

Dissertation: Hybrid Methods for Finite Element Meshing

Jack Bradbrook (psyjb4)

April 5, 2017

Contents

1	Introduction	3
2	Motivation and Background	4
2.1	Properties of finite element models	4
2.2	Limitations and general considerations	5
3	Related Work	7
3.1	Traditional Stress refinement methods	7
3.2	Uses of Artificial Intelligence and Machine Learning	7
3.3	Quality metrics	8
4	Description of the work	10
4.1	Aims and Objectives	10
4.2	System specification	10
4.3	Functional Requirements	10
4.4	Non-Functional Requirements	11
5	System Design	12
5.1	System Overview	12
5.2	Modular Architecture	12
5.3	Input Files	13
5.4	Simulation Data Model	13
5.5	Remeshing methods approach	14
5.5.1	Combining methods	14
5.5.2	Multi Threading	15
6	Software Implementation	16
6.1	Third Party FE Application	16
6.2	LISA	16
6.3	Languages and platforms	16
6.4	Implementation Methodology	17
6.5	Implementation of Subsystems	17
6.5.1	Re-meshing using hierarchical refinement	17
6.6	Fast node lookup and update of nodes	18
6.7	Sorting Element Nodes	19
6.8	Stress Based Refinement	21
6.9	Heuristic/Rule Based Refinement	21
6.10	Mesh Quality Assessment	22
6.11	Implementation Issues	22

6.11.1 Attempts to Automatically Define edges within models	22
7 Evaluation of Project	23
7.1 Validation against Functional Requirements	23
7.2 Validation against Non Functional	23
7.3 Unit Testing	23
7.4 Software Management	23
7.5 Software Quality and Maintainability	24
7.6 Documentation	24
7.7 Increase in performance through parallel execution	24
7.8 Evaluation Of System For Model Simulations	24
7.8.1 Suspension Bridge structure	24
7.8.2 Hybrid Evaluation	26
7.8.3 Analysis of and Addition of Metric(s)	26
7.9 Evaluation of Subsystems based on model evaluation	28
7.9.1 Analysis of Methods Using Metrics	28
7.9.2 Evaluation of Stress Refinement Effectiveness	28
7.9.3 Evaluation of Heuristic Refinement Effectiveness	28
7.9.4 Evaluation Issues	28
7.10 Strengths and Weaknesses	29
7.11 Evaluation summary	29
7.12 Successes and Limitations of project	29
8 Further Work	30
8.1 Gathering feedback from experienced engineers	30
8.2 Improving usability through a web interface	30
9 Personal Reflections and Summary	30
References	32
A Element Types within LISA	33
B Unit Testing	34
C Input and output files	34
D Project Layout in Solution Explorer	34
E Software Quality Metrics	34
F Bridge Cross Loading Simulation Results	34
F.1 Heuristic Refinement	35
F.2 Stress Refinement	35
G Bridge Base Loading Simulation Results	35
H Paper Mill Simulation Results	35
I Gantt Chart for Project Time Management	36

1 Introduction

The overall aim of this project is to perform the task of refining a Finite Element Mesh (FEM) using a stress based method as is typical of FEM refinement in conjunction with an approach that uses techniques from Artificial Intelligence and Machine Learning. It should be possible to reason about the quality of a mesh produced using both methods analytically in order to evaluate the success of the approach and potential for continued use.

Finite Element Analysis (FEA) is a method widely used across different engineering domains to simulate structures under certain conditions. The method works by taking a geometry defined in continuous terms, discretize it into a mesh system before calculating the property values for each of the discrete elements. This allows engineers to observe the effect the conditions have on the entire structure, see figure 1.

Success of the project will be determined by implementation of a system that is able to combine the two approaches described above in order to refine a mesh to of a quality comparable to that of a successful stress based method.

2 Motivation and Background

Over the past forty years FEA has emerged as a prominent technology for simulating complex real world engineering problems [1, 5]. FEA works by solving a system of differential equations with each equation representing a single element in a geometric mesh. By doing this FEA is able to generate highly accurate approximations for the properties of complex physical systems [5] [18]. The method can also be highly computationally expensive with the complexity typically increasing exponentially with the model size [5]. Analysis therefore proves to be highly time consuming and costly for individuals and organisations to conduct [7]. [16]

2.1 Properties of finite element models

Although this is a computer science as opposed to a mechanical engineering dissertation it's still important to briefly outline the general principals and properties that underpin the FE method in order to have a general appreciation for how the design and evaluation of the final software system was conducted.

Finite Element Models have several key properties that need to be specified by the engineers who create them, the configuration of these properties greatly determine the results obtained from the models execution. The first of these properties that an FE model possesses is the mesh. The mesh is constructed out of nodes, points which act as intersections between the second component- elements which are either a polygon or a polyhedron between the nodes. Nodes and elements are important concepts as they provide the theoretical framework for reasoning about the other properties of a mesh and hence the overall quality of the model [18].

In addition to the nodes and elements FE models also contain inputs and constraints. Inputs can be thought of as the phenomena from the outside world which is acting on the structure and consequently inducing some kind of physical effect on it. Inputs are used by engineers to model the conditions under which the structure will be expected to perform under when it is manufactured and enters operation.

Constraints are another fundamental concept that describe where the model is attached to the outside world. When computing stresses induced through the model there needs to be an area through which the stresses are assumed to leave. FEA is only able to calculate the path of the stresses through the model and thus the overall stress at given points using the law of conservation of energy i.e. energy cannot be created or destroyed meaning that any energy provided to the system as input through for example force needs a means by which to leave it, the constraint.

For example in figure 6 showing a suspension bridge model the simulation is to be run with the forces are induced upon the cables and the towers in the negative x direction as represented by the green vectors. The corresponding constraint area through which the force must leave is specified as the base of each bridge pillar and represented by multiple red arrows on each corresponding corner.

The final piece of information needed in order to calculate the stresses through the model is material data. Material data is associated with the models elements and is usually defined using two main parameters which are:

- Youngs Modulus - The ratio of stress over strain for a given material, i.e. for an amount of internal force endured by a material how much does it deform, a material such as rubber therefore has a low value for Young's modulus while diamond has a high value[30]
- Poissons ratio - Amount of deformation that occurs perpendicular to the force that is applied to the material.

For the sake of simplicity all structures used to evaluate the final software solution have assumed steel as their material property. Steel is a pretty common material used within manufacturing of many mechanical

components and does not exhibit any abnormal properties. This is beneficial for evaluation as it removes variability in the results that could arise from selecting a more complex material.

2.2 Limitations and general considerations

An important consideration when conducting FEA is the trade-off of a models accuracy against the time in which it can be solved. A major variable determining this trade-off is the models mesh structure which discretizes the problem so that simulation can be run on it. A mesh that is finer is more computationally expensive but also produces results of greater accuracy. It is therefore desirable to generate a mesh which is fine where accuracy is most needed but coarse where it is not [17].

In every type of analysis that the FE method is used for (thermal, structural, fluid flow, electrical) there is a specific differential equation associated with each of the elements. In order to achieve overall convergence of the model the equations must be solved simultaneously to achieve a value for each of the discrete elements [18]. For this dissertation project attention will be given specifically to the problem of FEA meshing in the context of static structural analysis where the value calculated across each element is its stress. It makes sense to work on hybrid mesh refinement in the context of static structural analysis as it is likely the most common engineering application of the method and as such has the largest body of research that is relevant [5][18].

For engineers the value obtained through computing the stresses under a particular set of conditions is feedback on the quality of their design. Ideally the results from an analysis will provide a good understanding of where the design is weak and how concentrated this weakness is. This information is used to either help verify the designs' quality or alternatively inform changes to its geometry or material properties so as to reduce stress on subsequent analysis [20].

To understand the gradient of stress within a part of the model the mesh needs to be designed carefully. As each element can only display values calculated from its edge nodes a smooth gradient requires a higher concentration of elements in areas under higher stress. A high quality mesh is therefore considered to have a higher concentration of elements in areas of predicted high stress while retaining lower concentration elsewhere, thereby revealing weaknesses in the design while minimising the models runtime.

Traditionally the automated mesh refinement process consists of computing stresses for a model with an initial coarse mesh and low computation cost, once rough stresses have been computed the elements in areas of higher stress can be divided recursively into additional elements in order to achieve smoother gradients on further executions [16]. Figure 1 shows a mesh which has been refined in an area of higher stress thus providing a clearer indication of a components weakness.

Unfortunately for many large models this method for refining a mesh is still excessively costly [4]. It therefore seems worth investigating use of alternative approaches posed by the field of computer science and artificial intelligence that could support the traditional high stress meshing approach.

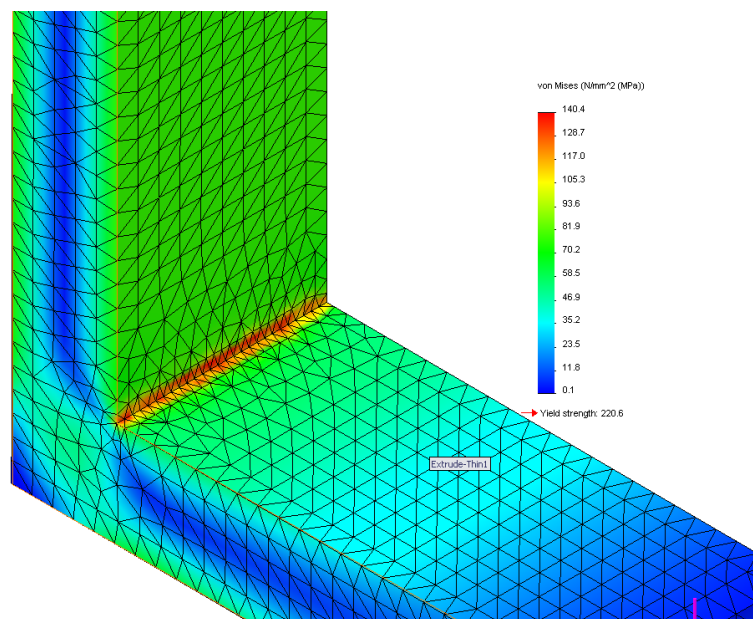


Figure 1: Mesh refinement in corner under high stress image source: ([28])

3 Related Work

Many approaches have so far been taken in an attempt to improve a computer's ability to perform the task of finite element meshing. The following subsections present an overview of the research conducted for the various aspects of the project which are more explicitly outlined under "Aims and Objectives"

3.1 Traditional Stress refinement methods

A number of methods exist to perform mesh refinement based on a stress gradient for a mesh, with the most common being higherarchical refinementent also known as h-refinement and relocation refinement or r-refinement[2] [3]

h-refinement: H-refinement is the process of recursively refining a mesh by splitting elements into additional sub elements. This process can be performed for elements with both both triangular and quadrilateral shapes. [2].

r-refinement: R-refinement is a method which attempts to improve the quality of the stress gradient without the alteration of the mesh element count and thus the computational cost. This is achieved by the relocation of elements within the mesh which effectively increases the size of elements in areas of low stress, while reducing the size in areas where stress is high [3].

3.2 Uses of Artificial Intelligence and Machine Learning

Within the domains of AI and machine learning methods such as neural networks [9], case based reasoning [10] and inductive logic programming [5] have all been adopted to facilitate generation of meshes comparable to that of human experts. Similarly there has also been effort to combine multiple numerical methods simultaneously for solving re meshing problems [8] although effort to combine the two approaches does not appear to have so far been made.

Due to the difficulty of obtaining meshes which hold commercial interest the majority of researchers working in this field have had to result to the use of training sets developed within academia [4]. The primary issue associated with this is that often the training data does not exhibit the level of complexity that you would expect in many industrial sectors. Many researchers must accept this as a limitation or agree commercial terms with an organisation in order to gain access to their models [14].

Having reviewed a variety of different AI based applications to FE the use of Inductive Logic Programming (ILP) used by Bojan Dolsak et al is of greatest interest. ILP is a machine learning method first presented by Stephen Muggleton in his 1991 paper "Inductive Logic Programming" [11]. Muggleton suggests that the traditional approaches of machine learning which rely on use of extensive data sets and statistical analysis are poor in the case of many real world problems for which data is not available [13]. Muggleton cites Human learning as an example of use of ILP style techniques where understanding of new concepts is achieved not through crunching large volumes of data points but instead the use of induction on a relatively concise set of background facts and examples obtained from previous life experiances [13].

ILP uses three types of input information in order to hypothesise additional facts about the system. These three types of input information are:

- Positive examples of what constitutes an area that is well meshed
- Negative example of areas that are poorly meshed
- Background facts

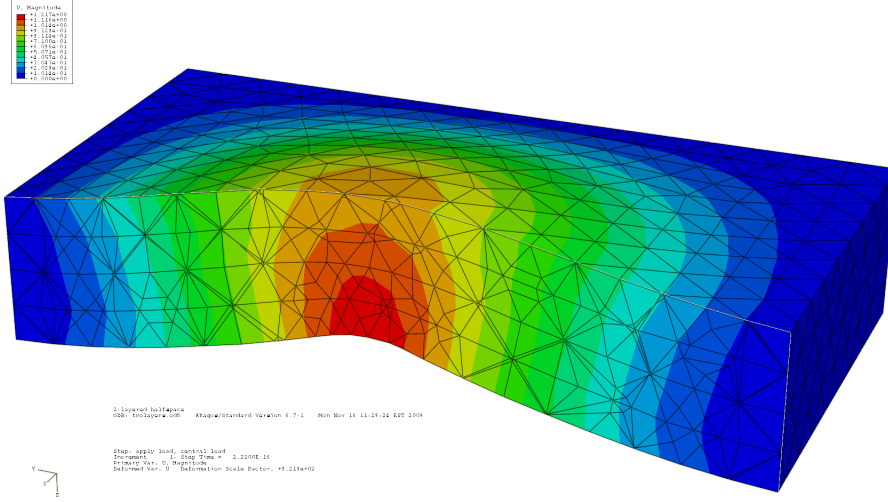


Figure 2: Example of how elements can be distorted in order to fit a geometry which will result in deterioration of gradient quality (image source: [15])

Using this information ILP is capable of hypothesising rules by determining which rules can exist within the system where given the set of background facts all positive examples are satisfied while few or none of the negative examples are. Although ILP requires a body of additional metadata associated with each mesh this is easier to obtain making ILP a highly practical solution. Along with his publication Muggletons also released his implementation of an ILP algorithm as a program titled "Golem" [12], Golem was applied by Dolsak to the problem of mesh refinement with a training set of just five meshes [5]. The resulting rule set when applied to subsequent models was able to correctly classify and re mesh areas with an average accuracy of 78% for a range of geometries [5] [6].

