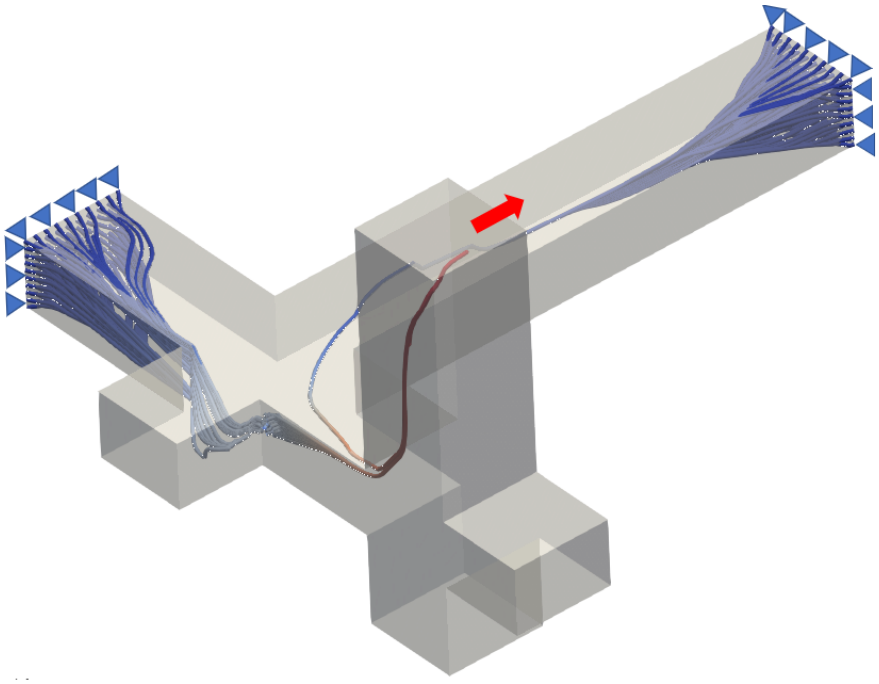
CHALMERS
UNIVERSITY OF TECHNOLOGY



Load Path Visualization in Aero-engine Structures Using U* Index Method

Master's thesis in Applied Mechanics

RAJESH RAMESH

DEPARTMENT OF INDUSTRIAL AND MATERIALS SCIENCE - IMSX30

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

Master's thesis 2021

Load Path Visualization in Aero-engine Structures Using U* Index Method

RAJESH RAMESH

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Industrial and Materials Science
Division of Material and Computational Mechanics
Chalmers University of Technology
Gothenburg, Sweden 2021

Load Path Visualization in Aero-engine Structures Using U* Index Method
RAJESH RAMESH

Academic supervisor: Ragnar Larsson, Professor, Head of Division Material and Computational Mechanics, Department of Industrial and Materials Science, Chalmers University of Technology

Industrial supervisor: Visakha Raja, Solid Mechanics Engineer, Department of System Analysis and Intellectual properties, GKN Aerospace Sweden AB.

Examiner: Ragnar Larsson, Professor, Head of Division Material and Computational Mechanics, Department of Industrial and Materials Science, Chalmers University of Technology

Master's Thesis 2021
Department of Industrial and Materials Science
Division of Material and Computational Mechanics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Load Path Visualization of a 3D complex structure from the developed routine.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Load Path Visualization in Aero-engine Structures Using U* Index Method
RAJESH RAMESH
Department of Industrial and Materials Science
Chalmers University of Technology

Abstract

In the aerospace industry, there exists an extensive demand for improving the performance of mechanical structures in the aircraft components. Such performance can be improved by various measures such as reducing the weight and improving the stiffness of the existing structures. To achieve this, it is important to study how the mechanical load is distributed in the structure which can be done by analyzing so-called load paths. The load paths are the trajectories that represent the flow of load within the structure. Existing stress-based concepts generally suffer from stress concentrations due to geometrical features, such as joints and curved sections. Hence, stress concentrations can provide misleading information when applied to study load paths. The load transfer index U^* is a relatively modern concept, proposed by previous researchers to study the load distribution in the structures. It expresses the internal stiffness between loading and arbitrary points within the structure. Unlike many other approaches, U^* index is unaffected by the presence of geometrical features. The load paths based on U^* can provide useful insights that would help in stiffness modification and design evaluation. Thus, this method has the potential to play a key role in structural mechanics. However, the U^* index concept-based analysis is not widely used in engineering problems since the calculation and visualization routines are not widely incorporated in finite element software. The purpose of this thesis is to visualize load paths based on U^* and enhance design on existing aero-engine structures.

This thesis focuses on developing computational methods to calculate the U^* index and routines to visualize load paths for structural engineering problems. Different software were used to develop the methodology in this thesis. The necessary finite element method calculations were performed by using ANSYS APDL and the results are post-processed in ParaView to visualize the load paths. The load paths are visualized for 2D and 3D structural problems. The influence of boundary conditions, types of loading, mesh elements and mesh sensitivity analysis were studied. This thesis sets the platform for wider application of the load path visualization concept in structural engineering problems which is of particular importance at GKN Aerospace Sweden AB, Trollhättan where the thesis work was carried out.

Keywords: Computational Mechanics, Design Optimisation, Finite Element Method, Load Path, Load Transfer, Structural Analysis, Structural Design, U* index .

Acknowledgements

This master thesis was performed at GKN Aerospace Sweden AB in Trollhättan, Sweden. Hence, I would like to show my sincere gratitude and respect to the people at the company for offering me the opportunity to work on this thesis project despite the pandemic. It has been a great time working at the company that provided me with a trove of knowledge.

First of all, I would like to thank my industrial supervisor Visakha Raja for supporting me throughout the project through his guidance, encouragement and valuable inputs which are crucial for this thesis.

Further, I would like to thank my academic supervisor Ragnar Larsson for his valuable and timely feedback on different issues during this thesis.

At last, I would like to thank my parents for their constant support and motivation to complete this thesis.

Rajesh Ramesh, Gothenburg, 6 2021

Contents

List of Figures

List of Tables

Nomenclature

List of Acronyms

2D	Two Dimension
3D	Three Dimension
APDL	Ansys Parametric Design Language
DOF	Degrees of Freedom
FEA	Finite Element Analysis
FEM	Finite Element Method
VTK	Visualization Toolkit

List of Notations

n_{load}	Load nodes
$n_{boundary}$	Boundary nodes
n_{free}	Free nodes
d_{load}	Displacement at load nodes
$\mathbf{K}$	Global Stiffness Matrix
$\mathbf{P}$	Load Vector
$\mathbf{d}$	Displacement Vector
U	Total Strain Energy of the original load case
U'	Total Strain Energy of the modified load case
U^*	Index expressing the internal stiffness
U^{**}	Index expressing the internal compliance

1
Introduction

This chapter gives an overview of the company's background, project background, project description, aim and limitations of the project.

1.1 Company Background

GKN Aerospace Sweden AB is a subsidiary of the company group GKN Aerospace Engine Systems within *GKN Aerospace* who are pioneers in manufacturing aero-engines, wing structures and other components of the aircraft. The company was formed in 2012 when Volvo Aero was integrated into the GKN group of companies. The headquarters is located at Trollhätan, Sweden. At Trollhättan, the company develops and manufactures structure components for commercial aero-engines for aircraft and space rockets and carries out engine maintenance. GKN Aerospace Engine Systems collaborates with major aircraft engine manufacturers such as *Rolls-Royce, Pratt & Whitney, Snecma,* and *General Electric* on large civil aircraft engines. Currently, *GKN Aerospace* engine components are used in about 90 per cent of all newly designed aircraft worldwide [1].

1.2 Project Background

One of the problems in the modern engineering industry is how to improve the structural design of the structures taking account of the product's performance and manufacturing cost during the initial stages of the product development. An improved structural design should have high strength to weight ratio or high stiffness to weight ratio. Although lightweight materials can be used for this purpose, an improved structural design will make the product structurally stronger and efficient. While designing the structure, it is important to study how the imposed load passes through the structure to the supporting points. It can be done by analyzing so-called load paths. The load paths are the trajectories that represent the flow of load within the structure. Hence, the load path visualization is considered in this thesis. Conventional stress-based concepts already exist in the structural mechanics for this motive. However, these concepts are vulnerable to factors like stress concentration due to the geometry features of the design such as bends, curves and holes. Hence, there exists a need for an analysis based on alternative methods that are unaffected to such stress concentration.

The index U^*, which defines the internal stiffness of the structure, expresses the degree of connection between an arbitrary point and loading point within the structure [13]. This index is independent to stress concentrations. The load paths based on the U^* field provides information about the load distribution, regions of high stresses. By knowing the load distribution, the material at locations that do not contribute to the load distribution can be removed and material can be added where the stiffness modification is required. Thus, the stiffness can be improved to achieve a high strength-weight ratio or a high stiffness-weight ratio. Hence, by taking account of this information, the structural design can be modified according to its purpose.

Previous research conducted at *GKN Aerospace* around the architecture and complexity of aero-engine components [2] examined the possibility of developing load path analysis for aero-engine structures. This included development of analysis scripts as well as looking into existing software.

A software developed by *FRONE Corporation* [4] exists to perform the necessary pre&post processing to visualize U^*. Due to the problems of integration with existing software practices at *GKN Aerospace*, the software was not used. Thus, the consequence led to the origin of this thesis.

1.3 Project Description

The purpose of the project to use load path visualization in existing aero-engine structures and thereby enable design improvements. Figure 1.1 illustrates how the load path visualization can be used in the design optimisation of the structure.

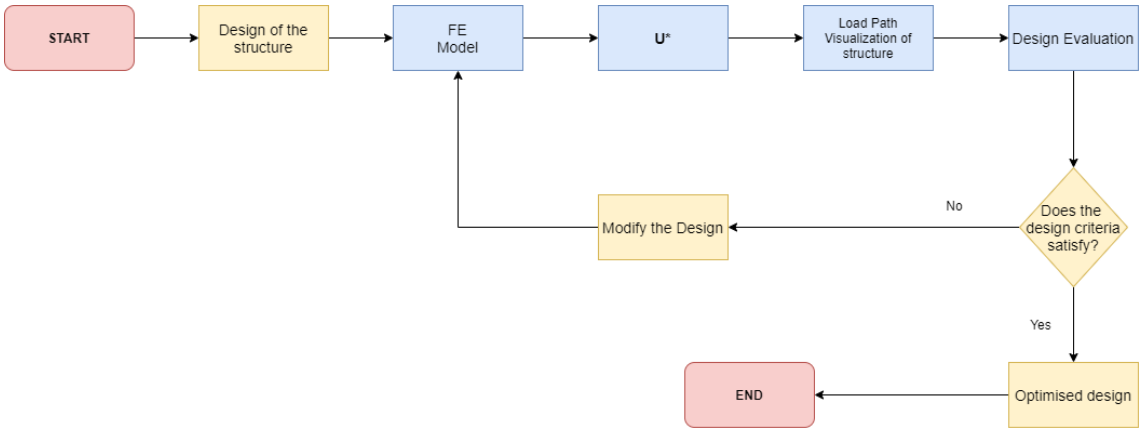


Figure 1.1: Flowchart of design optimisation based on load paths using U^*

The method takes advantage of the Finite Element Method technique to calculate the U^*. The contours and load paths are plotted based on U^* data field. The design can be evaluated based on the criteria from the load paths explained in section 2.3.3 and modified until the design satisfies the criteria. Thus, the structurally improved design can be obtained.

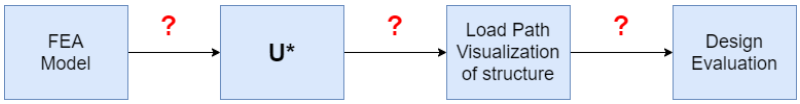


Figure 1.2: Missing links for load path visualization at *GKN Aerospace*

Although the procedure to visualize load paths and design evaluation are known, the methods to calculate U^* from the finite element model, visualizing the load paths and evaluating design based on the load path are not widely available in commercial FE software. Hence, this thesis focuses on the implementation of these methods with existing commercial software and for the wider use of load path visualization in *GKN Aerospace* to analyse and enable design improvement in the aero-engine structures.

1.4 Aim and Objectives of the Project

The thesis aims to carry out methods to do the necessary calculations and to plot load paths by the U^* index method in the aero-engine structures in 3D space. To achieve this, the following objectives are needed to be completed.

1. Implementation of the U^* calculation routine in commercial FEA software used at *GKN Aerospace*.
2. Produce contour plot of U^* for the physical problem.
3. Produce methods to visualize the Load paths within the structure.
4. Establish methods to evaluate load paths.

1.5 Limitations

Due to the limits on time for completion of this thesis, the scope had to be reduced and few limitations exist in this thesis. They are the following,

1. The thesis is limited to purely static mechanical loaded problems.
2. Only linear isotropic elastic material model is considered.
3. Only the routine for the direct method calculation of U^* is developed, the time reduction methods [11] are kept as future work.

2

Theory

This chapter illustrates the basic theory to understand the Finite element method and how it is used to visualize load path. This chapter also covers the theory of the U^* index method. The supporting literature is also presented to discuss relevant studies carried out in a similar field of work.

2.1 Introduction to Finite Element Method

The Finite Element Method (FEM) is a numerical procedure for solving partial differential equations. It is generally used to obtain solutions to boundary value problems in mathematical physics and engineering analysis. This method employs discretizing a continuous domain into several smaller, simpler subdomains called finite elements interconnected at points (nodes) common to two or more elements, each of which represents the unknown function with simple interpolation functions with unknown coefficients. This method formulates the equations for each finite element and combines them to obtain the solution of the whole body [5]. Thus, The solution of the whole body is approximated by a finite number of unknown coefficients. FEM is applied when the geometry is complex, where the analytical solution can not be implied. The study of a phenomenon with FEM referred to as Finite Element Analysis (FEA). It is widely used in several areas of engineering analysis and research.

The general procedure for finite element analysis (FEA) can be categorised in three steps as follows

1. Pre-processing
2. Solution
3. Post-processing

The Pre-Processing involves Geometry definition, Material property definition, Mesh generation (discretization of the domain), Selection of element types and physical constraints definition (Boundary conditions and loading conditions). The solution solves the system of equations in the form of matrices using numerical techniques to get unknown variables such as displacements, stresses, reaction forces and heat flow. Post-Processing is the analysis and evaluation of the results in the form of plots and graphs to make decisions. Usually, Post-Processing is done using computer programs/software.

Figure 2.1: Stages of Finite Element Analysis

2.2 Load Transfer Analysis

Load transfer analysis is one of the important analyses of an engineering structure that deals with the study of how the imposed load is being transferred to the supporting points [3].

Literature survey provided the information that there are various approaches to find the load paths in the structures such as Load path methods based on the direction of principal stresses, load flow, transferred & potential transferred force method and U^* field. Kun Marhadi and Satchi Venkataraman [8] did a comparative study of these methods used for Load Paths. They evaluated these methods considering that an ideal method should satisfy three qualities. They are

1. Complete visualization of load paths from the points of load applied to the points of supports or reactions.
2. The method should point out the regions/volume which requires stiffness modification.
3. The method should point out the region of interest such as regions with high stresses.

From their conclusion, each method had a drawback lacking one of the qualities that an ideal method should possess. The load path methods based on stress provide information that is almost similar to the conventional stress analysis. The stress concentration can influence and misguide the prediction of load paths due to geometry features. Hence, these methods only provide a limited amount of information to optimise/modify the design. As a result, it became necessary to look for alternate methods to detect load paths resistant to the influence of stress concentration. It was investigated that the U^* method was suitable for this criterion.

