,3)+kb2*lend2(1,3))/(ka2+
            kb2)];
        SLEsplit2(div1a,:) = [(ka2*slend2(2,1)+kb2*slend2(1,1))/(ka2+kb2) (ka2*
            slend2(2,2)+kb2*slend2(1,2))/(ka2+kb2) (ka2*slend2(2,3)+kb2*slend2
            (1,3))/(ka2+kb2)];
        SEsplit2(div1a,:) = [(ka2*send2(2,1)+kb2*send2(1,1))/(ka2+kb2) (ka2*send2
            (2,2)+kb2*send2(1,2))/(ka2+kb2) (ka2*send2(2,3)+kb2*send2(1,3))/(ka2+
            kb2)];
        DEsplit2(div1a,:) = [(ka2*dend2(2,1)+kb2*dend2(1,1))/(ka2+kb2) (ka2*dend2
            (2,2)+kb2*dend2(1,2))/(ka2+kb2) (ka2*dend2(2,3)+kb2*dend2(1,3))/(ka2+
            kb2)];
    end

    INRncount = 0;
    if size(LEsplit2,1)≥2,
        for div2a = 2:size(LEsplit2,1),
            for div3a = 1:(div2a-1),
                INRncount = INRncount+1;
                kaa2 = div3a; kbb2 = div2a-div3a;
                INRnodes(INRncount,:) = [(kaa2*LEsplit2(div2a,1)+kbb2*SLEsplit2(
                    div2a,1))/(kaa2+kbb2) (kaa2*LEsplit2(div2a,2)+kbb2*SLEsplit2(
                    div2a,2))/(kaa2+kbb2) (kaa2*LEsplit2(div2a,3)+kbb2*SLEsplit2(
                    div2a,3))/(kaa2+kbb2)];
            end
        end
    elseif size(LEsplit2,1)==1,
        tripoints2 = [tripts(1,1) tripts(1,2) tripts(1,3); tripts(2,1) tripts
            (2,2) tripts(2,3);tripts(3,1) tripts(3,2) tripts(3,3)];
        tricentroid2 = [sum(tripoints2(:,1))/3 sum(tripoints2(:,2))/3 sum(
            tripoints2(:,3))/3];
        INRnodes=tricentroid2;
    else
        INRnodes =[];
    end

    INRncount2 = 0;
    if size(LEsplit2,1)≥2,
        for div4a = 1:size(LEsplit2),
            for div5a = 1:(div4a*2)-1,
                kaa2a = div5a; kbb2a = (div4a*2)-1-div5a+1;
                INRncount2 = INRncount2+1;
                INRnodes1(INRncount2,:) = [(kaa2a*SEsplit2(div4a,1)+kbb2a*
                    DEsplit2(div4a,1))/(kaa2a+kbb2a) (kaa2a*SEsplit2(div4a,2)+
                    kbb2a*DEsplit2(div4a,2))/(kaa2a+kbb2a) (kaa2a*SEsplit2(div4a
                    ,3)+kbb2a*DEsplit2(div4a,3))/(kaa2a+kbb2a)];
            end
        end
    elseif size(LEsplit2,1)==1,
        INRnodes1=[];
    end
    %DataPts2 = [DataPts2;lnode1;lnode2;lnode3;LEsplit2;SLEsplit2;SEsplit2;
        INRnodes;INRnodes1];
    DataPts2 = [DataPts2;INRnodes;INRnodes1];
else
        tripoints2 = [tripts(1,1) tripts(1,2) tripts(1,3); tripts(2,1) tripts
            (2,2) tripts(2,3);tripts(3,1) tripts(3,2) tripts(3,3)];
        tricentroid2 = [sum(tripoints2(:,1))/3 sum(tripoints2(:,2))/3 sum(
            tripoints2(:,3))/3];
        DataPts2=tricentroid2;

end

%RUN THE FINDVERTEXNODES.M CODE HERE TO POPULATE THE DATAPTS2 MATRIX
FindConvexNodes;

```

```
DataPts2 = [DataPts2;DataPts4];


% fidpoInTriangle = fopen('PointsInMasterTriangle.cmf','w');
for ppts = 1:size(DataPts2,1),
    fprintf(fiddtp2,'*createpoint(%8.3f,%8.3f,%8.3f,0)\n',DataPts2(ppts,1),
        DataPts2(ppts,2),DataPts2(ppts,3));
end
%fclose(fidpoInTriangle);
```

E.5 FindConvexNodes.m

```
ElemNorm1 = []; ElemNorm2=[] ; ElemNorm3=[]; ConcaveNodeInfo=[]; UsedNodes=[];
CornerVertex = []; ConvexEdgeVertex = []; ConcaveEdgeVertex=[]; DataPts4=[];
%Pick the tria to be processed
pts = [tria(findmedialpt,3) tria(findmedialpt,4) tria(findmedialpt,5)];
EDGE = [pts(1) pts(2); pts(2) pts(3); pts(3) pts(1)];
for CVitr = 1:3,
    DataPts3=[];
    % find the edge of another element that contains nodes1 and 2 from the tria
        file
    edge = find((EDGE(CVitr,1)==tria(:,3)|EDGE(CVitr,1)==tria(:,4)|EDGE(CVitr,1)
        ==tria(:,5)) & (EDGE(CVitr,2)==tria(:,3)|EDGE(CVitr,2)==tria(:,4)|EDGE(
        CVitr,2)==tria(:,5)));

    lendax=[nd(EDGE(CVitr,1),2) nd(EDGE(CVitr,1),3) nd(EDGE(CVitr,1),4);
        nd(EDGE(CVitr,2),2) nd(EDGE(CVitr,2),3) nd(EDGE(CVitr,2),4)];
    if numnodes>1,
        for div1ax = 1:numnodes-1,
            ka12 = div1ax; kb12 = numnodes-ka12;
            edgesplit(div1ax,:) = [(ka12*lendax(2,1)+kb12*lendax(1,1))/(ka12+kb12
                ) (ka12*lendax(2,2)+kb12*lendax(1,2))/(ka12+kb12) (ka12*lendax
                (2,3)+kb12*lendax(1,3))/(ka12+kb12)];
        end
    else
        edgesplit = [(lendax(2,1)+lendax(1,1))/2 (lendax(2,2)+lendax(1,2))/2 (
            lendax(2,3)+lendax(1,3))/2];
    end

```

```
% find the angle the master element makes with this element
if size(edge,1)>1,
    if edge(1)==findmedialpt, SlaveElement = edge(2); else SlaveElement = edge(1); end
    % if the angle is greater than AngleTolerance, then go ahead
    P1 = [nd(tria(SlaveElement,3),2)  nd(tria(SlaveElement,3),3) nd(tria(SlaveElement,3),4)];
    P2 = [nd(tria(SlaveElement,4),2)  nd(tria(SlaveElement,4),3) nd(tria(SlaveElement,4),4)];
    P3 = [nd(tria(SlaveElement,5),2)  nd(tria(SlaveElement,5),3) nd(tria(SlaveElement,5),4)];