Dolsak's choice of metadata for the ILP method to generate mesh rules is based on the classification of edges within the FE model. Dolsak recognises that edges act as an important intersections within the model and as such provide useful items of reference when designing heruristics with which to reason about the model [5] [6]. For example if it is know that an edge has a force applied close to it based on the initial model conditions then other edges that intersect it should be additionally remeshed meshed [4] [6].

The format of the rules also make them attractive for experimenting with as part of a hybrid method since the method determines how to refine the mesh based on the arrangement of edges. This detail of analysis of the mesh is at a comparable detail to that of a traditional splitting method such as h-refinement which will likely to improve the ease at which the two methods can be combined simultaneously in the latter stages of the project.

3.3 Quality metrics

Finally work has also been done on establishing valid metrics for assessing the quality of a mesh automatically [14, 9] Metrics for meshes have been research far more extensively that AI methods due to their use for comparing different stress based refinements. There are also cases of common metrics being used for industrial meshing applications [14]. Although there are metrics for assessing a mesh on a global level such as element count score the consensus is that due to the variation in meshes this is less reliable than assessing quality based on the properties of individual elements within the mesh [14]

Localised metrics associated with the quality of each element have shown to be accurate for predicting the overall quality of a mesh when taking the average for each metric across all elements [14]. The quality of an elements shape is important since the stress values which are computed for the area within the element are calculated using the stress values at each of the nodes which enclose it [18]. Elements are typically deformed near parts of the geometry where its shape simply does not allow a uniform element to be placed, an example of where this has occurred can be seen in figure 2.

Some key shape metrics identified by Dittmer et al include (ideal values are for elements of quadrilateral type as used in the current prototypes):

- A Aspect ratio – longest side / shortest side, ideal value is 1
- B Maximum corner angle - widest internal angle to element, ideal is 90°
- C Maximum parallel deviation - how skewed the element is, ideal is 0°

4 Description of the work

4.1 Aims and Objectives

The aim of this project was to design, build and analyse a system for refining a mesh by combining a method from the domain of AI or machine learning with a stress based refinement method. The desirable end result will be a hybrid method of meshing which produces reasonably good solution accuracy whilst requiring less user intervention and reduced computational cost.

The project can be broken into three main sections of research and implementation which can each be considered a high level objective:

- A The first objective is to research and develop both a traditional stress refinement and an AI re meshing algorithm developed by industry or by academia. These algorithms should be able to run independently on a set of example models.
- B Secondly a process will need to be devised for combining the two re meshing methods to varying degrees will be required. This will make it possible to test the effects of a hybrid method across the same set of models. Through this it can be established whether there is notable benefit to combining the different approaches and if so to what degree is it effective.
- C The third objective will be to use justifiable metrics for assessing the quality of a given mesh. This will allow for a much greater range of hybrid variations to be tried without requiring inspection from an expert.

4.2 System specification

To demonstrate success in achieving the objectives of the project it is important to have tractability from the requirements through to the solution and lastly verification and validation. This section describes the current systems *functional requirements* (what the system will do) and *non-functional* requirements (its constraints) based upon evaluation of the research I have conducted in conjunction with discussions with the project supervisor: Dr Jason Atkin. Functional requirements have primarily been listed under their respective high level subsystems that are responsible for encapsulating their functionality.

Although it has not been developed as part of the project the application responsible for solving the finite element models has been included as part of the systems requirements since it highly influences the overall scope of the project and much of the design associated with other subsystems which were developed for the project.

4.3 Functional Requirements

1. FE integration: System will be able to interface with a third party finite element application
 - 1.1. The finite element applications solver must be able to solve a mesh based on its model configuration.
 - 1.2. The finite element applications solver must be able to execute a model programmatically
 - 1.3. The finite element applications solver must be able to output stress data at different points on the mesh.
 - 1.4. The finite element application will provide a graphical representation of the model.
 - 1.5. It will be possible to manipulate the model that the finite element application uses programmatically.

- 1.6. It should be possible to manipulate the model that the finite element application uses from within its graphical user interface.
2. Mesh refinement: System will be able to perform different kinds of finite element mesh refinement
 - 2.1. The system will be able to refine a finite element mesh using a stress based refinement method
 - 2.2. The system will be able to refine a finite element mesh using a non-stress based refinement method
 - 2.3. A non stress based refinement method will adapt the mesh using background information about mesh design which has been previously trained.
 - 2.4. The system will be able to combine the two methods to produce a coherent mesh which the FE application is able to successfully solve in order to obtain results for stress and displacement.
 - 2.5. The system will be able to combine both methods to varying degrees will be performed automatically by the system without direct user intervention.
 - 2.6. The system will re mesh using both stress and non-stress based refinement using quadrilateral elements.
 - 2.7. System will adapt weighting associated with each method based upon the metrics computer for the mesh in the systems previous iteration.
3. Quality assessment: System will provide the operator with results about the quality of meshes based on metrics obtained from research.
 - 3.1. An assessment will be conducted automatically for every mesh iteration that occurs.
 - 3.2. System will assess quality based on a variety of metrics to ensure overall robustness of measurement.
 - 3.3. The metrics will be computed for both individual elements within the model and for the entire mesh.

4.4 Non-Functional Requirements

Design: The system architecture will be developed using the object oriented design principals of SOLID to allow for clear interfaces between the different functional components. Functional programming practices will be adopted through use of the .NET Language Integrated Query or LINQ framework. This will help to simplify the code and improve reliability. Where functions and classes are written their length will be kept to a minimum to reduce complexity and allow for reuse wherever possible.

Documentation: The system will be comprehensively documented at both a code level and at an architecture level. At a code level C# doc comments will be written to provide a comprehensive summary of each function. This will allow the tool Doxygen [25] to generate a full set of developer documentation upon completion of the software implementation which will be included as an appendix. Doc comments will also help to encourage small functions.

General applicability: In order to demonstrate that hybrid methods are a feasible means of approaching meshing problems the resulting software should be able to successfully execute on a range of models with varying geometry. The range of geometries should be representative of typical structural variation.

5 System Design

5.1 System Overview

Determining the overall design of the system was initially hard since it was not clear exactly how many subsystems would be needed to mesh, evaluate and interface with LISA, what was clear was that the system would essentially be performing an optimisation procedure and as such needed to be driven iteratively towards a goal. The complexity and uncertainty surrounding the project meant ensuring the architecture remained modular was essential with well defined interfaces allowing components could easily be added or modified as the project progressed.

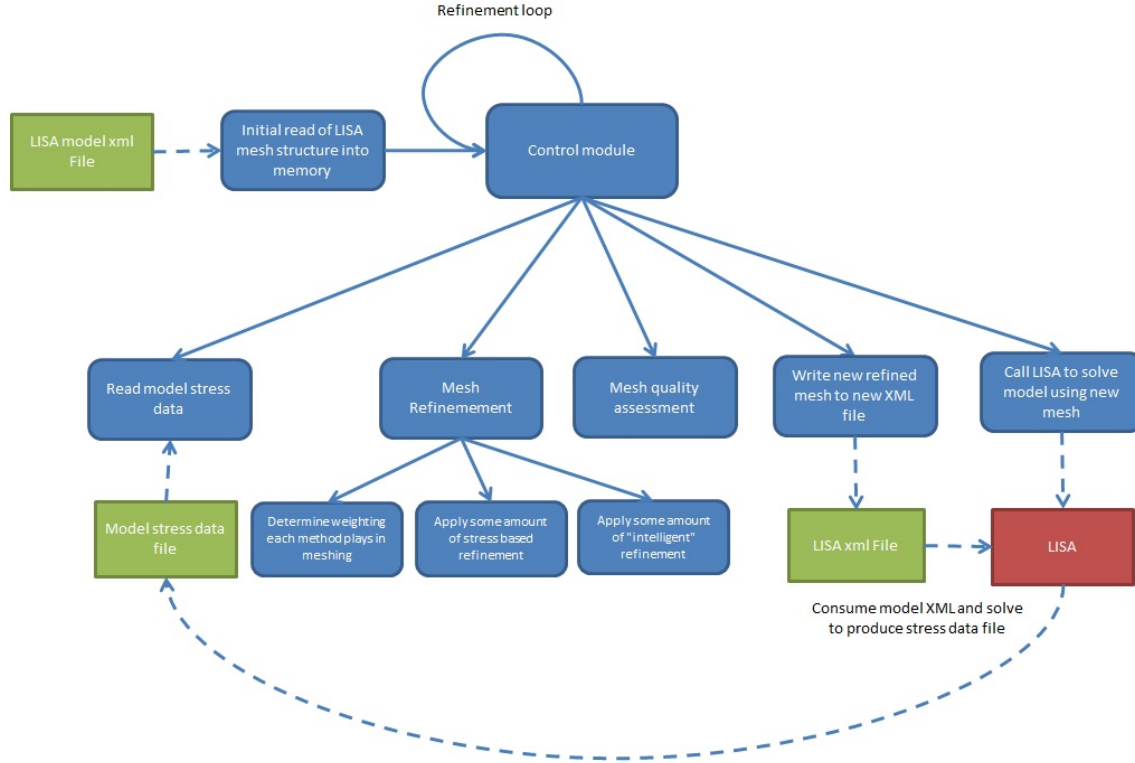


Figure 3: High level design of the system with its different modules

5.2 Modular Architecture

The modular architecture was crucial for allowing meshing algorithms and quality metrics to be replaced as necessary and was an important consideration in the design of the system. At best the quality of the output could only be predicted for each method before it was integrated into the system and executed in a range of different scenarios. To have tightly coupled these individual components would have rendered the overall system a failure in the event that any one of them performed greatly worse than expected. Instead the loose coupling of the architecture has enabled the system to be considered as more of a framework for testing the effects of combining heuristic knowledge of a Finite element problem with a stress based refinement method.

Although the system was highly modular It was also still desirable to maintain a hierarchy within the

classes so that smaller components could be developed independently and contained within the larger ones. Composition was therefore generally favoured over inheritance as a means of building the architecture. Static classes and methods were also used when needing to write utility functions that were required by multiple high level subsystems and therefore did not fit especially well into any particular one. Examples of these are generic vector algebra operations such as dot product, matrix determinants and normal vectors.

At the highest level namespaces were used to break down the class groups appropriately namespaces also naturally structure as folders within the Visual Studio (VS) solution explorer which made navigating the project and finding components much easier as the system expanded.

5.3 Input Files

The system requires three basic input files which should be placed within a directory that is given to the program as a parameter, these are files are:

- A structural model represented as a .liml file which LISA can solve
- An initial stress data file generated manually so the system has some idea about where to start working
- A JSON file containing important edges as identified by an engineer along with associated meta data.

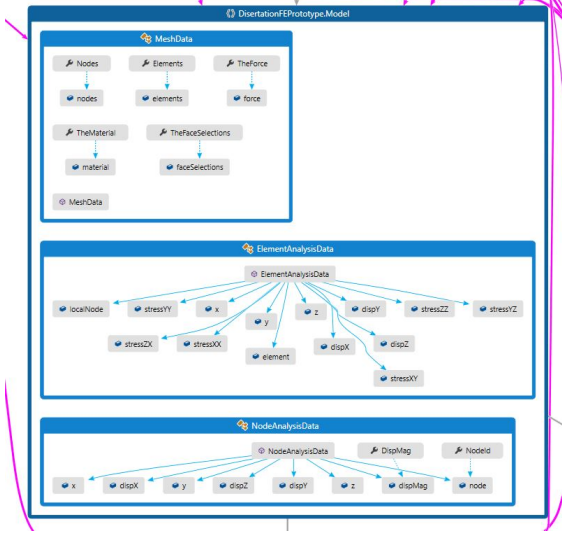
An example of the content and format for each of these input files can be seen in Appendix B

5.4 Simulation Data Model

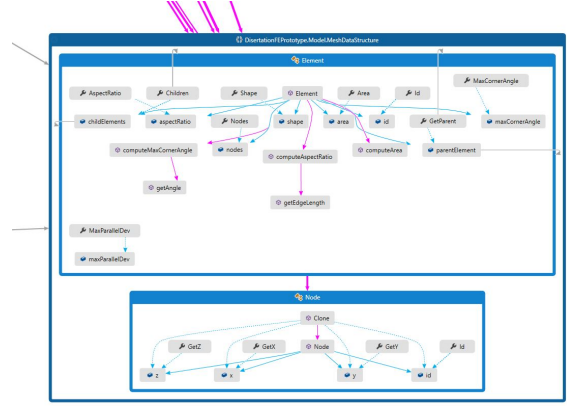
Writing an API for LISA was the first clear stage of development for my project for which a design had to be considered. The API was crucial in order to program the more complex aspects using basic operations and avoid having to regularly perform direct string manipulation of the input files in order to manipulate the model.

When the first re-meshing iteration occurs the system needs to read the input .liml file into an equivalent class model which closely resembles the files schema, diagrams for which can be seen in figure 4 below. Each class in this model contains corresponding data and methods used to represent and manipulate the model. These methods are then used by each of the refinement approaches to easily alter the mesh in a controlled manner. Once the mesh has been adapted however is required it can then be assessed by the modules responsible for validating its quality before finally being written back to a .liml file for LISA to solve on the subsequent iteration. Designing the data model so that it closely resembled the LISA schema not only made the higher level programming less confusing but also made serialisation of the data back to .liml format much simpler thus reducing the number of bugs arising from inconsistencies between different representations of the same data.