2.3 The U^* index

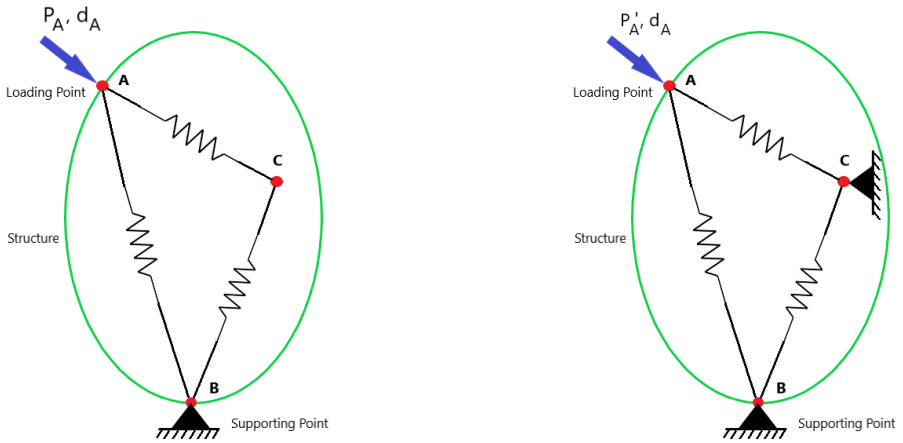
The U^* index theory for load transfer was introduced by a group of researchers, Kunihiro et al [13] in the year 1995 as a procedure to represent load transfer in the vehicle structures.

The U^* index method provides a numerical index that expresses the degree of connection between the loading points and arbitrary points in the structure [13]. In

other words, it provides information about the participation of the arbitrary point in the structure towards the load distribution. This method is based on the total strain energy of the structure under different boundary conditions.

2.3.1 Method

Let us consider a simple elastic body structure as shown in the figure 2.2 which has three points A, B and C. A is the point of loading, B is the point of support and C is an arbitrary point in the structure.



(a) Original loading case **(b)** Modified loading case

Figure 2.2: Spring model of the problem for U^* analysis

Since the body is elastic, the connection between these points can be represented by linear springs. The force-displacement relation of the system can be written as

$$\mathbf{P} = \mathbf{K} \quad \mathbf{d}$$

Where **P** and **d** are the load/force vector and displacement vector respectively. **K** is the stiffness matrix that points out the force-displacement relationship of each point due to external load and the mutual relationship between two arbitrary points within the structure [14]. The expression of the force-displacement relation for the problem in figure 2.2a can be written as

$$\begin{bmatrix} P_A \\ P_B \\ P_C \end{bmatrix} = \begin{bmatrix} K_{AA} & K_{AB} & K_{AC} \\ K_{BA} & K_{BB} & K_{BC} \\ K_{CA} & K_{CB} & K_{CC} \end{bmatrix} \begin{bmatrix} d_A \\ d_B \\ d_C \end{bmatrix}$$

The suffices in the vectors and matrix refers to the corresponding points in the structure.

If P_A is the load applied at point A, d_A is the displacement due to the loading at point A, then the total strain energy of the system is defined by

$$U = \frac{1}{2} P_A \, d_A$$

Since the point B is fixed (d_B=0),

$$U = \frac{1}{2} \left(K_{AA} d_A + K_{AC} d_C\right) d_A$$

The loading case/physical constraint is modified by replacing the force by enforcing the displacement d_A and constraining the arbitrary point C as shown in figure 2.2b. The total strain energy of the modified system can be written as

$$U' = \frac{1}{2} P'_A \, d_A$$

where P'_A is the load required to enforce displacement d_A. Since the point B and C are fixed,

$$U' = \frac{1}{2} \left(K_{AA} d_A\right) d_A$$

According to the U^* theory, the U^* index is defined as

$$U^* = 1 - \frac{U}{U'} \tag{2.1}$$

$$U^* = 1 - \frac{\left(K_{AA} d_A + K_{AC} d_C\right)}{\left(K_{AA} d_A\right)} \tag{2.2}$$

From equation 2.2, it is evident that U^* index of an arbitrary point depends on the stiffness between the loading point A and arbitrary point C. It points out the degree of connection between these points. Following the same procedure of constraining arbitrary points, the U^* distribution within the structure can be obtained by calculating U^* at other arbitrary points within the structure.

2.3.2 Visualization

According to the U^* theory, the path tracing through the regions with the highest internal stiffness within the structure is called the Load path. In other words, the load path connects the loading points to the supporting points through the highest U^* index values (points having the highest degree of connection).

If a vector is introduced such that

$$\lambda = -grad \, U^*$$

then, the line traced along the lowest λ value is the load path [9].

It is the path that passes through the highest points of the contour lines obtained from the U^* field within the structure. Figure 2.3 illustrates the load path (Blue

solid line) connecting loading points from the supporting points through the ridge of the contours of the simple structure considered in figure 2.2.

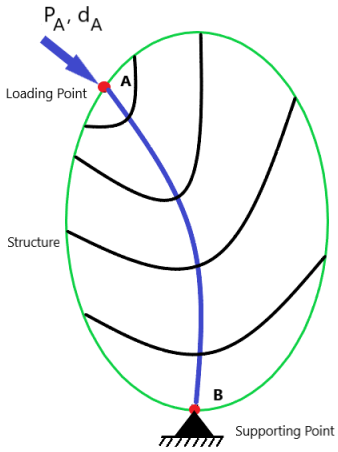


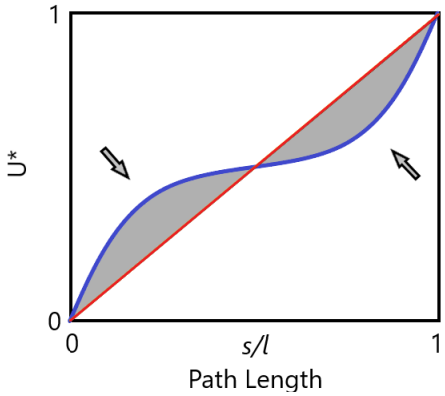
Figure 2.3: Load path using U^* index

2.3.3 Design Criteria

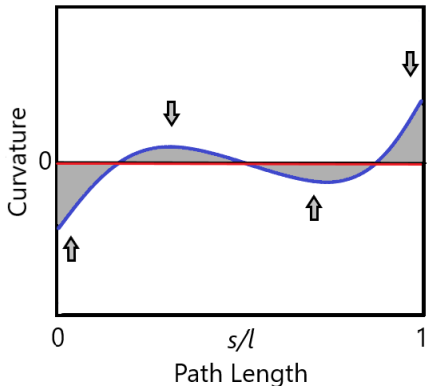
H.Hoshino et al [7] proposed three key criteria for design evaluation based on the U^* theorem, which expresses the global behaviour of a structure. An ideal structure with optimal stiffness distribution must satisfy the following criteria,

1. Uniformity: The load path of the structure containing U^* values must have a linear variation of U^* along the length of the load path.
2. Continuity: The load path of the structure containing U^* values must have the curvature of U^* variation to be constantly zero.
3. Consistency: The load paths traced from loading points and supporting points of the structure should coincide.

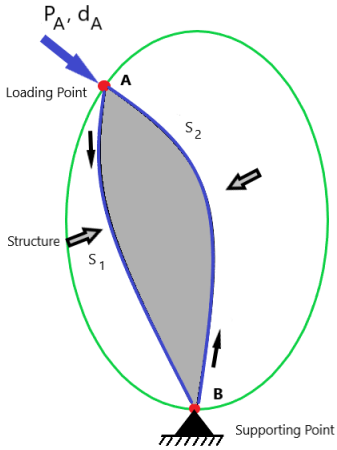
The solid red line in figures 2.4a and 2.4b represents the variation of the properties of the load path for the ideal structure. However, the engineering structures may have deviations from the ideal case. The solid blue line represents the variation of the properties of the load path of such structures. The deviations of the properties of structures from the ideal case are represented as shaded areas in figure 2.4. The large deviations in figure 2.4b (curvatures of U^* variation) indicates the regions of high stresses. Hoshino et al [7] used these criteria to modify the stiffness of automotive structures for vibration reduction. To optimise the stiffness of a structure, they redesigned the structure that minimizes the shaded area in figures 2.4a, 2.4b and 2.4c.



(a) Uniformity



(b) Continuity



(c) Consistency

Figure 2.4: Illustration of design criteria based on U^*

These design criteria helps the designers in evaluating the design to find out the critical regions in the structure or to modify the physical constraints that would make the structure stiffer.

2.3.4 Drawbacks

Although this U^* index method has great significance over the conventional stress concepts, it has some drawbacks which are needed to be addressed.

1. Since the calculation of U^* index within a structure involves solving the problem for multiple physical constraints, the computational time becomes heavy when the problem has more nodes. In other words, for fine mesh, the computational time becomes enormously heavy which is a serious problem in using this U^* index method. However, alternative methods of calculating the U^* was proposed by Sakurai T et al [11] which was proven to drastically reduce the computational time.

2. Initially, the concept of load path using U^* was proposed for the concentrated/point load. When this concept was applied to the distributed load, it produced unreliable results. As the contour near the loading has the highest gradient at the middle of the loading, the load paths connect to the middle of the loading. Thus, it does not provide reliable information about the stress near the points of loading. Sakurai T et al. [12] continued the calculation of U^* for multiple loading conditions. Wang et al. [18] introduced a new concept U^{**}, complementary to U^* for distributed loading conditions provided reliable results.

2.4 Streamlines

A streamline is defined as a line whose tangent at any point is in the direction of a scalar at that point [6]. It is generally used in fluid mechanics to represent the path of a massless fluid element at any given time.secIn this project, the streamlines are used to represent the load paths.

Let us consider a two dimensional scalar field data as shown in the figure 2.5. Consider a scalar data V at P(x,y,) . Let u, v be the components of the scalar data V in x, y directions. The mathematical equation that defines the streamline is given by

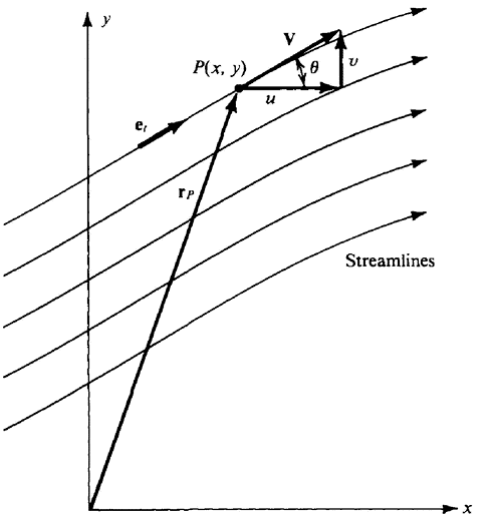


Figure 2.5: Components of the scalar. Picture taken from [6]

Since the streamlines is tangent to the direction of the scalar,

$$\tan(\theta) = \frac{u}{v} = \frac{dx}{dy}$$

The equation of the streamline in 2D,

$$\frac{dx}{u} = \frac{dy}{v}$$

similarly the equation of the streamline in three dimensions can be written as

$$\frac{dx}{u} = \frac{dy}{v} = \frac{dz}{w} \tag{2.3}$$

where w is the component of the scalar variable in z direction.

The equation 2.3 is solved to plot streamlines tangent to the vector of the scalar variable. In this thesis, the streamlines are used to represent the load paths based on the U^* contour.

2.5 Ansys Parametric Design Language

The Ansys Parametric Design Language (APDL) is a finite element solver used for various finite element simulations such as Thermal, Structural, Acoustic, Piezoelectric, Electrostatic and Circuit Coupled Electromagnets. It provides the user with many boundless features such as parameterization, macros, branching and looping, and complex math operations. The commands in the APDL enables the user to define user-defined routines and calculations which are not standard in APDL. Using these advantages, the U^* calculation routine is created by using the APDL commands.

2.6 Visualization of Streamlines in ParaView

ParaView is an open-source, multi-platform scientific data analysis and interactive visualization tool that enables analysis and visualization of extremely large datasets [19]. it is both a general-purpose, end-user application with a collection of tools and libraries for various applications including scripting (using Python), web visualization (through ParaViewWeb), or in-situ analysis (with Catalyst). It also supports scripting and batch processing using Python.

ParaView uses the Visualization Toolkit (VTK), to provide the cornerstone for visualization and data processing. ParaView has file readers that support various file formats typically used in computational software. VTK contributes a valuable data model that ParaView uses to efficiently interpret data from diverse fields with varying features. File readers in ParaView build a data type that is suitable for expressing the data in the files. ParaView permits to create and add filters to transform data based on the data type. In computational science, it is commonly used for post-processing the results obtained from the simulations.

2.6.1 Filters

The filters are the algorithms that read in the data, transform the data, export them to interpret results through data visualization or further processing. There are various filters in-built in ParaView which can be used to create pipelines to perform different operations, processing types and desired task by connecting these filters in a pipeline.

2.6.1.1 Gradient Of Unstructured datagrid

It is an in-built feature in ParaView which is used to calculate the gradients of the scalar field of cell data or point data. The filter functions by iterating over all the cells and calculating the local gradients based on parametric coordinates inside the cells. The gradients are obtained as vectors. The gradients obtained is used by the filter are used by the Streamtracer filter to plot the streamlines to visualize the load paths.

2.6.1.2 Streamtracer

Stream Tracer filter is an in-built feature in ParaView which generates streamlines for vector fields from the collection of seed points. The algorithm functions by reading an array of points, known as seed points, in the dataset and then integrating the streamlines starting at these seed points. This filter has options that lets the user tune the streamlines using the type of integrator, the direction of integrator, length of the streamline. This filter can be accessed from the *Filters* menu. This filter is used in this project to visualize the load-paths from the U^* index vector field.

2.7 The Visualization Tool Kit (VTK)

The Visualization Tool Kit (VTK) is one of the commonly used scientific data file formats. It is an open-source system for scientific visualization, 3D computer graphics and image processing and volume rendering [20]. It is produced by the company *Kitware*. VTK file formats are used to import the sets of data to ParaView .

VTK file format has two different styles. One is Simple Legacy format and the other is XML format. In this project, the Simple legacy format is used to communicate the data (Geometry, nodes, element topology, nodal results) from ANSYS APDL to ParaView. The Simple Legacy file format has five parts. The first parts correspond to file version and identifier. The second section refers to the header, and it describes the data and any other relevant information. The third part is where the type of file format is described between ASCII and BINARY. The fourth part refers to the dataset structure. This is where the geometry, nodes, element topology are defined. The last part describes the dataset attributes which is nothing but the scalar data, vector data at elements and nodes from the results are defined. The figure 2.6 illustrates all these parts.

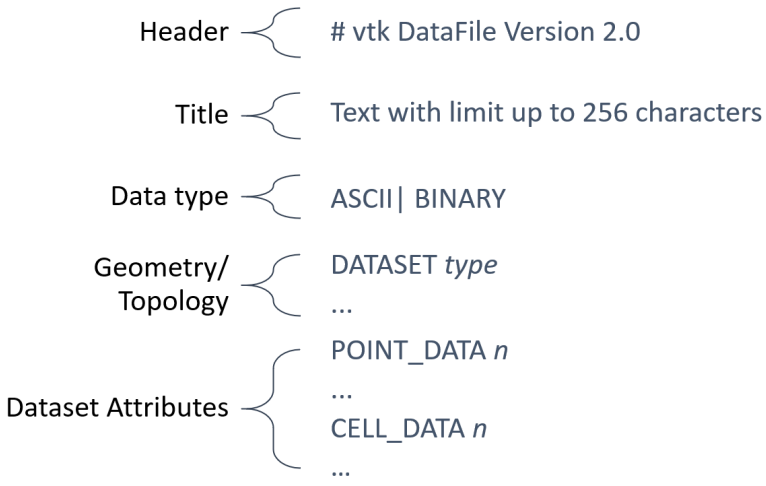


Figure 2.6: General structure of VTK file

Generally, the VTK file format supports has five types of dataset formats such as

1. Structured Points: defines a dataset of points having equal spacing in every three directions (X, Y, Z).
2. Structured Grid: defines a dataset of a grid of points having equal spacing in every three directions (X, Y, Z).
3. Unstructured Grid: defines a dataset of points having unequal spacing with any possible cell type.
4. Polygonal Data: defines a dataset of points associated with regular topology, polygonal geometry.
5. Rectilinear Grid: defines a dataset of the grid of points associated with regular topology, and semi-regular geometry aligned along the x-y-z coordinate axes.
6. Field: defines a dataset of points or cells without topological and geometric structure and a particular dimensionality.