    SlaveNormal = cross(P1-P2,P1-P3); % normal of the elements connected to the tri element
    a2 = SlaveNormal(1);
    b2 = SlaveNormal(2);
    c2 = SlaveNormal(3);

    ndot = dot(normal,SlaveNormal);
    theta =ndot/(sqrt(a^2+b^2+c^2)*sqrt(a2^2+b2^2+c2^2));
    angle = 180-((acos(theta))*(180/pi));

    lendax=[nd(EDGE(CVitr,1),2) nd(EDGE(CVitr,1),3) nd(EDGE(CVitr,1),4);
        nd(EDGE(CVitr,2),2) nd(EDGE(CVitr,2),3) nd(EDGE(CVitr,2),4)];
    for div1ax = 1:numnodes-1,
        ka12 = div1ax; kb12 = numnodes-ka12;
        edgesplit(div1ax,:) = [(ka12*lendax(2,1)+kb12*lendax(1,1))/(ka12+kb12) (ka12*lendax(2,2)+kb12*lendax(1,2))/(ka12+kb12) (ka12*lendax(2,3)+kb12*lendax(1,3))/(ka12+kb12)];
    end

    if angle == 0 || angle ≤ AngleTolerance || (180-angle) ≤ AngleTolerance,
        if numnodes>1 ,
            DataPts3=[edgesplit; lendax(1,:);lendax(2,:)];
        else
            DataPts3=[lendax(1,:);lendax(2,:)];
        end
    else
        % Now find the intersection point
        %centroid of the master element
        cme = [sum(tripts(:,1))/3 sum(tripts(:,2))/3 sum(tripts(:,3))/3];
        % node coordinates of the slave element stored in "elmcomedge"
        triptsSlvElm = [P1 ; P2 ; P3];
        %centroid of the slave element
        cse = [sum(triptsSlvElm(:,1))/3 sum(triptsSlvElm(:,2))/3 sum(triptsSlvElm(:,3))/3];
        %midpoint between the two centroids
        cmid = [(cme(1)+cse(1))/2 (cme(2)+cse(2))/2 (cme(3)+cse(3))/2];
        % normal of the master element
        Vec1=normal;
        %normal vector of the plane formed by the slave triangle is defined by
        Vec2 = [a2 b2 c2];
        axy = (cross((cse-cme),Vec2))/(cross(Vec1,Vec2));
        intersectingPoint = cme + axy*Vec1;
        testResult=TetrahedronTest(intersectingPoint,econTetra,tetra,nd) ;
        cmidResult=TetrahedronTest(cmid,econTetra,tetra,nd) ;
        if testResult==1 || cmidResult == 1,
            if numnodes==1, ConvexEdgeVertex = [ConvexEdgeVertex; EDGE(CVitr,1) lendax(1,:); EDGE(CVitr,1) lendax(2,:)];
            else
                edgesplit1=[];
                for edgs = 1:size(edgesplit),
                    edgesplit1=[edgesplit1; EDGE(CVitr,1) edgesplit(edgs,:)];
                end
                ConvexEdgeVertex = [ConvexEdgeVertex; edgesplit1; EDGE(CVitr,1) lendax(1,:); EDGE(CVitr,1) lendax(2,:)];
            end
        else
            DataToStoreConcavePointsMod4;
                edgesplit1=[];
                for edgs = 1:size(edgesplit),
                    edgesplit1=[edgesplit1; EDGE(CVitr,1) edgesplit(edgs,:)];
                end
                ConcaveEdgeVertex = [ConcaveEdgeVertex; edgesplit1; EDGE(CVitr,1) lendax(1,:);EDGE(CVitr,1) lendax(2,:) ];
                ConcaveNodeInfo1 = [];
                TmpC = [edgesplit;lendax];
                for tmpcve = 1:size(TmpC,1),
                    for vec = 1:size(VecMat,1),
                        ConcaveNodeInfo1 = [ConcaveNodeInfo1; p1 p2 VecMat(vec,1) TmpC(tmpcve,1) TmpC(tmpcve,2) TmpC(tmpcve
                                ,3) VecMat(vec,2) VecMat(vec,3) VecMat(vec,4) VecMat(vec,5) VecMat(vec,6) VecMat(vec,7) VecMat(vec,8)];
                        end
                    end
                    %ConcaveNodeInfo=[ConcaveNodeInfo;ConcaveNodeInfo1];
                %end
                ConcaveNodeInfo=[ConcaveNodeInfo;ConcaveNodeInfo1];
            end
        end
    else
        lendax=[nd(EDGE(CVitr,1),2) nd(EDGE(CVitr,1),3) nd(EDGE(CVitr,1),4);
            nd(EDGE(CVitr,2),2) nd(EDGE(CVitr,2),3) nd(EDGE(CVitr,2),4)];
        if numnodes>1 ,
            for div1ax = 1:numnodes-1,
                ka12 = div1ax; kb12 = numnodes-ka12;
                edgesplit(div1ax,:) = [(ka12*lendax(2,1)+kb12*lendax(1,1))/(ka12+kb12) (ka12*lendax(2,2)+kb12*lendax(1,2))/(ka12+kb12) (ka12*lendax(2,3)+kb12*lendax(1,3))/(ka12+kb12)];
            end
            DataPts3=[edgesplit; lendax(1,:);lendax(2,:)];
        else
            DataPts3=[lendax(1,:);lendax(2,:)];
        end
    end
         DataPts4 = [DataPts4 ;DataPts3];
end
% if the intersection point is inside, then go ahead and split the edge and store info into ConvexPoints else into Concave POints

% Do this for all three edges and all elements

for cvx = 1:size(ConvexEdgeVertex,1),
    fprintf(fidcvx,'%d %8.3f %8.3f %8.3f\n',ConvexEdgeVertex(cvx,1), ConvexEdgeVertex(cvx,2), ConvexEdgeVertex(cvx,3), ConvexEdgeVertex(cvx,4));
end

for cve = 1:size(ConcaveEdgeVertex,1),
    fprintf(fidcve,'%8.3f %8.3f %8.3f\n',ConcaveEdgeVertex(cve,2), ConcaveEdgeVertex(cve,3),ConcaveEdgeVertex(cve,4));
```

```
end

for cvetxt = 1:size(ConcaveNodeInfo,1),
    fprintf(fidcvetxt,'% d %d %d %8.3f %8.3f %8.3f %8.3f %8.3f %8.3f %8.3f %8.3f %8.3f %8.3f\n',ConcaveNodeInfo(cvetxt,:)); %ConcaveNodeInfo(cvetxt,2),...
    %         ConcaveNodeInfo(cvetxt,3), ConcaveNodeInfo(cvetxt,4), ConcaveNodeInfo(cvetxt,5), ConcaveNodeInfo(cvetxt,6),...
    %         ConcaveNodeInfo(cvetxt,7), ConcaveNodeInfo(cvetxt,8), ConcaveNodeInfo(cvetxt,9), ConcaveNodeInfo(cvetxt,10),...
    %         ConcaveNodeInfo(cvetxt,11), ConcaveNodeInfo(cvetxt,12), ConcaveNodeInfo(cvetxt,13));
end
```

E.6 TetrahedronTest.m