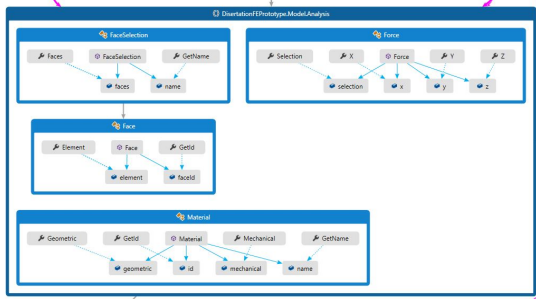
One aspect of the data models design which greatly adds to the systems flexibility is the hierarchical design for representing the various Element types. At the root of this structure is the IElement interface, all new Element types must adhere to this in order for the various refinement methods to refine that element. Implementing the interface are a range of abstract classes such as "SquareBasedElem" and "TriangleBasedElem" These classes are designed to contain methods that are generally applicable for calculating metrics and re meshing individual elements where the elements fit this abstract category but their concrete implementation specifies their dimensionality and number of nodes. This is powerful since computing metrics and performing subdivision for a 3d element is simply a reduction using the code to do this for a 2d element but over every face comprising the 3d one.



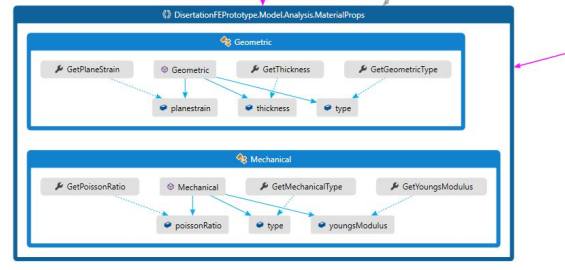
(a) Model classes



(b) Element and Node classes



(a) Model Analysis classes



(b) Material Property classes

Figure 5: Class model to represent .liml file structure used by LISA

5.5 Remeshing methods approach

Due to the modular design of the system the low level methods for performing refinement for each element type were contained within the elements abstract class or if more bespoke its subclass, see Figure 6. This made it much easier to decouple both the stress and heuristic methods allowing each of them to simply have the task of selecting elements which they considered beneficial to refine before delegating the refinement to each element itself. This allowed both of these high level refinement approaches to utilise the same low level functionality which greatly improved code reuse and simplified the design so much so that the high level meshing methods are relatively concise.

5.5.1 Combining methods

Since each refinement method could be executed independently of the other it made sense when testing combinations both to simply enumerate the combinations that each could be applied for each iteration before running each specific combination iteratively on a model. This results in a set of two valued tuples up to some value :

$$\{(a, b) \mid a, b \in \mathbb{N}a, b < k\}$$

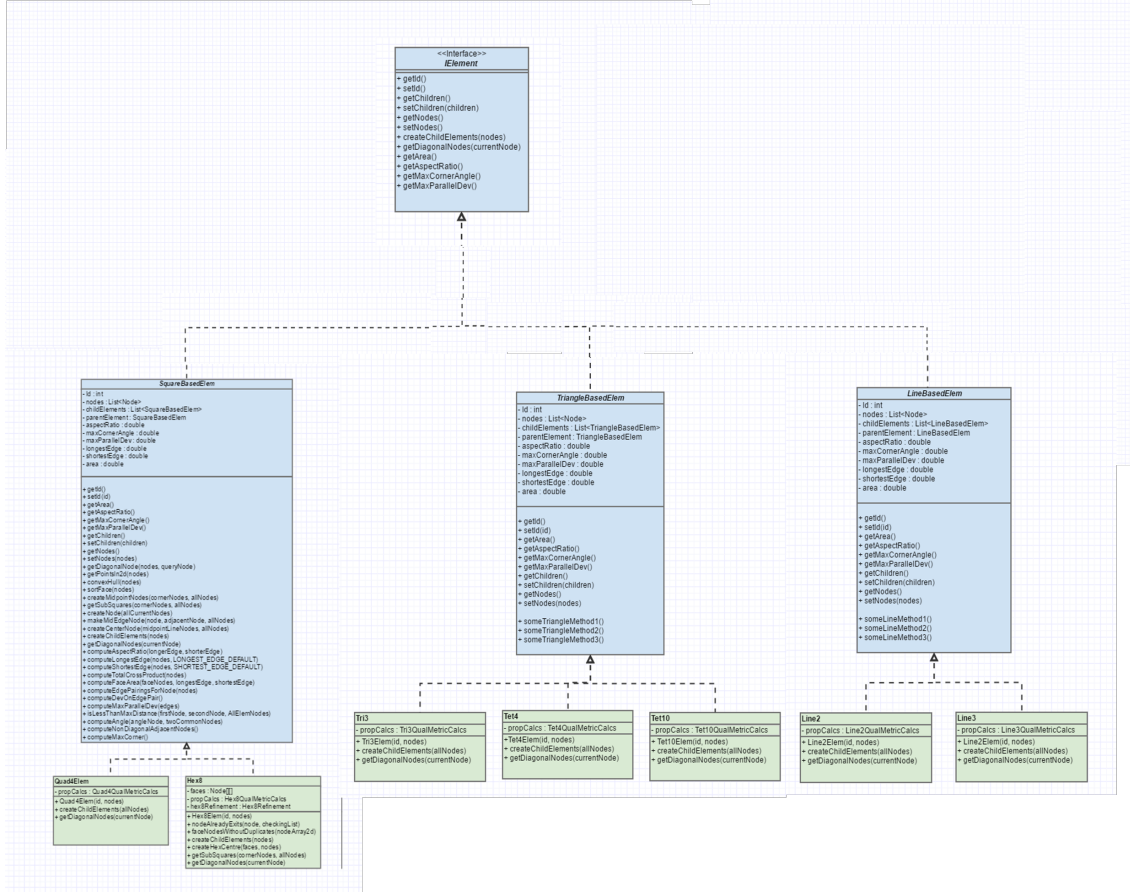


Figure 6: Class diagram showing the hierarchy of element classification within the data model, due to time limitations I was not able to implement the respective classes for triangle and line based elements, to see image representations of each element type within this class diagram refer to element type appendix

Although this approach can be used to combine both methods simultaneously upon testing the system later on

5.5.2 Multi Threading

To allow allow for better performance when evaluating a range of different weightings of the stress and heuristic methods multi threading was used to allow each configuration to be run independently of the others. When started each thread creates its own directory which it copies the three input files to and runs for its designating weighting configuration. Use of multi threading made assessment of the system much more rapid in the later stages of the p

6 Software Implementation

The following subsections detail the implementation of the final software solution that has been written to meet the objectives posed at the beginning of this dissertation.

6.1 Third Party FE Application

In order to demonstrate the potential feasibility of the hybrid approach it was first important to obtain a finite element solver which could be given a FE model containing data about forces, materials and the mesh structure and then execute the model programmatically so as to obtain stress results.

A multitude of commercial FE tools exist with there being a wide variety in both the complexity and cost associated with each tool. Finite element software is typically very expensive due to its high development cost and small customer base. Tools used within industry such as ANSYS typically require a great deal of time in order to become proficient in their usage and can cost in excess of five thousand pounds a year for a single licence [27]. It was therefore important to find a tool which was both affordable while also powerful enough to demonstrate a working prototype of the re meshing method.

6.2 LISA

After reviewing several FE applications used within industry in addition to a variety of less well known ones used within academia and by hobbyists LISA was selected as the solver application for which to implement the systems prototypes.

Strengths: LISA as a FE tool which allows the user to run models of up to 1300 element for free; This was beneficial in allowing me to experiment with the software and gauge the feasibility of my projects concept before requiring a full version. Once at a stage in the project where each problem had been solved for small models containing less than 1300 elements an academic licence for the software was purchased for the projects use.

LISA also provides a GUI which allows visual inspection of the model and its mesh; This is particularly useful for observing the output of the meshing algorithms which can often provide a human with a much better understanding of how the method has performed and whether or not there are obvious bugs. in the implementation of the meshing procedures

Weaknesses: Due to LISA's simplicity it does not come with an extensive API allowing for easy programmatic use of its inbuilt features, however it is still possible to interface with LISA through less direct means [19]. LISA models are stored in .liml files which use XML as a meta mark-up format. The model files contain all the information about the model including the materials used as well as loads and constraints and of course the mesh. It is therefore possible to manipulate a .liml file having parsed its contents before writing a new version of the file which LISA can be called to solve. In order to more easily alter the model it made sense to write a wrapper for the .liml files to abstract the manipulation of their content.

6.3 Languages and platforms

The final system has been written entirely using the C# programming language (version 5.0) with Visual Studio 2015 as the development environment on a Windows 10 system. C# is an application development language built on the .NET framework. Although any number of programming languages could have been used to implement the solution C# offered a good compromise for developing a system with both structural rigidity through static typing and object orientation in addition to functionality to allow for rapid prototyping. C# does this well through use of LINQ, a part of the standard library that provides a large number of higher order functions which allow for operations to be performed over any data structure that

implements the built in IEnumerable interface. Given that much of the code within the project performs the same operation on collections of nodes and elements stored in Lists arrays and dictionaries which all implement IEnumerable the ability to write much of the project using this capability dramatically reduced the number of errors encountered and increased development speed.

6.4 Implementation Methodology

Growing size of the software meant it was important to work systematically to continuously drive the project in the right direction and avoid the introduction of unnecessary complexity. This was achieved through regularly reviewing and refactoring the code which dramatically helped to reduced the amount of bugs introduced.

For the duration of the project the spiral methodology was adhered to. This enforced multiple deliverable stages that were concluded with a supervisor meeting every one or two weeks. Adopting the spiral methodology also provided flexibility regarding the order in which tasks were able to take place outside of a spiral iteration. This was necessary when conducting a research driven project where direction of work for subsequent development iterations was largely driven by the findings of the work in the previous ones.

Tasks were chosen every week for the project, the number of tasks and their complexity was determined using a combination of factors including their complexity, the criticality of the task e.g. Did it need to be completed for other important tasks to be started and the time available to me as the individual undertaking the project (More tasks typically performed on weeks when less work was due for other modules.

6.5 Implementation of Subsystems

This section describes the implementation details of the various different subsystems which combine to form the overall solution.

6.5.1 Re-meshing using hierarchical refinement

After reviewing both h-refinement [2] and r-refinement [3] it was concluded h-refinement would be best the best approach to adopt due to its simplicity and more widespread use [2] despite the fact that the mesh is usually more computationally expensive than it would have been if it was created using r-refinement [3]

Elements within traditional FEA can typically be classified as either triangle or square based elements, each of these provide different strengths and weaknesses when required to mesh and solve models. Within industry triangular elements are typically preferable since it is always possible to generate an initial triangular mesh from any arbitrary CAD geometry algorithmically. This is done by simply making smaller triangles until all gaps along the edge of the geometry are filled [32]. The same cannot always be said when meshing using square elements. For proof of the solutions concept however it was concluded that square based elements were preferable to triangular ones with since the steps required for a basic refinement are much simpler, just take the corners of an element that already exists, add their x, y and z components before dividing each component by two to achieve the coordinate for a node which is halfway between the two.

In addition to refinement it is also significantly easier to define edges which the ILP rules can be applied to when edges naturally form within a structure through a chain of nodes along the edges of square elements. Unfortunately Triangular meshes also generally incur a higher computational cost than an equivalent square element mesh due to added complexity of performing the calculations required to remesh in addition to requiring more elements over a given area to achieve the same accuracy.

From an implementation standpoint writing a square based remeshing algorithm was also substantially easier if given a mesh of exclusively square elements since the primary task to be performed is to repeatedly divide

each element into four sub elements, by contrast methods uses to re mesh triangular meshes are typically more complex and have corresponding initial meshes that are harder to generate manually by human operators [?].

6.6 Fast node lookup and update of nodes

A key requirement for the design of the data model generated by the hierarchical re meshing process was the need to perform fast lookup of nodes already present in the mesh. Lookup is important within the meshing methods as a means of checking whether a node that is about to be created already exists within the model, in the event that no such node already exists a new one can be created however if it does then instead of creating a new node the node that already exists needs to be connected to a node in an adjacent element that is currently being refined. If nodes are not linked correctly form correct elements the physics solver is unable to assume the stress moves through one element to another despite both having nodes at the same coordinates, this results in inaccurate output or potentially an error being thrown by LISA.

This issue arose as a result of partly as a result of the systems design, as previously mentioned subdivision for every individual element is the responsibility of that element which from a software engineering perspective is very good since it means the low level meshing process for each different type of element could be written within that elements class. This avoids the need for much heavier generalised refinement classes that would have needed to know how to perform the meshing for all elements in the model at once and for each of the different potential element types. A consequence of this was despite every Element being capable of meshing itself perfectly adjacent elements that also requiring refinement needed the ability to reconnect the new nodes along their edges to those that had created by the adjacent element. This can be seen below in figure x

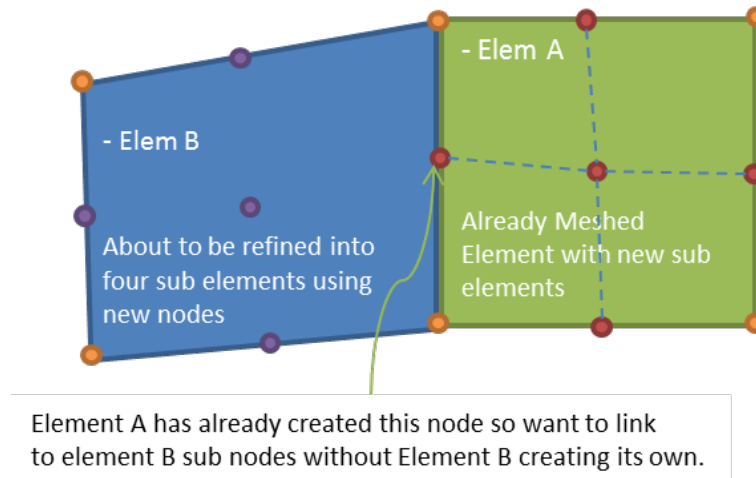


Figure 7: The need for an element to check for existing adjacent nodes when subdividing itself during refinement,

Orange Nodes - An original node for one or more elements

Red Nodes - new nodes made by Elem A

Purple Nodes - new nodes made by Elem B

The solution to this problem was to store all the nodes in the mesh model within a C# dictionary structure a reference to which is passed to each element within the model. The dictionary can be indexed using a Tuple of the x, y and z coordinates for the new potential element which will either return a node already at

that location or indicate that no such node exists, in which case that element is then responsible for creating the node as its first instance. Dictionaries in C# represent a generalised instance of a hash table ensuring that lookup and insert are both constant time on average.