3

Methodology

In this chapter, the method of approach to visualize the load paths for 3D structures is described. Further, the methods to evaluate the design based on the load paths are also described.

3.1 Problem Approach

The aim, scope of the thesis were studied and the milestones are set considering the time frame. A Gantt Chart is created and appropriate days are fixed as a deadline for each milestone. A systematic approach for the thesis to achieve the milestones was framed as shown in figure 3.1.

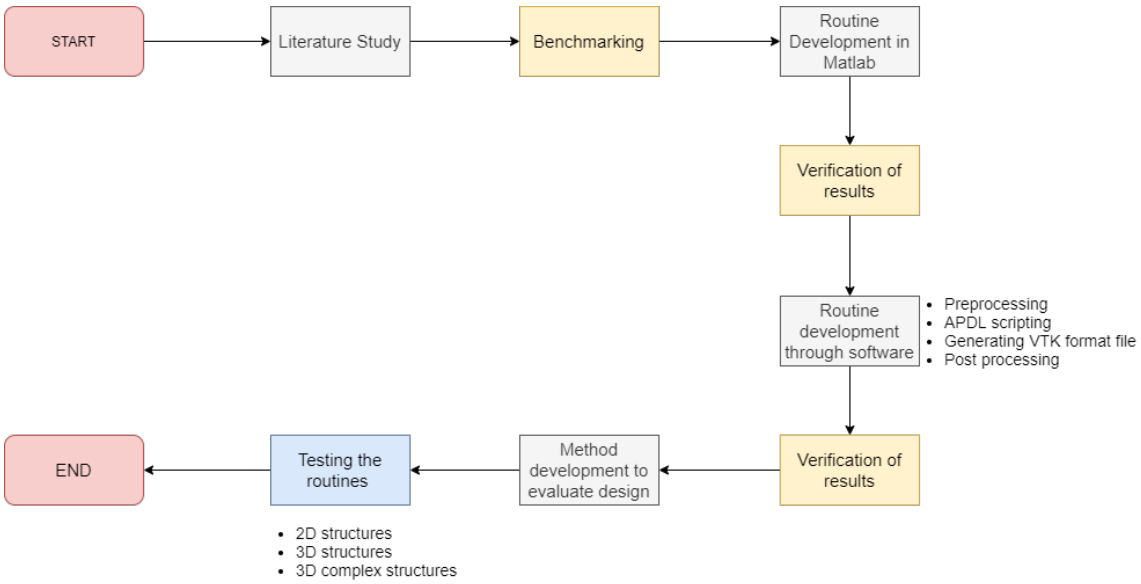


Figure 3.1: Systematic approach to the problem

The approach begins with the literature study. The literature study included the study of Load transfer analysis and its application, programming using APDL commands and creating VTK format files. The books, journals, research papers related to load transfer analyses gave exposure to a breadth of concepts in U^* theory, FEM calculations and provided plenty of insights to proceed with the thesis. Then, certain results from the literature are benchmarked and a Matlab program is developed to do the calculations and to visualize the load paths. Initially, a simple 2D structure

(cantilever problem) is considered for the ease of developing routines. The results obtained from Matlab are verified with the literature results.

After, the thesis is concentrated on developing routines through software. Ansys Workbench, Ansys Mechanical APDL, ParaView and Matlab are used for this purpose. A detailed description of how these software used in this thesis is in section 3.3. Once the routines are developed, the load paths are visualized and verified with the literature results. After verifying the results, the method to evaluate the design based on the load paths is developed. Finally, the complete routine is tested on 2D problems and 3D problems including complex shapes for different boundary conditions, different mesh elements and different mesh size. The results and discussions are provided in the chapter 4.

3.2 Load Path Visualization - Routine development for 2D problems in Matlab

With the knowledge gained from the literature study, a routine to visualize the load paths for simple 2D structures is created by scripting in Matlab and to replicate literature results. This is done to have a deep perception of all relevant aspects to plot load paths using U^* theory. The streamline function in Matlab plotted the ridgeline through the contours within the structure. The theory to streamlines is explained in the section 2.4.

Figure 3.2 illustrates the routine to visualize the load paths for a structural problem. The routine can be generally categorised into Pre-processing, Solution and Post-processing. The blocks in yellow colour correspond to the Pre-Processing of the load path visualization.

The blocks in green colour correspond to the U^* calculation algorithm (Solution). The blocks in red colour correspond to the Post-processing of the load path visualization. The algorithm in figure 3.2 is explained in the following.

1. The routine starts with the general pre-processing of a physical problem.
2. The solver does the calculation routine for U^* field in a structure.
3. First, the original load case is solved. The total strain energy of the structure U and the displacements at the loading nodes are calculated.
4. A loop is created that runs for all freenodes. Freenodes are an array of nodes that are a complement to the nodes that contribute to the loading and boundary nodes from total nodes.
5. In each loop, the load case is modified such that the DOFs of a Freenode is constrained in addition to the supporting nodes.
6. The force applied to the system is replaced by the enforced displacements which are calculated from the results of the original loading case.
7. The modified load case is solved and the strain energy of the modified system **U'** is calculated.

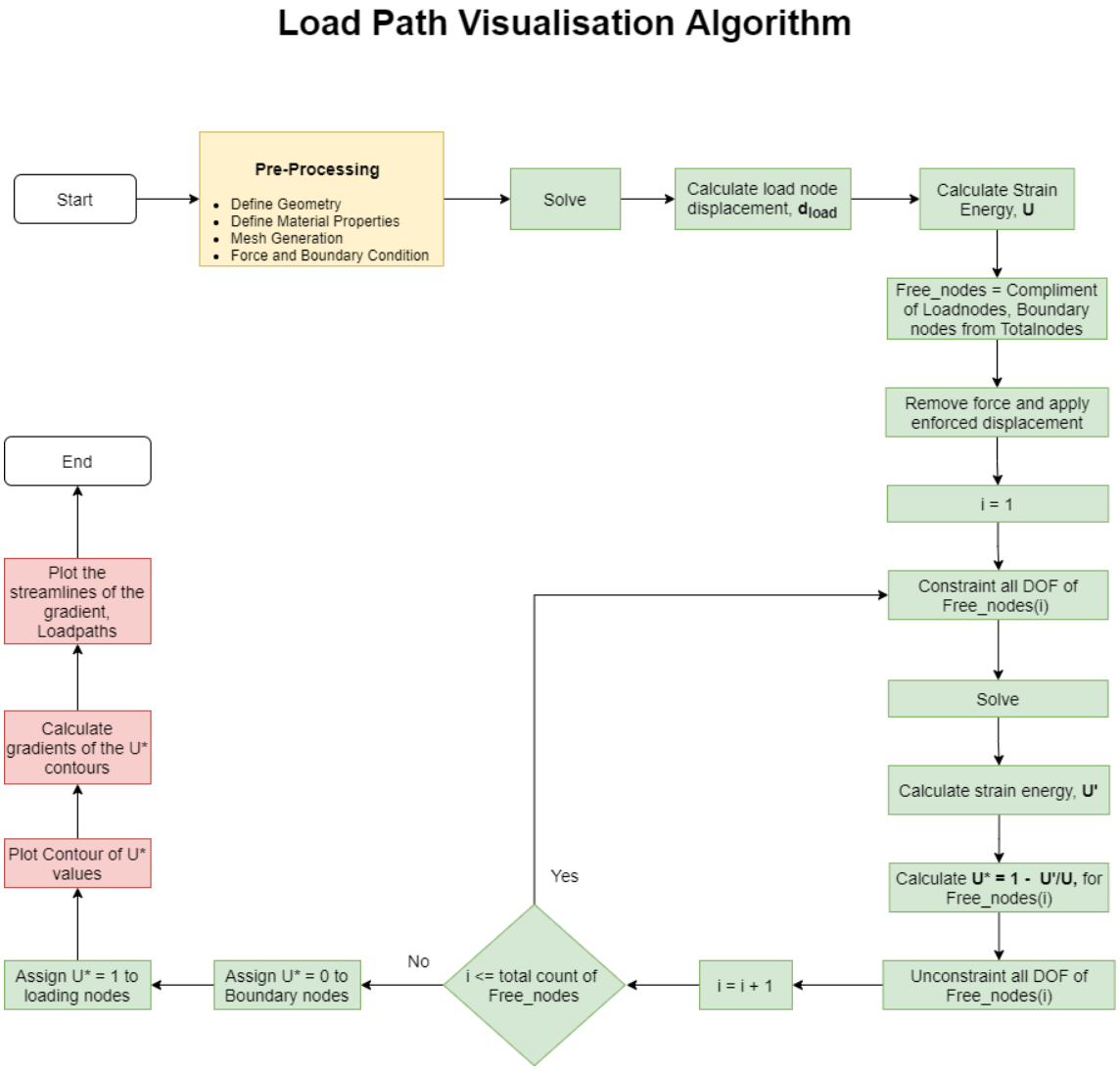


Figure 3.2: The algorithm to visualize load paths using U^* index method

8. The constrained DOFs of freenode is removed and the loop continues with the next freenodes.

9. Once the loop gets done, the U^* index values to the loading points and supporting points are assigned as 1 and 0.

10. The load paths are visualized by post-processing the results. To visualize the load paths, the streamlines can be plotted. The streamline plots line over the ridge of the contour lines. The streamlines require components of the scalar quantity (U^* index) in three dimensions. The components of the U^* index can be derived by taking the gradients of the U^* index in directions concerning their nodal coordinates.

3.3 Load Path Visualization - Routine Development through software

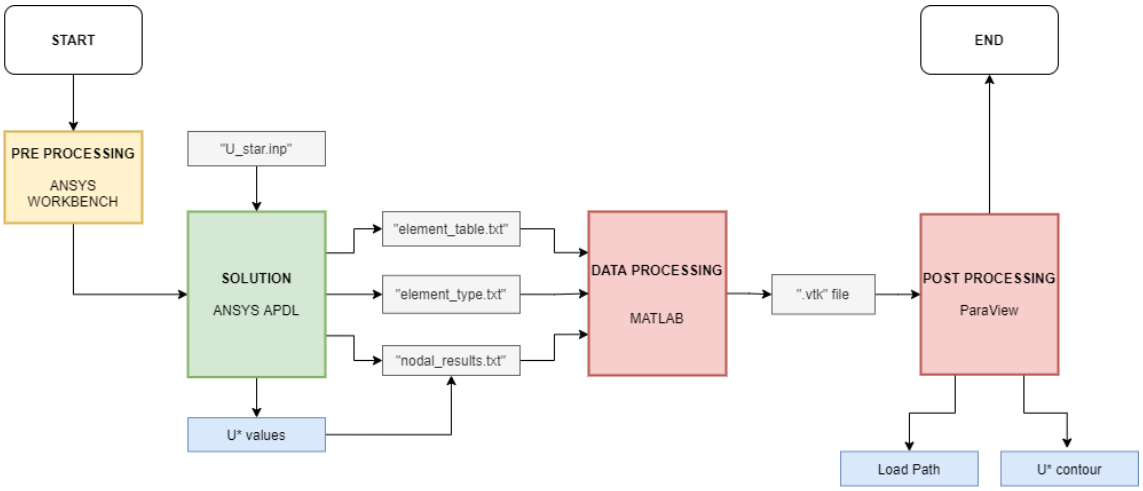


Figure 3.3: Load Path visualization routine in Software

Unlike traditional stress-strain analysis, the U^* star calculation requires a looping of solution routine under different modified physical constraints. To calculate the U^* field, a routine has to be defined in FEM software. The methods to implement the calculation of U^* field in the structure was investigated in various software. The Ansys Mechanical APDL was chosen to be a solver since it has great features that permit the user to define a user-defined calculation routine. Although the U^* can be calculated by using the Ansys Mechanical APDL, post-processing the result to visualize the contour and load paths are difficult using APDL. Hence, ParaView is chosen as a postprocessor is to visualize the contour of U^* and load paths.

Figure 3.3 illustrates the workflow carried in this thesis to visualize the load paths of a structure.

3.3.1 Preprocessing

The general preprocessing of a physical problem is carried out using commercial software Ansys Workbench where the geometry, material, mesh, element type are defined. Instead of applying the boundary and loading conditions, their corresponding nodes are selected as components using specific names as shown in the figure 3.4.

The “loadnodes” and “fixnodes” in the model tree in figure 3.4 corresponds to the nodes that are associated with loading and boundary conditions respectively. The load magnitude and orientation are applied using the Ansys Mechanical APDL.
The preprocessed model is exported to Ansys Mechanical APDL for the SOLUTION (U^* calculation) as shown in the figure 3.5.

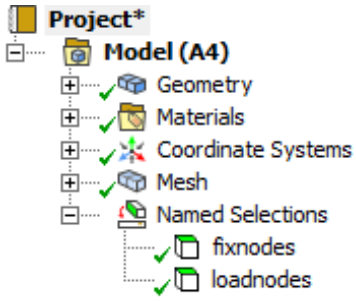


Figure 3.4: Model tree from Ansys Workbench with named selections used for Support and Loading nodes

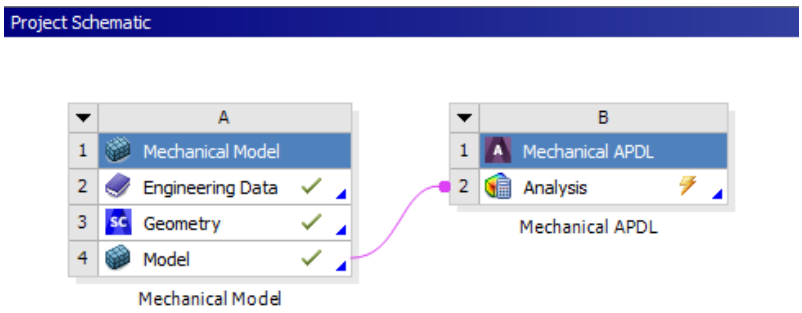


Figure 3.5: Transferring Preprocessed model to Ansys Mechanical APDL

3.3.2 Solution

APDL commands are used to define a user-defined to calculate the U^* field within the structure. A program with APDL commands is created in the ".inp" file format through which the commands can be imported to the Mechanical APDL solver. The program reads the load-nodes n_{load}, boundary-nodes $n_{boundary}$ data from the set of components created during pre-processing. The commands are programmed such that it follows the calculation of U^* routine.

Initially, the original loading case is solved. The strain energy U and load node displacements d_{load} are calculated. Then, for each free node n_{free}, a load step is written with a modified loading case (removing force and enforcing load node displacement d_{load}). So, the total number of load steps equals the total number of free nodes n_{free}. The programmed APDL commands solve all the load steps and the strain energy U' for all the load steps are calculated. From the set of strain energies for modified loading cases $U^{'}$, the index U^* is calculated by using the equation 2.1 for all nodes.