```
% To check if a Point is inside a set of Tetrahedron
function [testResult] = TetrahedronTest(x,econTetra,tetra,nd)
%stic
onTetfaceTol = 0.001;
testResult=[]; % '0' means outside, '1' means inside

tpt = x; % test point


for itt = 1:size(econTetra,1),
    vert1 = [nd(tetra(econTetra(itt),1),2) nd(tetra(econTetra(itt),1),3) nd(tetra(econTetra(itt),1),4)];
    vert2 = [nd(tetra(econTetra(itt),2),2) nd(tetra(econTetra(itt),2),3) nd(tetra(econTetra(itt),2),4)];
    vert3 = [nd(tetra(econTetra(itt),3),2) nd(tetra(econTetra(itt),3),3) nd(tetra(econTetra(itt),3),4)];
    vert4 = [nd(tetra(econTetra(itt),4),2) nd(tetra(econTetra(itt),4),3) nd(tetra(econTetra(itt),4),4)];

    D0 = det([vert1(1) vert1(2) vert1(3) 1; vert2(1) vert2(2) vert2(3) 1; vert3(1) vert3(2) vert3(3) 1; vert4(1) vert4(2) vert4(3) 1]);
    D1 = det([tpt(1) tpt(2) tpt(3) 1;       vert2(1) vert2(2) vert2(3) 1; vert3(1) vert3(2) vert3(3) 1; vert4(1) vert4(2) vert4(3) 1]);
    D2 = det([vert1(1) vert1(2) vert1(3) 1; tpt(1) tpt(2) tpt(3) 1;       vert3(1) vert3(2) vert3(3) 1; vert4(1) vert4(2) vert4(3) 1]);
    D3 = det([vert1(1) vert1(2) vert1(3) 1; vert2(1) vert2(2) vert2(3) 1; tpt(1) tpt(2) tpt(3) 1;       vert4(1) vert4(2) vert4(3) 1]);
    D4 = det([vert1(1) vert1(2) vert1(3) 1; vert2(1) vert2(2) vert2(3) 1; vert3(1) vert3(2) vert3(3) 1; tpt(1) tpt(2) tpt(3) 1]);

    D5 = D1+D2+D3+D4;
%
    if D0==0, % if D0==0, then this tetrahedron is degenerate
        testResult = 0;
    elseif D0<0 && D1<0 && D2<0 && D3<0 && D4≤onTetfaceTol, % if D0 is lesser than zero and any of the other "Di" is equal to zero then the point lies on this tetra's face
        testResult = 1;
        break
    elseif D0<0 && D1<0 && D2<0 && D3≤onTetfaceTol && D4<0,
        testResult = 1;
        break
    elseif D0<0 && D1<0 && D2≤onTetfaceTol && D3<0 && D4<0,
        testResult = 1;
        break
    elseif D0<0 && D1≤onTetfaceTol && D2<0 && D3<0 && D4<0,
        testResult = 1;
        break
    elseif D0>0 && D1≤onTetfaceTol && D2>0 && D3>0 && D4>0, % if D0 is greater than zero and any of the other "Di" is equal to zero then the point lies on this tetra's face
        testResult = 1;
        break
    elseif D0>0 && D1>0 && D2≤onTetfaceTol && D3>0 && D4>0,
        testResult = 1;
        break
    elseif D0>0 && D1>0 && D2>0 && D3≤onTetfaceTol && D4>0,
        testResult = 1;
        break
    elseif D0>0 && D1>0 && D2>0 && D3>0 && D4≤onTetfaceTol,
        testResult = 1;
        break
    elseif D0>0 && D1>0 && D2>0 && D3>0 && D4>0, % if D0 is greater than zero and all "Di" are the same sign as D0, then point lies inside tetra
```

```
        testResult = 1;
        break
    elseif D0<0 && D1<0 && D2<0 && D3<0 && D4<0, % if D0 is lesser than zero and
        all "Di" are the same sign as D0, then point lies inside tetra
        testResult = 1;
        break
    else
        testResult=0;
    end
end
%toc
```

E.7 DataToStoreConcavePoints.m

```
clear VecMat
MasterElemID = findmedialpt;
SecondElemID = SlaveElement;
EdgeMedian = [sum(lendax(:,1))/2 sum(lendax(:,2))/2 sum(lendax(:,3))/2];
k1 = 1; k2 = 2;
New4thPtx = ((k1*intersectingPoint(1)) - (k2*EdgeMedian(1)))/(k1-k2); % xp = (k1*
    xb-k2*xa)/(k1-k2) - always use xa as the point on the boundary and not the
    intersecting point for this workflow
New4thPty = ((k1*intersectingPoint(2)) - (k2*EdgeMedian(2)))/(k1-k2);
New4thPtz = ((k1*intersectingPoint(3)) - (k2*EdgeMedian(3)))/(k1-k2);
New4thPt = [New4thPtx New4thPty New4thPtz];
NewNormal = [EdgeMedian(1)-New4thPt(1), EdgeMedian(2)-New4thPt(2), EdgeMedian(3)-
    New4thPt(3)];
MstrElm4thPt = fourthpt;
SlvElm4thPt1 = Find4thNodeMAsurf(tria(SecondElemID,3), tria(SecondElemID,4), tria
    (SecondElemID,5), tetra);
SlvElm4thPt = [nd(SlvElm4thPt1,2) nd(SlvElm4thPt1,3) nd(SlvElm4thPt1,4)];
Mstrnewd = newd;
Slvnewd = (Vec2(1)*(-SlvElm4thPt(1)))+ (Vec2(2)*(-SlvElm4thPt(2)))+(Vec2(3)*(-
    SlvElm4thPt(3)));
MedianNewd = (NewNormal(1)*(-New4thPt(1)))+ (NewNormal(2)*(-New4thPt(2)))+(
    NewNormal(3)*(-New4thPt(3)));

% 2 points between intersecting point and the centroid
kza1=1; kzb2=2; CMEsplit=[]; CSEsplit=[];
```

```
CMEinter = [cme;intersectingPoint];
CSEinter = [cse;intersectingPoint];
    for kzz = 1:2,
        kza1 = kzz; kzb2 = 3-kzz;
        CMEsplit = [CMEsplit;(kza1*CMEinter(2,1)+kzb2*CMEinter(1,1))/(kza1+kzb2)
            (kza1*CMEinter(2,2)+kzb2*CMEinter(1,2))/(kza1+kzb2) (kza1*CMEinter
            (2,3)+kzb2*CMEinter(1,3))/(kza1+kzb2)];
        CSEsplit = [CSEsplit;(kza1*CSEinter(2,1)+kzb2*CSEinter(1,1))/(kza1+kzb2)
            (kza1*CSEinter(2,2)+kzb2*CSEinter(1,2))/(kza1+kzb2) (kza1*CSEinter
            (2,3)+kzb2*CSEinter(1,3))/(kza1+kzb2)];
    end


interCME = [(intersectingPoint(1)+cme(1))/2 (intersectingPoint(2)+cme(2))/2 (
    intersectingPoint(3)+cme(3))/2];
interCSE = [(intersectingPoint(1)+cse(1))/2 (intersectingPoint(2)+cse(2))/2 (
    intersectingPoint(3)+cse(3))/2];

interCME4thPtA = [((k1*interCME(1)) - (k2*EdgeMedian(1)))/(k1-k2) ((k1*interCME
    (2)) - (k2*EdgeMedian(2)))/(k1-k2) ((k1*interCME(3)) - (k2*EdgeMedian(3)))/(
    k1-k2)];
interCSE4thPtA = [((k1*interCSE(1)) - (k2*EdgeMedian(1)))/(k1-k2) ((k1*interCSE
    (2)) - (k2*EdgeMedian(2)))/(k1-k2) ((k1*interCSE(3)) - (k2*EdgeMedian(3)))/(
    k1-k2)];

interCMEnormal4A = [EdgeMedian(1)-interCME4thPtA(1), EdgeMedian(2)-interCME4thPtA
    (2), EdgeMedian(3)-interCME4thPtA(3)];
interCSEnormal4A = [EdgeMedian(1)-interCSE4thPtA(1), EdgeMedian(2)-interCSE4thPtA
    (2), EdgeMedian(3)-interCSE4thPtA(3)];

interCMEnewd = (interCMEnormal4A(1)*(-interCME4thPtA(1)))+ (interCMEnormal4A(2)*(
    -interCME4thPtA(2)))+(interCMEnormal4A(3)*(-interCME4thPtA(3)));
interCSEnewd = (interCSEnormal4A(1)*(-interCSE4thPtA(1)))+ (interCSEnormal4A(2)*(
    -interCSE4thPtA(2)))+(interCSEnormal4A(3)*(-interCSE4thPtA(3)));

interCME4thPt = [((k1*CMEsplit(1,1)) - (k2*EdgeMedian(1)))/(k1-k2) ((k1*CMEsplit
    (1,2)) - (k2*EdgeMedian(2)))/(k1-k2) ((k1*CMEsplit(1,3)) - (k2*EdgeMedian(3))
    )/(k1-k2)];
interCME5thPt = [((k1*CMEsplit(2,1)) - (k2*EdgeMedian(1)))/(k1-k2) ((k1*CMEsplit
    (2,2)) - (k2*EdgeMedian(2)))/(k1-k2) ((k1*CMEsplit(2,3)) - (k2*EdgeMedian(3))
    )/(k1-k2)];
```

```
interCSE6thPt = [((k1*CSEsplit(1,1)) - (k2*EdgeMedian(1)))/(k1-k2) ((k1*CSEsplit
    (1,2)) - (k2*EdgeMedian(2)))/(k1-k2) ((k1*CSEsplit(1,3)) - (k2*EdgeMedian(3))
    )/(k1-k2)];
interCSE7thPt = [((k1*CSEsplit(2,1)) - (k2*EdgeMedian(1)))/(k1-k2) ((k1*CSEsplit
    (2,2)) - (k2*EdgeMedian(2)))/(k1-k2) ((k1*CSEsplit(2,3)) - (k2*EdgeMedian(3))
    )/(k1-k2)];

interCMEnormal4 = [EdgeMedian(1)-interCME4thPt(1), EdgeMedian(2)-interCME4thPt(2)
    , EdgeMedian(3)-interCME4thPt(3)];
interCMEnormal5 = [EdgeMedian(1)-interCME5thPt(1), EdgeMedian(2)-interCME5thPt(2)
    , EdgeMedian(3)-interCME5thPt(3)];
interCSEnormal6 = [EdgeMedian(1)-interCSE6thPt(1), EdgeMedian(2)-interCSE6thPt(2)
    , EdgeMedian(3)-interCSE6thPt(3)];
interCSEnormal7 = [EdgeMedian(1)-interCSE7thPt(1), EdgeMedian(2)-interCSE7thPt(2)
    , EdgeMedian(3)-interCSE7thPt(3)];

interCMEnewd4 = (interCMEnormal4(1)*(-interCME4thPt(1)))+ (interCMEnormal4(2)*(-
    interCME4thPt(2)))+(interCMEnormal4(3)*(-interCME4thPt(3)));
interCMEnewd5 = (interCMEnormal5(1)*(-interCME5thPt(1)))+ (interCMEnormal5(2)*(-
    interCME5thPt(2)))+(interCMEnormal5(3)*(-interCME5thPt(3)));
interCSEnewd6 = (interCSEnormal6(1)*(-interCSE6thPt(1)))+ (interCSEnormal6(2)*(-
    interCSE6thPt(2)))+(interCSEnormal6(3)*(-interCSE6thPt(3)));
interCSEnewd7 = (interCSEnormal7(1)*(-interCSE7thPt(1)))+ (interCSEnormal7(2)*(-
    interCSE7thPt(2)))+(interCSEnormal7(3)*(-interCSE7thPt(3)));



if VectorsAlongDihedralAngle == 7,
    VecMat = [MasterElemID, NewNormal, MedianNewd, New4thPt;
    MasterElemID, Vec1, Mstrnewd, MstrElm4thPt;
    MasterElemID, Vec2, Slvnewd, SlvElm4thPt;
    MasterElemID,interCMEnormal4,interCMEnewd4,interCME4thPt;
    MasterElemID,interCMEnormal5,interCMEnewd5,interCME5thPt;
    MasterElemID,interCSEnormal6,interCSEnewd6,interCSE6thPt;
    MasterElemID,interCSEnormal7,interCSEnewd7,interCSE7thPt];

elseif VectorsAlongDihedralAngle == 5,
VecMat = [MasterElemID, NewNormal, MedianNewd, New4thPt;
    MasterElemID, Vec1, Mstrnewd, MstrElm4thPt;
    MasterElemID, Vec2, Slvnewd, SlvElm4thPt;
    MasterElemID, interCMEnormal4A,interCMEnewd,interCME4thPtA;
    MasterElemID, interCSEnormal4A,interCSEnewd,interCSE4thPtA];
else
    VecMat = [MasterElemID, NewNormal, MedianNewd, New4thPt;
    MasterElemID, Vec1, Mstrnewd, MstrElm4thPt;
    MasterElemID, Vec2, Slvnewd, SlvElm4thPt];
end
```

E.8 DihedralSpheres.m

```

fidcenters2 = fopen('OptimumSpheres-DihedralSpheres.cmf','a');
fidpft2 = fopen('PointsForTriangulation-DihedralSpheres.txt','a');
fidpftcmf2 = fopen('PointsForTriangulation-DihedralSpheres.cmf','a');

nd = dlmread('RMSSfull-nodeData.txt'); tria = dlmread('RMSSfull-NoEdgesSurfData.
    txt'); tetra = dlmread('RMSSfull-tetraData.txt'); %tria2=dlmread('RMSSfull-
    RunOnlyTheseSurfData.txt');
OptimumSphere2=[]; VertexPoint=[]; elem=[]; itercount=0; etimes=[]; PointsUsed
    =[];
MaxRadius = 5.0;
elemsAcrossThickness = 2; % Number of tetra along the thickest section of the
    geometry
NumberOfNodes=11; % Number of parts each edge of the triangle element will be
    split into for DataPts (never to be set less than 1)
numnodes=1; % Number of elements inside a single triangle element for processing
    spheres (never to be set less than 2) and should be set to atleast 2 for
    coarse meshes
AngleTolerance = 15; % For use in FindConvexNodes.m
VectorsAlongDihedralAngle = 7; % Should be set to 5 or 3 - For use in
    DataToStoreConcavePointsMod3.m - '5 & 3' includes the master and slave
    vectors

cleanfileNew;
disp('Cleaned Concave Nodes file');

v2=CNIsph3;
elemsAcrossThickness = elemsAcrossThickness+1;
runcount=0;

```

```
for findmedialpt2 = 1:size(v2,1), % v2 is the cleaned form of the input v2 loaded
     into matlab
    t2=clock;
    disp(['Total concave iterations : ', num2str(size(v2,1))]);
    disp(['Current iteration is : ', num2str(findmedialpt2)]);
    currentpt=v2(findmedialpt2,4:6);
    fourthpt =v2(findmedialpt2,11:13);
    UD.currentpt=currentpt;
    UD.a=v2(findmedialpt2,7);
    UD.b=v2(findmedialpt2,8);
    UD.c=v2(findmedialpt2,9);
    midpt=currentpt+(-currentpt+fourthpt)./8;
    newd0 = (UD.a*(-fourthpt(1,1)))+ (UD.b*(-fourthpt(1,2)))+(UD.c*(-fourthpt
        (1,3)));
    %midpt=currentpt;
    pointsInsideSphere=[];

    p1 = tria(v2(findmedialpt2,3),3); p2 = tria(v2(findmedialpt2,3),4); p3 = tria
        (v2(findmedialpt2,3),5);
    p4 = Find4thNodeMAsurf(p1,p2,p3,tetra);

    runcount=runcount+1;
    if runcount==1,
        disp('********** Finding Data Points *************');
    ConnectedElements;
    ForDataPts;
    elseif VectorsAlongDihedralAngle==7 && runcount==7,
        runcount=0;
    elseif VectorsAlongDihedralAngle==5 && runcount==5,
        runcount=0;
    elseif VectorsAlongDihedralAngle==3 && runcount==3,
        runcount=0;
    end

    newd = (UD.a*(-midpt(1,1)))+ (UD.b*(-midpt(1,2)))+(UD.c*(-midpt(1,3)));
    newd = max(newd,newd+sign(newd0)*0.1*maxdist);

    UD.DataPts=DataPts;
    UD.maxdist = maxdist;
    UD.tetra=tetra;
    UD.econTetra=econTetra;
    UD.nd=nd;
    try
        %keyboard
        t1=clock;
        x=linsearch(newd,UD);
        eti=etime(clock,t1);
        %disp(['linsearch finished in ', num2str(eti)]);

    catch
        disp(lasterr);
        %keyboard
    end;
    [npts,pts,centerpt,sphereRadius]=checksphere(x,UD);

    %centertest = inhull(centerpt,DataPts);
    centertest = TetrahedronTest(centerpt,econTetra,tetra,nd);
    [centerpt(1) centerpt(2) centerpt(3) sphereRadius centertest];
    if centertest == 1,% && sphereRadius>0.05% && sphereRadius<10
        OptimumSphere2 = [OptimumSphere2;centerpt(1) centerpt(2) centerpt(3)
            sphereRadius];
        fprintf(fidcenters2,'*createpoint(%8.3f,%8.3f,%8.3f,0)\n',OptimumSphere2(
            end,1),OptimumSphere2(end,2),OptimumSphere2(end,3));
        fprintf(fidcenters2,'*surfacemode(4)\n');
        fprintf(fidcenters2,'*solidspherefull(%8.3f,%8.3f,%8.3f,%8.3f)\n',
            OptimumSphere2(end,1),OptimumSphere2(end,2),OptimumSphere2(end,3),
            OptimumSphere2(end,4));
    end
            eti=etime(clock,t2);
        disp(['This iteration finished in : ', num2str(eti),'sec']);
end
```

E.9 linsearch.m

```
function [xbest,fbest]=linsearch(x0,USERDATA)
x=x0;
xbest=x;
[npts,pts,centerpt,sr]=checksphere(x,USERDATA);
f=abs(npts-1);
fbest=f;
```

```
inc=USERDATA.maxdist/1000;
x=x+inc;
trace=[];

while 1
    if abs(inc)<USERDATA.maxdist/10000,break;end;
    %keyboard
    fold=f;
    srold=sr;
    [npts,pts,centerpt,sr]=checksphere(x,USERDATA);
    f=npts-1;
    %tr=[x f fold fbest sr srold inc]
    if abs(f)≤fbest,
        fbest=abs(f);
        xbest=x;
    end;
    if f==fold&&sr≠0,
        if sr<srold,
            inc=-inc;
        end;
        inc=inc*1.2;
    end;
    if f>fold || sr==0,
        x=x-sign(x0)*2*inc;
        inc=inc/2;
    end;
    x=x+sign(x0)*inc;
end;
```

E.10 CheckSphere.m

```
function [npts,pts,centerpt,sr]=checksphere(newd,UD)
if ¬isfield(UD,'maxdist'),
    UD.maxdist = 1.5;
end;

centerpt = newcenter(UD.currentpt,UD.a,UD.b,UD.c,newd);

if ¬TetrahedronTest(centerpt,UD.econTetra,UD.tetra,UD.nd),
% tr=TetrahedronTest(centerpt,econTetra,tetra,nd) ;
% if tr==0,
    npts=1000;
    pts=[];
    sr=0;
    return
end;
sr = ecd(UD.currentpt,centerpt);
distcenter=ecd(centerpt,UD.DataPts);
pts=(find(distcenter≤sr));
dp=UD.DataPts(pts,:);
distcurrent=ecd(UD.currentpt,dp);
%np=find(distcurrent>0);
pts=pts(distcurrent>sr);
if ¬isempty(pts)&&length(pts)<100,
    dp=UD.DataPts(pts,:);
    rr=remove_duplicates_allcolumns(dp);
    pts=pts(rr);
end;
npts=length(pts);
```

E.11 ecd.m

```
function [d]=ecd(p1,p2)
d = sqrt((p1(1)-p2(:,1)).^2+(p1(2)-p2(:,2)).^2+(p1(3)-p2(:,3)).^2);
```

E.12 cleanfile.m

```
v1=load('ConcaveEdgePointsCleaned.txt');
v2=load('ConcaveNodeInfo.txt');
ivtol = 0.0001;
CNIsph1=[]; CNIsph3=[];

for ai = 1:size(v1,1),
    CNIsph1=[];
    for bi = 1:size(v2,1),
        if abs(v1(ai,1)-v2(bi,4))≤ivtol;
            if abs(v1(ai,2)-v2(bi,5))≤ivtol;
```

```
                if abs(v1(ai,3)-v2(bi,6))≤ivtol;
                    CNIsph1 = [CNIsph1;v2(bi,:)];
                end
            end
        end
    end
    if VectorsAlongDihedralAngle == 7,
        CNIsph1(8:end,:)=[];
        CNIsph3=[CNIsph3;CNIsph1];
    elseif VectorsAlongDihedralAngle == 5,
        CNIsph1(6:end,:)=[];
        %CNIsph3=[CNIsph3;CNIsph2];
        CNIsph3=[CNIsph3;CNIsph1];
    else
        CNIsph1(4:end,:)=[];
        CNIsph3=[CNIsph3;CNIsph1];
    end
end

if exist('ConcaveNodeInfo-Cleaned.txt','file'), delete('ConcaveNodeInfo-Cleaned.txt');end;
fidcni = fopen('ConcaveNodeInfo-Cleaned.txt','a');
for cni=1:size(CNIsph3,1),
    fprintf(fidcni,'% d %d %d %8.3f %8.3f %8.3f %8.3f %8.3f %8.3f %8.3f %8.3f %8.3f %8.3f\n',CNIsph3(cni,:));
end
fclose(fidcni);
```

E.13 MyCrustOpen.m

```
%% Run  program
[t]=MyCrustOpen(p);

fidtriangulate = fopen('Triangulation-TsecNB.cmf','w');
for ppp = 1:size(p,1)
    fprintf(fidtriangulate,'*createnode(%8.3f,%8.3f,%8.3f,0,0,0)\n',p(ppp,1),p(ppp,2),p(ppp,3));
end
for ttt = 1:size(t,1)
```

```
    fprintf(fidtriangulate,'*createlist(node,1) %d %d %d\n',t(ttt,1),t(ttt,2),t(
        ttt,3));
    fprintf(fidtriangulate,'*createelement(103,103,1,1)\n');
end
fclose(fidtriangulate);
```

Bibliography

[1] P. Boart, A. Andersson, and B.O. Elfström. Knowledge Enabled Pre-processing for structural analysis. In *1st Nordic Conference on Product Lifecycle Management-NordPLM*, volume 6, pages 25–26, 2006.

[2] L. Rajagopal. Rapid load path Design of Jet engine component using Knowledge Based Engineering. Master's thesis, 2005.

[3] C.B. Chapman and M. Pinfold. The application of a knowledge based engineering approach to the rapid design and analysis of an automotive structure. *Advances in engineering software*, 32(12):903–912, 2001.

[4] PACElab. `http://www.pace.de`.

[5] L.L. Phan and E. Garcia. Formulation and implementation of an Aircraft–System–Subsystem interrelationship model for technology evaluation. In *25th International Congress On The Aeronautical Sciences, ICAS2006*, volume 21, pages 35–36.

[6] Transcendata Europe. `http://www.transcendata.com/products/cadfix`.

[7] M.W. Beall, J. Walsh, and M.S. Shephard. Accessing CAD geometry for mesh generation. In *12th International Meshing Roundtable*, pages 31–42. Citeseer, 2003.

[8] IST World European research projects. `http://www.ist-world.org/ProjectDetails.aspx?ProjectId=4b91ca3fd36e4cd4939a894a6cea3e1b`. Last accessed on 21/10/2010.

[9] I.I. Voutchkov, A.J. Keane, and R. Fox. Robust structural design of a simplified jet engine model, using multiobjective optimization. *American Institute of Aeronautics and Astronautics*, 2006.

[10] VIVACE Project. `http://www.vivaceproject.com`, . Last accessed on 21/10/2010.

[11] C. Dye, J.B. Staubach, D. Emmerson, and C.G. Jensen. CAD-Based Parametric Cross-Section Designer for Gas Turbine Engine MDO Applications. *Computer-Aided Design & Applications*, 4(1-4):509–518, 2007.

[12] Deflection analysis of Aero Gas Turbine structure during prototype development. `www.mscsoftware.com/support/library/conf/auc97/p02197.pdf`. MSC Aerospace Users Conference, Last accessed on 21/10/2010.

[13] Carcass Deflection analysis of a Turbo Fan Engine. `http://www.mscsoftware.com/support/library/conf/wuc83/p02083.pdf`. MSC Aerospace Users Conference, Last accessed on 21/10/2010.

[14] J.A. DeCastro and K.J. Melcher. A study on the requirements for fast active turbine tip clearance control systems. In *Proceedings of the 40 th Joint Propulsion Conference and Exhibit, AIAA-2004-4176*, 2004.

[15] A.J. Keane and J.P. Scanlan. Design search optimization in aerospace engineering. *Philosophical transactions of the Royal Society*, 365:2501–2529, 2007.

[16] CATIA, Dassault systems. `http://www.3ds.com/products/catia`.

[17] Siemens NX4.0. `http://www.plm.automation.siemens.com/`.

[18] ProEngineer. `www.ptc.com/products/proengineer/`.

[19] SolidWorks. `http://www.solidworks.co.uk`.

[20] ANSYS. `http://www.ansys.com/`.

[21] NASTRAN. `http://www.mscsoftware.com/Products/CAE-Tools/MSC-Nastran.aspx`.

[22] LS-Dyna, Livermore software. `http://www2.lstc.com/lspp/content/overview.shtml`.

[23] iSight FD, Engineous Software. `http://www.simulia.com/products/isight.html`.

[24] K.Y. Lee, M.A. Price, C.G. Armstrong, M. Larson, and K. Samuelsson. *CAD-TO-CAE Integration through automated model simplification and adaptive modelling.* Chalmers Finite Element Centre, Chalmers University of Technology, 2003.

[25] S.J. Leary, A. Bhaskar, and A.J. Keane. A knowledge-based approach to response surface modelling in multifidelity optimization. *Journal of Global Optimization*, 26(3): 297–319, 2003.

[26] P.G. Young, G. Tabor, T. Collins, J. Richterova, E. Dejuniat, and T. Beresford-West. Automating the Generation of 3D Finite Element Models Based on Medical Imaging Data. *Society of Automotive Engineers, 400 Commonwealth Dr, Warrendale, PA, 15096, USA*, 2006.

[27] K. Shimada. Current trends and issues in automatic mesh generation. *Computer-Aided Design & Applications*, 3(6):741–750, 2006.

[28] S.J. Leary, A. Bhaskar, and A.J. Keane. A constraint mapping approach to the structural optimization of an expensive model using surrogates. *Optimization and Engineering*, 2(4):385–398, 2001.

[29] S.J. Leary, A. Bhaskar, and A.J. Keane. Screening and approximation methods for efficient structural optimization. In *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, Georgia, 2002.

[30] A.I.J. Forrester, A. Sóbester, and A.J. Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A*, 463(2088):3251–3269, 2007.

[31] R.J. Donaghy, W.M. Cune, S.J. Bridgett, C.G. Armstrong, D.J. Robinson, R.M. McKeag, D. Robinson, et al. Dimensional reduction of analysis models. *Citeseer*, 1996.

[32] R.J. Donaghy, C.G. Armstrong, and M.A. Price. Dimensional reduction of surface models for analysis. *Engineering with Computers*, 16(1):24–35, 2000.

[33] ANSYS Design Space. `http://www.Ansys.com/products/structural-mechanics/products.asp`.

[34] B.H. Couteau, Y. Payan, and S.H. LavalleHe. The mesh-matching algorithm: an automatic 3D mesh generator for finite element structures. *Journal of Biomechanics*, 33:1005–1009, 2000.

[35] K. Miyoshi and T. Blacker. Hexahedral mesh generation using multi-axis cooper algorithm. In *9th International Meshing Roundtable Proceedings*, pages 89–97. Citeseer, 2000.

[36] Vivace Reports. `http://www.vivaceproject.com/content/engine/reports.php`, . Last accessed on 21/10/2010.

[37] K.W. Shim, D.J. Monaghan, and C.G. Armstrong. Mixed dimensional coupling in finite element stress analysis. *Engineering with Computers*, 18(3):241–252, 2002.

[38] R.W. McCune, C.G. Armstrong, and D.J. Robinson. Mixed-dimensional coupling in finite element models. *International Journal for Numerical Methods in Engineering*, 49(6):725–750, 2000.

[39] T.T. Robinson, C.G. Armstrong, G. McSparron, A. Quenardel, H. Ou, and R.M. McKeag. Automated mixed dimensional modelling for the finite element analysis of swept and revolved CAD features. In *Proceedings of the 2006 ACM symposium on Solid and physical modeling*, pages 117–128. ACM, 2006.

[40] D.J. Monaghan, I.W. Doherty, et al. Coupling 1D beams to 3D bodies. In *Proc. 7 th Int. Meshing Roundtable, Sandia National Laboratories.* Citeseer, 1998.

[41] S. Bournival, J.C. Cuillière, and V. François. A mesh-geometry based approach for mixed-dimensional analysis. *Proceedings of the 17th International Meshing Roundtable*, pages 299–313, 2008.

[42] M. Rezayat. Midsurface abstraction from 3D solid models: general theory and applications. *Computer-Aided Design*, 28(11):905–915, 1996.

[43] H. Lee, Y.Y. Nam, and S.W. Park. Graph-based midsurface extraction for finite element analysis. In *Computer Supported Cooperative Work in Design, 2007. CSCWD 2007. 11th International Conference on*, pages 1055–1058. IEEE, 2007.

[44] M. Ramanathan and B. Gurumoorthy. Generating the Mid-Surface of a Solid using 2D MAT of its Faces. *Computer Aided Design and Applications. v1*, pages 665–674.

[45] R. Stolt. A sectioning method for constructing the mid-surface of thin-walled die-cast and injection moulded parts. *Research Report: School of Engineering, ISSN*, 1404 (0018):2, 2006.

[46] C.G. Armstrong, D.J. Robinson, R.M. McKeag, T.S. Li, S.J. Bridgett, R.J. Donaghy, and C.A. McGleenan. Medials for meshing and more. In *4th Int. Meshing Roundtable*, pages 277–288. Citeseer, 1995.

[47] K. Suresh. Automating the CAD/CAE dimensional reduction process. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 76–85. ACM, 2003.

[48] K. Suresh. Generalization of the mid-element based dimensional reduction. *Journal of Computing and Information Science in Engineering*, 3:308, 2003.

[49] R.A. Katz and S.M. Pizer. Untangling the Blum medial axis transform. *International Journal of Computer Vision*, 55(2):139–153, 2003.

[50] F. Preparata. The medial axis of a simple polygon. *Mathematical Foundations of Computer Science 1977*, pages 443–450, 1977.

[51] H.I. Choi, S.W. Choi, and H.P. Moon. Mathematical theory of medial axis transform. *Pacific Journal of Mathematics*, 181(1):57–88, 1997.

[52] M. Sinha and K. Suresh. Simplified engineering analysis via medial mesh reduction. In *Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 207–212. ACM, 2005.

[53] Matlab File Exchange. `http://www.mathworks.com/matlabcentral/fileexchange/12399`. Last accessed on 21/10/2010.

[54] L. Prasad. Morphological analysis of shapes. In *CNLS newsletter*. Citeseer, 1997.

[55] L. Prasad. Rectification of the chordal axis transform and a new criterion for shape decomposition. In *Discrete Geometry for Computer Imagery*, pages 263–275. Springer, 2005.

[56] W.R. Quadros and K. Shimada. Hex-layer: layered all-hex mesh generation on thin section solids via chordal surface transformation. In *Proceedings of 11th international meshing roundtable*, pages 169–180, 2002.

[57] W.R. Quadros. An approach for extracting non-manifold mid-surfaces of thin-wall solids using chordal axis transform. *Engineering with Computers*, 24(3):305–319, 2008.

[58] A. Telea and A. Vilanova. A robust level-set algorithm for centerline extraction. In *Proceedings of the symposium on Data visualisation 2003*, pages 185–194. Eurographics Association, 2003.

[59] A. Telea and J.J. Van Wijk. An augmented fast marching method for computing skeletons and centerlines. In *Proceedings of the symposium on Data Visualisation 2002*, page 251. Eurographics Association, 2002.

[60] M.S. Hassouna and A.A. Farag. Robust Centerline Extraction Framework Using Level Sets. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Volume 1-Volume 01*, pages 458–465. IEEE Computer Society, 2005.

[61] Computational Geometry Algorithms Library. `http://www.cgal.org/`. Last accessed on 21/10/2010.

[62] Mesecina. `http://www.balintmiklos.com/mesecina/download.html`. Last accessed on 21/10/2010.

[63] B. Miklos, J. Giesen, and M. Pauly. Medial axis approximation from inner Voronoi balls: a demo of the Mesecina tool. In *Proceedings of the twenty-third annual symposium on Computational geometry*, pages 123–124. ACM, 2007.

[64] J. Giesen, B. Miklos, and M. Pauly. Medial axis approximation of planar shapes from union of balls: A simpler and more robust algorithm. In *Proc. Canad. Conf. Comput. Geom*, pages 105–108, 2008.

[65] SphereTree Algorithm. `http://isg.cs.tcd.ie/spheretree/`. Last accessed on 21/10/2010.

[66] G. Bradshaw and C. O'Sullivan. Sphere-tree construction using dynamic medial axis approximation. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 33–40. ACM, 2002.

[67] S. Yoshizawa, A.G. Belyaev, and H.P. Seidel. Free-form skeleton-driven mesh deformations. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 247–253. ACM, 2003.

[68] SM03Skeleton. `http://www.riken.jp/brict/Yoshizawa/Research/Skeleton.html`. Last accessed on 21/10/2010.

[69] Power Crust. `http://userweb.cs.utexas.edu/users/amenta/powercrust/welcome.html`. Last accessed on 21/10/2010.

[70] N. Amenta, S. Choi, and R.K. Kolluri. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, pages 249–266. ACM, 2001.

[71] E.B. Rudnyi and J.G. Korvink. Model order reduction for large scale finite element engineering models. *ECCOMAS CFD, The Netherlands*, 2006.

[72] P. Alliez and C. Gotsman. Recent advances in compression of 3D meshes. *Advances in Multiresolution for Geometric Modelling*, pages 3–26, 2005.

[73] A. Thakur, A.G. Banerjee, and S.K. Gupta. A survey of CAD model simplification techniques for physics-based simulation applications. *Computer-Aided Design*, 41(2): 65–80, 2009.

[74] Harpoon. `http://www.sharc.co.uk/`.

[75] ANSYS Tips and Tricks. `http://ansys.net/?mycat=tips`. Last accessed on 21/10/2010.

[76] Investigation of a Robust Method for Connecting Dissimilar 3D Finite Element Models by David Trujillo. `http://home.earthlink.net/~trucomp/RobustMethod_V1.pdf`. Last accessed on 21/10/2010.

[77] A. Pantano and R.C. Averill. A penalty-based finite element interface technology. *Computers & Structures*, 80(22):1725–1748, 2002.

[78] M.A. Puso. A 3D mortar method for solid mechanics. *International Journal for Numerical Methods in Engineering*, 59(3):315–336, 2004.

[79] C.R. Dohrmann and S.W. Key. A transition element for uniform strain hexahedral and tetrahedral finite elements. *International Journal for Numerical Methods in Engineering*, 44(12):1933–1950, 1999.

[80] C.R. Dohrmann, S.W. Key, and M.W. Heinstein. Methods for connecting dissimilar three-dimensional finite element meshes. *International Journal for Numerical Methods in Engineering*, 47(5):1057–1080, 2000.

[81] C.R. Dohrmann, S.W. Key, and M.W. Heinstein. A method for connecting dissimilar finite element meshes in two dimensions. *International Journal for Numerical Methods in Engineering*, 48(5):655–678, 2000.

[82] ANSYS Commands Reference, Release 10.0 Documentation for ANSYS. `http://www1.ansys.com/customer/content/documentation/100/ansys/acmd100.pdf`. Last accessed on 21/10/2010.

[83] I.I. Voutchkov and A.J. Keane. Multi-objective optimization using surrogates. *Computational Intelligence in Optimization*, pages 155–175, 2010.

[84] M. Price, C. Stops, and G. Butlin. A medial object toolkit for meshing and other applications. In *Proceedings of 4th International Meshing Roundtable*, pages 219–229. Citeseer, 1995.

[85] Transcendata, Europe website. `http://www.fegs.co.uk/medial.html`. Last accessed on 21/10/2010.

[86] C.G. Armstrong, R.M. McKeag, H. Ou, and M.A. Price. Geometric processing for analysis. In *Geometric Modeling and Processing 2000. Theory and Applications. Proceedings*, pages 45–56. IEEE, 2002.

[87] C.G. Armstrong, S.J. Bridgett, R.J. Donaghy, R.W. McCune, R.M. McKeag, and D.J. Robinson. Techniques for interactive and automatic idealisation of CAD models. *Numerical Grid Generation in Computational Field Simulation, Mississippi State University, MS*, pages 643–662, 1998.

[88] Dimensional and Geometry constraints in NX. `http://www.cad-resources.com/Sketches.pdf`. Last accessed on 21/10/2010.

[89] A. Petik. Some aspects of using STL file format in CAE systems. In *International Workshop CA Systems and Technologies*, page 80. Citeseer, 1999.

[90] W. Rust and K. Schweizerhof. Finite element limit load analysis of thin-walled structures by ANSYS (implicit), LS-DYNA (explicit) and in combination. *Thin-walled structures*, 41(2-3):227–244, 2003.

[91] MAT 2D code. `http://www.mathworks.de/matlabcentral/fileexchange/12399`. Last accessed on 21/10/2010.

[92] IGES to Matlab. `http://www.mathworks.com/matlabcentral/fileexchange/13253`. Last accessed on 21/10/2010.

[93] R. Tam and W. Heidrich. Feature-preserving medial axis noise removal. *Computer VisionECCV 2002*, pages 755–756, 2002.

[94] D. Shaked and A.M. Bruckstein. Pruning medial axes. *Computer Vision and Image Understanding*, 69(2):156–169, 1998.

[95] A. Sud, M. Foskey, and D. Manocha. Homotopy-preserving medial axis simplification. In *Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 39–50. ACM, 2005.

[96] H. Sakai and K. Sugihara. A method for stable construction of medial axes in figures. *Electronics and Communications in Japan (Part II: Electronics)*, 89(7):48–55, 2006.

[97] A.D. Ward and G. Hamarneh. The groupwise medial axis transform for fuzzy skeletonization and pruning. *IEEE transactions on pattern analysis and machine intelligence*, 2009.

[98] R. Ogniewicz. Automatic medial axis pruning by mapping characteristics of boundaries evolving under the euclidean geometric heat flow onto voronoi skeletons. *Harvard Robotics Laboratory Technical Report*, pages 95–4, 1995.

[99] K. Deb. *Optimization for engineering design: Algorithms and examples.* PHI Learning Pvt. Ltd., 2004.

[100] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2): 182–197, 2002.

[101] R. Wang, K. Zhou, J. Snyder, X. Liu, H. Bao, Q. Peng, and B. Guo. Variational sphere set approximation for solid objects. *The Visual Computer*, 22(9):612–621, 2006.

[102] Y. Xie and J. Xu. Efficient algorithm for approximating maximum inscribed sphere in high dimensional polytope. In *15th Annual Fall Workshop on Computational Geometry and Visualization*, page 73. Citeseer, 2005.

[103] E.C. Sherbrooke, N.M. Patrikalakis, and E. Brisson. Computation of the medial axis transform of 3-D polyhedra. In *Proceedings of the third ACM symposium on Solid modeling and applications*, pages 187–200. ACM, 1995.

[104] S.P. Li and K.L. Ng. Monte Carlo study of the sphere packing problem. *Physica A: Statistical Mechanics and its Applications*, 321(1-2):359–363, 2003.

[105] Y.G. Lee and K. Lee. Computing the medial surface of a 3-D boundary representation model. *Advances in Engineering Software*, 28(9):593–605, 1997.

[106] H. Du and H. Qin. Medial axis extraction and shape manipulation of solid objects using parabolic PDEs. In *Proceedings of the ninth ACM symposium on Solid modeling and applications*, pages 25–35. Eurographics Association, 2004.

[107] H. Ledoux. Computing the 3D Voronoi diagram robustly: an easy explanation. In *Voronoi Diagrams in Science and Engineering, 2007. ISVD'07. 4th International Symposium on*, pages 117–129. IEEE, 2007.

[108] I. Boada, N. Coll, N. Madern, and J. Antoni Sellares. Approximations of 2D and 3D generalized Voronoi diagrams. *International Journal of Computer Mathematics*, 85(7): 1003–1022, 2008.

[109] M.R. Jones, M.A. Price, and G. Butlin. Geometry management support for automeshing. In *Proceedings of 4th International Meshing Roundtable*, pages 153–164, 1995.

[110] Y. Choi, K. Kim, and S. Lee. Shape reconstruction from unorganized points using voronoi diagrams. *The International Journal of Advanced Manufacturing Technology*, 21(6):446–451, 2003.

[111] R. Fabio. From point cloud to surface: the modeling and visualization problem. In *International Workshop on Visualization and Animation of Reality-based 3D Models*, pages 446–451. Citeseer, 2003.

[112] SYCODE. `http://www.sycode.com/products/point_cloud_rh/index.htm`.

[113] Rhinocerous. `http://www.rhino3d.com/`.

[114] MyCrustOpen. `http://www.mathworks.com/matlabcentral/fileexchange/22185-surface-reconstruction-from-scattered-points-cloud-part1`. Last accessed on 21/10/2010.

[115] MyRobustCrust. `http://www.mathworks.com/matlabcentral/fileexchange/22595-surface-reconstruction-from-scattered-points-cloud-part2`. Last accessed on 21/10/2010.