6.7 Sorting Element Nodes

One issue faced when working with LISA was an interface requirement specified requiring nodes for each type of element to be sorted in a specific geometric order. The general rule for node ordering within LISA is to have them form a perimeter around the edge of an element in 3d space without edges crossing one another internal to the element.

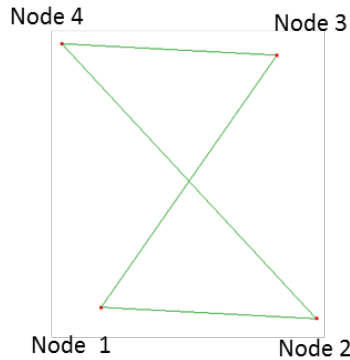


Figure 8: Element with 3d skew resulting in edges between diagonals being shortest by a small amount.

When addressing this problem for simple models constructed from Quad4 elements most straightforward approach was to simply traverse each of the nodes in the order specified by LISA and with the resulting traversal list being ordered for LISA. The resulting traversal process resembles the following:

```

while  $\exists node \in UnsortedNodes$  do
    Get distance between current node and the next two nodes in UnsortedNodes;
    if Nodes left == 1 then
        else
        end
        if distanceToNode1 < distanceToNode2 then
            currentNode Node1;
            sortedNodes.Add(Node1);
            UnsortedNodes.Remove(Node1);
        else
            currentNode Node2;
            sortedNodes.Add(Node2);
            UnsortedNodes.Remove(Node2);
        end
    end
end

```

Algorithm 1: How to write algorithms

For the most part this approach was both fast and correct for Quad4 elements although in cases where elements were particularly skewed in 3d space it was sometimes possible for the internal diagonals to be shorter than the actual sides as seen in Figure 9 below. This proved to be a significant flaw in the method and brought about the realisation that a reliable approach would not be able to depend simply upon highly variable properties such as node distances.

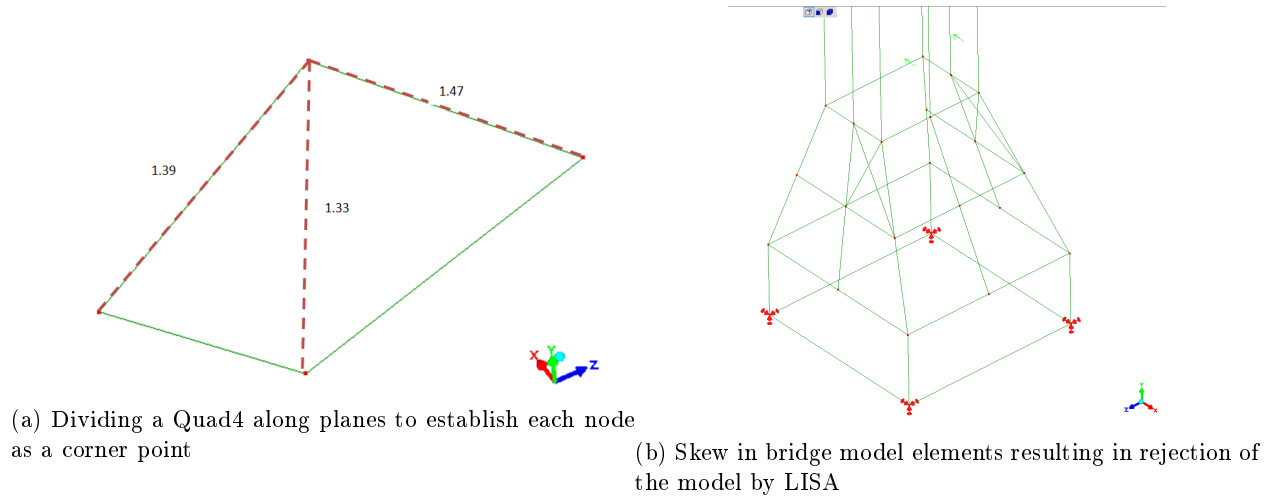


Figure 9: Incorrectly sorted elements arising from failure of traversal routine for skewed elements

Attempting to arrive at a more general solution focus was directed towards sorting the more complex Hex8 element type as this represented a more complete instance of the problem. Analysis of this led to the realisation that the most important task in sorting nodes for an arbitrary type is to simply establish the corner nodes relative to that type. Having established corners correctly sorting then simply required adding them to a list in the order specified by LISA.

The subsequent method which was used to successfully establish corners for both Quad4 and Hex8 elements was to split nodes for each element using planes running along the x, y and z axis as can be seen in Figure 9 below.

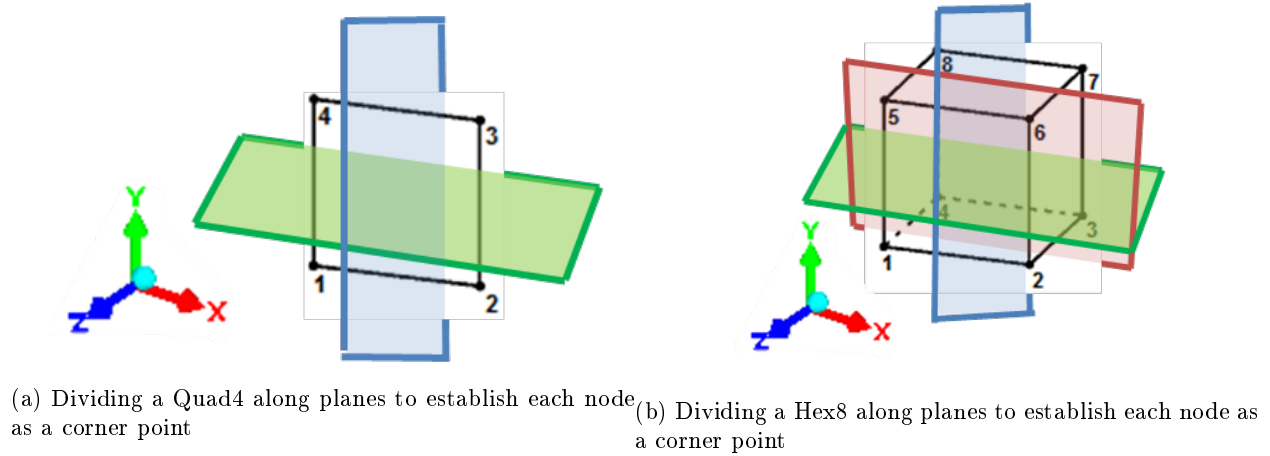


Figure 10: Splitting Element points using x, y and z planes in order to perform ordering for LISA

Although this approach resolved the initial problems resulting from simply trying to traverse the nodes based on length it was still not a complete solution. Even more complex non square elements such as four point tetrahedrons (see appendix A) cannot always have their nodes divided evenly in cases where several reside on single plane.

In order to resolve this issue I needed to research and apply a convex hull algorithm to this problem. The subsequent solution was the following:

Despite the existence of algorithms for generating a convex hull in three dimensions due to the small number of points it made sense to assess the effectiveness of a 2d method for solving the problem first. Every Quad4 element therefore needed to be flattened to a 2d representation by calculating the maximum delta between the greatest and smallest value on each axis and eliminating the axis with the smallest delta. These new 2d points could be given to the algorithm which when used in conjunction with the approach already taken was able to solve all instances of node ordering within the models.

This method has $O(n \log n)$ time complexity however due to the size of n being 4 in all cases the complexity of sorting an individual element is constant, with the overall complexity of sorting all elements in the model being $O(n)$ where n is the number of elements.

6.8 Stress Based Refinement

To focus meshing in areas of high stress each iteration needed to parse the results file from the previous iterations execution of LISA. LISA result files are in csv format by default and contain the displacements and stresses associated with each node within the model once it has been solved.

Once the data in the output file has been parsed the nodal values for which displacement is known for can be cross referenced against those in the current model by intersecting the lists of node data on node Ids. An evaluation function is then able to determine whether or not any element handed to it meets the criteria for refinement by simply looking at the sum of the stress at its given nodes. If an element is determined to be over the threshold to justify refinement the elements “createChildElements()” method is called to subdivide it further.

6.9 Heuristic/Rule Based Refinement

Each rule is represented as a function within the implementation, this closely resembles the format presented by Dolsak [4, 5, 6] [7]. The rules resides within the “RuleManager” class and each take a number of the defined edges as parameters.

When an instance of the RuleManager is created it parses the edges file provided by the user into a list of edges that the rules can then be executed on.

Every rule then checks the properties of a particular edge against properties which have been identified through the ILP learning algorithm as being important when the model executes. In cases where the rules accept more than one edge as an argument the system attempts to apply the rule to each pair of different edges in the edge list giving a time complexity of $O(n^2)$ where n is the total number of defined edges.

If a rule detects a relationship in the model the edge is assigned a criticality rating as defined by the rule, the value is then used by the meshing procedure to determine how many times it should re mesh the elements along that edge.

The properties that can exist between two edges when compared are the following:

- Edges opposite one another - the edges run alongside one another closely, look at the distance between each of the corresponding nodes and check whether this distance is less than some threshold amount
- Edges possess the same form -
- Edges are considered the same - to meet this requirement both edges must be almost the same length, opposite one another and possess the same form.

Since this system relied on a persistent definition of edges across multiple refinement iterations another challenge was to correctly redefine edges in terms of the newly created nodes so that after meshing had occurred the rules could be re applied to a refined edge to potentially refine it further.

```
1 reference | 0 changes | 0 authors, 0 changes
private void rule7(Edge edgeA, Edge edgeB)
{
    const int INVOLVED_EDGES = 3;

    bool b1 = edgeA.GetEdgeType() == Edge.EdgeType.important;
    bool b2 = edgeA.GetLoadType() == Edge.LoadingType.notLoaded;
    bool b3 = edgeA.isNeighbour(edgeB);
    bool b4 = edgeB.GetEdgeType() == Edge.EdgeType.importantShort;

    if(b1 && b2 && b3 && b4)
    {
        edgeA.ElementCount = INVOLVED_EDGES;
    }
}
```

(a) Code implementation of rule 7 provided by dolsak within the RuleManger class

mesh(A,3) :-
 important(A),
 not_loaded(A),
 neighbour(B,A),
 important_short(B).

(b) Rule 7 as stated by dolsak in his papers [6]

6.10 Mesh Quality Assessment

Dittmers rules for computing the quality of both individual elements and the entire mesh are built into their own "MeshQualityAssessments" and "ElementQualityMetrics" classes, the latter of which is encapsulated within an element object, like with refinement this allows each element to assess its own quality removing the need for additional utility classes and static methods.

Since each element is initialised with the nodes that comprise it, it is also possible to derive all the geometric characteristics and thus its quality metrics upon its initialisation. This allows the metrics for each node to also be calculated upon its initialisation removing the risk of elements returning null when asked for them.

6.11 Implementation Issues

Implementation of the design was not without its difficulties, many of which arising as a consequence of unforeseeable complications when implementing the well understood theoretical aspects.

6.11.1 Attempts to Automatically Define edges within models

As discussed under evaluation one of the main issues faced was identifying where the system behaved poorly through weakness in the methods or poor design and implementation as opposed to poor output generated as a consequence of poor input by the operator. In order to avoid this issue it was desirable to try and remove user intervention besides the configuration of the initial model. The obvious way by which to do this was automatic identification of interesting edges within the model. A crucial property which made this approach appear promising is the fact that it is known that edge importance directly correlates with the size of the edge and how much force is applied near it, since both this information exists within the data model it should be possible to identify edges from it and generate them automatically for Dolsaks rules to process.

In practice there are multiple complications surrounding this, several of these arise from ambiguity in Dolsaks paper regarding what constitutes a an edge which is for example "long" or "important". As a result edges are only able to be defined to the extent that a user has confidence in their understanding of these concepts.

Having considered a method using these properties of the mesh several days were then spent attempting to implement it with poor success achieved.

7 Evaluation of Project

Evaluation of the project consisted of multiple stages ranging from verification of the functional requirements through simple black box testing to evaluation of refined mesh quality using methods provided by Dittmer [14].

7.1 Validation against Functional Requirements

In order to validate the system against many of the functional requirements the system only needs to be run on several basic models with different input configurations.

output clearly demonstrates the systems ability to evaluate the quality of meshes using a range of metrics and refinement occurring as a result of both the stresses induced by the user and on their categorisation of edges.

7.2 Validation against Non Functional

Validation of non functional requirements was made simple due to the limited number of them, this was partly a consequence of the system not being designed for a specific user base resulting in expectations regarding the systems design to improve usability and guide interaction. It was also not possible to define the general accuracy and performance of the system during the requirement elicitation phase since this could only be determined once the required research and trial on the finished system gave indication to both of these.

In the case of quality for the systems design and documentation evidence is present to indicate that this adheres to the requirements specified with the project submission containing detailed documentation in the form of a Deoxygen guide and sophisticated use of object oriented and functional software design as seen within the codebase. General applicability of functionality has also been demonstrated through evaluation using a variety of both models and conditions when performing simulations.

7.3 Unit Testing

Holistic evaluation provided evidence of the overall systems effectiveness however without verification of individual components it would not have been possible to assess the accuracy of the results generated by the underlying methods. Unit testing was also conducted from within Visual Studio using the NUnit framework and structured as a separate VS project. This provided the guarantee that the system was not able to interact with the tests and that testing was conducted through the class and function interfaces provided by the solution. Tests were grouped into classes with each test class corresponding approximately to one class within the system. Each test class then contains a number of test functions each of which performing the asserts necessary to deem its associated function as correct. This layout provided clear traceability from each item of function to its associated test making assessment of the test coverage much easier.