Further, the program is developed to export some information such as mesh element type, element nodal connectivity and U^* data along with nodal coordinates in ".txt" file formats as "element_type.txt", "element_table.txt" and "nodal_results.txt" respectively. These ".txt" file formats are exported for post-processing the results obtained from the Ansys Mechanical APDL solver.

In figure 3.3, “U_star.inp” contains the APDL commands, which is inputted to the Solver Ansys Mechanical APDL. The “U_star.inp” is appended to this report in Appendix A.1.

3.3.3 Post Processing

The solution is followed by post-processing of the results to visualize the load paths. Usually, the Ansys Mechanical APDL results are exported in the “.rst” file format. But, the ParaView has no reader to read the results directly from the “.rst” file format. However, the results obtained from the Ansys Mechanical APDL can be fed into the ParaView through “.vtk” file format. Thus, the result data from the Ansys Mechanical APDL are exported as “.txt” file formats.

3.3.3.1 Data Processing

A programming script is developed in Matlab to construct a “.vtk” format file. The Matlab program reads the results from “element_type.txt”, “element_table.txt” and “nodal_results.txt” files and generates a “.vtk” format file to transform results from the solver Ansys Mechanical APDL to post-processor ParaView.

The “.vtk” file format has a structure through which the data can be inputted to the ParaView. The general structure of the VTK file format is described in the section 2.7 in figure 2.6. However, the detailed structure of the VTK file format that is used in this thesis is described in the figure 3.6. ASCII type of dataset is chosen. From the five types of dataset formats described in section 2.7, the unstructured grid has been chosen since it supports the non-equidistant nodes.

A sample structure of the “.vtk” file generated by the Matlab used in this thesis is presented in the figure 3.6.

Table 3.1: Description of terms in VTK file structure in figure 3.6

SI	Variables	Description
1	n_{nodes}	Total number of nodes
2	n_{elem}	Total number of elements
3	U_n^*	U^* of nth node
4	$type_n$	Type of mesh element of nth element
5	P_{nx}	x-coordinate of nth node
6	P_{ny}	y-coordinate of nth node
7	P_{nz}	z-coordinate of nth node

In figure 3.6, the keywords are colour-coded to differentiate the keywords and variables for the explanation. The keyword “POINTS” is used to define the locations of the nodes requires two parameters (Total number of nodes and type of data). The keyword “CELLS” defines the element-nodal connectivity, requires two parameters

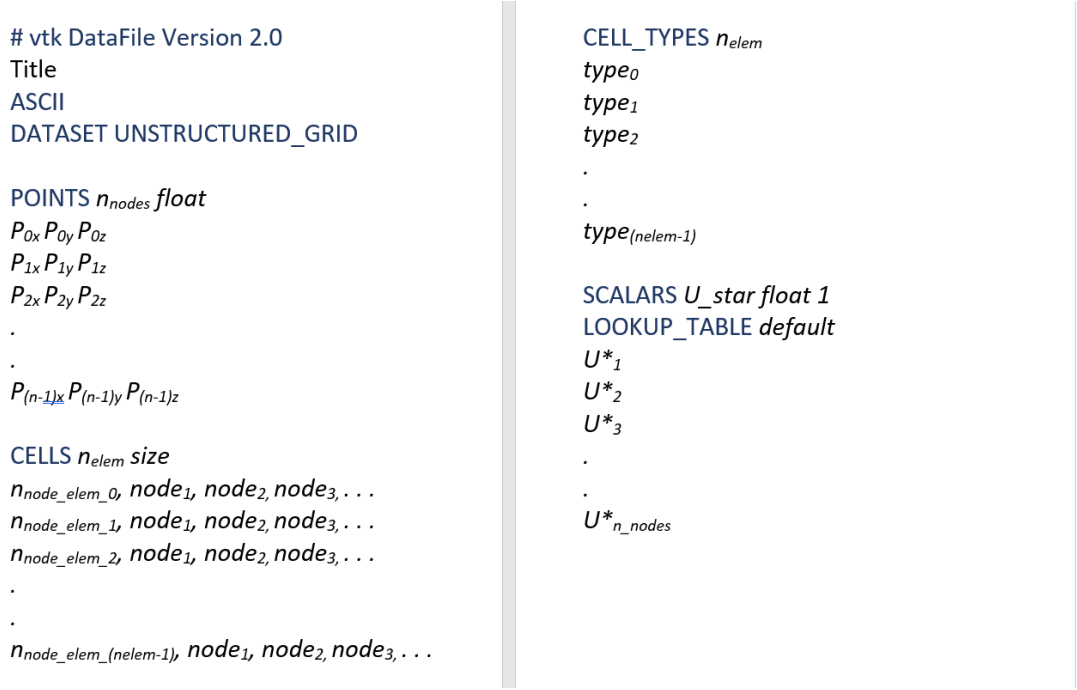


Figure 3.6: General Structure of the VTK structure for the ParaView input

(Total number of elements and total number of integer values required to represent the list). The keyword "CELL_TYPES" defines the type of element, requires one parameter (Total number of elements). The legacy type of VTK file format has a separate list of cell types. Hence, the APDL element type number and the matching cell type number must be studied and chosen accordingly. The keyword "SCALARS" is used to define the U^* values, requires two parameters (name of the scalar and type of the data).

The Matlab program was developed with some limitations as it supports only certain mesh elements from Ansys Mechanical APDL. The supported mesh elements are listed in table 3.2.

Table 3.2: Supported mesh element for data transfer

SI No	Mesh Element	Description
1	SHELL181	4-Node Structural Shell
2	PLANE182	2-D 4-Node Structural Solid
3	PLANE183	2-D 8-Node or 6-Node Structural Solid
4	SOLID185	3-D 8-Node Structural Solid
5	SOLID186	3-D 20-Node Structural Solid
6	SOLID187	3-D 10-Node Tetrahedral Structural Solid

The Matlab code for transferring the result from Ansys Mechanical APDL to ParaView is appended to this report in Appendix A.2.

3.3.3.2 Visualization

The ".vtk" format file generated from the Matlab is inputted to the ParaView. By reading the ".vtk" file, the ParaView plots the contour of U^* data set. The gradients are calculated by using the filter "GradientOfUnstructuredDataSet". From the gradients, the streamlines are plotted using the filter "StreamTracer". While plotting the streamlines, the locations of seed points for the streamlines should be chosen at the location of supporting nodes.

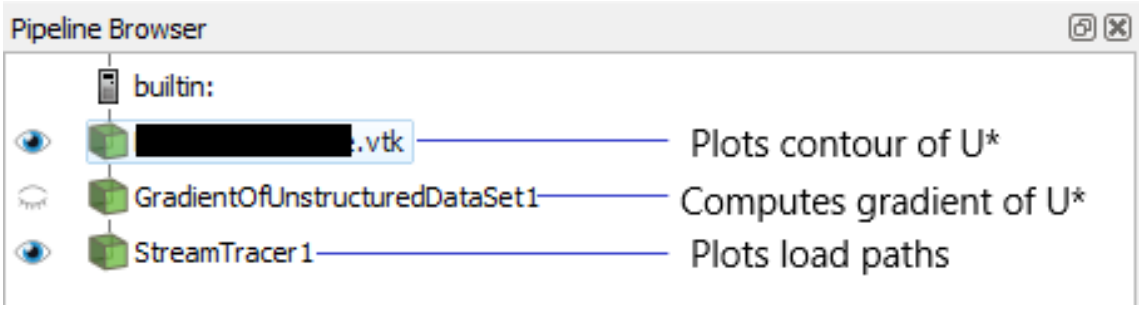


Figure 3.7: ParaView pipeline browser

Figure 3.7 shows the pipeline browser illustrating how the filters are used to visualize the load paths. The plot may show many streamlines. Thus, the streamlines flow from the nodes of support to the nodes of loading. But, not all the streamlines connect the supporting nodes to loading nodes. The streamlines that connect supporting nodes and loading nodes should be concentrated.

At the end of the flowchart on figure 3.3, the first three objectives mentioned in the section 1.4 are achieved.

3.4 Design evaluation based on the principle load path

According to the U^* theory, the structural design of a structure should be optimised based on the insights obtained from the evaluation of load paths. The load paths are to be evaluated using the design criteria as explained in section 2.3.3 which requires continuity plot and uniformity plot. Hence, this section explains how the Uniformity and continuity plots are developed for load paths.

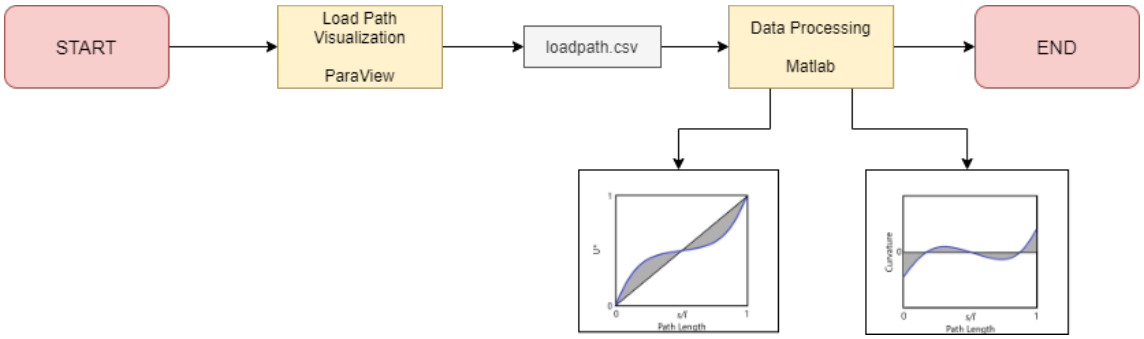
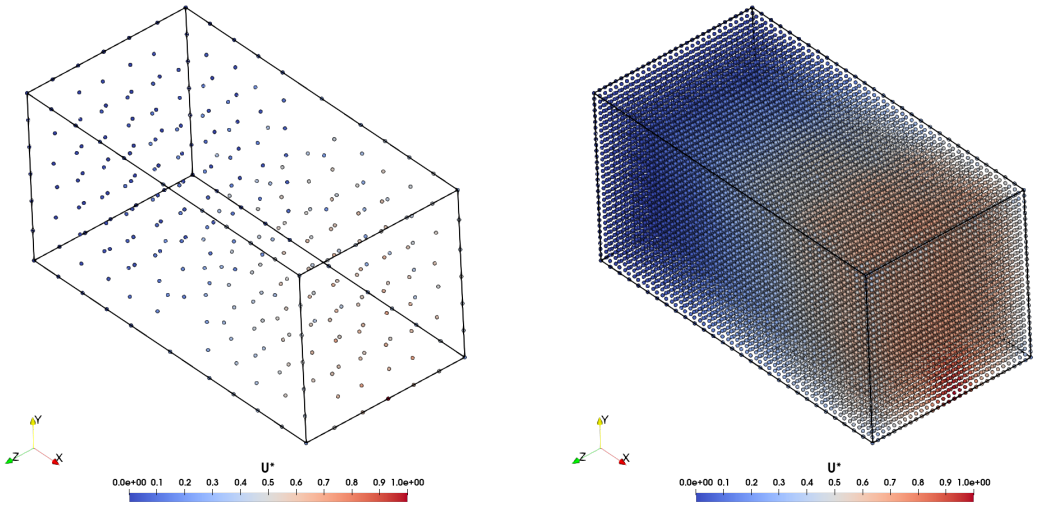


Figure 3.8: Routine for Load Path evaluation

To plot the continuity plot and uniformity plot, Matlab is used in this thesis. The data of the principal load path containing the U^* values are extracted from ParaView and exported in the ".csv" file format. The ".csv" file is imported into Matlab and programmed to plot continuity and uniformity plots.

3.5 Visualization Enhancement

In this study, an approach is developed to enhance the visualization of the load paths and avoid the discontinuities of the load paths. The approach includes the construction of more nodes with interpolation of the nodal values from the solution of U^*. Hence, the streamlines take advantage of more nodal locations and produce a smooth visualization of load paths. This can be achieved can using the in-built filter "ResampleToImage" in ParaView.



(a) Actual nodes **(b)** Additional nodes

Figure 3.9: Generation of additional nodes in ParaView

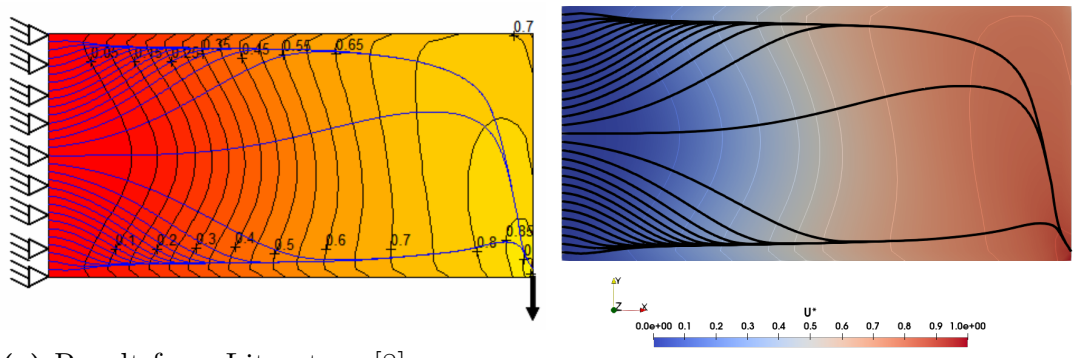
4

Results and Discussion

In this chapter, the results obtained from developed routine is presented. The results includes contour of U^*, visualization of load paths. The influence of mesh element, mesh size, boundary condition, loading conditions are explained.

4.1 Verification of results for 2D problem

As mentioned in the section 3.1, simple 2D problems are considered to develop routines to visualize load paths and to verify with the results from the literature. Figure 4.1a shows the comparison between the result of a 2D cantilever problem from the literature [8] and the result for the same problem obtained from the developed routine explained in section 3.3.



(a) Result from Literature [8]

(b) Result from developed methodology

Figure 4.1: Load Path visualization on 2D cantilever beam

Plane stress condition with thickness and quadrilateral mesh element were considered in the FEM calculation for figure 4.1b. It is obvious that load paths in figure 4.1b are in correspondence with load paths in figure 4.1a. Thus, it can be concluded that the developed routine delivers the desired results for the 2D structural problem.

4.2 Routine test - 3D Cantilever problem

Once the results obtained from the developed routine are verified for 2D problems, the routines are tested for 3D problems. For simplicity, a 3D cantilever problem is studied to understand the behaviour of load paths under different mesh elements, different element size and different types of loading cases. Further, the routines are tested for 3D complex shape.

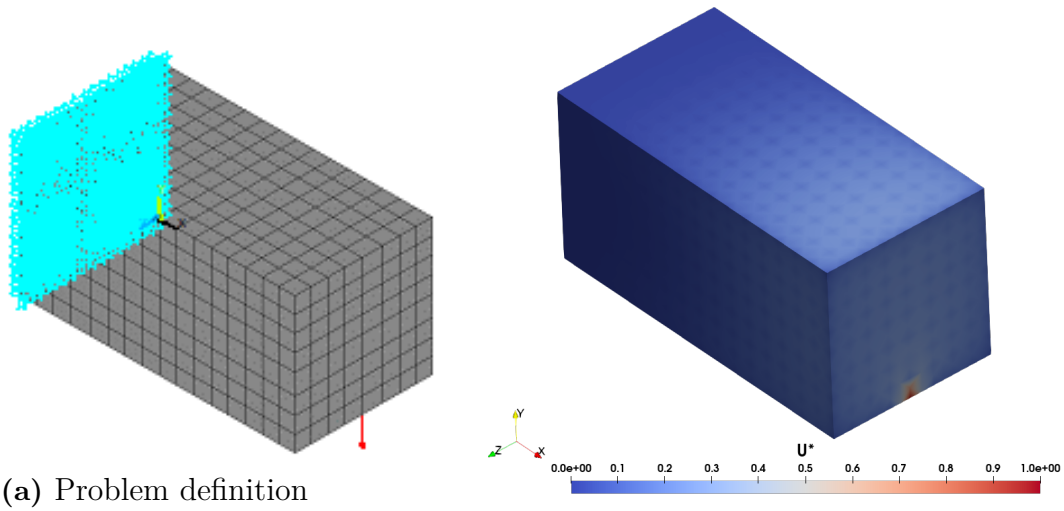
load paths in the 3D cantilever beam are investigated under different loading conditions such as point load, distributed load on the edge and distributed load on the face to study the influence of types of loading on load paths. Also, the influence of the type and size of the mesh elements and computational time are studied.

4.2.1 Type of loading

Point Load

A 3D cantilever beam is point-loaded as shown in the figure. A study on the influence of the type of mesh element is carried out for the point load loading condition. Two mesh elements were investigated for the point loaded cantilever problem. First, the results for second-order hexahedral mesh is studied. A fine mesh of second-order hexahedral mesh element is used to visualize load paths. The developed routine is used to calculate the U^* and visualize load paths.

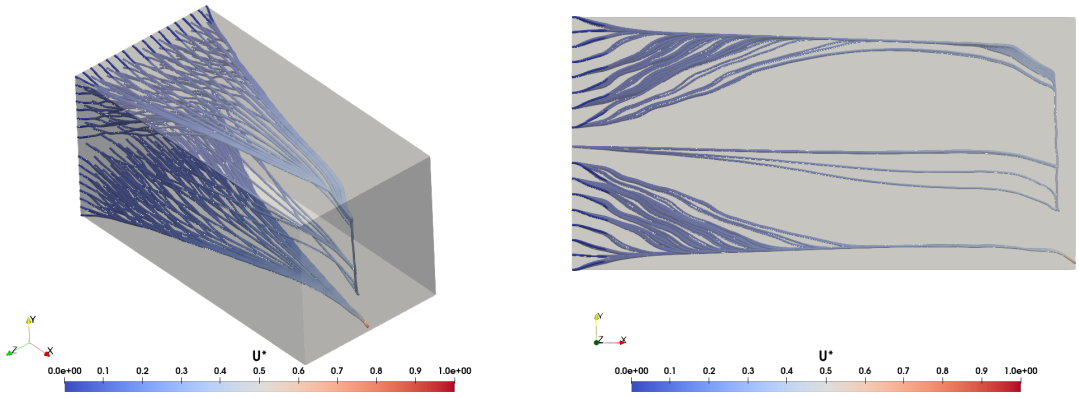
The U^* contour and load paths obtained from the developed routine for the point loaded cantilever beam with second-order hexahedral mesh are presented in the figures 4.2 and 4.4.



(a) Problem definition

(b) U^* contour

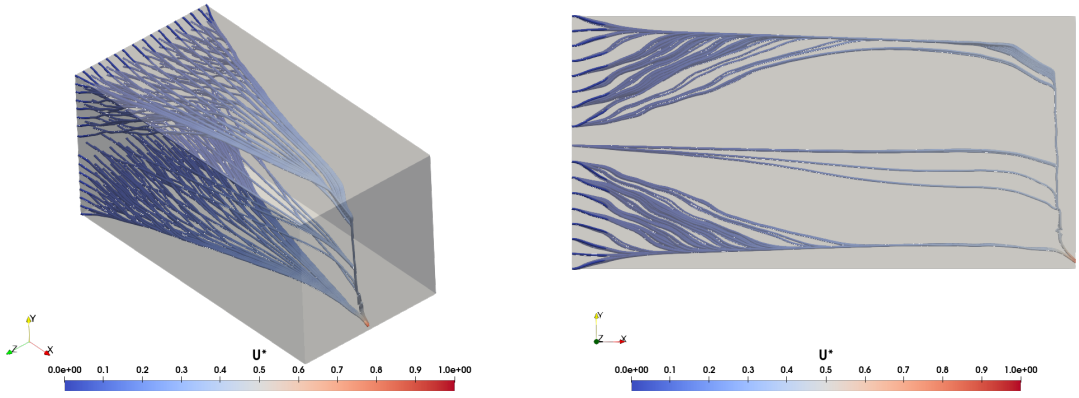
Figure 4.2: Cantilever beam point load [Hexahedral mesh]



(a) Isometric view **(b)** Front view

Figure 4.3: Discontinuity of Load paths

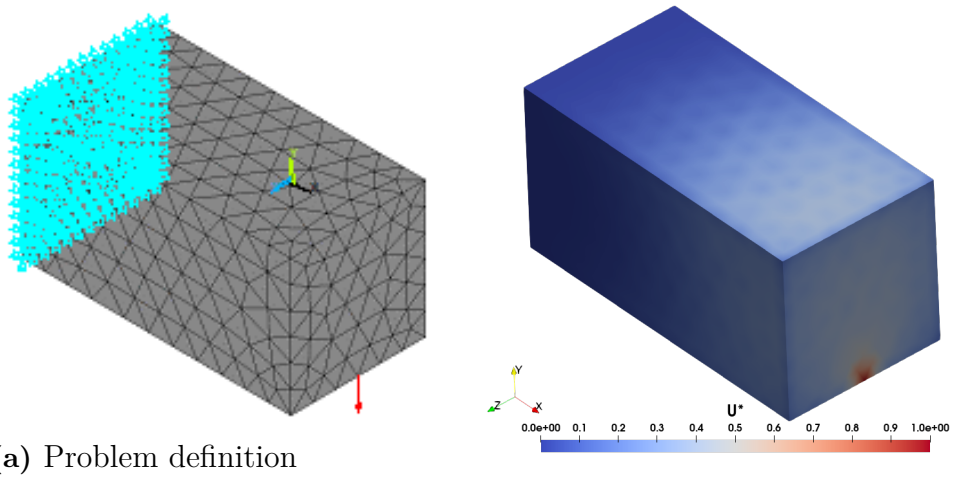
It is observed from the figure 4.3, that there exists a discontinuity in load paths arising from the boundary nodes near the loading node. This discontinuity of load paths may be due to the size of the mesh is inefficiently larger to take gradient as the U^* values take a sharp increase near the loading node. However, this discontinuity can be fixed using plotting streamlines seeding from near the point of discontinuity. The complete visualizations of load paths for point loaded cantilever beam are presented in figure 4.4.



(a) Isometric view **(b)** Front view

Figure 4.4: Load paths visualization for cantilever beam point loaded [hexahedral mesh element]

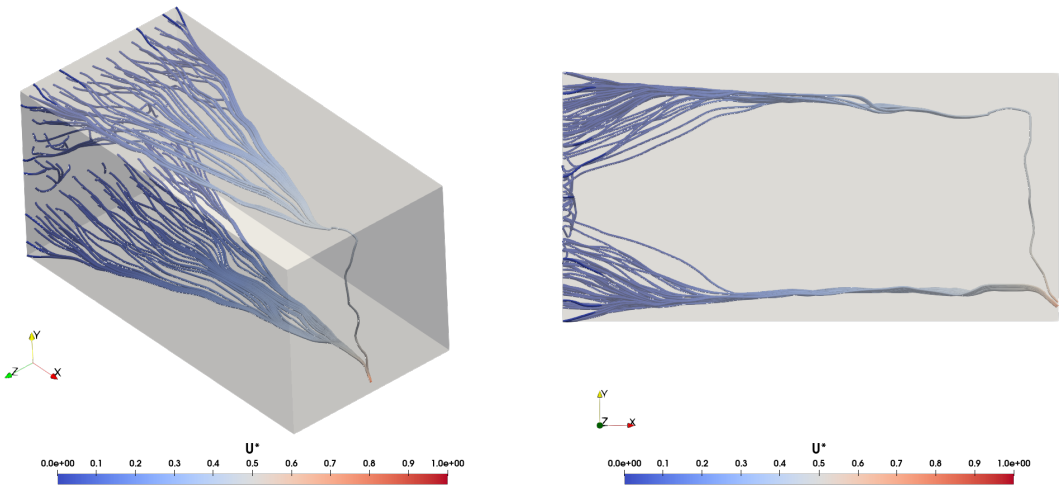
Further, the same problem is studied with the second-order tetrahedral mesh element. The U^* contour and load paths obtained from the developed routine for the point loaded cantilever beam with second-order tetrahedral mesh are presented in the figures 4.5b and 4.6.



(a) Problem definition

(b) U^* contour

Figure 4.5: Cantilever point loaded [Tetrahedral mesh]



(a) Isometric view

(b) Front view

Figure 4.6: Load path visualization for cantilever beam point loaded [Tetrahedral mesh element]

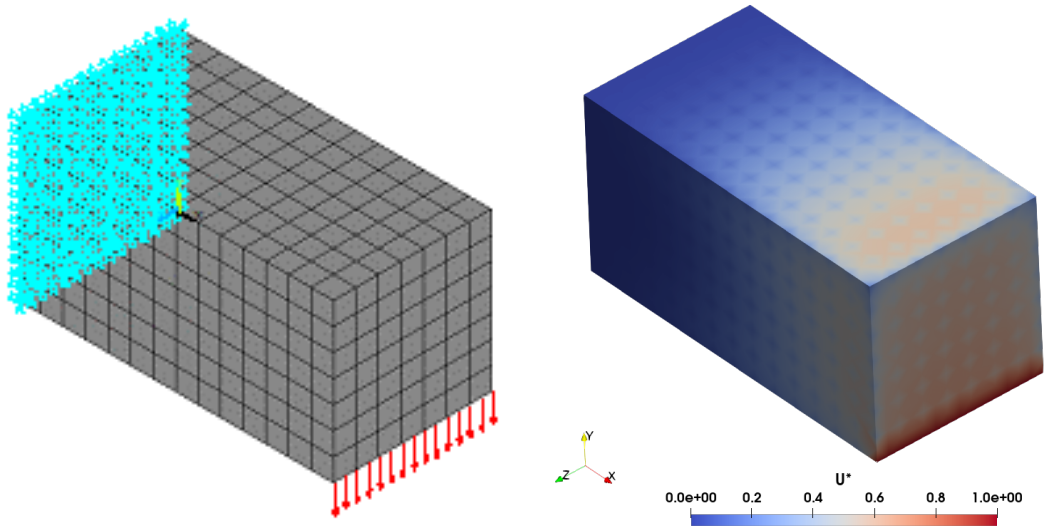
Although the U^* contour for both mesh elements are similar, some dissimilarities are observed for load paths. load paths in the figure 4.6 are irregular and unsmooth. In addition to that, some of the load paths that originate from supporting nodes are not connected to the loading node. Instead, they get diverted back to the supporting nodes. Whereas, for the hexahedral mesh (see figure 4.4), all load paths that originate from the support point connects to the loading point which provided comparatively reliable results. Thus, the second-order tetrahedral mesh produces poor results whereas the second-order hexahedral mesh provides regular and smooth results. Hence, it can be concluded that the second-order hexahedral mesh is the suitable mesh element for the visualization of load paths.

Distributed Load

The developed routine is tested on a 3D cantilever beam for the distributed load. This is done to study the behaviour of the load path for distributed loading in 3D space. For distributed loading, two loading cases were considered. First, the distributed load is applied vertically to the bottom edge of the free end of the cantilever beam as shown in figure 4.7a. Then, the distributed load is applied vertically on the face of the free end of the cantilever beam.

Edge load

The figure 4.7b shows the contour of U^* distribution for the first distributed loading case.

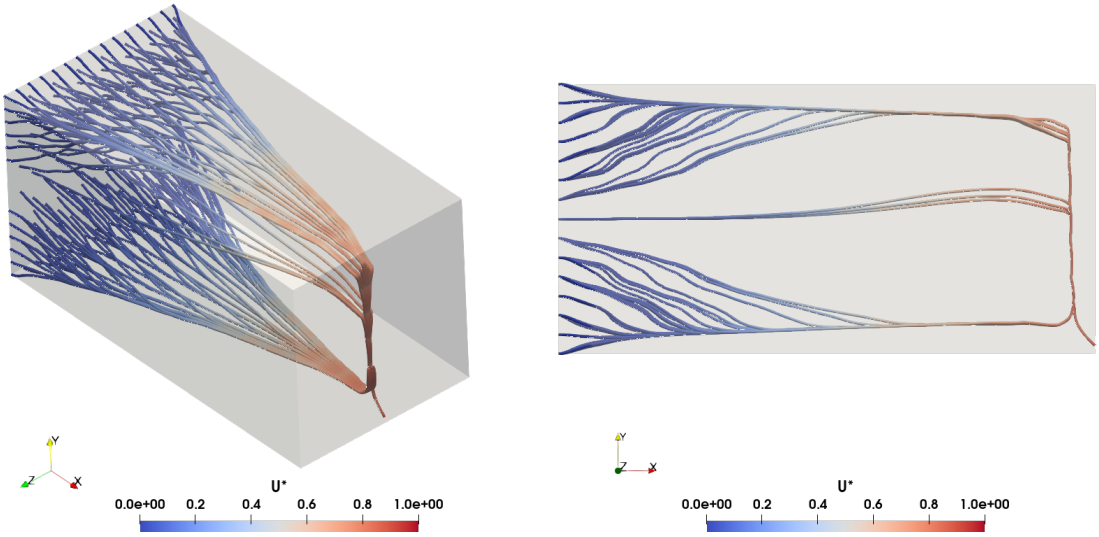


(a) Problem definition

(b) U^* contour

Figure 4.7: Cantilever beam with distributed load along the edge.

Figure 4.8 represents two different views of visualization of load paths obtained from the developed routine. load paths in figure 4.8a provide a controversial argument for the distributed loading as follows. Figure 4.8a represents load paths that originate from the support nodes that get converged to a single loading node. This convergence of load paths is not acceptable for the distributed load, as it portrays the inactive participation of the rest of the loading nodes. The results obtained are similar to the case of point load. The reason is as follows.



(a) Isometric view **(b)** Front view

Figure 4.8: Load path visualization for cantilever beam with distributed load on the edge.