When designing the tests it was important to not only to

The design and execution of the tests did manage to reveal a bug within the system .

Much of the software

this allowed test names to replicate their partner functions making the

7.4 Software Management

In order to ensure progress was responsibly backed up and features could be easily managed a private Github repository was set up and all progress made to the project pushed every couple of days. This proved invaluable on one occasion where a bug was accidentally introduced and despite efforts could not be removed manually.

7.5 Software Quality and Maintainability

The quality of the design and implementation of the system reflects my experience not only as a computer science undergraduate but as a developer with one year industrial experience, although not directly effecting the execution of the program properties such as appropriate variable naming, loose coupling of classes, use of abstractions and descriptive error messages make the software easier to read and debug for any potential future developers.

Visual Studio also enabled calculation of various software quality metrics for the code base automatically. This made selecting parts of the codebase for refactoring much easier when time was allocated for this. Upon completion of the project the average maintainability index across all modules was 75 with the lowest score for any high level module being 60 and the highest 92. According to the Microsoft Developer Network (MSDN) website code with an index of between 0 and 9 indicates low maintainability, 10 to 19 indicates moderately maintainable and 20 to 100 high maintainability [36].

7.6 Documentation

The process of continuously writing descriptive documentation was important to the success of the project and was treated as an integral part to meeting to the goals of the project development methodology which aimed to reduce the systems complexity and improve readability. Through the writing Doc comments corresponding to every function within the codebase it was possible to generate documentation files automatically through use of the tool Doxygen. This allows anyone with the solution to view descriptions of each of its functions either in the codebase or alternatively through the manual produced automatically by Doxygen.

7.7 Increase in performance through parallel execution

The following

The average speed up (Time of serial execution / Time of parallel execution) was

The Efficiency of the parallel execution (speedup / number of processors) with an Intel i5 processor with 4 cores the average efficiency was calculated as

7.8 Evaluation Of System For Model Simulations

several models have been created resembling basic equivalents of that used for real engineering applications.

do this reliably and validate the heuristic component against the results obtained in academic papers two of the three models were based on those previously used by Dolsak (Paper mill and Cylinder structures). In addition to these two a suspension bridge model was developed to demonstrate the systems capacity to work beyond Dolsaks basic validation examples.

These initial models were constructed manually using LISA's graphical user interface using measurements by Dolsak and in the case of the suspension bridge from documents available on the web [4] [?].

7.8.1 Suspension Bridge structure

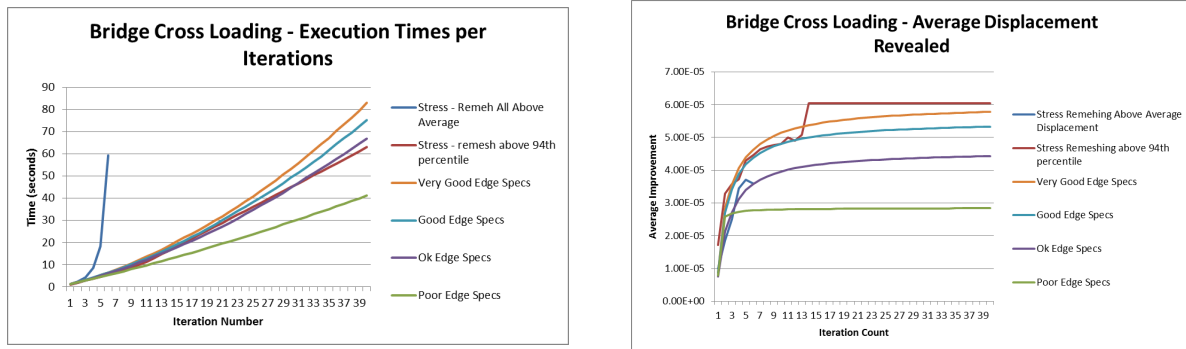
The main model used for testing the methods was a suspension bridge structure. The bridge seemed like a good model to perform initial experimentation on since its structure is

Constructed manually out of Quad4 LISA elements it has four key constraint points at the base of each supporting column. The bridge itself initially consists of 196 elements and 212 nodes before remeshing, which is considered coarse for such an FE structure.

When testing the system with the suspension bridge model the first step was to test each method independently of the other in order to evaluate both approaches before attempting to combine the two. Testing each method individually with multiple configurations provided a better understanding of the variation within

each allowing more accurate high level reasoning about the results of a hybrid.

Variation of inputs was particularly important when testing Heuristic method as experimentation revealed a very high amount of variation in the results both in terms of runtime and the methods ability to reveal high stress areas, this can be seen in Figure 9 below. Four main tests were run for the heuristic approach with each test having a different edge specification file of some level of a designated level of quality. With the very good and good edge specifications I made sure to run the model first using stress refinement and then observe areas of high displacement, this allowed specification of edges around areas I knew in advance would benefit from being specified. The idea being a more experienced FE engineer would be able to do this without having to run the stress refinement first. In addition to the two good quality edge specification sets an OK one was also produced where some edges were specified as existing over the highest stress area while others focused mainly on the lower stress areas, and finally a set of poor edges none of which overlapped the high stress area.



(a) Execution time in seconds for each method to reach 40(b) Average amount of displacement and thus stress revealed by each approach as number of iterations increases

Figure 12: Execution time increase compared to the amount of information revealed for the different approaches

For stress refinement the parameter varied was the threshold used to determine whether elements were considered under high stress and should be further refined. The main consequence of varying this parameter was change in both the node count and the programs runtime. Despite time complexity for subdivision being $O(n^2)$ for all methods using a low threshold results in large values of n and consequently a rapid increase in both runtime and element count.

In order to perform reasonable comparison between both the stress and heuristic refinement methods it was therefore important to specify the threshold such that approximately equal amounts were performed by each approach per unit of weighting preference per iteration (see combining methods under System Design), thus allowing each method to be evaluated on its merit to select elements appropriately and a hybrid method through its designated weightings. To do this the increase in element count was tracked for the different heuristic methods, as can be seen in Figure 10. The average increase in elements per iteration could then be calculated as 6% of the total number of elements within the model. This meant stress refinement could be reasonably compared to heuristic refinement if for each use of the refinement it only re meshed those elements considered to be above the 94th percentile in terms of displacement.

Evaluation of the data Shows that in general refinement based on displacement and stress consistency results in well refined meshes although depending on the threshold this can be highly costly to compute. By contrast heuristics have shown to retain low computational overhead although are volatile when needed to produce meshes of high quality. Where edges are well specified for the heuristics quality can be seen to

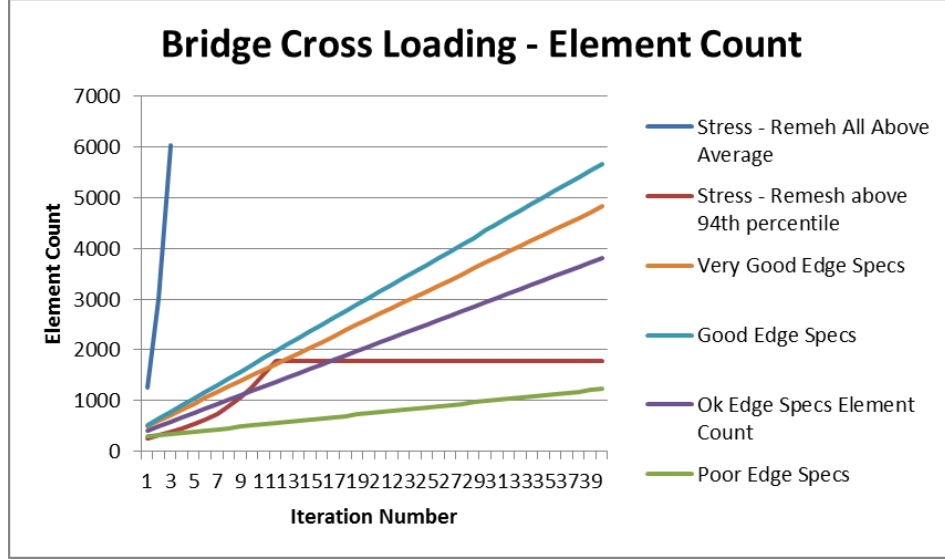


Figure 13: Execution time in seconds for each method to reach 40 refinement iterations

increase more rapidly than refinement based on displacement data. Heuristic refinement slows down with subsequent iterations which is likely a result of the regions yielding improvement with refinement becoming increasingly small and thus increasing the distance between the edges originally specified and the target meshing location.

Heuristics therefore appear preferable in cases where they can be confirmed as good and where very high accuracy is not as important as reduced computational overhead. By contrast displacement refinement works well when high accuracy is needed although incurs a higher computational overhead per gain in accuracy early on.

The idea hybrid configuration will therefore exhibit the following two characteristics:

Edge Specification assessment: Establish quickly if the edges specified are any good when meshing, if they aren't stop continued use and switch to displacement refinement.

Heuristic Refinement Halting: If the heuristic refinement does appear to be working well establish at what point it is beginning to become inferior and then switch methods.

Re executing the system with a variety of hybrid weightings

7.8.2 Hybrid Evaluation

7.8.3 Analysis of and Addition of Metric(s)

When looking at the initial results of the quality metrics for both the various simulations there were several key observations which greatly informed the remaining evaluation to be conducted.

The maximum internal angles to elements are one good indication of general quality for the mesh and indeed gradual improvement with additional refinement iterations increasingly subdividing skewed elements into more uniform ones. Figure 11 below shows greater uniformity towards the optimum of 90 degrees for elements of type Quad4 is achieved for all but the poorest edge specification. Rate of improvement is fast for the first few iterations but then gradually declines

Looking at the Figure its also clear that as a result of the heuristics being described in terms of edges

Although the Dittmer metrics providing a good indication of general improvement in quality across a mesh or a particular element in the majority of cases they gave little insight into the strengths of a particular element arrangement within a given mesh. Consequently it was possible to conclude using Dittmers metrics

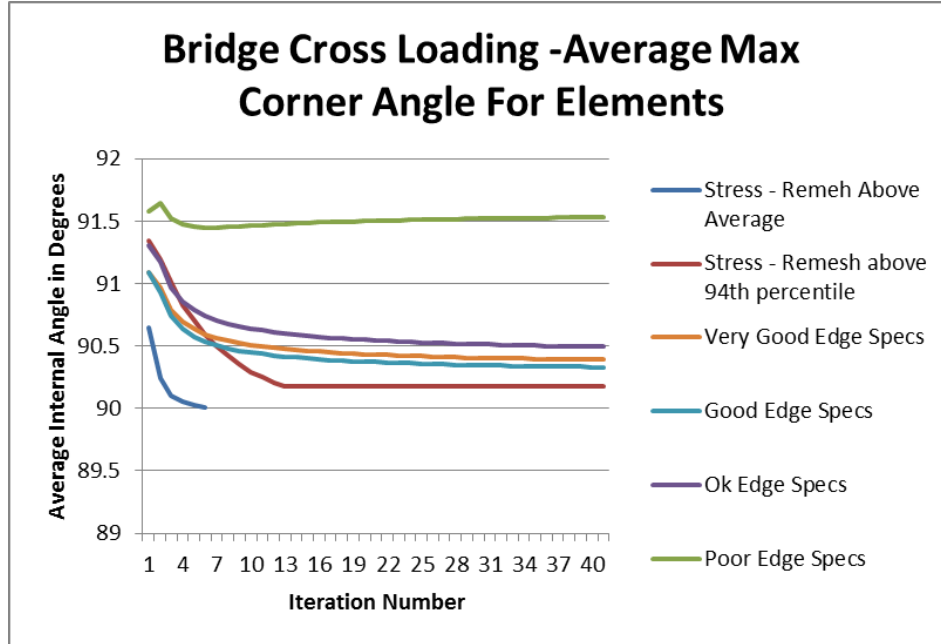


Figure 14: Execution time in seconds for each method to reach 40 refinement iterations

that the general quality of the resultant meshes was good with this quality being at least retained and often improved, but not that the method refining the mesh had made a better selection of elements to mesh. [22].

It became apparent that for a more conclusive evaluation another metric was also required. To design such a metric I started by reflecting on the fact that each method should provide as much information as possible to an engineer while retaining the lowest computational overhead possible, essentially what much additional information is revealed through each additional iteration of a given approach. It can also assume in this case that the concentration of displacement and therefore stress is what we are interested in finding. I concluded that the best means of evaluating each method was to cross reference the resultant stress data for each iteration against the newly refined regions generated by each approach, the overall stress revealed within the output as a result of that part of the mesh being refined could then be summed and divided by the number of nodes created in order to reveal it through evaluation. This new metric became the average displacement revealed metric as can be seen in Figure 9b.

In order to evaluate general consistency in these results the experiments were rerun for Results for which can be seen in

7.9 Evaluation of Subsystems based on model evaluation

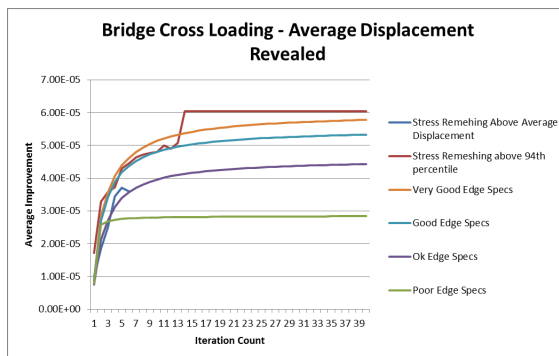
Running the system for the above models gave clear indications as to the overall effectiveness of both methods

Having run the system on the range of models described above it was possible to begin assessing it's ability to mesh and the use of Dittmers metrics for assessing the quality of each mesh.

7.9.1 Analysis of Methods Using Metrics

When evaluating Dittmers metrics for distinguishing between meshes of different qualities there were several key observations that resulted in re assessment of how to evaluate the mesh qualities in general.

Having coded this into the MeshQualityAssessment class the results for this metric could be plotted to evaluate each approach over a number of iterations for the different simulations. I also varied the selection of edges provided to heuristic refinement method which highlighted just how dependent the heuristic approach was on the quality of user expertise.



(a) Bridge Cross Loading -Average Displacement Revealed Through Creation of Node per Iteration the plateau experienced by Stress Remeshing above 94th percentile is a consequence of all subsequent meshing being focused on an area already very highly meshed, in other words

7.9.2 Evaluation of Stress Refinement Effectiveness

Comparisons across multiple models suggests that for general purpose scenarios where results of the simulation are highly unpredictable and the geometry irregular stress based refinement will continue to provide engineers with reliable results within a predictable although potentially costly amount of time. This scenario likely encapsulates the majority of use cases for FE analysis in the real world and therefore is likely to remain the prominent approach by practitioners.

It is clear however that the decline in performance is dependent upon the threshold used to determine meshing

7.9.3 Evaluation of Heuristic Refinement Effectiveness

The most crucial observation which can be made through analysis of the simulation data is that heuristic equation based on model edges has resulting mesh quality has high variability. Despite this it is also clear that despite high variation in results for different edge specifications the general rate of growth or decline for the resultant metric values remains consistent.

7.9.4 Evaluation Issues

A significant issue faced in attempting to demonstrate the effectiveness of the system was to provide an indication of how well the system worked without taking into account the ability of the user who may be

providing the edge rules for a particular model. Not taking this into account would result in an inaccurate representation of its ability.

7.10 Strengths and Weaknesses

The resulting system successfully satisfied both the functional and non functional requirements in addition to providing insights into the possibilities of a hybrid technique for effective finite element meshing, something that was optimistic at the start of the project but highly desirable. The project was well managed with deliverables consistent met as per the initial time plan quality maintained through application of good software engineering practice.

One of the great strengths of the system would be its modular architecture which has allowed for a great amount of potential extendibility in the future. Although little focus has so far been given to the systems usability it can ideally be developed and distributed as tool for experimentation with hybrid meshing with limited additional effort.

Another strength of the system is its ability to accept any heuristic definition in terms of edges within a mesh structure. Theoretically this means the final system is also capable of using the same types of edge specifications for any type of FE analysis such as fluid flow or heat transfer given a corresponding rule set by which to mesh with.

A downside of the current design is manual specification of the edges by the user directly into the JSON input file which is both time consuming and prone to error despite the relatively small size of the models analysed in this dissertation. Comparing the size of these with those those used in industry it is clear that this process is simply not practical for engineers conducting FE analysis. To change this better tools are required that will allow engineers to automatically generate edge specifications quickly, most likely through some GUI or a bespoke high level language capable of combining knowledge about the mesh structure and different types of edges to generate specific rules. Again this is beyond the scope of the project and could potentially be a dissertation in its own right.

Although the system had a strong subsystem and class level architecture many of its greater weaknesses could be attributed to prioritisation of its rapid prototyping over efficient implementation of the various algorithms and methods described in this dissertation. Much of this a consequence of overusing the functional programming capabilities within the C# LINQ library. Widespread adoption of functional programming practices was stated as a desirable aspect of the final system implementation within the original non functional requirements. This has largely been adhered to with higher order and lambda functions widespread throughout the codebase. In the later stages of the project however it became clear that in many cases over reliance on these features had resulted in reduced readability for many of underlying algorithms described in this dissertation.

In many cases over reliance on map resulted in

As a consequence of trying to embrace the functional programming style as part of a practical software development project

Additionally the system demonstrated also proved capable of providing interesting insights into the possibilities of performing hybrid and even demonstrated that

7.11 Evaluation summary

7.12 Successes and Limitations of project

Having reflected on the the results of the methods implemented within the system through the execution of

8 Further Work

This section details some areas which given additional time to work on the project would at the very least be investigated, if not implemented. Each of these areas would hopefully provide some benefit in assisting to demonstrate the possibilities of hybrid methods.

8.1 Gathering feedback from experienced engineers

Approaching the end of the project it became clear that in order to better identify the systems strengths and weaknesses would require additional user testing by engineers who have experience conducting this type of analysis. Despite a lack of available time obtaining feedback from engineers with extensive applied industrial experience along with that of academics would have hopefully allowed for a more conclusive analysis of the systems and its ability to work across a greater number of general case scenarios. User feedback was largely not obtained within the duration of the project as a result of time constraints and the complexity inherent in simply implementing and validating the project for a selection of basic models. As such even if time had been available the ethical clearance required to collect user feedback at the start was not obtained.

8.2 Improving usability through a web interface

Although possible to visit various engineers in order to conduct feedback the process is both time consuming on my part and inconvenient for the participant as a rigid time for which to meet must be scheduled and a laptop containing the working software brought to them which they must design or transfer their model to before running it multiple times to obtain results. This scenario is at best inconvenient for the participants and pressures them into arriving at a conclusion within a relatively small time of experimenting with it.

Instead by facilitating interaction with the system by means of a web interface the engineers would be able spend as little or as much time as they like experimenting with the system and allow them to submit feedback digitally allowing feedback to be obtained and aggregated from a much wider range of different sources separated by significant geographic distance.

To use the interface an engineer would simply need to submit a model they have already created along with a JSON file containing edges they have designated as important for their model. LISA supports imports from multiple CAD formats including Standard for the Exchange of Product model data (STEP) and Initial Graphics Exchange Specification (IGES)) [19]. Upon receiving the request the web server would the current project with their input data and having finished allow them to download the re meshed model along with the calculated stress data for analysis.

9 Personal Reflections and Summary

Overall I felt the project was a success. The final software solution was evidently capable of facilitating execution and therefore allowing comparison to be made of both heuristic and stress based mesh refinement techniques. Not only did the final system allow this comparison for the two specific approaches that were coded by me in order to validate the system it provided a flexible framework allowing for a potentially unlimited set of configurations when performing future experimentation.

From my perspective I wanted to use this project as an opportunity to improve my understanding of a technology that I previously had limited knowledge of through its use on my industrial placement year. My prior experience with with FE analysis was very much confined to that of a typical engineer making use of the method through a licensed desktop application with many of the technicalities that are of most interest to a computer scientist hidden. I therefore found the project highly enjoyable as an opportunity to learn

more about the underlying processes through both research and practical experimentation. As a means of facilitating my own learning as an individual I therefore also consider the project a success.

Despite working on larger software projects during my year within industry this was certainly the most complex I have undertaken as an individual. As the lead software developer on my own project I encountered many challenges which as a junior developer within industry were not my responsibility but which I observed team leaders and senior developers encountering regularly. Such tasks were those that required high level analysis of the overall solution in order to continuously steer the project in the right direction and ensure successful delivery within the specified time scales. In many ways the project therefore gave me a good appreciation of the overall difficulties associated with delivering a software project in its entirety and one that has the potential change fluidly throughout.

In particular I found the research and evaluation phases particularly difficult. Upon completion I came to realise this was largely due to my lack of formal education in mechanical engineering which meant I had to work a lot harder both to understand the initial problems associated with the methods and subsequently to correctly evaluate the results I obtained. Had I chosen a more traditional computer science topic I believe both of these aspects would have been much easier.

Having completed the research too much time was then spent concerned with specifics of the implementation.

A negative consequence of the bespoke nature of the system was that obtaining details of designs results for comparable systems published by the wider academic community was both challenging and highly time consuming.

Throughout the majority of the project I felt organisation of time and planning of activities was done well. Work on the project began early with the goal of easing pressure in the later stages and work continued despite deadlines for coursework associated with other modules. A crucial mistake made was to reduce effort two months before the deadline having completed the software implementation and written much of the initial sections of the dissertation despite not completing evaluation of the results. Evaluation of the software turned out to be substantially more time consuming than expected in addition stresses towards the deadline.

Given the chance to redo the project I would have liked to conducted more research to better assess an even wider scope of research surrounding meshing methods. In many cases the academic literature surrounding bespoke meshing techniques became complicated very quickly and was clearly aimed at individuals with a high degree of prior experience.

By contrast I found the system design and programming components
planning and research components

managerial architectural aspects which I had not previously had that great an involvement with

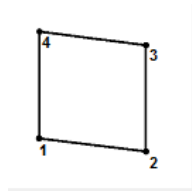
References

- [1] Max D. Gunzburger, Janet S. Peterson *Finite Element Methods* https://people.sc.fsu.edu/~jburkardt/classes/fem_2011/chapter1.pdf
- [2] Adaptive Finite Element Techniques <http://www.cs.rpi.edu/~flaherje/pdf/fea8.pdf>
- [3] Scott McRae *r-Refinement grid adaptation algorithms and issues*
- [4] Bojan Dolsak and Anton Jezernik *Mesh generation expert system for engineering analysis with FEM*
- [5] Bojan Dolsak, Anton Jezernik *A knowledge base for finite element mesh design* Artificial Intelligence in Engineering 9 (1994)
- [6] Bojan Dolsak, Stephen Muggleton *The Application of Inductive Logic Programming to Finite Element Mesh Design*
- [7] Bojan Dolsak, Frank Reig, Reinhard Hackenschmidt *Consultative Rule-Based Intelligent System for Finite Element Type Selection* Research Gate 2016
- [8] Paul Dvorak *Two meshing methods are better than one* <http://machinedesign.com/archive/two-meshing-methods-are-better-one>
- [9] Larry Manevitz, Malik Yousef, Dan Givoli *Automatic Mesh Generation (for Finite Element Method) Using Self-Organising Neural Networks*
- [10] Abid Ali Khan, Imran Ali Chaudhry2 & Ali SaroshCase *Case Based Reasoning Support for Adaptive Finite Element Analysis: Mesh Selection for an Integrated System*
- [11] Stephen Muggleton *Inductive Logic Programming*
- [12] <http://www-ai.ijs.si/~ilpnet2/systems/golem.html>
- [13] Stephen Muggleton *Logic based and Probabilistic Symbolic Learning* <https://www.youtube.com/watch?v=4Cwd05dWW98>
- [14] Jeremy P. Dittmer, C. Greg Jensen, Michael Gottschalk, and Thomas Almy *Mesh Optimisation Using a Genetic Algorithm to Control Mesh Creation Parameters*
- [15] <http://danielpeter.github.io/rays.html>
- [16] Lina Vasiliauskiene, Romualdas BAUŠYS *Intelligent Initial Finite Element Mesh Generation for Solutions of 2D Problems* INFORMATICA, 2002, Vol. 13, No. 2, 239–250 2002
- [17] E.Bellengera,Y.Benhafidb, N.Troussierb *Framework for controlled cost and quality of assumptions in finite element analysis* Finite Elements in Analysis and Design 45 (2009) 25–36
- [18] G. P. Nikishkov *INTRODUCTION TO THE FINITE ELEMENT METHOD* <http://homepages.cae.wisc.edu/~suresh/ME964Website/M964Notes/Notes/introfem.pdf>
- [19] <http://www.lisafea.com/pdf/manual.pdf>
- [20] Nam-Ho Kim *STRUCTURAL DESIGN USING FINITE ELEMENTS* http://web.mae.ufl.edu/nkim/eas6939/Opt_FEM.pdf
- [21] *Type of Finite Elements and Steps in FEA Process* <http://highered.mheducation.com/sites/dl/free/0073398144/934758/Ch07TypesOfFiniteElementsAndStepsInFEAProcess.pdf>
- [22] *Element Quality And Checks*http://www.altairuniversity.com/wp-content/uploads/2012/04/Student_Guide_211-233.pdf
- [23] <https://www.quora.com/What-is-the-cantilever-beam-What-is-the-advantages-and-disadvantages-of-it>
- [24] <http://www.lisafea.com/purchase.html>
- [25] <http://www.stack.nl/~dimitri/doxygen/>

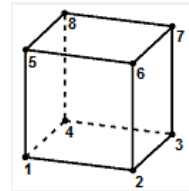
- [26] <https://caeai.com/blog/will-poorly-shaped-elements-really-affect-my-solution>
- [27] <http://mscnastrannovice.blogspot.co.uk/2013/04/how-much-does-ansys-cost.html>
- [28] <http://www.engineeringanalysiservices.com/moving-mesh-fea-analysis.php>
- [29] <http://loyc.net/2014/2d-convex-hull-in-cs.html>
- [30] <http://physicsnet.co.uk/a-level-physics-as-a2/materials/young-modulus/>
- [31] <http://silver.neep.wisc.edu/~lakes/PoissonIntro.html>
- [32] Jonathan Richard Shewchuk *Delaunay Refinement Algorithms for Triangular Mesh Generation*
<https://people.eecs.berkeley.edu/~jrs/papers/2dj.pdf>
- [33] Eliannah Hunderfund and Professor Ernesto Gutierrez-Miravete Rensselaer *Finite Element Analysis of a Rotating Disk*
- [34] <http://www.pulpandpapercanada.com/news/coupling-changes-cut-paper-machinery-maintenance-1000108660>
- [35] http://www.engineeringtoolbox.com/centripetal-acceleration-d_1285.html
- [36] <https://msdn.microsoft.com/en-gb/library/bb385914.aspx>

A Element Types within LISA

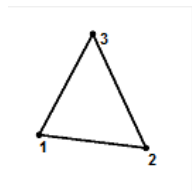
Here are shown the the visual specifications LISA provides for the ordering and layout of nodes for defining each type of element supported. Each of these types can be classified using the



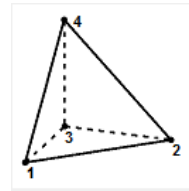
(a) quad4 element



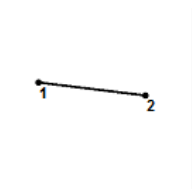
(b) hex8 element



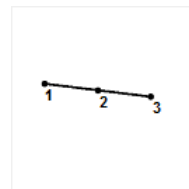
(a) Specification for node ordering of tri3 element within LISA



(b) tet4 element



(a) line2 element



(b) line3 element

B Unit Testing

[35]