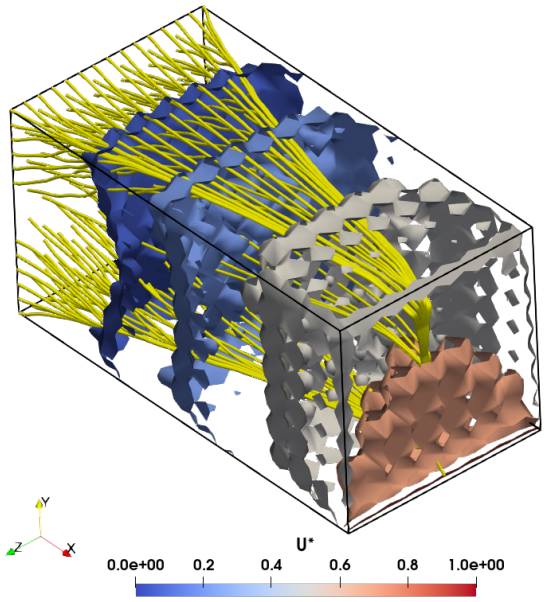
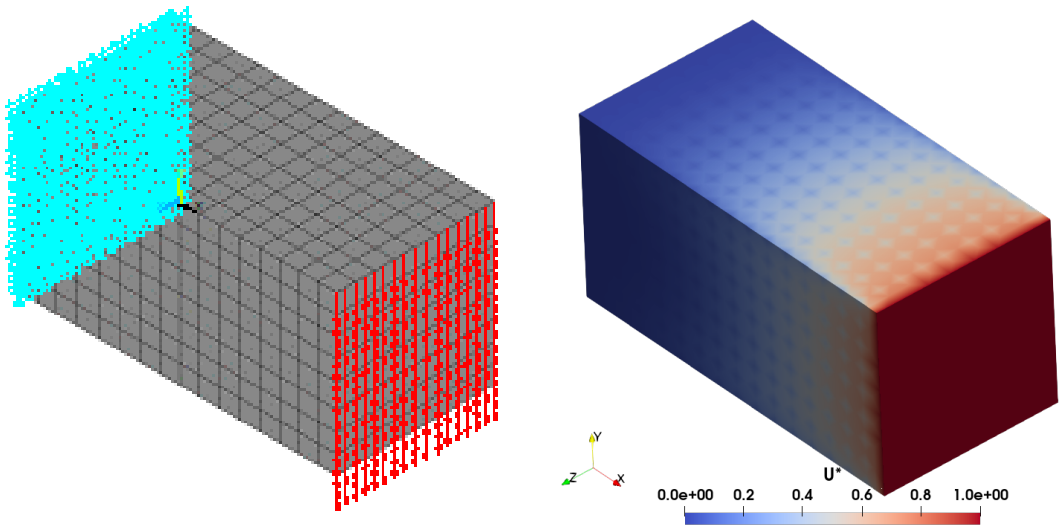


Figure 4.9: Representation of load paths passing through the contour planes for the cantilever beam with edge distributed load.

Figure 4.9 illustrates how load paths passes through the contour planes. The yellow lines in figure 4.9 represents load paths . Since the loading nodes has U^* index value of 1, the contour planes except near loading nodes take concave shape. When load paths pass through the least/highest gradient of the contour, they tend to converge. Further, it is observed that the contour plane with U^* index of 0.9 is flat and has no gradient. Thus, load paths are connected to the middle of the contour plane. This is the reason for the convergence of load paths to a single loading node.

Surface load

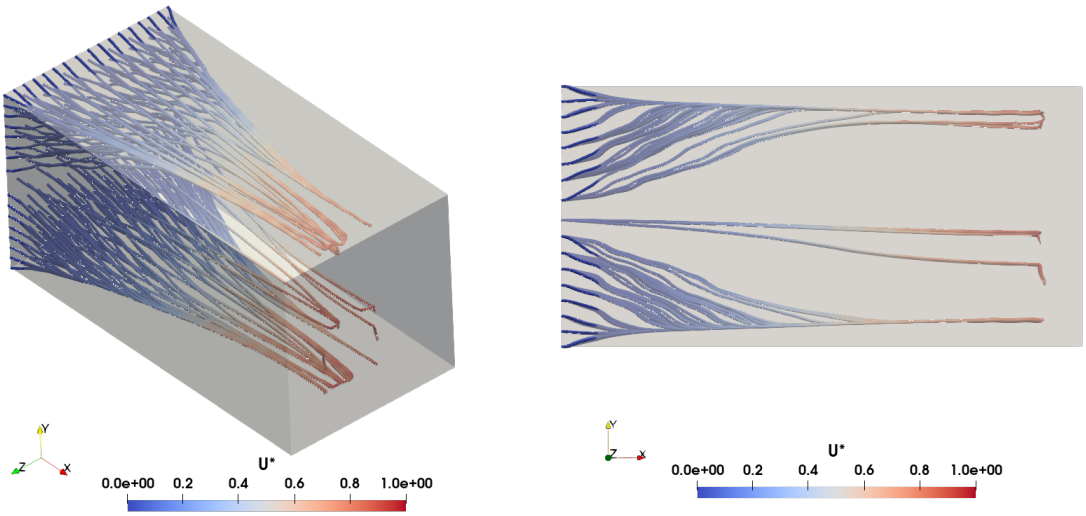
The figure 4.10b shows the contour of U^* distribution for the the second distributed loading case.



(a) Problem definition

(b) U^* contour

Figure 4.10: Cantilever beam with distributed load along the surface.



(a) Isometric View

(b) Front View

Figure 4.11: Load path visualization for the surface loaded cantilever beam

Figure 4.11 visualizes load paths for the surface loaded cantilever beam in two different views. It can be observed that load paths do not connect to the loading nodes. Thus, it is observed from both loading cases, load paths are experiencing problem near the loading nodes due to the contour plane shapes of the U^*. The contour planes for the surface loaded cantilever problem is in the figure 4.12. Since

the contour plane at the loading surface is flat, the gradients are zero. Hence, the streamlines do not connect to the loading nodes.

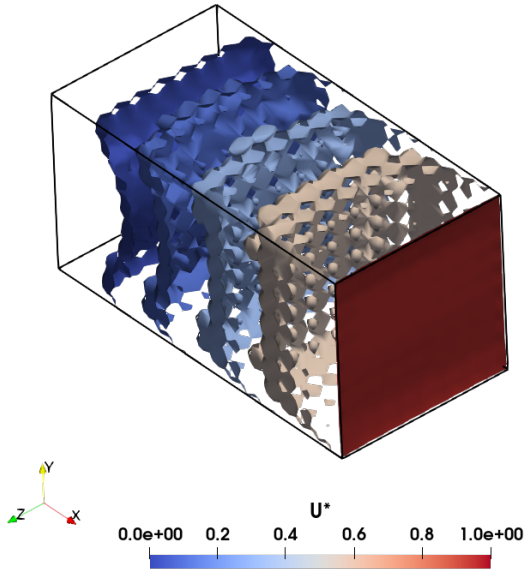
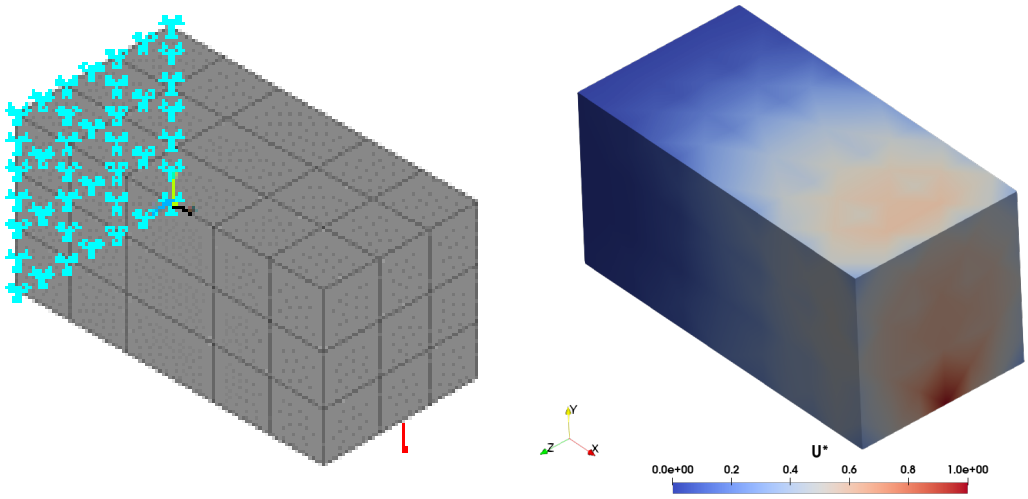


Figure 4.12: Contour plane for the cantilever beam with distributed load along the surface

Thus, the U^* is not suitable for the distributed load. For distributed load, a concept U^{**} [9] is introduced which is an extension of the U^* concept. The U^{**} is based on the complementary energy which deals with the internal compliance between loading points and arbitrary points within the structure.

4.2.2 Mesh Size

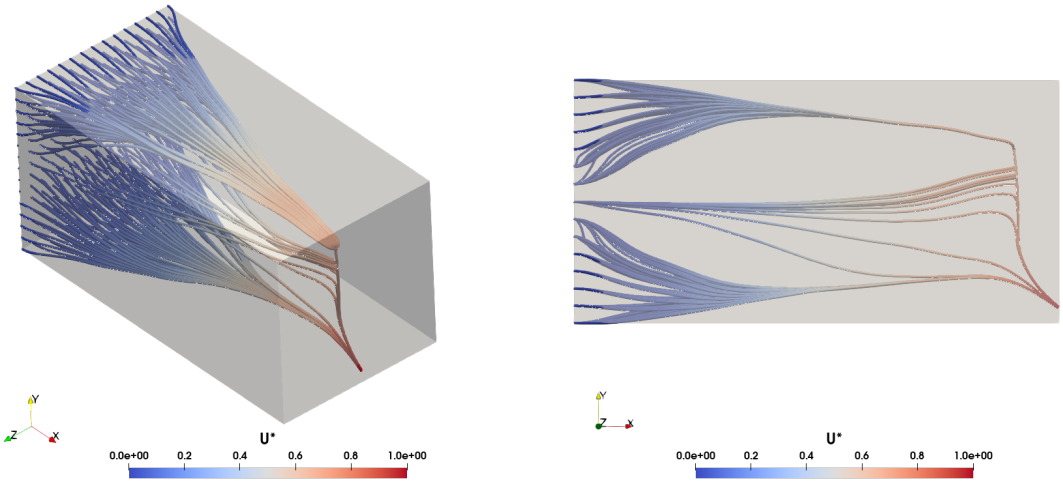
The results from point loaded cantilever beam for fine second-order hexahedral mesh is compared with the results from point loaded cantilever beam for a coarse second-order hexahedral mesh. The representation of considered coarse mesh is in figure 4.13a. Further details are provided in the table 4.1.



(a) Problem definition

(b) U^* contour

Figure 4.13: Cantilever beam point loaded (coarse mesh)



(a) Isometric view

(b) Front View

Figure 4.14: Load path visualization for the point loaded cantilever beam (Coarse mesh)

Despite slight variation, load paths in figure 4.14 are similar to load paths for fine mesh in figure 4.4. However, the quality of the information that load paths provide has to be studied. It can be done by plotting the uniformity plot for both meshes. The uniformity plot (discussed in section 2.3.3) is plotted for load paths that arise from a lower support node for both fine mesh and coarse mesh. The comparison of uniformity plot are in figure 4.15.

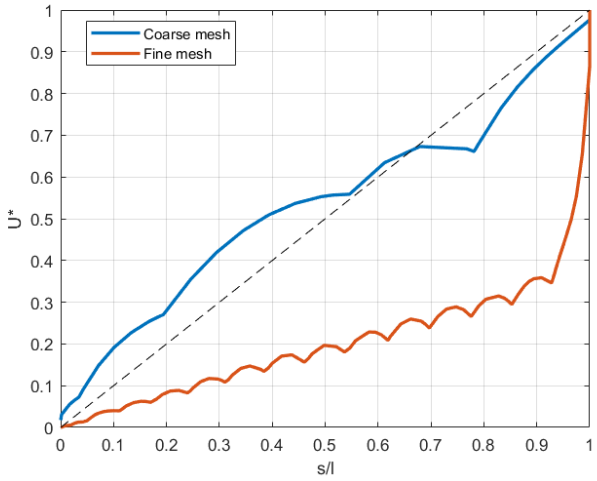


Figure 4.15: Uniformity plot for load paths for Fine and Coarse mesh

It is observed from the figure 4.15, there exists a significant difference in U^* variation between fine mesh and coarse mesh. Since the cantilever beam is point loaded, high stresses exist close to the loading node. This information can be figured out from the uniformity plot, as the U^* takes a sharp increase at the end. On the other end, for coarse mesh, the U^* exhibits comparatively linear variation which is not acceptable/reasonable for the point load. Thus, it can be concluded that coarse mesh provides the visualization of load paths with slight inaccuracy. But, it provides an unreliable result for the stiffness distribution.

Table 4.1: Comparison between fine mesh and coarse mesh

Properties	**Fine Mesh**	**Coarse Mesh**
Total Number of nodes	5121	320
Fixed nodes	225	40
Load nodes	1	1
Free nodes	4895	279
CPU time [s]	3805	25.3

Although the fine mesh provides desired results, from table 4.1, it is observed that the CPU time is enormous for the fine mesh when compared to the coarse mesh.

4.2.3 Computational Time

Since the routine to calculate U^* includes looping of the solution under different modified conditions, the computational cost for the visualization of load paths has to be considered seriously. For the point loaded cantilever beam, the computational time to solve for U^* is recorded for various mesh sizes. The computational time is plotted against the total number of nodes in figure 4.16.

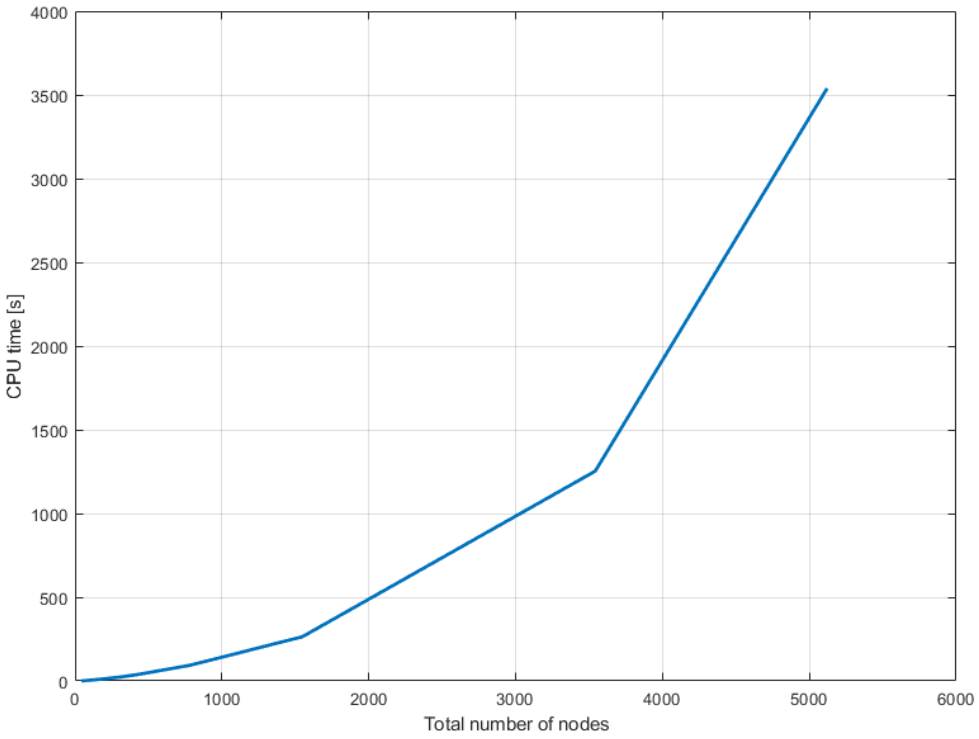


Figure 4.16: Computational time vs Total number of nodes.

From figure 4.16, it can be inferred that, the computational time required to get a reliable result increases exponentially with the total number of nodes in the geometry. From the conclusion of section 4.2.2, it is advisable to have a fine mesh to get clear detailed insights to improve the design. Thus, it is observed for a simple cantilever problem, the solution becomes computationally heavy to obtain reasonable results. This exponential increase in computational time is a serious problem when comes to the application in aero-engine structures. As the finite element model of aero-engine structures may have a large set of nodes that may require months to visualize load paths. Although it is desired to have a fine mesh for good results, the effect of computational time creates a contradiction for the use of fine mesh.