C Input and output files

Below can be seen the format of the input files for the system, a LISA .liml and a .json edge definition file

```
1 <liml>
2 <analysis type="880" />
3 <node nid="1" x="0" y="0" z="0" />
4 <node nid="2" x="0" y="0" z="20" />
5 <node nid="3" x="100" y="0" z="20" />
6 <node nid="4" x="100" y="0" z="0" />
7 <node nid="5" x="0" y="100" z="0" />
8 <node nid="6" x="0" y="100" z="20" />
9 <node nid="7" x="100" y="100" z="20" />
10 <node nid="8" x="100" y="100" z="0" />
11 <node nid="9" x="0" y="0" z="100" />
12 <node nid="10" x="0" y="0" z="120" />
13 <node nid="11" x="0" y="100" z="100" />
14 <node nid="12" x="0" y="100" z="120" />
15 <node nid="13" x="100" y="100" z="100" />
16 <node nid="14" x="100" y="100" z="120" />
17 <elem name="Default" color="010000" material="Material">
18 <elem eid="1" shape="quad" nodes="1 4 3 2" />
19 <elem eid="2" shape="quad" nodes="1 2 3 4" />
20 <elem eid="3" shape="quad" nodes="5 2 1 4" />
21 <elem eid="4" shape="quad" nodes="5 4 1 2" />
22 <elem eid="5" shape="quad" nodes="7 4 3 2" />
23 <elem eid="6" shape="quad" nodes="7 2 3 4" />
24 <elem eid="7" shape="quad" nodes="10 4 3 2" />
25 <elem eid="8" shape="quad" nodes="10 2 3 4" />
26 <elem eid="9" shape="quad" nodes="11 4 3 2" />
27 </elem>
28 <file selection="BridgeBase" />
29 <force selection="Downward" />
30 <pin/ />
31 <end>
32
33 <!-- BridgeBase -->
34 <!-- BridgeBase -->
35 <!-- BridgeBase -->
36 <!-- BridgeBase -->
37 <!-- BridgeBase -->
38 <!-- BridgeBase -->
39 <!-- BridgeBase -->
40 <!-- BridgeBase -->
41 <!-- BridgeBase -->
42 <!-- BridgeBase -->
43 <!-- BridgeBase -->
44 <!-- BridgeBase -->
45 <!-- BridgeBase -->
46 <!-- BridgeBase -->
47 <!-- BridgeBase -->
48 <!-- BridgeBase -->
49 <!-- BridgeBase -->
50 <!-- BridgeBase -->
51 <!-- BridgeBase -->
52 <!-- BridgeBase -->
53 <!-- BridgeBase -->
54 <!-- BridgeBase -->
55 <!-- BridgeBase -->
56 <!-- BridgeBase -->
57 <!-- BridgeBase -->
58 <!-- BridgeBase -->
59 <!-- BridgeBase -->
60 <!-- BridgeBase -->
61 <!-- BridgeBase -->
62 <!-- BridgeBase -->
63 <!-- BridgeBase -->
64 <!-- BridgeBase -->
65 <!-- BridgeBase -->
66 <!-- BridgeBase -->
67 <!-- BridgeBase -->
68 <!-- BridgeBase -->
69 <!-- BridgeBase -->
70 <!-- BridgeBase -->
71 <!-- BridgeBase -->
72 <!-- BridgeBase -->
73 <!-- BridgeBase -->
74 <!-- BridgeBase -->
75 <!-- BridgeBase -->
76 <!-- BridgeBase -->
77 <!-- BridgeBase -->
78 <!-- BridgeBase -->
79 <!-- BridgeBase -->
80 <!-- BridgeBase -->
81 <!-- BridgeBase -->
82 <!-- BridgeBase -->
83 <!-- BridgeBase -->
84 <!-- BridgeBase -->
85 <!-- BridgeBase -->
86 <!-- BridgeBase -->
87 <!-- BridgeBase -->
88 <!-- BridgeBase -->
89 <!-- BridgeBase -->
90 <!-- BridgeBase -->
91 <!-- BridgeBase -->
92 <!-- BridgeBase -->
93 <!-- BridgeBase -->
94 <!-- BridgeBase -->
95 <!-- BridgeBase -->
96 <!-- BridgeBase -->
97 <!-- BridgeBase -->
98 <!-- BridgeBase -->
99 <!-- BridgeBase -->
100 <!-- BridgeBase -->
```

```
1 {
2   "Edges": [
3     {
4       "Id": 1,
5       "edgeType": "circuit",
6       "loadType": "oneSideLoaded",
7       "boundaryType": "free",
8       "nodePath": [42, 198, 197, 41]
9     },
10    {
11      "Id": 2,
12      "edgeType": "importantLong",
13      "loadType": "oneSideLoaded",
14      "boundaryType": "fixedTwoSides",
15      "nodePath": [70, 62, 61, 46, 45, 67]
16    },
17    {
18      "Id": 3,
19      "edgeType": "importantShort",
20      "loadType": "oneSideLoaded",
21      "boundaryType": "fixedTwoSides",
22      "nodePath": [41, 197]
23    },
24    {
25      "Id": 4,
26      "edgeType": "notImportant",
27      "loadType": "notLoaded",
28      "boundaryType": "fixedCompletely",
29      "nodePath": [197, 207]
30    },
31    {
32      "Id": 5,
33      "edgeType": "circuit",
34      "loadType": "continuousLoading",
35      "boundaryType": "fixedOneSide",
36      "nodePath": [88, 142, 148, 147, 140, 141, 135, 85]
37    }
38  ]
39 }
40
```

(a) Cut down .liml file to show general content which largely defined the schema for the systems data model by an engineer, this is parsed and the rules are applied to determine the models meshing based on the input

D Project Layout in Solution Explorer

Below show the Visual Studio Solution Explorer which provides a general idea of the layout of the project with namespace hierarchies from within an IDE.

E Software Quality Metrics

F Bridge Cross Loading Simulation Results

Below can be seen the results of executing both the heuristics and the stress refinement method on the suspension bridge model when loads are placed against the base of the bridge in the negative x direction.

F.1 Heuristic Refinement

F.2 Stress Refinement

G Bridge Base Loading Simulation Results

H Paper Mill Simulation Results

I Gantt Chart for Project Time Management

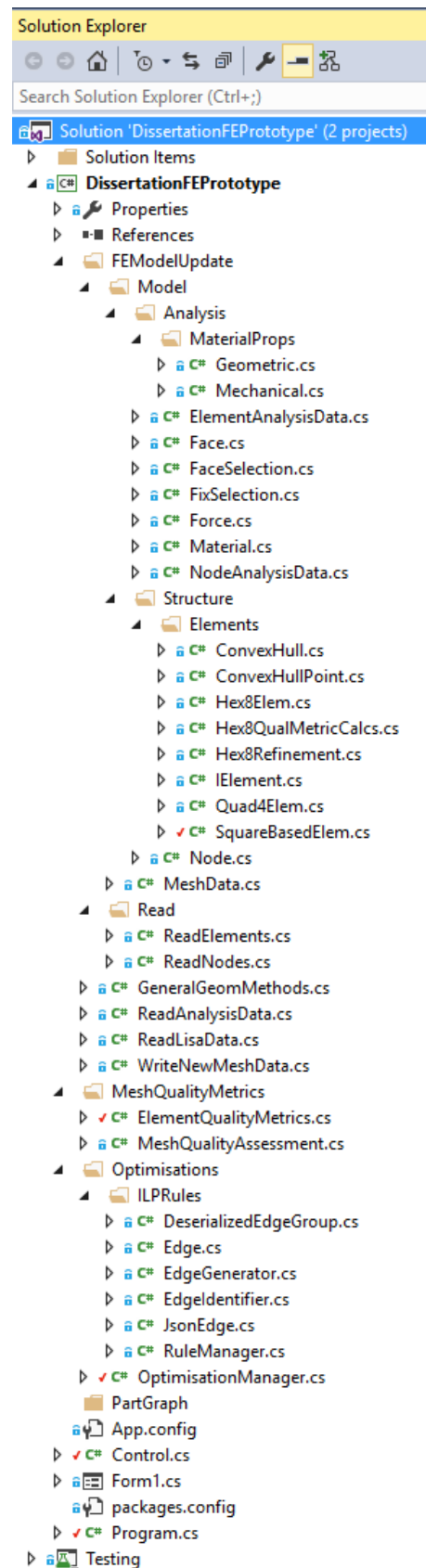


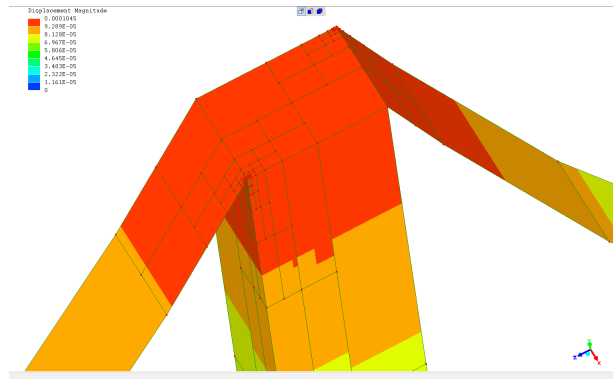
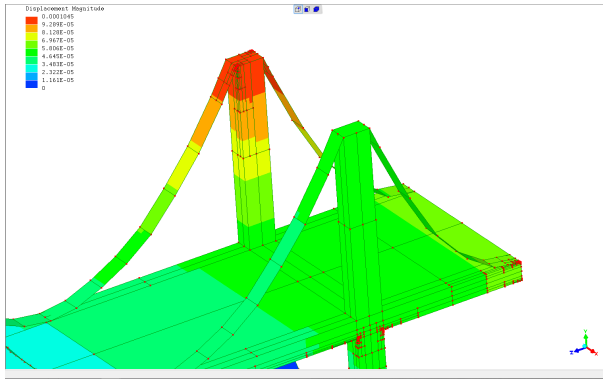
Figure 20: The metrics calculated by visual studio for all high level modules in the system

Code Metrics Results						
Filter: None						
Hierarchy		Maintainability Index	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
[-] DissertationFEPrototype (Debug)		75	808	7	125	1,871
[-] DissertationFEPrototype		62	24	7	39	126
[-] DissertationFEPrototype.FEModelUpdate		64	35	1	29	115
[-] DissertationFEPrototype.FEModelUpdate.Model		90	17	1	11	27
[-] DissertationFEPrototype.FEModelUpdate.Model.Structure		90	15	1	5	35
[-] DissertationFEPrototype.FEModelUpdate.Model.Structure.Elements		69	329	2	35	684
[-] DissertationFEPrototype.FEModelUpdate.Read		60	26	1	21	62
[-] DissertationFEPrototype.MeshQualityMetrics		71	43	1	18	78
[-] DissertationFEPrototype.Model		68	4	1	0	26
[-] DissertationFEPrototype.Model.Analysis		87	20	1	6	42
[-] DissertationFEPrototype.Model.Analysis.MaterialProps		92	8	1	0	14
[-] DissertationFEPrototype.ModelUpdate		62	37	1	32	129
[-] DissertationFEPrototype.Optimisations		53	57	1	29	161
[-] DissertationFEPrototype.Optimisations.ILPRules		82	193	1	34	372

Figure 21: The metrics calculated by visual studio for all high level modules in the system

Code Metrics Results						
Filter: None		Min:		Max:		
Hierarchy	Maintainability I...	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code	
[-] DissertationFEPrototype (Debug)	75	808	7	125	1,871	
[-] { } DissertationFEPrototype.Model.Analysis.MaterialProps	92	8	1	0	14	
[-] [-] Geometric	92	4	1	0	7	
[-] [-] Mechanical	92	4	1	0	7	
[-] { } DissertationFEPrototype.FEModelUpdate.Model	90	17	1	11	27	
[-] [-] MeshData	90	17	1	11	27	
[-] { } DissertationFEPrototype.FEModelUpdate.Model.Structure	90	15	1	5	35	
[-] [-] Node.Origin	100	0	1	0	0	
[-] [-] Node	80	15	1	5	35	
[-] { } DissertationFEPrototype.Model.Analysis	87	20	1	6	42	
[-] [-] Material	92	5	1	2	9	
[-] [-] FixSelection	87	2	1	1	4	
[-] [-] Face	86	4	1	1	8	
[-] [-] FaceSelection	86	4	1	2	8	
[-] [-] Force	85	5	1	0	13	
[-] { } DissertationFEPrototype.Optimisations.ILPRules	82	193	1	34	372	
[-] [-] Edge.BoundaryType	100	0	1	0	0	
[-] [-] Edge.EdgeType	100	0	1	0	0	
[-] [-] Edge.LoadingType	100	0	1	0	0	
[-] [-] DeserializedEdgeGroup	94	5	1	3	5	
[-] [-] Edge	80	25	1	11	55	
[-] [-] JsonEdge	73	49	1	4	63	
[-] [-] RuleManager	72	28	1	6	74	
[-] [-] EdgeGenerator	62	14	1	23	37	
[-] [-] EdgIdentifier	55	72	1	20	138	
[-] { } DissertationFEPrototype.MeshQualityMetrics	71	43	1	18	78	
[-] [-] MeshQualityAssessment	74	28	1	16	38	
[-] [-] ElementQualityMetrics	68	15	1	7	40	
[-] { } DissertationFEPrototype.FEModelUpdate.Model.Structure.Elements	69	329	2	35	684	
[-] [-] IElement	100	12	0	5	0	
[-] [-] ConvexHullPoint	82	3	1	1	8	
[-] [-] Hex8QualMetricCalcs	76	10	1	9	25	
[-] [-] SquareBasedElem	63	104	1	21	259	
[-] [-] Quad4Elem	62	4	2	11	27	
[-] [-] ConvexHull	61	15	1	7	27	
[-] [-] Hex8Elem	58	24	2	21	80	
[-] [-] Hex8Refinement	50	157	1	11	258	
[-] { } DissertationFEPrototype.Model	68	4	1	0	26	
[-] [-] NodeAnalysisData	78	3	1	0	11	
[-] [-] ElementAnalysisData	58	1	1	0	15	
[-] { } DissertationFEPrototype.FEModelUpdate	64	35	1	29	115	
[-] [-] GeneralGeomMethods	71	4	1	4	14	
[-] [-] ReadMeshData	56	31	1	25	101	
[-] { } DissertationFEPrototype	62	24	7	39	126	
[-] [-] Form1	74	7	7	12	22	
[-] [-] Program	56	10	1	13	43	
[-] [-] Control	55	7	1	20	61	
[-] { } DissertationFEPrototype.ModelUpdate	62	37	1	32	129	
[-] [-] ReadAnalysisData	62	9	1	6	49	
[-] [-] WriteNewMeshData	61	28	1	28	80	
[-] { } DissertationFEPrototype.FEModelUpdate.Read	60	26	1	21	62	
[-] [-] ReadNodes	63	10	1	7	26	
[-] [-] ReadElements	58	16	1	21	36	
[-] { } DissertationFEPrototype.Optimisations	53	57	1	29	161	
[-] [-] OptimisationManager	53	57	1	29	161	

Figure 22: The metrics calculated by visual studio for the all classes in the final system



(a) Important edges specified effectively to facilitate pre-emptive meshing of area which undergoes high displacement and thus stress (b) Close up view of refinement for high displacement areas

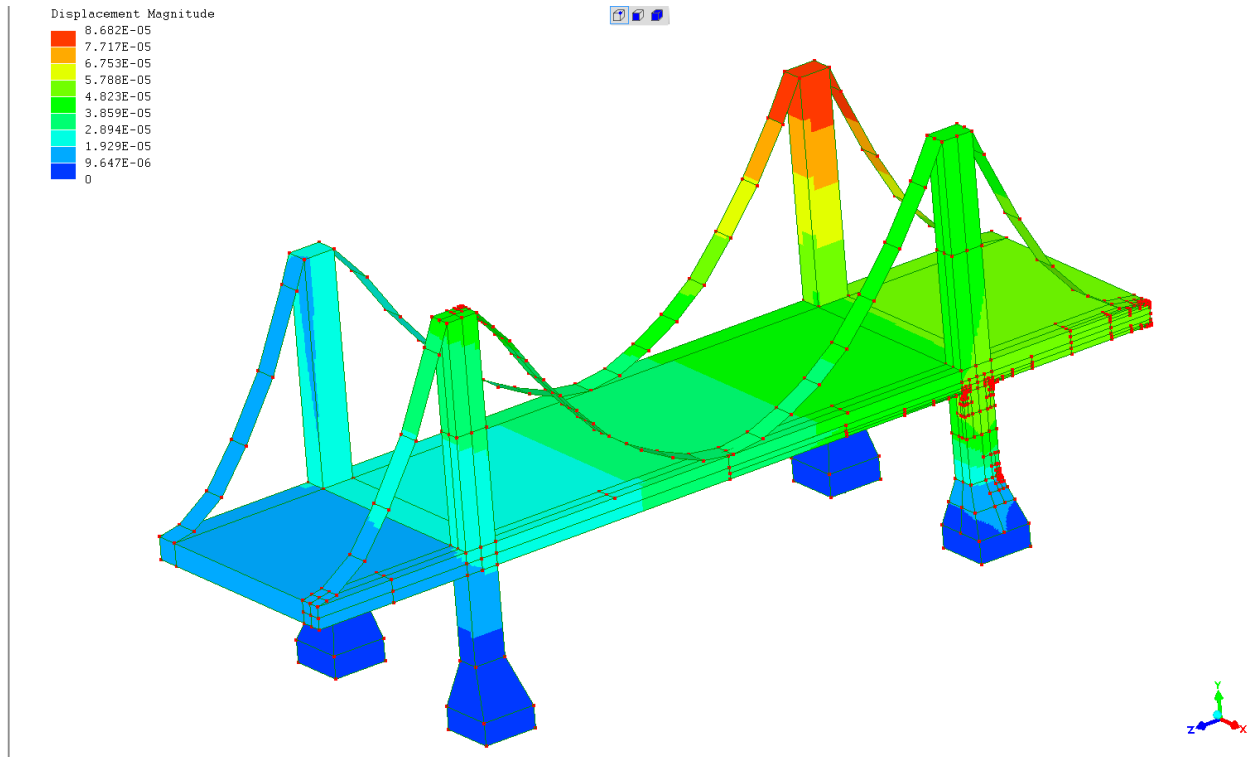


Figure 24: Important edges more poorly specified missing high displacement region on top of furthest suspension bridge tower

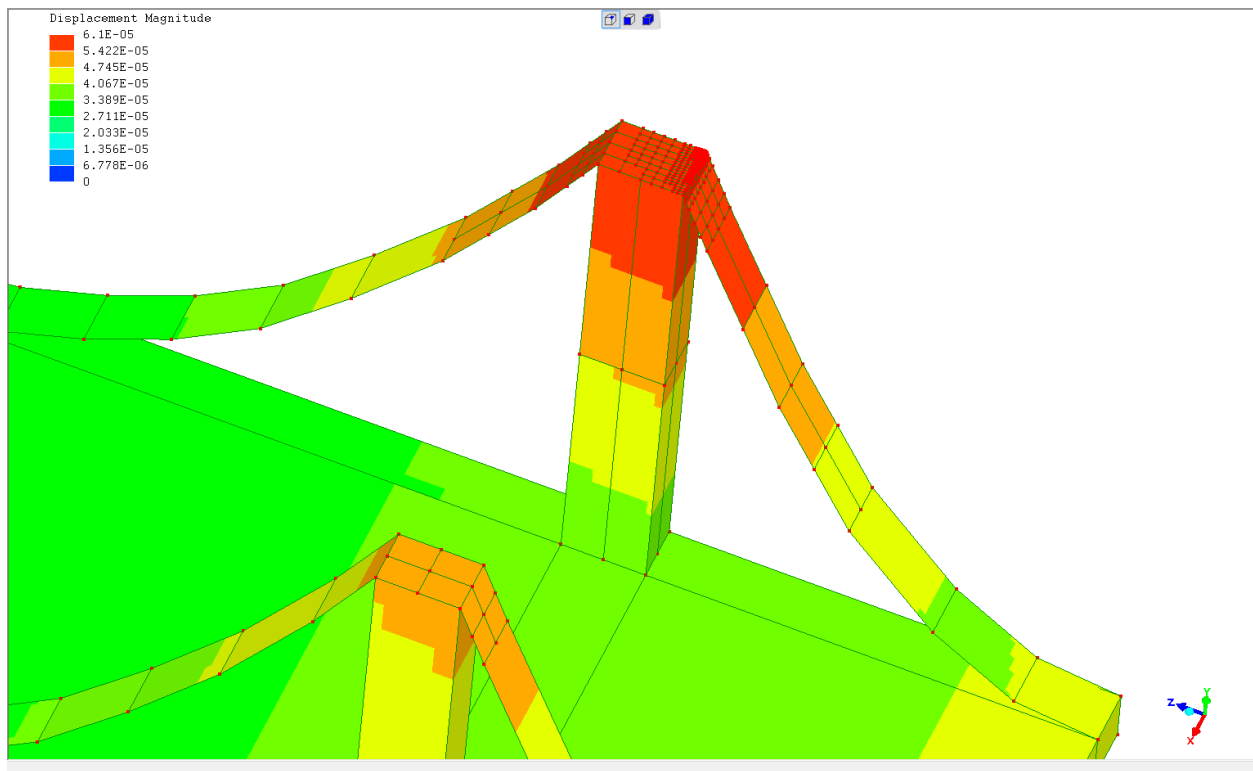


Figure 25: Iterative stress/ displacement refinement method used to focus meshing on the top 6% most displaced region of model

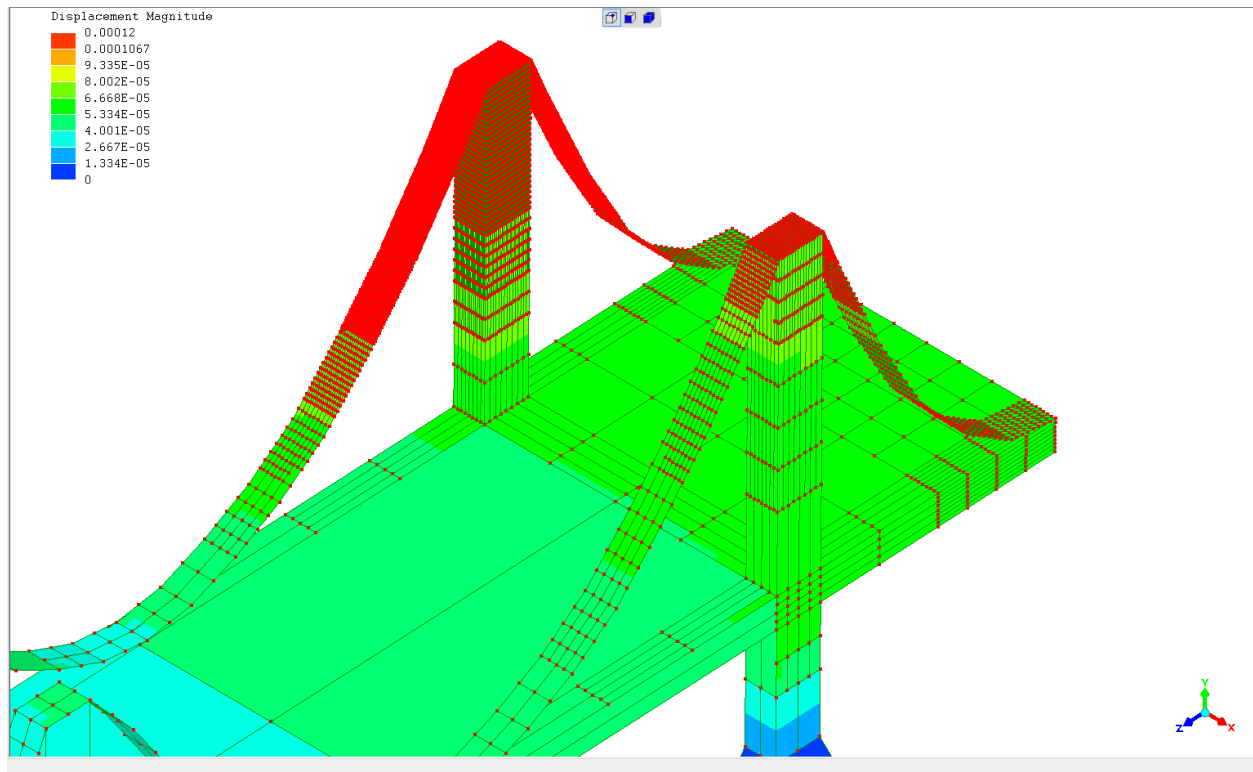


Figure 26: Iterative refinement of high displacement but with the remesh threshold specified as the average displacement across the whole model. A consequence of this is the gradient of refinement fidelity that can be seen corresponding to the importance of that part of the structure

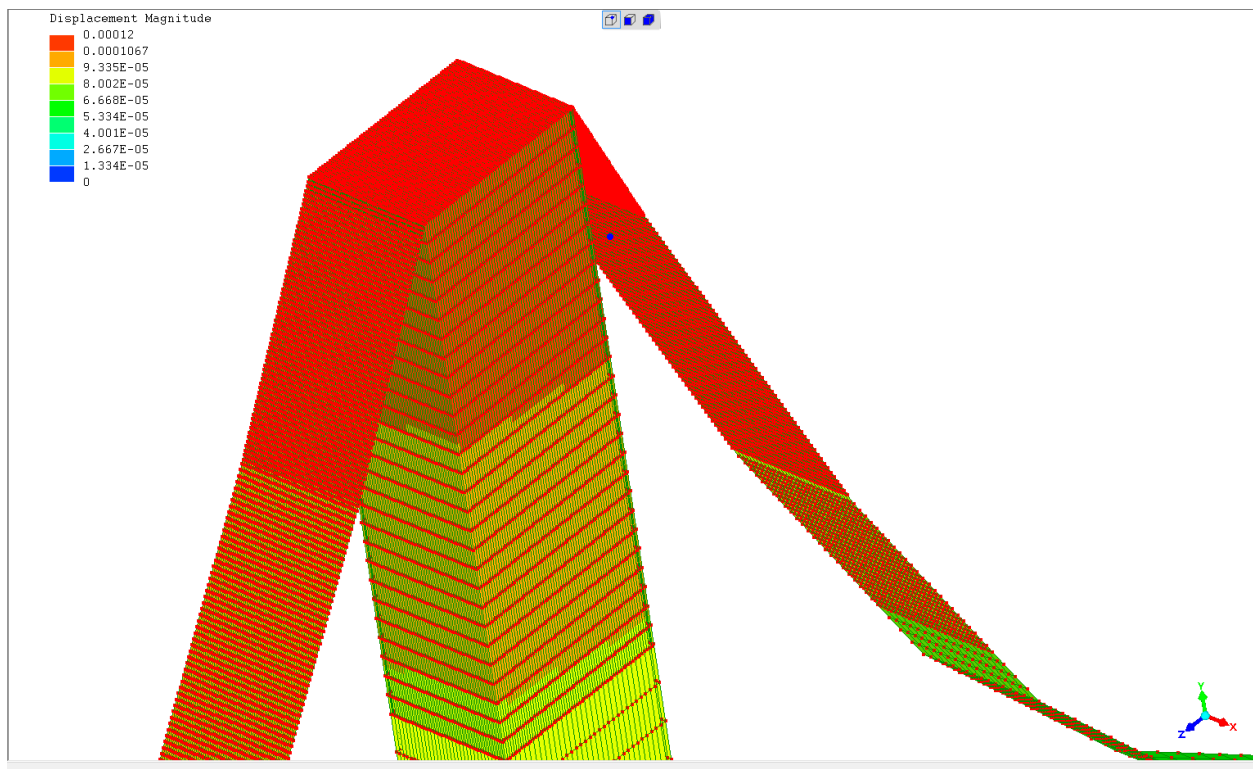


Figure 27: The metrics calculated by visual studio for the all classes in the final system