4.2.4 Conclusions

From the study on cantilever beam to visualise load paths under different cases, certain cases are filtered out that are most applicable for the U^* conce. They are the following,

1. Type of Loading: Point load
2. Type of mesh: Fine and Moderate
3. Type of mesh element: Hexahedral mesh element

4.3 Routine test - 3D Complex Structure

The robustness of the developed load path visualization routine has to be tested before its application in aero-engine structures. For this purpose, a complex structure from the previous research [3] at *GKN Aerospace* is considered. The geometry and the loading case of the structure are provided in the figure 4.17. The loading cases are assumed such that load paths would take diversions in three directions. From the study on the cantilever beam, a point load is applied, a moderate mesh is used and a second-order hexahedral mesh element is considered.

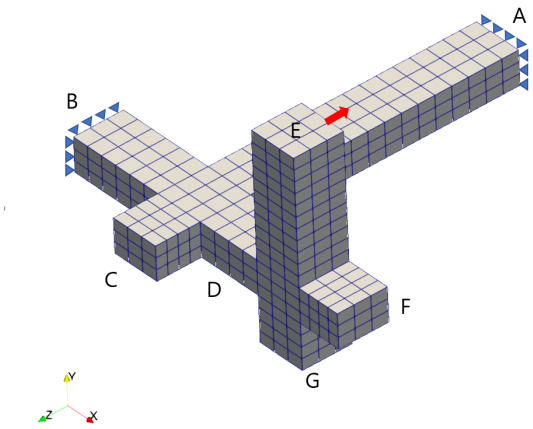
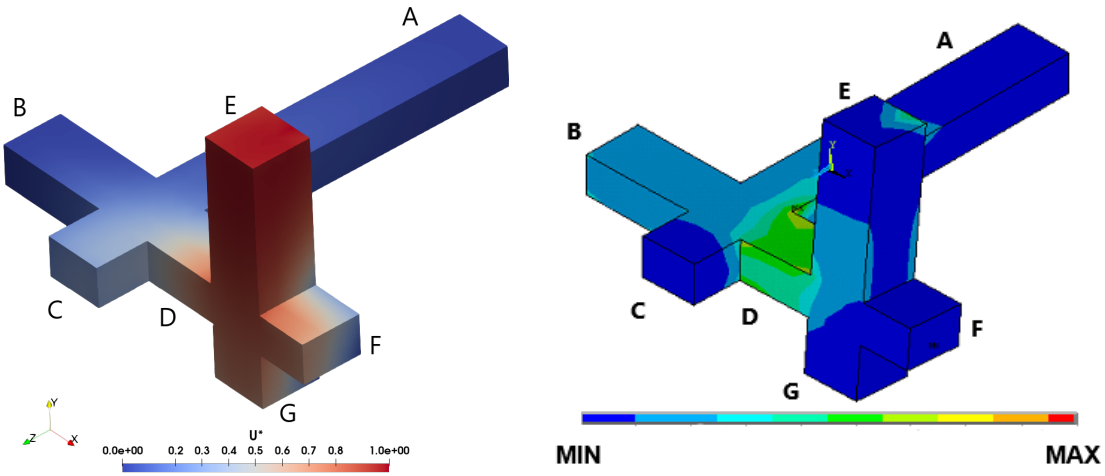


Figure 4.17: Problem definition for the complex structure

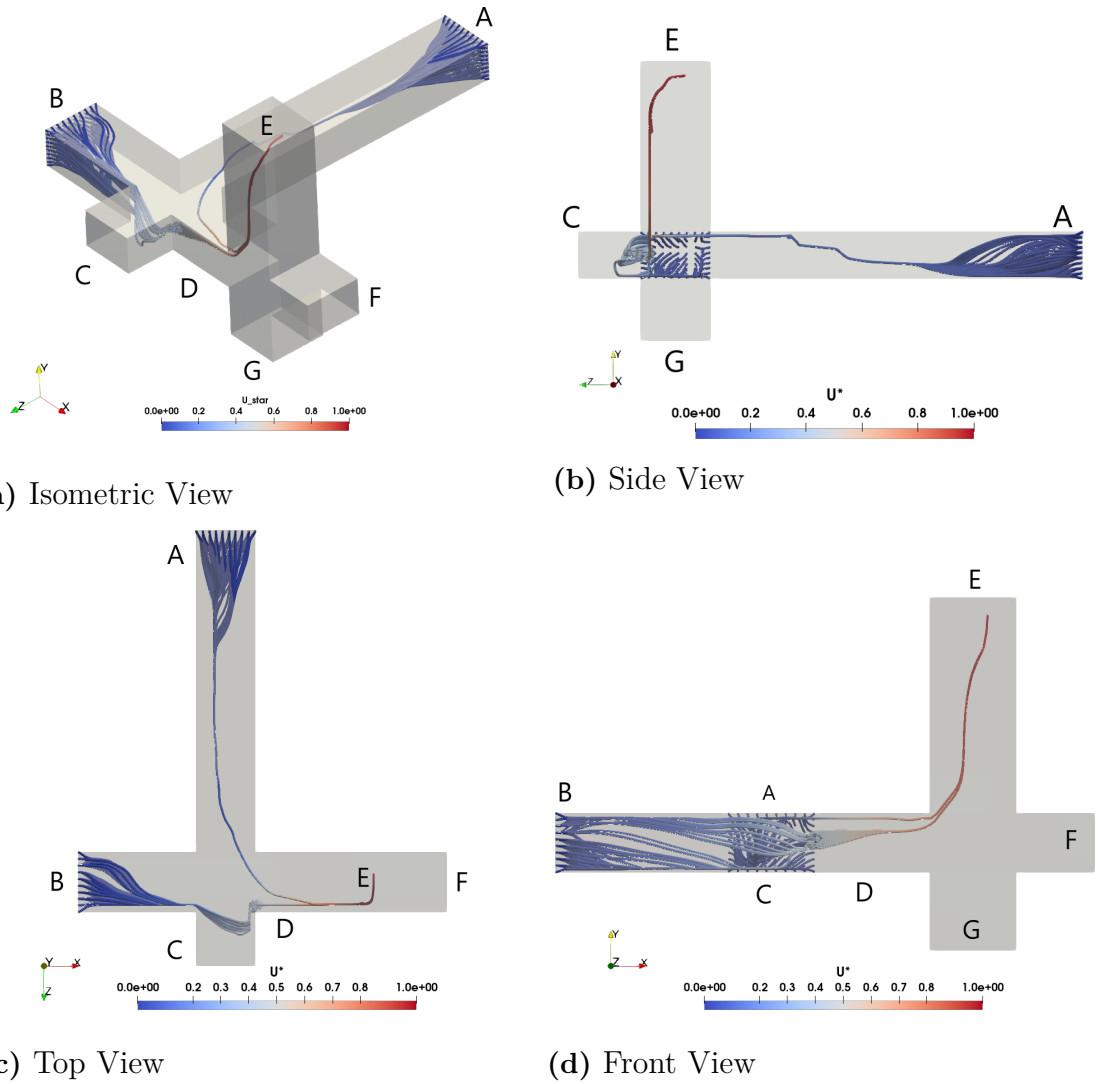


(a) U^* contour **(b)** Von mises stress contour

Figure 4.18: Comparison of U^* and Stress contours for the complex structure

Figure 4.18 shows the comparison between the U^* contour and stress contour for the complex structure. Stress concentrations can be observed in figure 4.18b at the region D due to the influence of bends in the structure.

The developed routine visualizes load paths for the complex structure for the assumed loading case. The visualization of load paths from different perspectives for the complex structure is provided in the figure 4.19.



(a) Isometric View

(b) Side View

(c) Top View

(d) Front View

Figure 4.19: Load path visualization for the complex structure

Design Improvement suggestions

Figure 4.19 clearly illustrates how the load is being transferred from the support areas to the loading point. Also, it indicates what regions of the structure actively contributes to the load distribution and inactive regions. This structure can be structurally improved by considering the design criteria as explained in the section 2.3.3. It would result in improvement of the stiffness of the structure.

Based on the load path visualization, the volume of the regions G and F in the figure 4.19a can be reduced, since it does not contribute to the assumed loading case. The

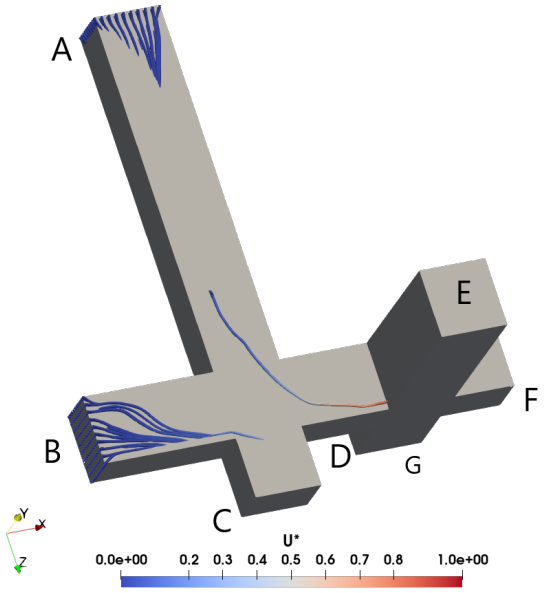


Figure 4.20: Load path taking surface for the load transfer

region D in the figure 4.19a requires attention because it provides a channel for load paths originating from both support ends to connect to the loading node. Load paths flowing through region C in the figure 4.19c has sharp curvatures which may cause improper stiffness distribution. Load paths should have smooth curvatures for better load transfer. Thus, the design can be improved by creating fillets on the adjacent sides of region C which enhance smoother curvatures for load paths.

It is observed from figures 4.19 and 4.18b wherever the regions that load paths passes close to the boundaries of the geometry except for the support regions, the regions experience high stresses. It may be due to load paths that tend to move out of the geometry space, get diverted by the surface constraint and passes along the surface. As a consequence, load paths takes the surface of the geometry as a medium for the load transfer which is evident in figure 4.20. This would subject the structure to higher risk. For instance, if a crack was initiated on the surface where the load path pass, it would increase by successive loading. Thus, load paths take another direction which may affect the stiffness distribution and result in failure of the structure. Hence, the design must be modified such that alternate load paths should exist in case of failure of one of the load paths.

5
Conclusion and Future work

This chapter includes the concluding remarks and some suggestions which can be considered for further development of the routines.

5.1 Concluding Remarks

Developing routines to calculate the U^* and to visualize load paths in structural components that could be used by *GKN Aerospace* for further use was the main interest in this thesis. It was achieved by using different software. The FEM calculations to calculate U^* were performed by developing a program script in Ansys Mechanical APDL. The visualization process is carried by both Matlab and ParaView. The developed routine was verified with literature results. For 3D structures, a study is carried out under different cases using the developed routine. From the study, it was suggested, a geometry with a fine mesh of second-order hexahedral mesh element with point load is more suitable for the visualization of load paths using the U^* concept. The mesh size and the computational time plays a significant factor in deriving insights in this thesis. Fine mesh provides desirable result but the computational cost becomes a major drawback for the U^* approach. Due to the effect of computational time, the routine was not performed on the aero-engine structures in this thesis as it may take a long duration or may require special computers to visualize load paths within the time frame of this thesis. The developed routine was tested on a complex structure and design improvement suggestions were provided.

5.2 Future Work

This thesis had few limitations and the scope of the thesis was reduced to fit the thesis within the time frame. Hence, the future work which has to be carried out at *GKN Aerospace* after this thesis should focus on those limitations and other work as follows.

- The Matlab script developed to transfer the results from Ansys Mechanical APDL to ParaView supports certain mesh elements as mentioned in table 3.2. The Matlab program script can be updated that supports more mesh elements.

- The routine to visualize the load paths based on the reaction forces for the Consistency plot (explained in the section 2.3.3) can be developed.

- The discontinuity in load paths visualization using ParaView makes hard to construct uniformity and continuity plots. Hence, future work should focus on methods to solve the discontinuity in load paths.

- Since the direct method of calculation U^* is computationally heavy, the future work should focus on alternative U^* calculation methods such as Inspection Loading method and Inverse Stiffness method [11].

- The existing routine can be extended that supports the concept U^{**} [9] to visualize load paths for distributed loading conditions.

- The existing routine can be extended to support non-linear material models [15].

Bibliography

[1] Facts about GKN Aerospace Sweden AB http://aerospace.se/wp-content/uploads/2013/12/FAKTABLAD_GKN-Aerospace-Sweden_131209.pdf

[2] Raja, Visakha, On the Design of Functionally Integrated Aero-engine Structures: Modeling and Evaluation Methods for Architecture and Complexity, Chalmers University of Technology, Doctoral thesis, 2019.

[3] Raja, V., Kokkolaras, M. & Isaksson, O. A simulation-assisted complexity metric for design optimization of integrated architecture aero-engine structures. Struct Multidisc Optim 60, 287–300 (2019). https://doi.org/10.1007/s00158-019-02308-5

[4] Frone corporation, The CAE solution provider https://frone.jp/

[5] Daryl L. Logan (2011). A first course in the finite element method. Cengage Learning. ISBN 978-0495668251.

[6] Granger, Robert A.. (1995). Fluid Mechanics - 8.2 Equation of a Streamline. (pp. 422-424). Dover Publications. Retrieved from https://app.knovel.com/hotlink/pdf/id:kt00B0E7C8/fluid-mechanics/equation-streamline

[7] H. Hoshino, T. Sakurai, K. Takahashi, Vibration reduction in the cabin of heavy-duty trucks using the theory of load transfer paths, JSAE Review, 24 165-171 (2003).

[8] Marhadi K, Venkataraman S. Comparison of quantitative and qualitative information provided by different structural load path definitions. International Journal for Simulation and Multidisciplinary Design Optimization. 2009 Jul 1;3(3):384-400.

[9] Wang, E.Y., Nohara, T., Ishii, H., Hoshino, H., Takahashi, K., 2010. Load Transfer Analysis Using Indexes U* and U** for Truck Cab Structures in Initial Phase of Frontal Collision. AMR 156–157, 1129–1140. https://doi.org/10.4028/www.scientific.net/amr.156-157.1129

[10] Naito T, Kobayashi H, Urushiyama Y. Application of load path index U* for evaluation of sheet steel joint with spot welds. SAE Technical Paper; 2012 Apr 16.

[11] Sakurai T, Takahashi K, Kawakami H, Abe M. Reduction of calculation time for load path U* analysis of structures. Journal of Solid Mechanics and Materials Engineering. 2007;1(11):1322-30.

[12] Sakurai T, Tada M, Ishii H, Nohara T, Hoshino H, Takahashi K. Load path U* analysis of structures under multiple loading conditions. Nippon Kikai Gakkai Ronbunshu A Hen(Transactions of the Japan Society of Mechanical Engineers Part A)(Japan). 2007 Feb;19(2):195-200.

[13] Shinobu M, Okamoto D, Ito S, Kawakami H, Takahashi K. Transferred load and its course in passenger car bodies. JSAE review. 1995 Apr 1;16(2):145-50.

[14] Naito T, Kobayashi H, Urushiyama Y, Takahashi K. Introduction of new concept U* sum for evaluation of weight-efficient structure. SAE International Journal of Passenger Cars-Electronic and Electrical Systems. 2011 Apr 12;4(2011-01-0061):30-41.

[15] Wang, Q., Pejhan, K., Telichev, I. et al. Extensions of the U* theory for applications on orthotropic composites and nonlinear elastic materials. Int J Mech Mater Des 13, 469–480 (2017). https://doi.org/10.1007/s10999-016-9348-z

[16] Pejhan, K., Wu, C.Q. and Telichev, I. (2015) 'Comparison of load transfer index (U*) with conventional stress analysis in vehicle structure design evaluation', *Int. J. Vehicle Design,* Vol. 68, No. 4, pp.285–303.

[17] Pejhan, K., Wang, Q., Wu, C.Q. and Telichev, I. (2017) 'Experimental validation of the U* index theory for load transfer analysis', *Int. J. Heavy Vehicle Systems*, Vol. 24, No. 3, pp.288–304.

[18] E. Wang, X. Zhang and K. Takahashi, "Load Path U* analysis of a square pipe under collision," in Society of Automotive Engineers of Japan Annual Congress, Yokohama, 2007.

[19] ParaView users's guide [Online]. Available : https://docs.paraview.org/en/latest/index.html

[20] The VTK user's guide [Online]. Available : https://www.kitware.com/products/books/VTKUsersGuide.pdf

A

Appendix

This chapter contains the programmed script used in Matlab and Ansys to construct the routine as discussed in section 3.3 to visualize load paths.

A.1 APDL program for U^* calculation

The FEM calculations to calculate the U^* is done by developing a program in Ansys APDL. The programmed script with APDL commands is the following.

```
1
2  /CLE
3  *DEL,ALL
4  ! Analyses input starts here
5  /NOPR   ! Suppress printing of UNDO process
6  FINISH  ! Make sure we are at BEGIN level
7  /PREP7 ! Enter the preprocessor
8
9
10 file_name = '-----' ! Enter the filename
11 NODES_PER_ELEMENT = 20 ! Enter Number of nodes per element
12 DOF_PER_NODE = 3 ! 2 - 2D problem ; 3 - 3D problem
13
14 CDREAD,DB,cantilever_3d_coarse,cdb
15 EPLOT
16
17 !---------- GET FREENODES & TOTALNODES ----------
18
19 nsel,all
20 nsel,u,,,fixnodes
21 nsel,u,,,loadnodes
22 CM,freenodes,node
23
24 nsel,all
25 cm,totalnodes,node
26
27 !---------- COUNT THE NODES ----------
28 NSEL,S,,,fixnodes
```

```
29 *GET,fixnodes_count,NODE,0,COUNT
30 NSEL,S,,,loadnodes
31 *GET,loadnodes_count,NODE,0,COUNT
32 NSEL,S,,,freenodes
33 *GET,freenodes_count,NODE,0,COUNT
34 NSEL,S,,,totalnodes
35 *GET,totalnodes_count,NODE,0,COUNT
36
37
38
39 !---------- GET THE NODE NUMBERS ----------
40
41 NSEL,S,,,loadnodes      ! Selecting load side nodes
42 *VGET, loadnodes_no, NODE, 0, nlist
43 NSEL,S,,,fixnodes       ! Selecting load side nodes
44 *VGET, fixnodes_no, NODE, 0, nlist
45 NSEL,S,,,freenodes
46 *VGET, freenodes_no, NODE, 0, nlist
47
48 ALLSEL
49 NSEL,S,,,freenodes      ! select nodes setdiff(all_nodes, [
    load_nodes, boundary_nodes])
50 *GET,min_node_no,NODE,0,NUM,MIN          ! Get and assign the
    least node number of freenodes in the varable "
    min_node_no"
51
52 *DIM,loaded_nodes_lc,ARRAY,2,freenodes_count
53
54 !---------- Apply original boundary conditions
    ----------
55
56 /PREP7                  ! Enter the preprocessor
57 ALLSEL
58 D,fixnodes,ALL,ALL   ! All dofs of the fix nodes to zero
59 F,loadnodes,Fy,-10   ! Apply force
60
61 !---------- Solve the problem
    ----------
62 /SOLU                   ! Enter the solution
63 SOLVE
64
65 !---------- Calculate the strain Energy
    ----------
66 /POST1
67 SET,
68 /POST26
```

```
69 ENERSOL,2,SENE,,STRAINENERGY
70
71 FILE,'%file_name%','rst','.'
72 /UI,COLL,1
73 NUMVAR,200
74 SOLU,191,NCMIT
75 STORE,MERGE
76 FILLDATA,191,,,,1,1
77 REALVAR,191,191
78
79 *GET,U_VALUE,VARI,2, REAL,1  ! Store the Strain energy value
80
81 !---------- Calculate the load node displacement
       ----------
82
83 *DIM, LOADNODE_DISP,ARRAY,DOF_PER_NODE,loadnodes_count
84
85 *IF,DOF_PER_NODE,EQ,2,THEN
86         *DO, I, 1,loadnodes_count
87             *GET,LOADNODE_DISP(1,I),NODE,loadnodes_no(I)
                   ,U,X
88             *GET,LOADNODE_DISP(2,I),NODE,loadnodes_no(I)
                   ,U,Y
89             !*GET,LOADNODE_DISP(3,I),NODE,loadnodes_no(I
                   ),U,Z
90         *ENDDO
91 *ELSEIF,DOF_PER_NODE,EQ,3,THEN
92         *DO, I, 1,loadnodes_count
93             *GET,LOADNODE_DISP(1,I),NODE,loadnodes_no(I)
                   ,U,X
94             *GET,LOADNODE_DISP(2,I),NODE,loadnodes_no(I)
                   ,U,Y
95             *GET,LOADNODE_DISP(3,I),NODE,loadnodes_no(I)
                   ,U,Z
96         *ENDDO
97 *ENDIF
98
99
100 !---------- Do the looping ----------
101
102 /PREP7 ! Enter the processorer
103
104 *DO,I,1,freenodes_count
105     DDELE,ALL
106         FDELE,ALL
107     ALLSEL
```

```
108         !Apply Normal boundary Condition
109     D,fixnodes,ALL,ALL     ! All dofs of the fix nodes to
            zero
110     ! Assign enforced displacement
111         *IF,DOF_PER_NODE,EQ,2,THEN
112             *DO,J,1,loadnodes_count
113                 D,loadnodes_no(J),UX,
                        LOADNODE_DISP(1,J)
114                 D,loadnodes_no(J),UY,
                        LOADNODE_DISP(2,J)
115             *ENDDO
116         *ELSEIF,DOF_PER_NODE,EQ,3,THEN
117             *DO,J,1,loadnodes_count
118                 D,loadnodes_no(J),UX,
                        LOADNODE_DISP(1,J)
119                 D,loadnodes_no(J),UY,
                        LOADNODE_DISP(2,J)
120                 D,loadnodes_no(J),UZ,
                        LOADNODE_DISP(3,J)

121             *ENDDO
122         *ENDIF
123     NSEL,S,,,freenodes
124     loaded_nodes_lc(1,I)=min_node_no
125         min_node_no=NDNEXT(min_node_no)
126     D,loaded_nodes_lc(1,I),ALL,ALL
127     ALLSEL
128     LSWRITE,I     ! Write loadstep
129     DDELE,ALL     ! delete all constrained dofs
130 *ENDDO
131 DDELE,ALL
132
133
134
135 !------------ SOLVE ALL THE SETS ------------------------------
136 *GET,TBEFORE,ACTIVE,,TIME,CPU
137 /SOLU
138 ALLSEL
139 LSSOLVE,1,freenodes_count
140 *GET,TAFTER,ACTIVE,,TIME,CPU
141 SOLUTION_TIME = (TAFTER-TBEFORE) !record the computational
        CPU time
142
143
144 !------------ CALCULATE STRAIN ENERGY FOR ALL SET
        ---------------------------
```

```
145
146 /POST26
147 /OUTPUT,U_prime,txt
148 ENERSOL,3,SENE,,U_PRIME
149 /OUTPUT            ! send the output back again to usual file
150
151 !————————— ASSIGN U* VALUES TO THE NODES
    ————————————————————
152
153 FILE,'%file_name%','rst','.'
154 /UI,COLL,1
155 NUMVAR,200
156 SOLU,191,NCMIT
157 STORE,MERGE
158 FILLDATA,191,,,,1,1
159 REALVAR,191,191
160
161 *DO,J,1,freenodes_count ! RECORDING U' PRIME VALUES FOR THE
    FREENODE
162     *GET,loaded_nodes_lc(2,J),VARI,3, REAL,J
163 *ENDDO
164
165 *DIM,U_STAR,ARRAY,1,freenodes_count
166
167
168 *DO,K,1,freenodes_count
169     U_STAR(1,K)= 1 - (U_VALUE/loaded_nodes_lc(2,K))   !
        ORIGINAL RATIO
170     !U_STAR(1,K)= 1 - (loaded_nodes_lc(2,K)/U_VALUE)  !
        REVERSE RATIO
171 *ENDDO
172
173 !————————— PLOTTING ——————————
174
175 /POST1
176 SET,LAST
177 /GRAPHICS,FULL
178
179 *DO,I,1,freenodes_count
180 DNSOL,freenodes_no(I),U,X,U_STAR(1,I)
181 *ENDDO
182
183 *DO,I,1,loadnodes_count
184 DNSOL,loadnodes_no(I),U,X,1
185 *ENDDO
186
```

```
187 *DO,I,1,fixnodes_count
188 DNSOL,fixnodes_no(I),U,X,0
189 *ENDDO
190 RAPPND,freenodes_count+1,freenodes_count+1
191
192 PLNSOL, U,X, 0,1.0
193
194 !---------- DATA PROCESSING ----------
195
196 ALLSEL
197 *GET,ELEM_NO_MIN,ELEM,0,NUM,MIN
198 *GET,ELEM_NO_MAX,ELEM,0,NUM,MAX
199 *GET,ELEM_COUNT,ELEM,0,COUNT
200
201
202 ALLSEL
203 *GET,ELEM_NO_MIN,ELEM,0,NUM,MIN
204 *GET,ELEM_NO_MAX,ELEM,0,NUM,MAX
205 *GET,ELEM_COUNT,ELEM,0,COUNT
206
207 *DIM,ELEMENT_TABLE,ARRAY,ELEM_COUNT,1+NODES_PER_ELEMENT ! 20
      NODES
208 *DO,I,ELEM_NO_MIN,ELEM_NO_MAX
209             *SET,ELEMENT_TABLE(I,1),I
210             *DO,J,1,NODES_PER_ELEMENT
211                     *SET,ELEMENT_TABLE(I,J+1),NELEM(I,J)
212             *ENDDO
213 *ENDDO
214
215
216 *DIM,TABLE_RES,ARRAY,totalnodes_count,5
217 *IF,DOF_PER_NODE,EQ,2,THEN
218             *DO,I,1,totalnodes_count
219                     *SET,TABLE_RES(I,1),I                    !
                          Node Number
220                     *GET,TABLE_RES(I,2),NODE,I,LOC,X !X-
                          coordinate
221                     *GET,TABLE_RES(I,3),NODE,I,LOC,Y !Y-
                          coordinate
222                     *SET,TABLE_RES(I,4),0    ! Z- coordinate
223             *ENDDO
224 *ELSEIF,DOF_PER_NODE,EQ,3,THEN
225             *DO,I,1,totalnodes_count
226                     *SET,TABLE_RES(I,1),I                    !
                          Node Number
```

```
227                     *GET,TABLE_RES(I,2),NODE,I,LOC,X ! X-
                            coordinate
228                     *GET,TABLE_RES(I,3),NODE,I,LOC,Y ! Y-
                            coordinate
229                     *GET,TABLE_RES(I,4),NODE,I,LOC,Z ! Z-
                            coordinate
230             *ENDDO
231 *ENDIF
232
233
234 *DO,J,1,freenodes_count
235                     *SET,TABLE_RES(freenodes_no(J),5),U_STAR(1,J
                            )
236 *ENDDO
237
238 *DO,J,1,loadnodes_count
239                     *SET,TABLE_RES(loadnodes_no(J),5),1
240 *ENDDO
241
242 *DO,J,1,fixnodes_count
243                     *SET,TABLE_RES(fixnodes_no(J),5),0
244 *ENDDO
245
246
247 !---------- EXPORTING DATA ----------------------
248 *cfopen,element_type.txt
249 ! list element types
250 *GET,net,ETYP,1,NUM,COUNT
251 *DO,i,1,net
252             *GET,ETname,ETYP,i,ATTR,ENAM
253 *VWRITE,'ET ',i,' ',ETname
254 (A3, F4.0, A1, F4.0)
255 *ENDDO
256 *cfclose
257
258 *MWRITE,ELEMENT_TABLE,element_table,txt
259 (F10.0,F10.0,F10.0,F10.0,F10.0,F10.0,F10.0,F10.0,F10
       .0,F10.0,F10.0,F10.0,F10.0,F10.0,F10.0,F10.0,F10.0,
       F10.0,F10.0)
260
261 *MWRITE,TABLE_RES,result_table,txt
262 (F10.0,F10.6,F10.6,F10.6,F10.6)
263 !6 digit precision. if you want more precision, increase 'x'
       in F10.x
```

A.2 Matlab program for data transfer

This sections contains the Matlab script to transfer the data from Ansys Mechanical APDL to ParaView.

```
clear
close all
clc

fileId1 = fopen('cantilever_beam_3d_pointload.vtk','w'); % open file for writing; discard existing contents
fprintf(fileId1,'# vtk DataFile Version 2.0 \n');
fprintf(fileId1,'Unstructured Grid - canitlever_beam_3d_pointload \n');

%read data
etable = dlmread('element_table.txt');% get the element-nodes connectivity
result_table= dlmread('result_table.txt');
element_type = fgets(fopen('element_type.txt'));
element_type = str2num(element_type(9:end));

%convert it to array
U_star = result_table(:,5);
nodal_coordinates = result_table(:,[2 3 4]);
etable = int64((etable));

etable(:,all(etable == 0))=[]; % remove all the zero columns
etable=etable-1; % because the indexing in paraview starts from 0 not from 1
etable_vtk = etable ;
nodes_per_element = length(etable(1,:))-1 ;
etable_vtk(:,1) = nodes_per_element; % insert the number of nodes for element in the first column

if  element_type == 181
    cell_type = 9;
elseif element_type == 182
    cell_type = 9;
elseif element_type == 185
    cell_type = 12;
elseif element_type == 186
    cell_type = 25;
elseif element_type == 187
    cell_type = 24;
end
%%
```

```
38 fprintf(fileId1,'ASCII \n');
39 fprintf(fileId1,'DATASET UNSTRUCTURED_GRID \n');
40 % Define points - nodal coordinates
41 fprintf(fileId1,'\nPOINTS %d float \n', length(
      nodal_coordinates(:,1)));
42 for i =1 :length(nodal_coordinates(:,1))
43     fprintf(fileId1,'%d ',nodal_coordinates(i,:));
44     fprintf(fileId1,'\n ');
45 end
46 % define cells - elements
47 fprintf(fileId1,'\nCELLS %d %d \n', length(etable(:,1)),
      size(etable,1)*size(etable,2));
48
49 for i =1 :length(etable_vtk(:,1))
50     fprintf(fileId1, '%d ', etable_vtk(i,:));   % order is
         changed because of the format
51     fprintf(fileId1,'\n ');
52 end
53 %%
54 % define cell data - element type
55 fprintf(fileId1,'\nCELL_TYPES %d \n',length(etable(:,1)));
56 for i =1 :length(etable(:,1))
57     fprintf(fileId1,'%d \n', cell_type );
58 end
59 % define scalar data - U* data
60 fprintf(fileId1,'POINT_DATA %d \n',length(U_star(:,1)));
61 fprintf(fileId1,'SCALARS U_star float 1 \n');
62 fprintf(fileId1,'LOOKUP_TABLE default \n');
63 for i= 1:length(U_star(:,1))
64     fprintf(fileId1,'%d \n', U_star(i) );
65 end
66
67 figure
68 plot3(nodal_coordinates(:,1),nodal_coordinates(:,2),
      nodal_coordinates(:,3),'*')
69 hold on
70 % text(nlist(:,2),nlist(:,3),nlist(:,4),string(nlist(:,1)))
71 axis equal
```

DEPARTMENT OF MECHANICS AND MARITIME SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY