Dissertation: Hybrid Methods for Finite Element Meshing

Submitted September 2015, in partial fulfilment of the conditions of the award of the degree

Computer Science

Jack Bradbrook

psyjb4

School of Computer Science
University Of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature:

Date:

I hereby declare that I have all necessary rights and consents to publicly distribute this dissertation via the University of Nottingham's e-dissertation archive.*

Contents

1 Introduction

The overall aim of this project is to perform the task of refining a Finite Element Mesh (FEM), (see Motivation and Background for description of FEM) using a stress based method as is typical of FEM refinement in addition to 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. As an engineer a typical iteration in the design process for mechanical components involves first designing the component within a CAD application before running a simulation on the design using knowledge of the conditions the is expected to perform under. Having defined the components geometry in continuous terms using CAD software a Finite Element System will discretize it into a representative mesh approximation before calculating values for each of the discrete elements when performing the simulation. This allows engineers to observe the effect the conditions have on the entire structure, see figure 1.

The success of the project will be determined by implementation of a system that is able to combine the stress and AI based refinement methods in order to refine a mesh to of a quality comparable to that of a successful stress based method.

This Dissertation starts with a review of the FEA process followed by looking at some of approaches people have taken to refine meshes automatically and gauge their quality before continuing to describe the design and discuss challenges of implementing the system. The system is then demonstrated as capable of correctly performing this task in the final evaluation section.

2 Motivation and Background

Over the past forty years FEA has emerged as a prominent technology for simulating complex real world engineering problems [14, 4]. 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 [4] [19]. The method can also be highly computationally expensive with the complexity typically increasing exponentially with the model size [4]. Analysis therefore proves to be highly time consuming and costly for individuals and organisations to conduct [6] [34].

2.1 Properties of Finite Element Models

Although this is a computer science as opposed to a mechanical engineering dissertation it is 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 has 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 [19].

Elements within FE models can have different types that define their shape. Different shaped elements are selected based on the type of structure that is being assessed and the simulation conditions, some typical examples would be a quadrilateral also known as a quad4 and a triangle (tri3). Appendix A shows the different element types supported by the FE solver used for this known as LISA.

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 [19]. 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. Stress is defined as force over a given area in an object and arises from some external force being applied. 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 [4][19].

In addition to the nodes and elements FE models also contain loads and constraints. Loads can be thought of as the phenomena from the outside world that is acting on the structure and consequently inducing some kind of physical effect on it. Loads are used by engineers to model the conditions 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 through the model there needs to be an area through which the stresses are assumed to leave. FEA is only able to calculate the the stresses through the model using the law of conservation of energy i.e. energy cannot be created or destroyed meaning that any energy entering the system as an input such as 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 [26].
- Poissons ratio - Amount of deformation that occurs perpendicular to the force that is applied to the material [11].

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. The system I have designed would also work with these more complex materials however the process of assessing the results could potentially be much harder without a more extensive engineering background and better knowledge of the specific materials.

2.2 Limitations and general considerations

An important consideration when conducting FEA is the trade-off of a model's accuracy against the time in which it can be solved. A major variable determining this trade-off is the model's 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 [1].

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 [9].

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 [34]. 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 [3]. It is therefore 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.

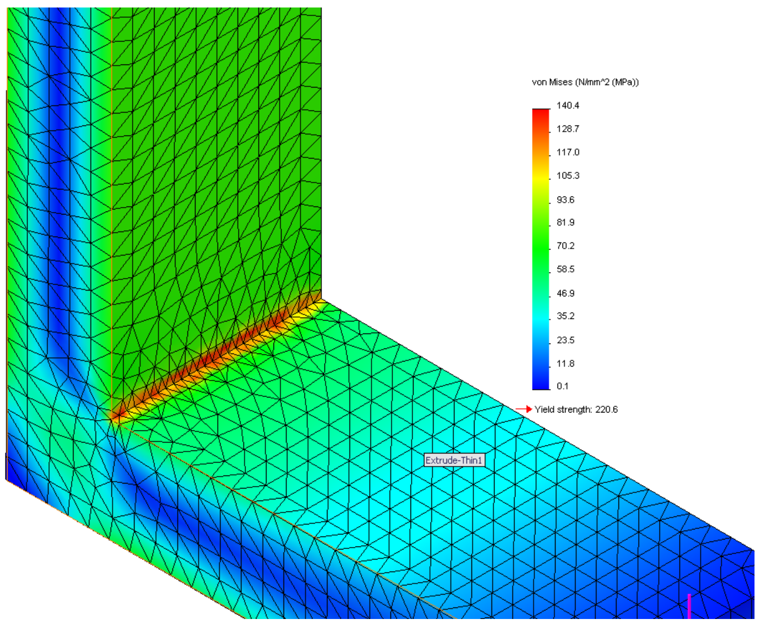


Figure 1: Mesh refinement in corner under high stress image source: ([31])

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 Subdivision Approaches

Multiple approaches exist for subdividing different types of meshing with the most common being hierarchical refinementent also known as h-refinement and relocation refinement or r-refinement[13] [15]

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 [13].

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 [15].

3.2 Stress Refinement

Refining a mesh based on some model parameter is consistent across all types of FE modelling. Execution of a model with a generic mesh is first required so as to obtain a set of results by which further refinement can be targeted. Using unique ids for nodes allows results from the previous iteration to be compared against the mesh and refinement to be focused on those nodes exhibiting a high amount of a specified property [30].

3.3 Uses of Artificial Intelligence and Machine Learning

Within the domains of AI and machine learning methods such as neural networks [12], case based reasoning [8] and inductive logic programming [4] have 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 [7] 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 resort to the use of training sets developed within academia [3]. 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 [2].

Having reviewed a variety of different AI based applications of 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" [18]. 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 [17]. 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 [17].

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

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" [16], Golem was applied by Dolsak to the problem of mesh refinement with a training set of just five meshes [4]. 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 [4] [5].

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 [4] [5]. 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 refined [3] [5].

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 is likely to improve the ease with which the two methods can be combined simultaneously in the latter stages of the project.

3.4 Quality metrics

Finally work has also been done on establishing valid metrics for assessing the quality of a mesh automatically [2, 12] Metrics for meshes have been researched far more extensively than 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 [2]. 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 [2].

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 [2]. 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 [19] a mesh containing a large number of distorted elements is therefore likely to yield inaccurate stress results when given to a FE solver such as LISA. 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.

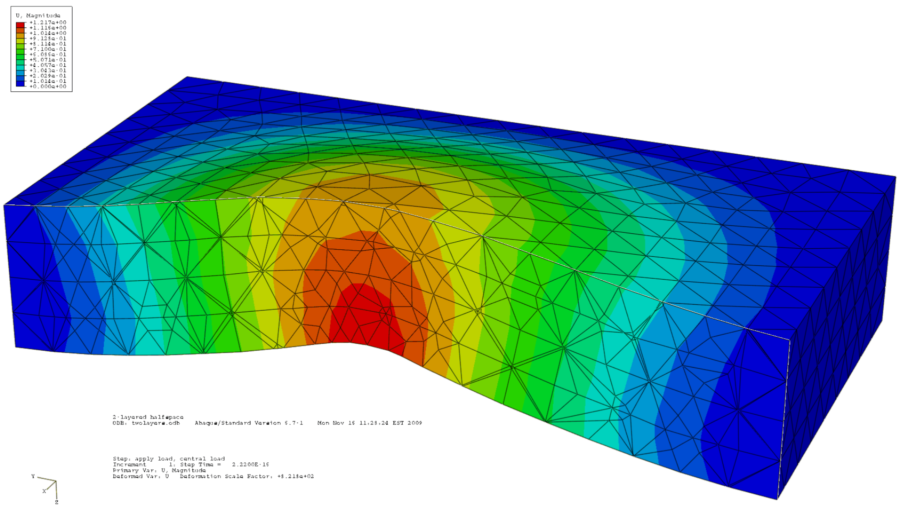


Figure 2: Example of how elements can be distorted in order to fit a geometry which will result in deterioration of gradient quality (image source: [20])

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 derived from the fields of AI or machine learning with a method relying purely on information already present in the model. The desirable end result will be a hybrid method of meshing which effectively prioritises those areas of importance whilst incurring a reduced computational cost.

The project can be broken down into thee main areas of research and implementation which have the following high level objectives:

A Research and implement both a traditional refinement procedure using data present within the model and an approach using techniques from AI developed and used by either industry or academia. These algorithms should be able to run independently on a set of example models.

B Secondly a process needs to be devised to combine the two re meshing methods to varying degrees. This will make it possible to evaluate and compare the effects of a hybrid meshing against each of the individual methods for a range of models. Through this it should be possible to establish whether or not there is potential benefit to using a hybrid approach and if so to what extent.

C The third objective will be to research and implement justifiable metrics for assessing the quality of a given mesh, this will allow objective comparisons to be made for the resulting meshes.

4.2 System specification

To demonstrate success in achieving the objectives of the project it is important to have traceability from the requirements through to the solution and lastly verification and validation. This section describes the systems initial *functional requirements* (what the system will do) and *non-functional* requirements (its constraints) based upon evaluation of the research 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 that 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 computed 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 element 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 by removing unnecessary state from the program. 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 [24] to generate a full set of developer documentation upon completion of the software implementation which will be included as an appendix. Ensuring that the majority of information is present within doc comments will also help to promote a reduction of loose comments within the code and hence function size.

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 encountered by engineers.

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 the finite element solver (see section 5.4 on LISA, the selected solver), 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 variable complexity and uncertainty surrounding the different parts of the project meant ensuring the architecture remained modular with well defined interfaces allowing components to easily be added or modified as the project progressed.

The system can be broken down into several main tasks that form the overall process:

Generate Initial Model: Using an initial input file containing a FEM build a class model which can easily be manipulated by the methods within the code.

Write Model To Solver: Write Model to a solver file that can be solved and call solver to obtain an output file for the model.

Read Solver Output: Read the stress data computed across the mesh from the output file back into the data model so it can be cross referenced with the current mesh

Apply Refinement Processes : Apply both the stress and alternative refinement process to improve the mesh such that on the subsequent iteration more stress is revealed.

Evaluate Refined Mesh : See how much stress was revealed in the last iteration by applying refinement and whether or not the quality of the elements improved.

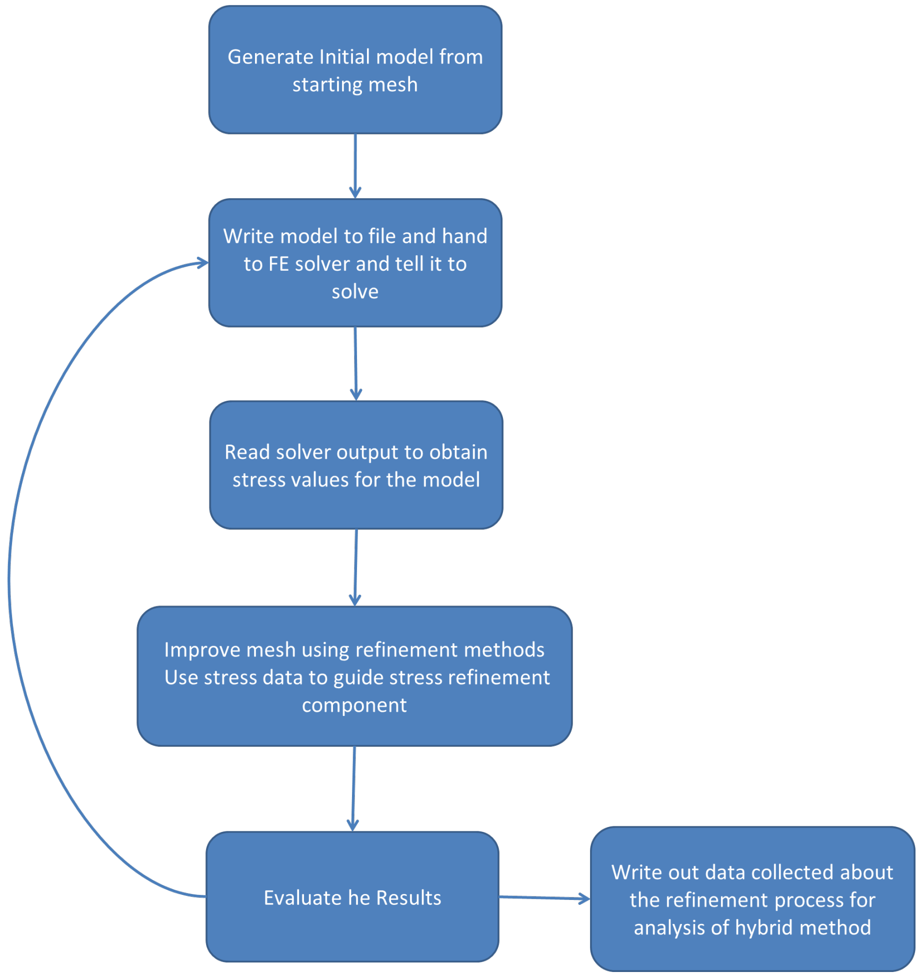


Figure 3: High level process of system

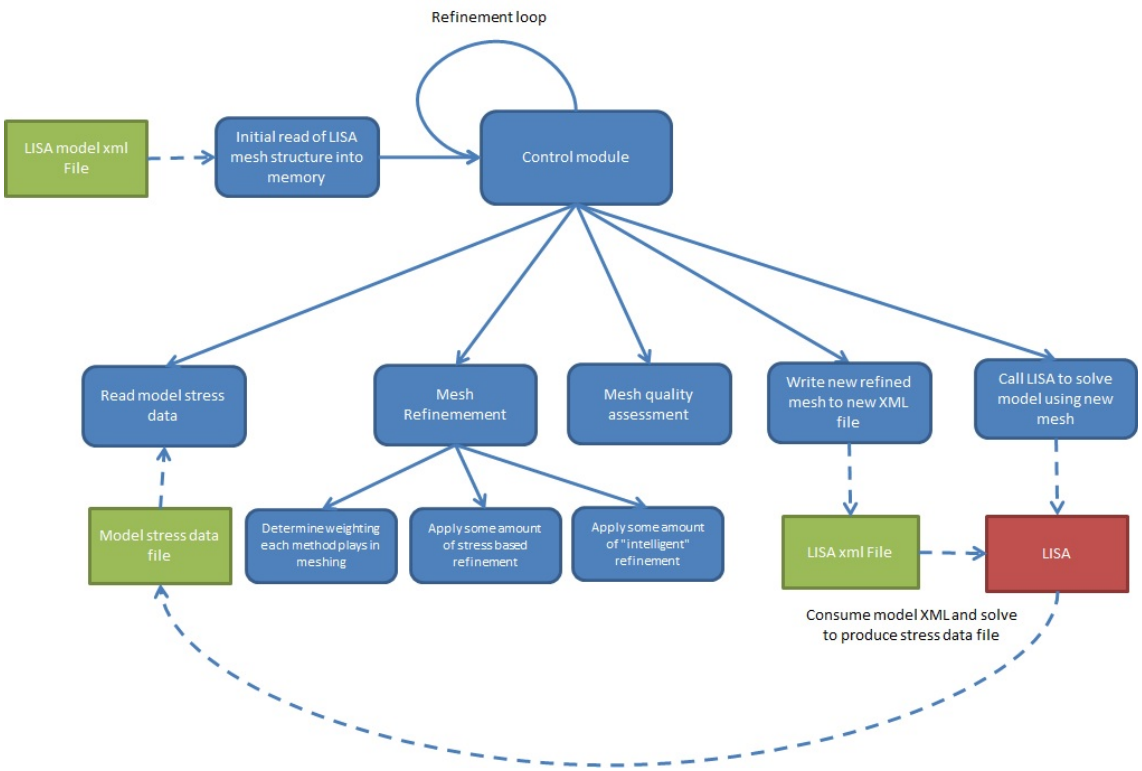


Figure 4: High level design of the system with its different modules

5.2 Modular Architecture

The modular architecture was crucial for allowing meshing algorithms and quality metrics to be replaced as necessary. At best the quality of the output could 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 failed. 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 different meshing approaches in order to generate a hybrid method.

Although the system was highly modular It was also still desirable to maintain an architecture hierarchy so that classes could be developed independently but easily integrated. 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 no fit especially well into any particular one. Examples of these are generic vector algebra operations such as dot product, matrix determinant and calculating surface normals.
At the highest level namespaces were used to break down the class groups appropriately, namespaces also naturally structured as folders within the Visual Studio (VS) solution explorer (see appendix D) which made navigating the project and finding components much easier as the system expanded in size.

5.3 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 [33]. 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.

5.4 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 is 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.

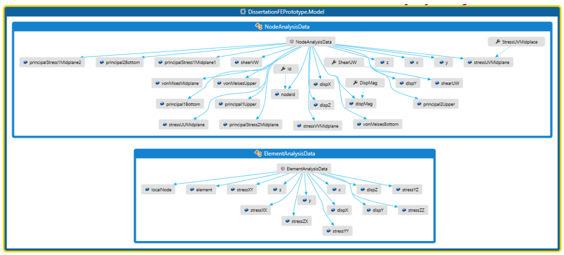
LISA also provides a GUI which allows visual inspection of the model and its mesh; this is particularly useful for observing the output of the meshing algorithms which can often provide a human with a much better understanding of how the method has performed and whether or not there are obvious bugs. 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 [25]. 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.

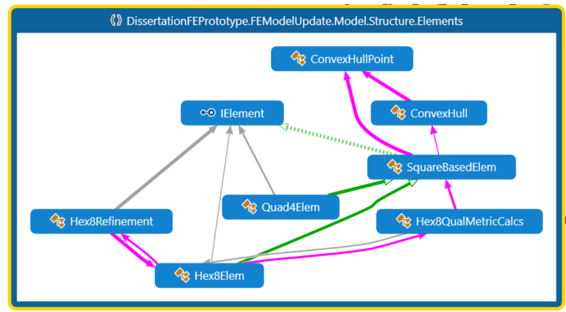
5.5 Simulation Data Model

Writing an API for LISA was the first 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.

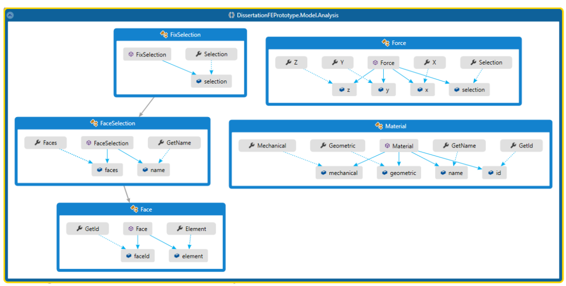
When the first re-meshing iteration occurs the system needs to read the input .liml file into an equivalent class model which closely resembles the files schema, diagrams for which can be seen in figure 4 below. Each class in this model contains corresponding data and methods used to represent and manipulate 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 it is required to 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.



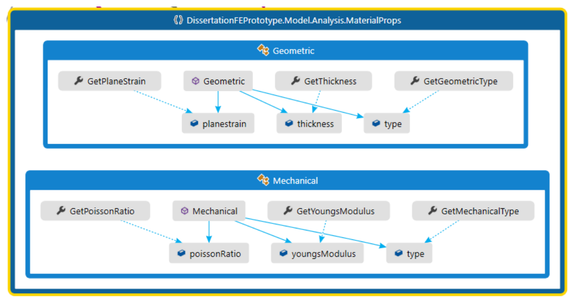
(a) Model classes



(b) Element Classes



(a) Model Analysis classes



(b) Material Property classes

Figure 6: Class model to represent .liml file structure used by LISA

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 request refinement of an element using its class. 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, see figure 6 below. This is powerful since computing metrics and performing subdivision for a 3D element is simply a reduction using the code for a 2D element but over every face comprising the 3D one.

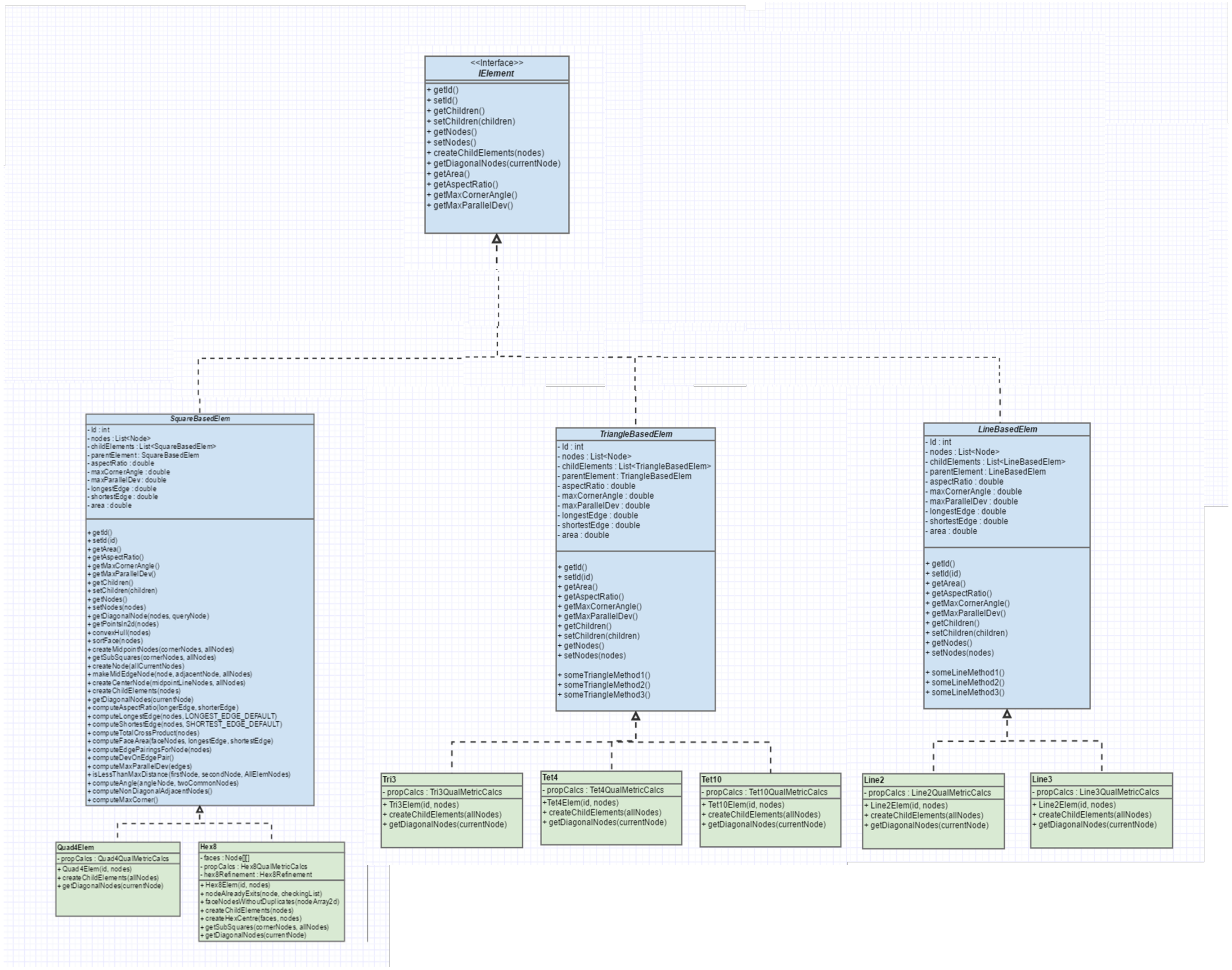


Figure 7: Class diagram showing the hierarchy of element classification within the data model, due to time limitations I was not able to implement the respective classes for triangle and line based elements, to see image representations of each element type within this class diagram refer to element type appendix

5.6 Remeshing Methods Approach

When developing multiple meshing processes it was advantageous to break down and separate aspects of system functionality so that the system would be able to successfully incorporate and new meshing procedures that may be added later. An FE mesh refinement system can be thought of as performing two distinct but not clearly separated tasks:

Element Refinement: How are elements going to be refined when it is known where is bes to refine within the mesh.

Meshing Strategy: Which parts of the mesh are going to be refined?

Both h and r refinement fall under the first task and can be thought of as simply taking an argument from a higher level process about where they should mesh.

By contrast stress refinement and the heuristic method that is also being used can be thought of as strategies. The goal of a strategy in this scenario is to maximise meshing in those areas where stress is likely to be high in advance. Having computed the stress this also acts as the means by which to assess the quality of the strategy, what we're attempting to assess with the system. Strategies are also much more general than subdivision processes and so it does not make sense to couple them to the subdivision functionality in any way. There is no reason that the same strategy shouldn't be used for a variety of meshes constructed out of different element types. The the change in element shapes will however have an affect on how it is able to be subdivided. As a result the same subdivision code cannot be used to divide both a quadrilateral and a triangle, see appendix A for element types.

The Design solution to this aspect of finite element refinement was to abstract each of these concepts such that both strategies and element subdivision processes could be interchanged without any issues arising. Since subdivision is associated with specific elements the code for performing this task was encapsulated within a specific element class, from the Strategies perspective however it was important to be able to ask any type of element to subdivide itself. This was achieved through the design of an interface referred to in section 5.5 above as the IElement interface. Each type of element must implement this interface which exposes all the methods required for the controlling strategy to be able to refine the model in any area it has chosen. Using the interface the strategy can refine all elements it has selected as beneficial for refinement by simply calling the elements createChildElements() method through the interface. This not only allowed a strategy to refine any type of element but it also meant any strategy could refine any type of element.

Selecting a Subdivision Approach: Having reviewed both h-refinement [13] and r-refinement [15] as techniques for performing element subdivision it was concluded h-refinement was preferable due to its simplicity and widespread use despite typically being more computationally expensive than r-refinement [13, 15]. Another advantage of selecting h-refinement is how contained the approach is by only needing information about one particular element in order to refine it. By contrast r-refinement also needs to know about the state of the entire mesh which also contains the sizes and types of all the other elements, this at best complicates the simple relationship of strategies simply having to delegate the refinement task to specific elements in those regions of interest. Finally it is also not clear whether when implemented as part of a hybrid approach whether r-refinement would loop over and repeatedly swap the same elements when refining under two strategies simultaneously. This seems likely since each strategy asserts its own priorities for the method and will need to take elements from wherever else in the model has lots of elements in order to further refine that area.

Having decided to adopt h-refinement it was important to consider how new elements would be created and re integrated back into mesh structure. Subdividing elements recursively naturally forms a tree structure with an element creating additional smaller elements of the same type inside itself. The element type also largely determines the branching factor of the tree since most shapes naturally divide evenly into a specific number of smaller instances of themselves such that the shape of the original element is preserved. For example dividing a quadrilateral into four quarters results in each of the sub elements retaining the same aspect ratio as its parent. Alternatively dividing the same quadrilateral into two half's results in an aspect ratio twice as big, This is bad since the accuracy of the results produced by the solver for a mesh is highly dependent upon properties of each elements shape, see section 3.4 [2]. Once new elements have been created by the parent these new elements need to be registered within the main model. Initially the only element the model can see directly in its element list is the parent. This process involves updating the model reference such that the leaf nodes are added and the parent removed. To find the leafs a simple depth first search is performed and those elements with the structure which do not contain any children are added to the list. The final stage of the process is to assign unique id values to the elements so they can be referenced by LISA. Having constructed the list the first element is assigned the id of the parent and the parent destroyed, the following elements in the list are assigned the next available id values within the model. The subdivision process can be repeated arbitrarily many times before flattening the tree depending on the level of refinement desired. An example of the tree and the eventual flattened list can be seen below in figure 8.

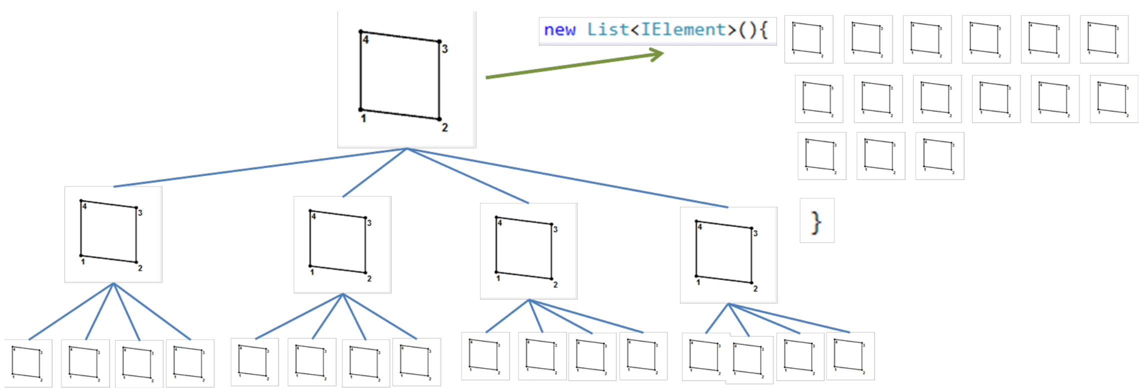


Figure 8: Process of flattening a refined element tree into a single list which can be handed back to LISA for processing

Stress and Heuristic Refinement: Deciding on the second strategy by which to perform mesh refinement was perhaps the most significant design decision for the entire project. Having assessed a range of options, see section 3.3 a clear realisation was that given the inherent complexity of the problem developing a sophisticated AI method within the available timescales was going to be too complex. Dolsaks expert system [3, 4], again reviewed in section 3.3 seemed most promising. As also outlined in section 3.3 unlike some of the other approaches this one offers a straightforward implementation of an expert system with clearly defined concepts rules that directly utilise the data already available within the LISA models. It's also important to note that the system was also generated automatically using a machine learning technique and has produced results that support the claims made of its capability, [3, 4, 5]. Success of this method as the second strategy within a hybrid would indicate the potential strength of AI as part of a hybrid approach to finite element meshing.

From a project delivery standpoint this also meant that a significant amount of time did not need to be invested into the development of the underlying machine learning approach for generating the rule set with focus instead directed towards integrating this and the various other components in the project.

5.7 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 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 a starting point.
- A JSON file containing important edges and associated meta data as identified by an engineer looking at the model.

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

5.8 Combining methods

Since each refinement method performed a discrete amount of subdivision every time it was called it made sense when developing a hybrid approach based on the two methods to define each potential hybrid as some weighted combination of the two methods. The simplest way to represent this appeared to be a two valued tuple containing the number of times each method should be applied for each iteration. One way to do this automatically would be to iterate through combinations of two integers up to some value k:

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

Using this weighting system each application of a method can be thought of as adding a depth of one to element refinement trees affected by the rule as seen in figure 7. Testing the hybrid weightings as different specifications meant it was also possible to improve the systems throughput by conducting evaluations simultaneously on different threads. When started each thread creates its own directory which it copies the three input files to and runs for its designating weighting configuration.

Another key consideration when comparing the different meshing approaches was to establish what the value of weighting unit and thus allowing comparisons and evaluation for each of the hybrids as a weighting specification. Balance of the weightings was achieved at the start of the evaluation process through observing the increase in element count for the different heuristic methods when run individually as can be seen in Figure 10. With the average increase in element count per iteration being calculated as 6% of the model total the stress refinement threshold could then be configured so as only to mesh those elements within the 94th percentile within the model in terms of stress.

6 Software Implementation

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

6.1 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.

6.2 Implementation Methodology

The 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 the time they were expected to take along with their criticality within the project. In a busy week requiring lots of work for other modules the tasks with lower time costs and higher criticality were typically selected over the others.

6.3 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.

6.4 Heuristic/Rule Based Refinement

Each rule is represented as a function within the implementation, this closely resembles the format presented by Dolsak [3, 4, 5, 6]. 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 posses the same form - The Edges share the same edge type as one another
- Edges are considered the same - Both edges must be almost the same length, opposite one another and posses the same form.

Each edge specification has several properties which Dolsak describes within his papers including:

- Id number - used to identify edge uniquely within the RuleManager
- Edge type - How would the engineer describe the edge, does it form a circuit, it an important aspect of the models design or is it along the edge of some hole?
- Load type - Is the edge between to areas with forces applied to them, is just one side of it or is it located elsewhere within the model.
- Boundary type - Does the edge run along a constraint point where the model is attached to the outside world.

Each of the three edge type properties have a set of recognised values defined by Dolsak within his paper which have been listed in appendix C. In addition to these properties used by the refinement system for deciding where to mesh each element also contains a list of nodes referencing those within the model which describe the path the edge takes along the mesh.

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 [5]

Having completed meshing of targeted areas it was also important for the heuristic refinement process to redefine the specified edges so that refinement along an edge could be performed again on a subsequent iteration taking into account the newly created nodes and their associated elements created along the original edge. The solution to this problem was yet another node traversal starting at the origin of an edge and moving through each of the node points defined along the edge while collecting newly created nodes lying between the two.

Since the only information available by which to determine the new edge nodes is the node path comprising the current edge it has to be assumed that new nodes linking the each current node pair will take a direct path between the two. Since there is no graph structure stored within memory that represents all the possible traversal moves and with the number of possible paths increasing exponentially with the number of nearby nodes considered it was not practical to generate it. Consequently he only way to effectively move towards the node end point consistently was to perform a greedy search.

The greedy search starts by computing the distance between the start and end node points for all nearby nodes, determine the nodes directly adjacent to the start and then of those nodes link the one which has the shortest distance to the end node to the path before repeating this process for that node. This approach was not the first used in an attempt to solve the problem however having completed its implementation it proved to work effectively.

A more detailed pseudo code description of this process can be seen below:

```
forall E ∈ refinedEdges do
    Get the list of potentially new nodes K as the set of all nodes previously created by refining
     elements that run along E;

    Create a new empty list containing the new edge path used for subsequent iterations EL;

    forall edgeNodePairsN ∈ E do
        Make a new sub list SL which will be the sorted path between the end of each node pairing in
         the current path N.FirstNode and N.SecondNode;
        Add N.FirstNode to S;

        Make two lists A and B which will contain tuples holding the distance from each potentially
         new node to N.FirstNode in A and n2.SecondNode in B along with a reference to the
         potential new path node;

        forall nearbyElemNode ∈ K do
            Compute euclidean distance from n1 and store with reference in A;
            Compute euclidean distance from n2 and store with reference in B;

            Sort A ascending;
            Sort B ascending;

            Make a list F of first n nodes from A i.e. the closest ones depending on the element type,
             more complex elements take higher number of nodes, in the case of quad4 take 4;

            forall n2Node ∈ n2 do
                if n2Node.Ref ∈ F then
                    Add F to S;
                    Set new n1 value as F;
                    Break;
                end
            end
        end
        Add N.SecondNode to S to complete that section of the path;
        Add SL to new edge path EL;
    end
end
```

Algorithm 0: Greedy search of shortest node path in 3D space to form new edge path after each iteration

6.5 Mesh Quality Assessment

Dittmers rules for computing the quality of both individual elements and the entire mesh are built into their own “MeshQualtyAssessments" 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 element to also be calculated upon its initialisation and thus removing the risk of null values being returned when other parts of the system request this information.

Coding the individual methods did not take too long since Dittmer provided a clear description for each metric calculation method many of the values needed to perform the calculations were also conveniently stored as properties within different parts of the model. A considerable part of calculating each metric was therefore the process of aggregating all of the necessary data from the model so that the individual values could be calculated.

Although the element shape metrics provided an indication of how much an elements shape deviated from its ideal understanding the differences between exactly what each metric implied was challenging, specifications for the metrics did not provide information on this besides describing cases of a good value and bad value. As a consequence the metric I decided to focus my evaluation primarily on out of the ones described was the maximum corner angles which provided a clear indication of element skew. This metric was also useful in attempting to identify elements which cause problems when needing to be sorted, see 6.6.2.

6.6 Implementation Challenges

Implementation of the system was not without its share of challenges, some of which required fundamentally re addressing the approach used to tackle the problem. This section outlines the main instances of such cases during the projects development where as a consequence a notable change to the implementation was made often requiring additional research.

6.6.1 Fast Node Lookup and Update

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 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 have been created by the adjacent element, this can be seen below in figure 9.

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.

6.6.2 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

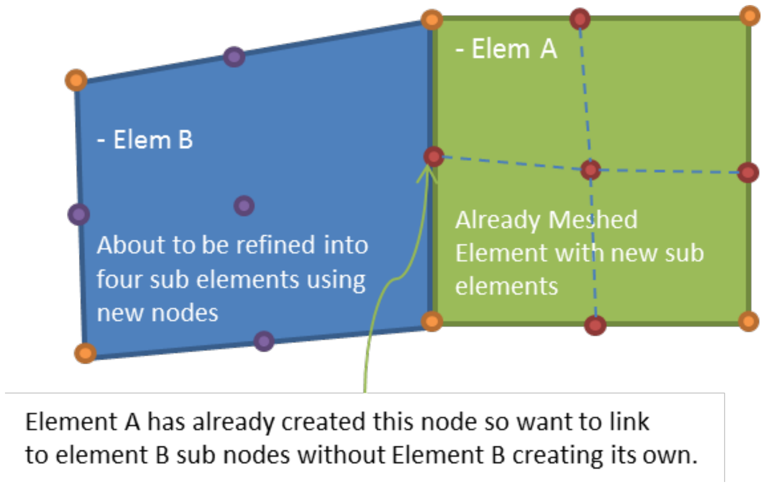


Figure 10: The need for an element to check for existing adjacent nodes when subdividing itself during refinement,

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

to have them form a perimeter around the edge of an element in 3d space without edges crossing one another internal to the element.

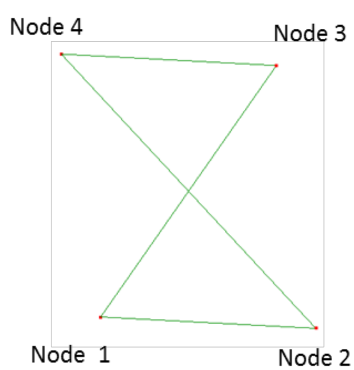


Figure 11: Element with 3d skew resulting in edges between diagonals being shortest by a small amount.

When addressing this problem for simple models constructed from Quad4 elements the 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:

make a list to contain sorted nodes called SN **while** $\exists node \in unstortedNodes$ **do**

Get distance between current node and the next two nodes in unstortedNodes;

if $sortedNodes.Count == 1$ **then**

Calculate the distance between the origin node and the penultimately unsorted node A Calculate the distance between the origin node and the last unsorted node B

if $lengthA > lengthB$ **then**

currentNode = A;

Add A to SN;

Remove A from unsortedNodes;

else

currentNode B;

Add B to SN;

UnstortedNodes.Remove(B);

end

else

Go through all unsorted nodes, compute distance to each, assign the node with the shortest distance as C.

currentNode = C;

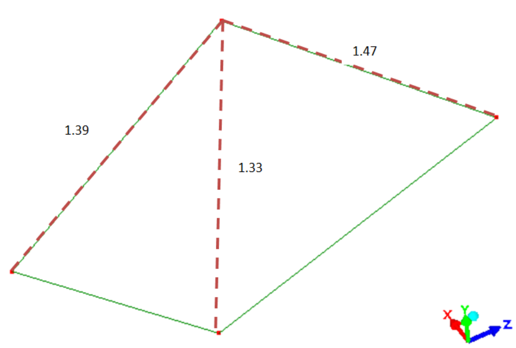
Add c to SN Remove C from unsortedNodes;

end

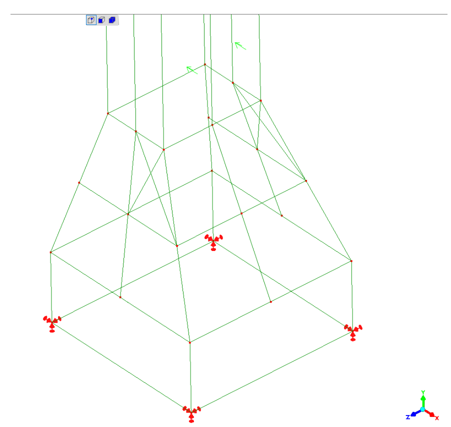
end

Algorithm 1: A basic traversal approach for sorting Quad4 elements, code for when one node has already been sorted used to ensure that in nearly all cases the diagonal from the origin is selected as the third node in the sequence.

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 an internal diagonal to be shorter than both of the edge sides as seen in Figure 9 below, this broke the traversal process which relied upon there being at least one side edge that was shorter than the diagonal. This proved to be a significant flaw in the approach and brought about the realisation that a reliable strategy for solving this problem would not be able to depend simply upon varying properties of the different elements9.



(a) Quad4 Element with diagonal shorter than both external edges

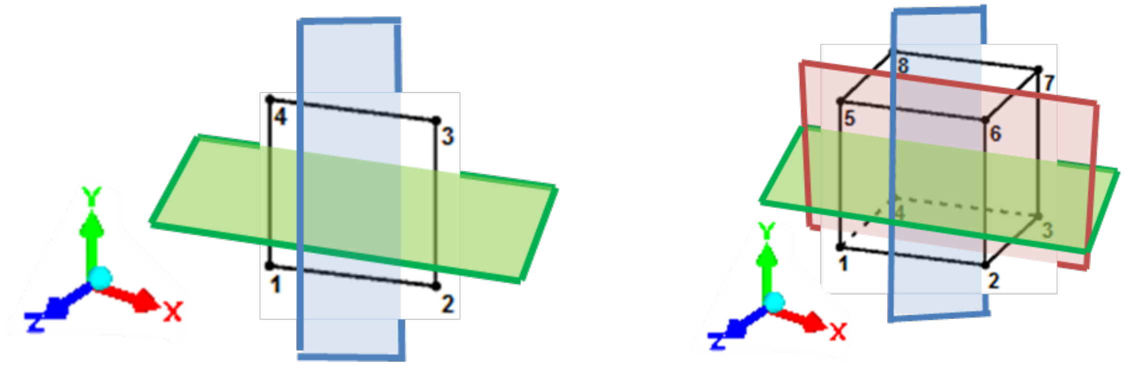


(b) Skew in bridge model elements resulting in rejection of the model by LISA

Figure 12: 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.



(a) Dividing a Quad4 along planes to establish each node as a corner point

(b) Dividing a Hex8 along planes to establish each node as a corner point

Figure 13: 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 it did not offer a strong general case solution to the problem with the code for a Hex8 element needing to be significantly different and more complex than that of a Quad4 and with the potential for the most complex FE element types such as wedge15, hex20, and pyr13 requiring implementations with even greater number of plane divisions and groupings in order to successfully identify every node.

Having already devised two solutions it seemed likely that there would be some body of research surrounding the problem worth investigating. with research leading to a set of possible alternaives known as convex hull algorithms. As the name suggests the goal of a convex hull algorithm is to generate a convex hull, convex hulls have several definitions but the simplest of these as described by [22] is for a set of points in some space a subset S of those points is convex if for any two points P and Q inside S the line between the two should also be inside S. This is directly applicable in the case of quad4 elements where the LISA sort order is the convex hull of the points, in the case of more complex elements the algorithm can be applied repeatedly to different faced divided though plane splitting before sorting the nodes at the end with knowledge of node ordering within each individual face.

After reviewing the different convex hull algorithms including Grapham scan [29] $O(n\ log\ n)$ and brute force scan $O(n^4)$ [22] [28] I decided to trial the following C# implementation of the Monotone Chain algorithm, also $O(n\ log\ n)$ [21].

The Monotone Chain algorithm algorithm was developed shortly after Graham scan and builds upon the concepts introduced in the former. Graham's scan works by initially finding the point in the data set with

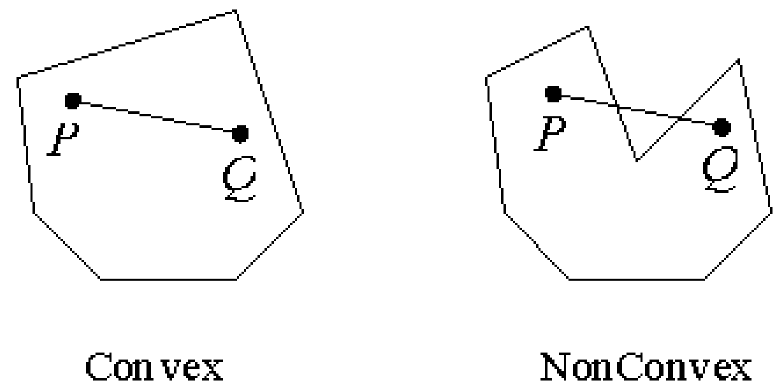


Figure 14: Illustration of convex hull definition, imace source: [22]

the lowest y coordinate which can be called P. Having found this point the other points in the set are sorted based on the angle created between them and P. Combining both these steps gives a complexity of $O(n\ log\ n)$, with $O(n)$ to find P and $O(n\ log\ n)$ to perform a general sort of the angles. Moving through each point in the sorted array Graham scan determines whether moving to this point results in making a right or left hand turn based on the two previous points. If a right turn is made then the second do last point has caused a concave shape which has violated the requirement of the convex hull path. In this scenario is repeated the algorithm excludes the point from the convex set and resumes with the previous two points being those on the path before the rejected point. A stack structure is therefore typically used to keep track of the point ordering as is the case with the Monotone Chain implementation within the system [23].

The Monotone Chain algorithm performs essentially the same procedure however instead of sorting using simply y values Monotone Chain sorts using both x and y values. This allows the algorithm to sort the points in two separate groups which form the top and the bottom of the hull and a reduction in the complexity of the sort comparison function.

```
Sort the points of P by x-coordinate (in case of a tie, sort by y-coordinate);
Make two empty lists I and L Lists hold vertices of upper and lower hull;
while i = 1; i < n; i++ do
  | while L contains at least two points and the sequence of last two points of L and the point P[i]
  |   does not make a counter clockwise turn do
  |   | Remove the last Point from L;
  | end
end
while i = n; i > 1; i- - do
  | while L contains at least two points and the sequence of last two points of L and the point P[i]
  |   does not make a counter clockwise turn do
  |   | Remove the last Point from U;
  | end
end
Remove the last point of each list (it's the same as the first point of the other list).;
Concatenate L and U to obtain the convex hull of P.;
Points in the result will be listed in counter-clockwise order.;
```

Algorithm 2: Monotone Chain algorithm for generating convex hull, pseudocode description credit: [32]

The additional complexity of implementing a 3D convex hull algorithm meant it was much easier to experiment the with the approach as a potential solution to the problem using a 2D implementation by simply reducing the problem to a 2D equivalent. This was for quad4 elements by calculating the maximum delta between the max and min 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. The only instances in which this approach failed were where highly elements would lie on a perfectly diagonal plane resulting in two axis of elimination using this method. This problem was avoid however by using the basic traversal to sort these elements.

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 [2].

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. Having run the system on a range of different models results clearly demonstrate the system's ability to evaluate the quality of meshes using multiple refinement processes based on the stresses induced by the user and their categorisation of edges within the model.

7.2 Validation Against Non Functional Requirements

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 system's 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 system's design and documentation evidence is present, (see appendices B, F and G) to indicate that this adheres to the requirements specified with the project submission containing detailed documentation in the form of a Deoxygen guide and 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 system's effectiveness however without verification of individual components it would not have been possible to assert the accuracy of the results produced. Unit testing was also conducted from within Visual Studio using the NUnit framework and structured as a separate VS project. This guaranteed 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 implemented solution. Tests were also 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. Appendix B shows the visual studio test explorer containing the various tests.

7.4 Software Quality and Management

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 [27] 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 maintainabilty.

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

7.5 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 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, for example see appendix.

7.6 Evaluation Of System For Model Simulations

In order to perform reasonable evaluation of the different methods it was important to carefully design tests for the system which would fairly evaluate its ability to generate a range of hybrid methods capable of performing meshing.

Firstly it was important to test the various methods individually for at least one model in order to verify that each of the methods individually performs as expected, this step does not produce particularly interesting results although is an key step in order to have trust in the results subsequently produced by the hybrids. When evaluating the hybrids the system also needed to be evaluated for several different FE models with varying simulation conditions. This demonstrates consistency in the results and de the systems ability to work for a range of different model inputs.

Since the system also builds hybrids from different weightings of the stress and heuristic refinement methods it was also important to ensure that when the systems performance could be partially attributed to the heuristic component the range of possible user inputs for the edge specifications was taken into account.

Three models were created in total for evaluation, with each of the models being a general simplification of some more complex model that could be expected within an industrial engineering setting. Each model has a manually constructed low fidelity mesh built using LISA's graphical user interface. The models also have a set of forces applied to them which are required so that stress is induced within the simulation. Constraints are also assigned to surfaces as described under section 2.1.

Different Quality Edge Specifications: Deliberate variation of inputs for the heuristic component was also important for making a general assessment of the heuristic method, see 6.4 for heuristic edge specifications. Since the users specify the edges that determine the meshing focus they directly affect the final result of the process, it is therefore important to consider the results produced by the system for a variety of different potential users. The effects of good and bad edge specifications can be observed both in execution times for the simulations and in the system's ability to mesh accurately where required as seen in figure 14 and the mesh structure in appendix F.

Since it was not possible to objectively compare the edge specifications for different models that were evaluated a basic criteria was developed so that comparisons could be drawn. For each model four sets of of edges were consequently constructed and given classifications of “Best”, “Good”, “Ok” and “Poor” With the following as general guidelines for defining each set:

Best: Approximately five edges specified directly over or adjacent to those areas of known high stress within the model - input potentially generated by a user with a high degree of expertise in evaluating the specific type of structure.

Good: Approximately three edges over or close to areas of high stress within the model - input potentially generated by a user with a high degree of general FE experience although potentially not specific to that type of structure.

Ok: Three to five edges some near high stress and other not - representing a user with some experience but by no means an expert.

Poor: Three to five edges none of which are close to areas of high stress - representing input as would be generated by an inexperienced user new to FE stress analysis.

As someone who would identify as an inexperienced user determining the areas of high stress in advance was important in order to successfully develop rule sets which satisfied each of these categories for the different simulations. The system was therefore initially run for each model without a heuristic component in order to provide some indication of what edges could be used for each set. Having designed four edge sets for each of the three evaluation models data could then be collected on the performance of both the individual heuristic and the combined hybrid method for each of these methods based on different levels of user expertise.

A clear drawback of not having the edge sets defined by an experienced FE engineer is a lack of a guarantee that the edge sets chosen by me accurately represent those similar to what would be specified by a typical user of the system. At best it can be asserted that the sets represent a range of deviations from the theoretical ideal for the edge specifications which it is reasonable to expect from users with varying degrees of skill. Consequently it is only possible to objectively judge each of the different sets on the basis of the results they produce. These results for the different specifications cannot be claimed to represent the output mesh of a user group.

7.6.1 Metrics Selections

Due to the complex nature of finite element models there are a huge number of potential methods that can be calculated from the available stress data and mesh. selecting appropriate metrics by which to draw conclusions from the different models was in itself challenging. Below I have described the metrics I eventually selected along with a description of what they indicate and why this was important to conclude the systems ability to meet its objectives.

For subsequent evaluation of hybrids the following metrics are then
Average Maximum Internal Corner Angle: This metric was cited by Dittmer [2]as one of the most consistent indicators by which to evaluate the quality of a mesh with gradual deviation from the optimum indicating a degradation in quality and the meshing processes inability to maintain quality and consequently insure accuracy of subsequent results.

Execution Times: Since it is important for all methods to run in a reasonable amount of time, measuring the increase in runtime with additional iterations provided a good indication of how costly each approach became and whether there were any points at which meshing became significantly more expensive.

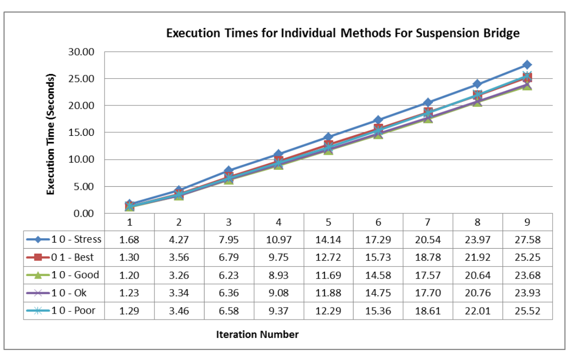
Average Stress Revealed for Each Iteration: In order to measure the different methods ability to reveal stress over time the average stress underneath an elements across the model was an effective metric. with increase in the average occurring as a result of more elements being placed on those areas of high stress and decrease occurring with creation of elements over areas of lower stress. Since this metric is a measurement of stress which is force over a given area stress is essentially a measure of pressure and as such the unit of measurement used is pascals.

7.6.2 Evaluation of Bridge Structure

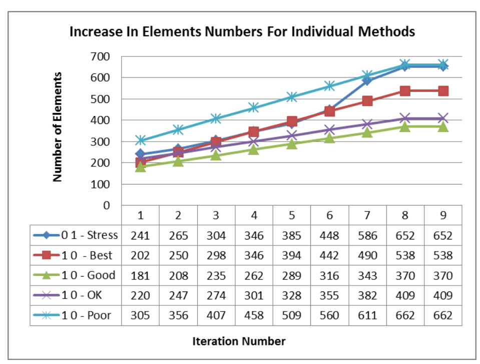
The suspension bridge model consists of 196 elements of quad4 type and 212 nodes which can be considered coarse given the size of the structure. Four constraint points were specified at the base of each supporting column and strong forces of applied across the structure along the negative x axis.

The first step taken before evaluating the hybrids was to find
The results in figure 14 below show the increase in time taken to complete execution and the number of elements generated having reached each iteration, the growth in both of these are linear which indicates that no individual method is doing significantly more meshing than another. This shows both the methods are closely balanced the importance of which is described in section 5.8. Looking at b it can also be seen that the quality of individual heuristics does not necessarily result in more meshing with it being possible to mesh more in undesirable areas given poor user input, a better edge specification therefore does not simply have an advantage over a poorer one by being able to conduct more meshing.

These tests indicate that subsequent evaluations for hybrid methods are conducted fairly with each weightings representing equatable in the number of elements created and consequently the execution time.



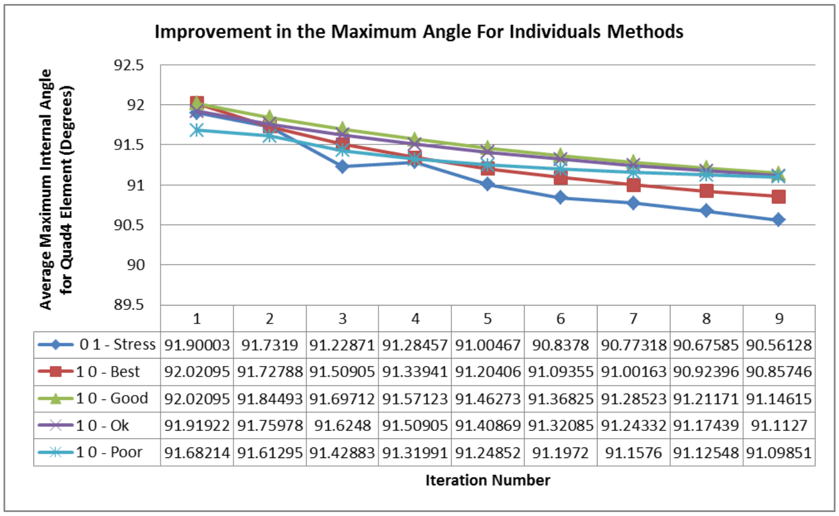
(a) Time taken to complete each iteration using the different methods



(b) Increase in number of elements for each of the individual refinement methods

Figure 15: Execution time increase compared to the amount of information revealed for the different approaches

The maximum internal corner angles can be seen as improving fairly linearly over time although with a the greatest rate of improvement occurring during the first few iterations for each method before the average for the mesh approaches the optimum, which for elements of type quad4 is 90 degrees. This means in general the refinement methods reduce skew present within the model through the creation of new elements and the calculated stress retaining its accuracy.



	1	2	3	4	5	6	7	8	9
0 1 - Stress	91.90003	91.7319	91.22871	91.28457	91.00467	90.8378	90.77318	90.67585	90.56128
1 0 - Best	92.02095	91.72788	91.50905	91.33941	91.20406	91.09355	91.00163	90.92396	90.85746
1 0 - Good	92.02095	91.84493	91.69712	91.57123	91.46273	91.36825	91.28523	91.21171	91.14615
1 0 - Ok	91.91922	91.75978	91.6248	91.50905	91.40869	91.32085	91.24332	91.17439	91.1127
1 0 - Poor	91.68214	91.61295	91.42883	91.31991	91.24852	91.1972	91.1576	91.12548	91.09851

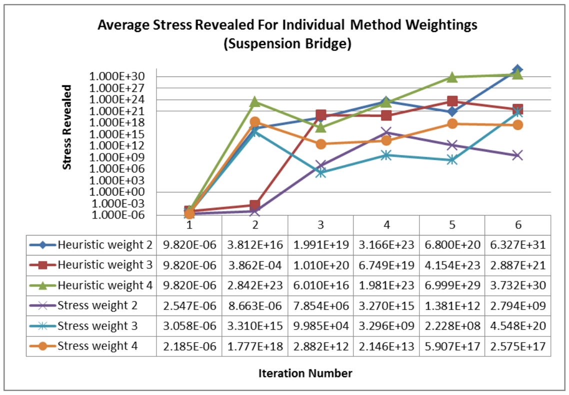
Figure 16: Approaching the ideal quad4 geometry for simulation data accuracy of 90 degrees using refinement with each of the different methods

Graphing this for multiple iterations gave an accurate representation of each methods effectiveness with the average being reduced if a method poorly selects an area under which to refine and increased if meshing occurs exclusively in areas of concentrated stress.

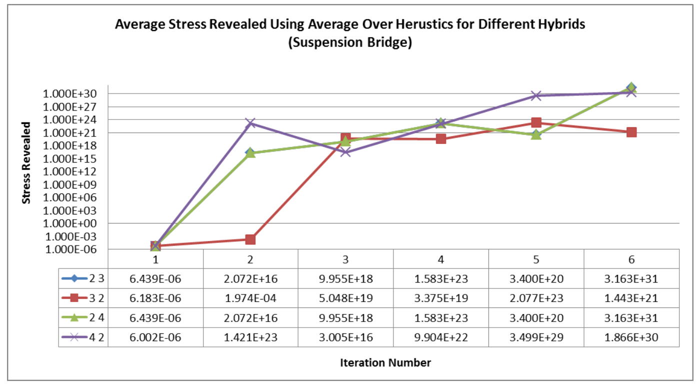
Evaluating each of the individual methods using this metric with various weightings produced the following results:

Need to re run one of the sets to re produce the graph because I got the input settings wrong for it

Having completed analysis for each of the individual methods it was reasonable to run some hybrid strategies for each of the models, successful execution of the models produced results that indicate rapid overall improvement with regards to finding stress as can be seen below in figure 17. Rapid improvement can be seen during the first few iterations in figure 17a before achieving a plateau was intitially highly surprising to the extent that for a period it seemed like the results must be incorrect. Re execution of the model with varying configurations including reduced force and alternative constraints resulted in minimal difference however. Conducting additional research then revealed that this is not in fact an uncommon property of stress gradients within FE models as stress will trend towards infinity at points of serious weakness within a design when very high force is exerted, see appendix I (figure 34) and figure 17b [10].



(a) Average stress revealed with different weightings for the individual methods as a component of a hybrid.



(b) Average stress revealed using multiple iterations of the different hybrid methods over time

Figure 17: Amount of stress revealed under mesh over multiple iterations

Looking at figure 16a above it is clear where there are weaknesses at predictable points the increasing speed at which meshing can be focused using heuristic refinement is significant, in particular during the first few iterations, see iteration 2 in figure 16a above. This suggests that the heuristics can on average perform better than a more general approach given a specific case. As expected however the extent to which this is true highly depends upon the users ability to correctly identify regions of interest in advance. Figure 16 below shows the bridge model undergoing relatively high stress at various points across the model but with exponential increase at specific points where structures join one another.

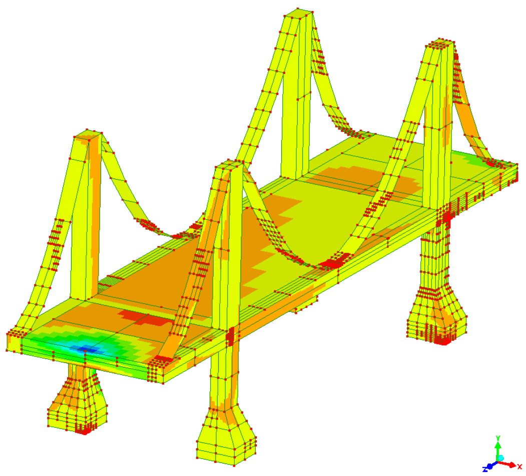
Figure 16b above indicates some unpredictability in the hybrid results from combining multiple methods. For example hybrid (3, 2) shows poor performance until iteration three while the others including hybrids with less precedence given to the hybrid with additional fluctuations between all four methods during the final four iterations. After closer observation of the models within LISA it seems the most likely cause of this is overlapping of areas between the different methods to the affect that a heuristic method often reveals just some of a high stress area which when observed by the stress refinement method triggers much more detailed refinement at that particular location. The alternative to this is the stress refiner has to do a lot more meshing relatively around the general area by itself in order to find this location. This effect can also be observed for different corner and edge points in figure 17a and b below.

This suggests that although heuristic refinement can potentially be unreliable if used individually, as a supplementary method to support stress based refinement it has potential to improve the overall speed when analysing an FE model.

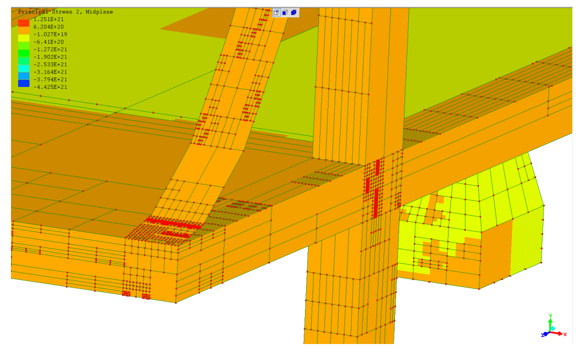
Having run the model with the hybrids the coloured stress gradients across each structure could be inspected as an engineer would, see figure 18 and appendices H, I and J below. LISA assigns colours to different ranges of stress based on the range of values within the model. It is therefore possible for green to potentially represent a degree of high stress in cases where high stress exists across the entire model and so this is the average, although in the majority of cases this will not be true with red areas representing those of highest stress followed by orange, yellow, green, light blue and eventually dark blue.

Looking at the gradient meshes in figure 18 it can be seen that refinement has been successfully focused in those areas of high stress with meshing becoming increasingly focused on smaller and smaller areas covering the highest stress this affect can be seen after as seen clearly after 8 iterations in figure 18b below and also in appendix H. This is supported by the data shown above in figure 17 above with the dramatic increase in the average stress revealed per iteration and thus convergence of the meshing procedures on these areas.

This effect is the desired result of applying mesh refinement as discussed in section 2 on the projects motivation and background.



(a) Stress Revealed through the initial highly coarse bridge mesh without running any iterations for either method



(b) Stress revealed within model after just 4 iterations with a heuristic stress method weighting of 3-2, edge heuristics also considered good in this case

Figure 18: Execution time increase to the amount of information revealed for the different approaches

7.7 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 all of the objectives being delivered as per the initial time plan. Quality was also maintained throughout the project by the application of good software engineering practice.

The modular architecture turned out to be the first great strength of the system allowing for a huge amount amount of potential extendibility in the future and simplifying the ease with which both of the methods could be integrated separately into the system. Although little focus has so far been given to the system's usability it could be developed and distributed as a public tool for experimentation with hybrid meshing with limited additional effort. Another highly flexible aspect is the system's ability to also 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 the need for the user to manually specify 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 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 would likely be a dissertation in its own right.

Although the system had a strong subsystem and class level architecture many of its weaknesses could be attributed to needing to prioritise the ability to perform 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 non-functional requirements so as to simply the design and reduce unnecessary state. This has largely been adhered to with higher order and lambda functions widespread throughout the codebase. In the later stages of the project it became apparent however that in many cases reliance on these features resulted in reduced readability and performance for many of underlying algorithms described in this dissertation.

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 applied industrial experience along with that of academics this would have 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 would need to have been acquired.

8.2 Improving Usability Through A Web Interface

Although it would have been possible to visit various engineers in order to conduct feedback the process would have been both time consuming on my part and inconvenient for the participant as a time at which to meet must be scheduled, also a laptop containing the working software would need to be brought to them on 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 short period of time after starting to experiment with it.

An alternative approach would be to develop a web interface so as to allow users to interact with the system in a more efficient manner. This approach would allow engineers to submit feedback digitally which could then be aggregated from a much greater range of users sources separated by significant geographical distance.

To use the web 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 the Standard for the Exchange of Product model data (STEP) and Initial Graphics Exchange Specification (IGES)) [25]. Upon receiving the request the web server would run the system using their input data and having finished allow them to download the re meshed model along with the calculated stress data for analysis.

8.3 Added Sophistication of Hybrid Generation

It has been shown the system can be used to effectively execute and evaluate discrete combinations of different methods it is clear this is an incredibly simple approach to demonstrating the working concept in reality the optimum meshing strategy is likely to be some fuzzy function of several meshing approaches with gradient weighting. As such this would be an exciting direction in which to take the project in future and would greatly increase the experimentation flexibility of the overall system.

9 Project Conclusion and Personal Reflections

Having used the system to successfully evaluate a range models and compare two individual methods for finite element meshing it has been shown that the project has been delivered to meet each of the three main objectives outlined under "Description Of The Work". The delivered system has been demonstrated capable of being able to effectively evaluate meshing approaches using both a traditional refinement approach and one derived from the domain of AI with effective comparisons between each. Simulation results from the suspension bridge above and the paper mill disk and cylinder (appendices I and J) have shown that there are significant potential benefits of using an alternative method such as an expert system in conjunction with traditional stress based refinement and that this can be applied without degradation of quality to the original mesh geometry. Although unlikely that an alternative refinement process will supersede stress based refinement in the near future the high computational cost for large models and the demonstrated potential of alternatives supports the case for conducting further research and development in this area.

From my own 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 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 personal 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 project 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 requiring high level analysis of the design and purpose of the system in order to continuously steer the project in the right direction. In many such cases the direction the project needed to take was not obvious making it hard to focus purely on implementation. Discussion and management of these decisions with my supervisor Jason Atkin ensured that the project was never stalled for too long and all tasks were successfully delivered within the specified time scales. As a result of these challenges I feel the project has provided me with a much better appreciation of the difficulties associated with delivering a software project in its entirety.

Throughout the majority of the project 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.

The research and evaluation phases were probably the most challenging for me personally, upon finishing I came to realise this was mostly due to a combination of my lack of prior experience with regards to academic research and formal education in mechanical engineering. Both of these factors meant I had to work a lot harder both to understand the initial problems associated with the methods and subsequently to perform reasonable evaluations of both my own results and those described within academic literature. One such example in this was the exponential increase in stress at particular points which took me by surprise having not stressed models to the point of breaking before. Overall had I chosen a more traditional computer science topic I believe both the research and evaluation stages would have been much easier and taken considerably less time.

As the project progressed the increase in scope also presented problems for me as the sole researcher and developer of the system. With a considerable body of research in the wider academic community about each of the specific problems the system needed to solve there was only time for me to survey the most popular papers for each subtopic. This in conjunction with much of the literature being highly specialist and requiring a postdoctoral level of understanding on finite element meshing meant that in the end it was only possible for me to write basic implementations for each of the subsystems given the time available to me.

I believe that having completed the research too much time was then spent concerned with the specifics of the implementation, much of which was associated with integrating the functionality of LISA into my system. Although LISAs simplicity was its great strength and helped in simplifying many of the initial design and testing aspects of the project its lack of an extensive API resulted in a large amount of the projects time being focused towards system integration issues which were not apparent during the design and research stages. Although these problems such as element sorting and data modelling proved interesting challenges solving them was considerably more time consuming than was initially predicted and thus reduced the amount of time that could be directed towards the other more theoretical components. Given the chance to repeat the project and having learnt a lot about of finite element systems I feel I would better placed to both use and evaluate a greater range of potential choices. Its likely I would have therefore changed the finite element application and instead tried to use time I may have saved to improve the system for combining methods to add sophistication.

Doing the project I would like to implement the refinement processes for a wider variety of different element types such as tri3 and tet4 as shown in appendix A. Although the system architecture would remain the same for the most part the potential for a more conclusive evaluation of the hybrid approaches using models of different element types would be interesting.

I think its also likely if repeating the project that I would continue to use Dolsaks ILP knowledge base, it proved effective as a way of refining the mesh in certain areas based on user input and met the criteria required for a second refinement strategy whilst not taking an excessive amount of time to implement. Working on the project this time around I ran out of time to fully experiment with designing edge specifications that triggered all the rules that are used for mesh refinement. This suggests that there is still a lot of potential for continued development of the overall system without altering this aspect of it.

In the end I was also glad that I selected C# as the language for system implementation and would do so again with the possible exception of Java so as to have better cross platform compatibility. Initially I was also considering Python although upon reflection I feel this would have been a mistake with implementation of the more object oriented aspects such as the element interface and subclass structure being made much more difficult by the language.

References

[1] E. Bellenger, Y. Benhafid, and N. Troussier. Framework for controlled cost and quality of assumptions in finite element analysis. *Finite Elements in Analysis and Design*, 45(1):25–36, 2008.

[2] J. P. Dittmer, C. G. Jensen, M. Gottschalk, and T. Almy. Mesh optimization using a genetic algorithm to control mesh creation parameters. *Computer-Aided Design and Applications*, 3(6):731–740, 2006.

[3] B. Dolšak and A. Jezernik. Mesh generation expert system for engineering analyses with fem. *Computers in Industry*, 17(2-3):309–315, 1991.

[4] B. Dolšak, A. Jezernik, and I. Bratko. A knowledge base for finite element mesh design. *Artificial intelligence in engineering*, 9(1):19–27, 1994.

[5] B. Dolsak and S. Muggleton. The application of inductive logic programming to finite element mesh design. In *Inductive logic programming*. Citeseer, 1992.

[6] B. Dolšak, F. Rieg, M. Novak, and R. Hackenschmidt. Consultative rule-based intelligent system for finite element type selection.

[7] P. Dvorak. Two meshing methods are better than one. http://machinedesign.com/archive/two-meshing-methods-are-better-one.

[8] A. A. Khan, I. A. Chaudhry, and A. Sarosh. Case based reasoning support for adaptive finite element analysis: mesh selection for an integrated system. *Applied Physics Research*, 6(3):21, 2014.

[9] N.-H. Kim. Structural design using finite elements. http://web.mae.ufl.edu/nkim/eas6939/Opt_FEM.pdf.

[10] F. Kreith. Stress concerntration fundamentals. http://www.engineersedge.com/material_science/stress_concentration_fundamentals_9902.htm.

[11] R. Lakes. Poisson intro. http://silver.neep.wisc.edu/~lakes/PoissonIntro.html.

[12] L. Manevitz, M. Yousef, and D. Givoli. Finite-element mesh generation using self-organizing neural networks. *Microcomputers in Civil Engineering*, 12(4):233–250, 1997.

[13] J. S. P. Max D. Gunzburger. Adaptive finite element techniques. http://www.cs.rpi.edu/~flaherje/pdf/fea8.pdf.

[14] J. S. P. Max D. Gunzburger. Finite element methods. https://people.sc.fsu.edu/~jburkardt/classes/fem_2011/chapter1.pdf.

[15] D. S. McRae. r-refinement grid adaptation algorithms and issues. *Computer Methods in Applied Mechanics and Engineering*, 189(4):1161–1182, 2000.

[16] Muggleton and Feng. Golem. http://www-ai.ijs.si/~ilpnet2/systems/golem.html.

[17] S. Muggleton. Logic based and probabilistic symbolic learning. https://www.youtube.com/watch?v=4CwdO5dWW98.

[18] S. Muggleton, R. Otero, and A. Tamaddoni-Nezhad. *Inductive logic programming*, volume 38. Springer, 1992.

[19] G. P. Nikishkov. Introduction to the finite element method. http://homepages.cae.wisc.edu/~suresh/ME964Website/M964Notes/Notes/introfem.pdf.

[20] D. Peter. ray tracing. http://danielpeter.github.io/rays.html.

[21] D. Piepgrass. The convex hull of a planar point set. http://loyc.net/2014/2d-convex-hull-in-cs.html.

[22] D. Sunday. The convex hull of a planar point set. http://geomalgorithms.com/a10-_hull-1.html.

[23] D. Sunday. Convex hull of a planar set of points. http://geomalgorithms.com/a10-_hull-1.html#chainHull_2D.

[24] D. team. Doxygen. http://www.stack.nl/~dimitri/doxygen/.

[25] L. team. Lisa manual. http://www.lisafea.com/pdf/manual.pdf.

[26] P. N. team. Youngs modulus. http://physicsnet.co.uk/a-level-physics-as-a2/materials/young-modulus/.

[27] V. S. Team. Visual studio maintaintainance index. https://msdn.microsoft.com/en-gb/library/bb385914.aspx.

[28] Unknown. Brute force closest pair and convex-hull. http://www.csl.mtu.edu/cs4321/www/Lectures/Lecture%206%20-%20Brute%20Force%20Closest%20Pair%20and%20Convex%20and%20Exhausive%20Search.htm.

[29] Unknown. The convex hull of a set of points. http://www2.lawrence.edu/fast/GREGGJ/CMSC210/convex/convex.html.

[30] Unknown. Finite element mesh refinement. https://www.comsol.com/multiphysics/mesh-refinement.

[31] Unknown. High stress corner. http://www.engineeringanalysisservices.com/moving-mesh-fea-analysis.php.

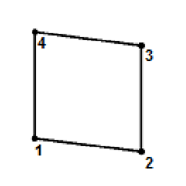
[32] Various. Algorithm implementation/geometry/convex hull/monotone chain. https://en.wikibooks.org/wiki/Algorithm_Implementation/Geometry/Convex_hull/Monotone_chain.

[33] Various. How much does ansys cost? http://mscnastrannovice.blogspot.co.uk/2013/04/how-much-does-ansys-cost.html.

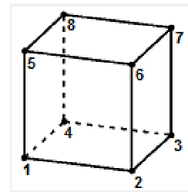
[34] L. Vasiliauskienė and R. Baušys. Intelligent initial finite element mesh generation for solutions of 2d problems. *Informatica*, 13(2):239–250, 2002.

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 element types can be classified using the

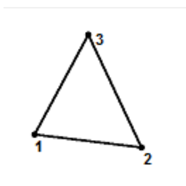


(a) quad4 element

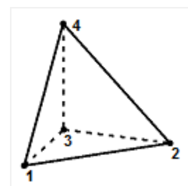


(b) hex8 element

Figure 19: Square based elements

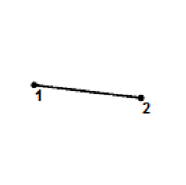


(a) tri3 element

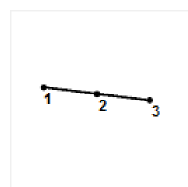


(b) tet4 element

Figure 20: Triangular based elements



(a) line2 element



(b) line3 element

Figure 21: Line based elements

B Unit Testing

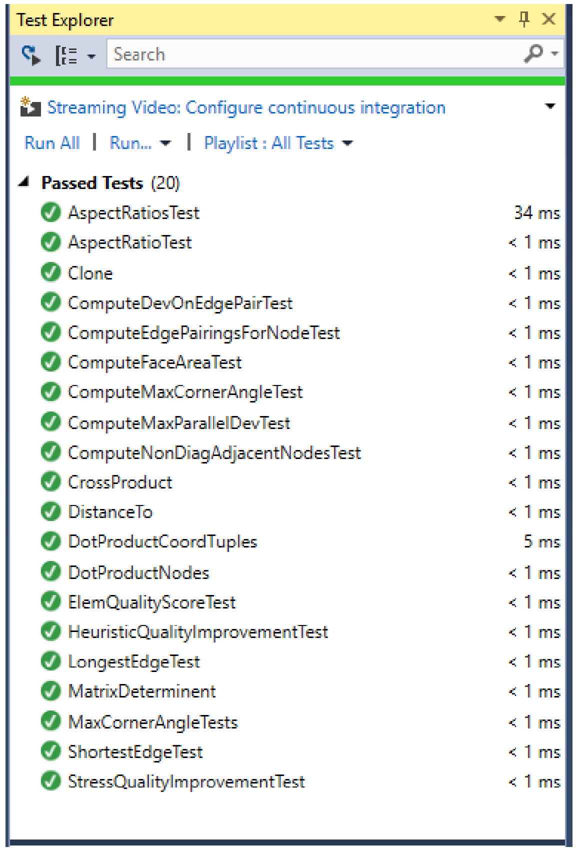


Figure 22: Visual Studio window showing the small suite of twenty tests for validating the core functionalyu of the system.

C Edge Definition Categories

Edge Type

- important long
- important
- important short
- not important
- circuit
- half circuit
- quarter circuit
- short for a hole
- long for a hole

- circuit hole
- half circuit hole
- quarter circuit hole

Boundary Type

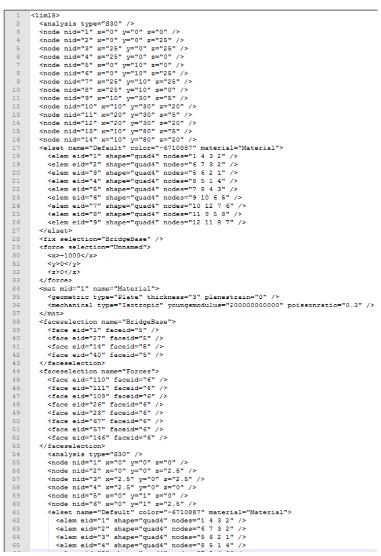
- free
- fixed on one side
- fixed on two sides
- fixed completely

Load Type

- no loading
- one side loaded
- two sides loaded
- continuous loading

D 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



```
<liml8>
  <analysis type="S30" />
  <node nid="1" x="0" y="0" z="0" />
  <node nid="2" x="0" y="0" z="25" />
  <node nid="3" x="25" y="0" z="25" />
  <node nid="4" x="25" y="0" z="0" />
  <node nid="5" x="0" y="10" z="0" />
  <node nid="6" x="0" y="10" z="25" />
  <node nid="7" x="25" y="10" z="25" />
  <node nid="8" x="25" y="10" z="0" />
  <node nid="9" x="10" y="30" z="5" />
  <node nid="10" x="10" y="30" z="20" />
  <node nid="11" x="20" y="30" z="5" />
  <node nid="12" x="20" y="30" z="20" />
  <node nid="13" x="10" y="80" z="5" />
  <node nid="14" x="10" y="80" z="20" />
  <elset name="Default" color="-6710887" material="Material">
    <elem eid="1" shape="quad4" nodes="1 4 3 2" />
    <elem eid="2" shape="quad4" nodes="6 7 3 2" />
    <elem eid="3" shape="quad4" nodes="5 6 2 1" />
    <elem eid="4" shape="quad4" nodes="8 5 1 4" />
    <elem eid="5" shape="quad4" nodes="7 8 4 3" />
    <elem eid="6" shape="quad4" nodes="9 10 6 5" />
    <elem eid="7" shape="quad4" nodes="10 12 7 6" />
    <elem eid="8" shape="quad4" nodes="11 9 5 8" />
    <elem eid="9" shape="quad4" nodes="12 11 8 7" />
  </elset>
  <fix selection="BridgeBase" />
  <force selection="Unnamed">
    <x>-1000</x>
    <y>0</y>
    <z>0</z>
  </force>
  <mat mid="1" name="Material">
    <geometric type="Plate" thickness="3" planestrain="0" />
    <mechanical type="Isotropic" youngsmodulus="200000000000" poissonratio="0.3" />
  </mat>
  <faceselection name="BridgeBase">
    <face eid="1" faceid="5" />
    <face eid="27" faceid="5" />
    <face eid="14" faceid="5" />
    <face eid="40" faceid="5" />
  </faceselection>
  <faceselection name="Forces">
    <face eid="110" faceid="6" />
    <face eid="111" faceid="6" />
    <face eid="109" faceid="6" />
    <face eid="26" faceid="6" />
    <face eid="23" faceid="6" />
    <face eid="67" faceid="6" />
    <face eid="57" faceid="6" />
    <face eid="146" faceid="6" />
  </faceselection>
    <analysis type="S30" />
    <node nid="1" x="0" y="0" z="0" />
    <node nid="2" x="0" y="0" z="2.5" />
    <node nid="3" x="2.5" y="0" z="2.5" />
    <node nid="4" x="2.5" y="0" z="0" />
    <node nid="5" x="0" y="1" z="0" />
    <node nid="6" x="0" y="1" z="2.5" />
    <elset name="Default" color="-6710887" material="Material">
      <elem eid="1" shape="quad4" nodes="1 4 3 2" />
      <elem eid="2" shape="quad4" nodes="6 7 3 2" />
      <elem eid="3" shape="quad4" nodes="5 6 2 1" />
      <elem eid="4" shape="quad4" nodes="8 5 1 4" />
```

(a) Cut down .liml file to show general content which largely defined the schema for the systems data model

```
{
    "Edges": [{
        "Id": 1,
        "edgeType": "circuit",
        "loadType": "oneSideLoaded",
        "boundaryType": "free",
        "nodePath": [42, 198, 197, 41]
    },
    {
        "Id": 2,
        "edgeType": "importantLong",
        "loadType": "oneSideLoaded",
        "boundaryType": "fixedTwoSides",
        "nodePath": [70, 62, 61, 46, 45, 67]
    },
    {
        "Id": 3,
        "edgeType": "importantShort",
        "loadType": "oneSideLoaded",
        "boundaryType": "fixedTwoSides",
        "nodePath": [41, 197]
    },
    {
        "Id": 4,
        "edgeType": "notImportant",
        "loadType": "notLoaded",
        "boundaryType": "fixedCompletely",
        "nodePath": [197, 207]
    },
    {
        "Id": 5,
        "edgeType": "circuit",
        "loadType": "continuiousLoading",
        "boundaryType": "fixedOneSide",
        "nodePath": [88, 142, 148, 147, 140, 141, 135, 85]
    }
    ]
}
```

(b) A json file containing the edges of interest specified by an engineer, this is parsed and the rules are applied to determine the models meshing based on the input

E 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.

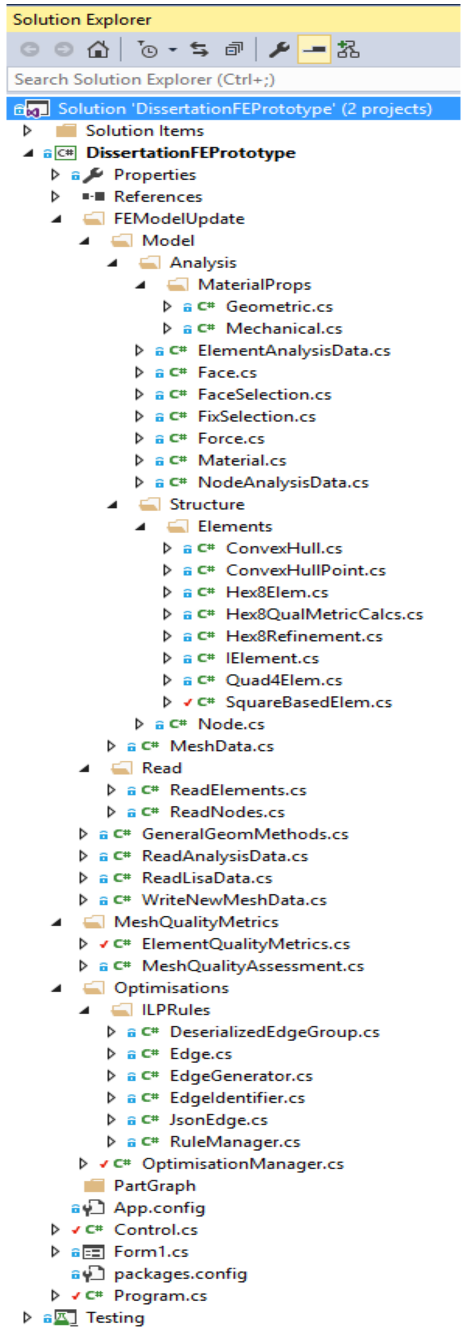


Figure 24: The metrics calculated by visual studio for all high level modules in the system

F Doxygen Documentation

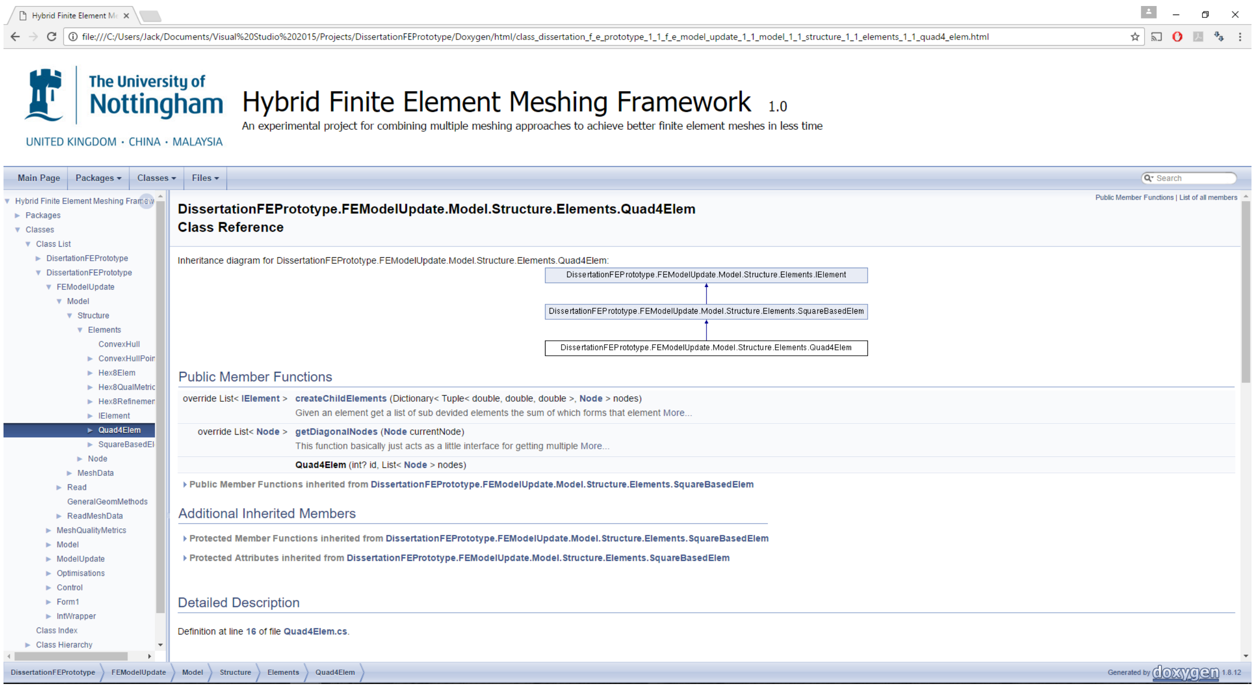


Figure 25: The manual page for the Quad4 element class with the class hierarchy and specific public methods

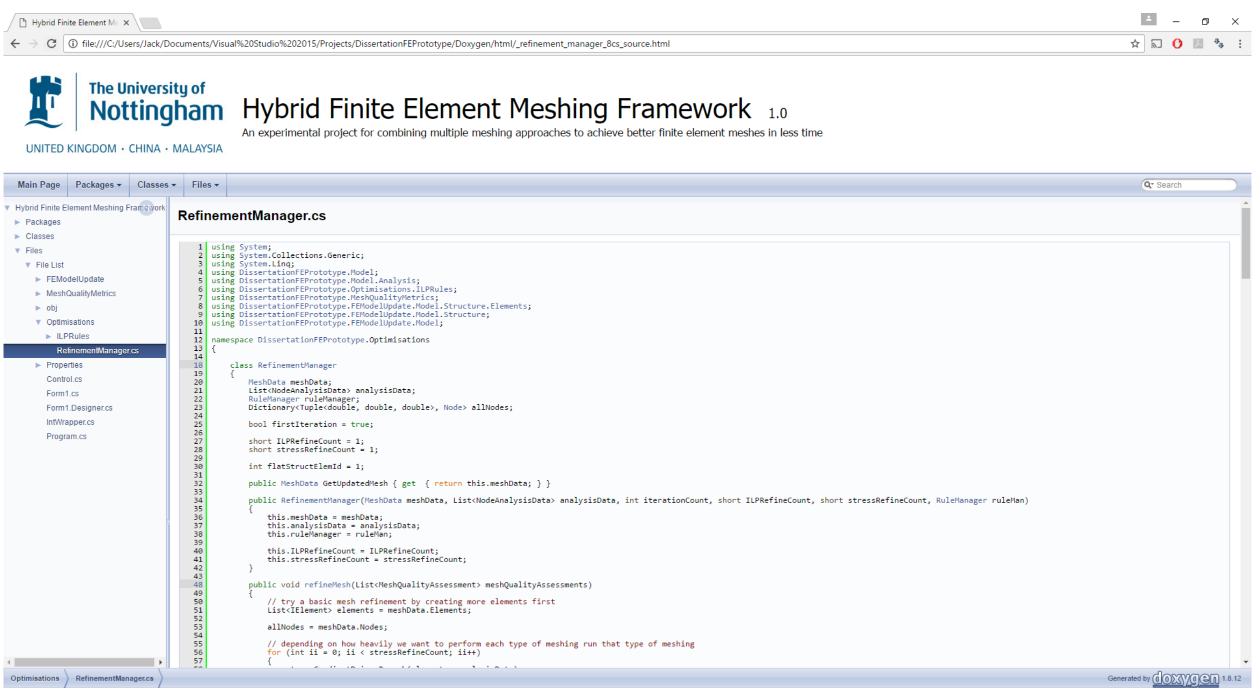


Figure 26: Code for the RefinementManager class viewed within the Doxygen UI

G Software Quality Metrics

Code Metrics Results

Filter: None Min: Max:

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

Code Metrics Results

Filter: None Min: Max:

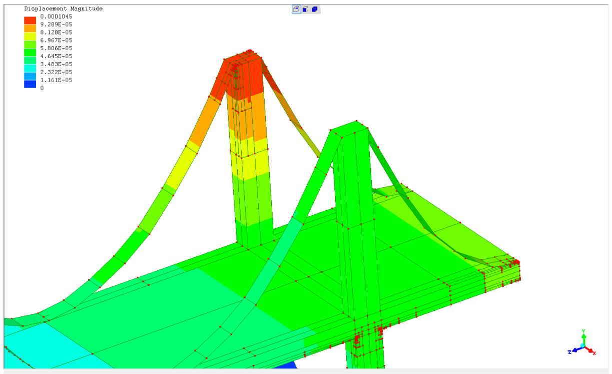
Hierarchy	Maintainability I...	Cyclomatic Complexity	Depth of Inheritance	Class Coupling	Lines of Code
DissertationFEPrototype (Debug)	75	808	7	125	1,871
DissertationFEPrototype.Model.Analysis.MaterialProps	92	8	1	0	14
Geometric	92	4	1	0	7
Mechanical	92	4	1	0	7
DissertationFEPrototype.FEModelUpdate.Model	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
EdgeIdentifier	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 28: The metrics calculated by visual studio for the all classes in the final system

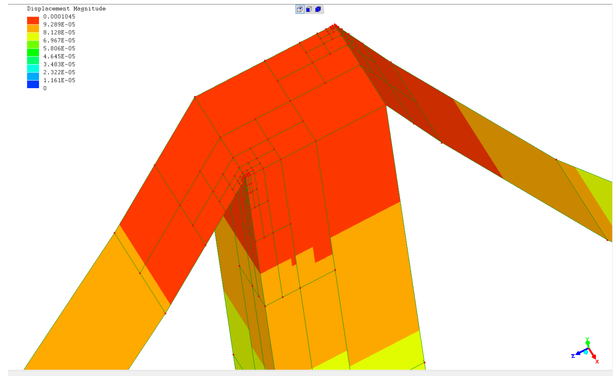
H Mesh Refinements

This appendix item attempt too show the general mesh that are formed using the heuristic and stress based refinement strategy with examples of where a heuristic has been placed well and poorly and where there is also variation in the threshold used to decide whether elements are meshed with the stress variable. Although in the rest of the models we are looking for stress since this is the primary variable of interest displacement has been selected as the analysis variable for displacement due to it producing a clearer gradient than stress.

H.1 Heuristic Refinement



(a) Important edges specified effectively to facilitate pre-emptive meshing of area which undergoes high stress



(b) Close up view of refinement for high displacement areas

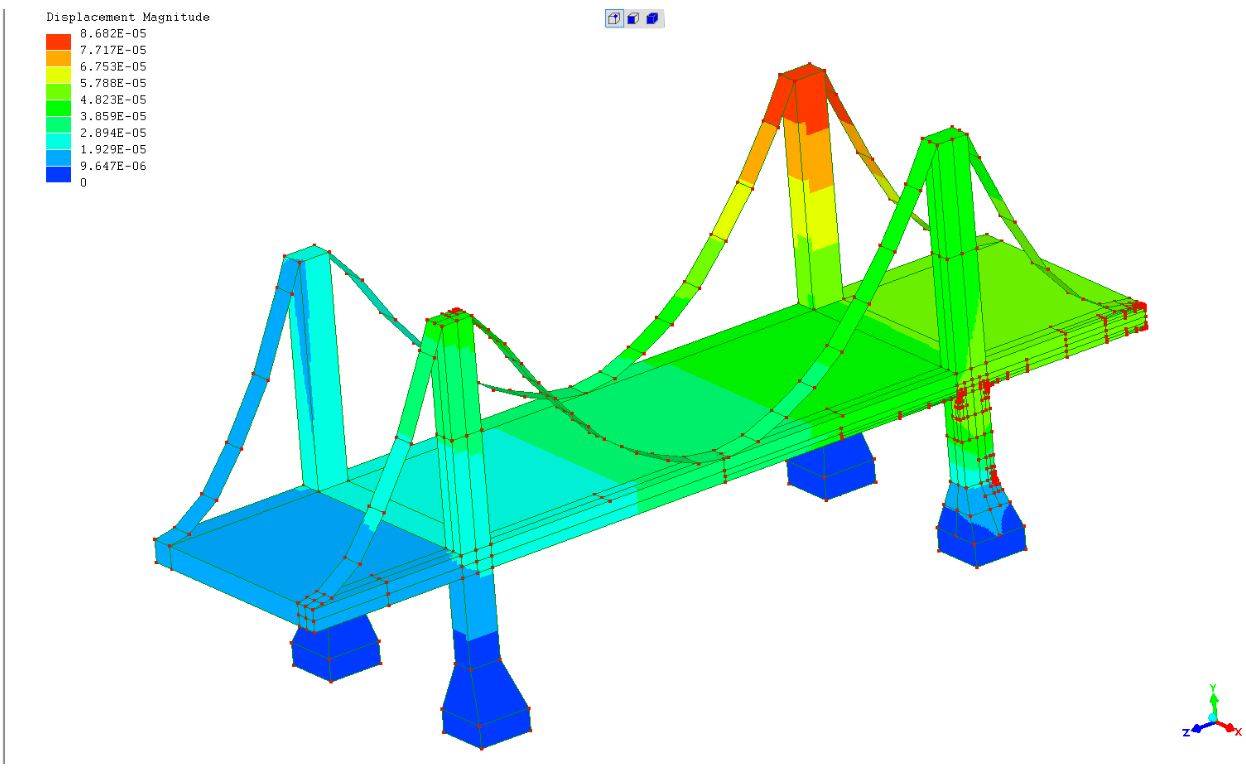


Figure 30: Important edges more poorly specified missing high displacement region on top of furthest suspension bridge tower

H.2 Stress Refinement

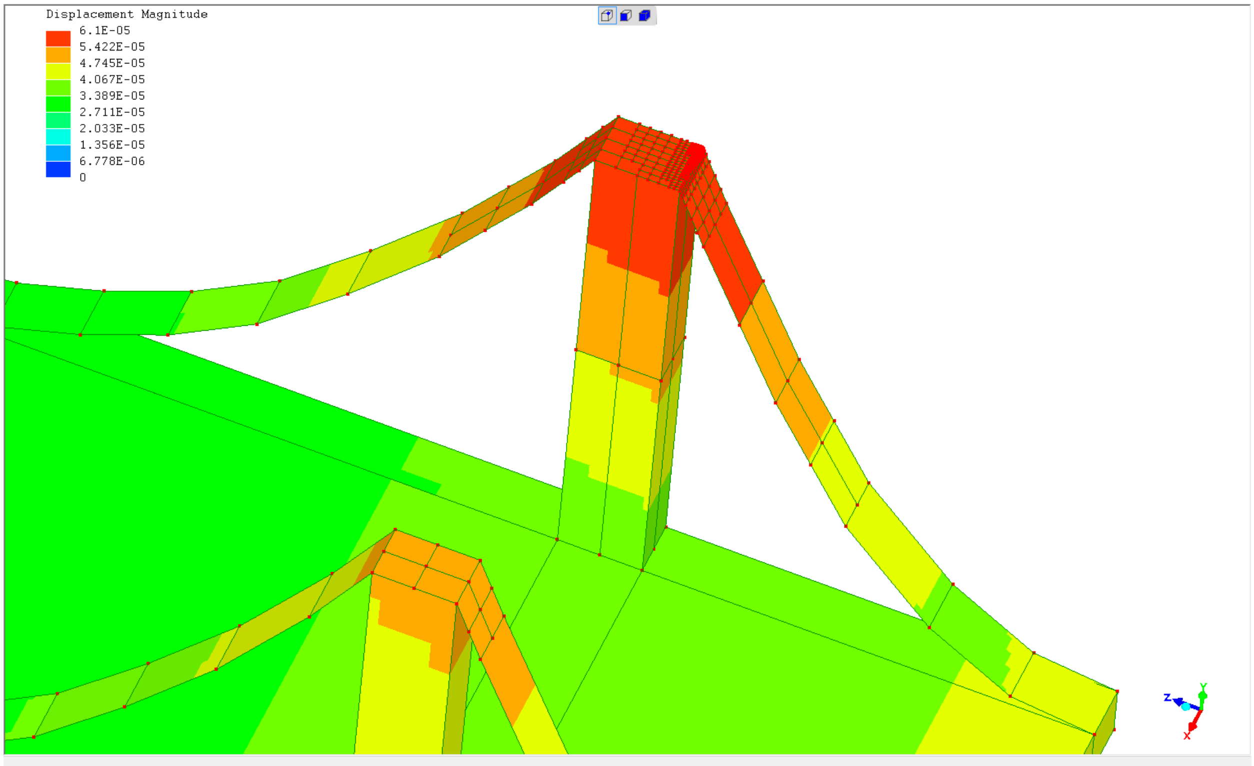


Figure 31: Iterative stress/ displacement refinement method used to focus meshing on the top 6% most displaced region of model

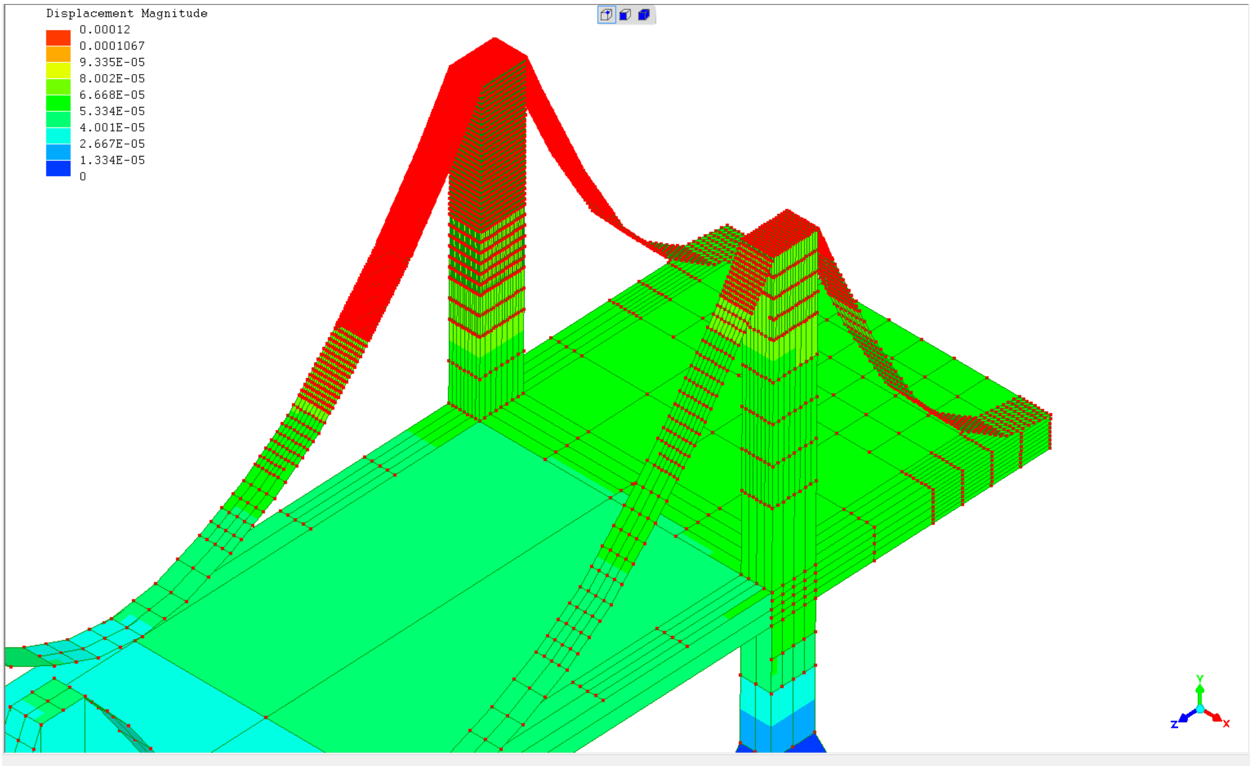


Figure 32: Iterative refinement of high displacement but with the remesh threshold specified as the average displacement across the whole model. A consequence of this is the gradient of refinement fidelity that can be seen corresponding to the importance of that part of the structure

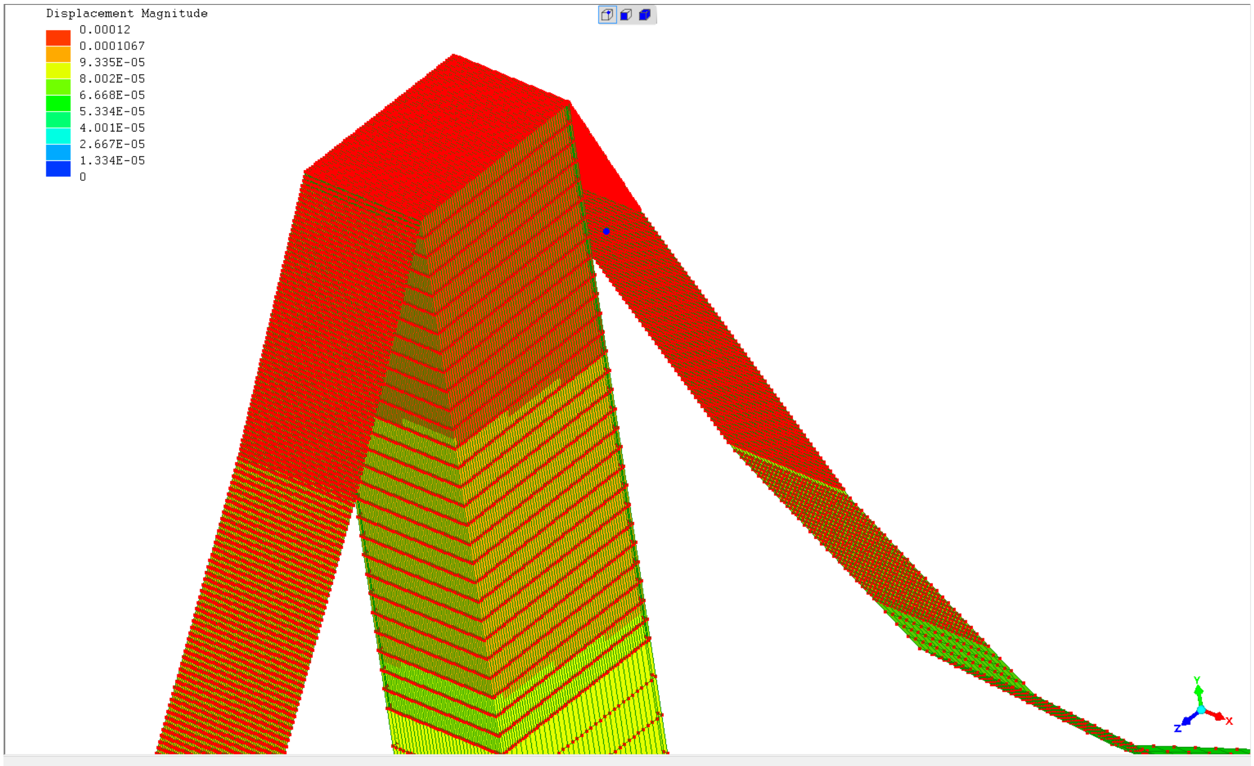
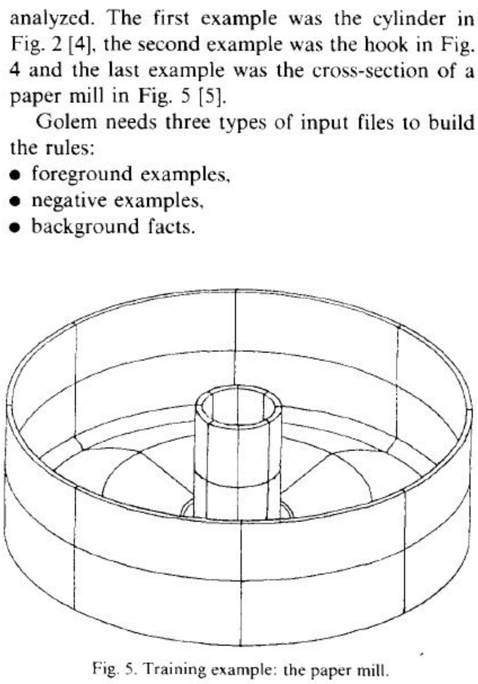


Figure 33: Closer view of very high meshing intensity

I Paper Mill Simulation Results

For the paper mill simulation angular forces were set up around the outside of the disk so as to simulate the effect of the disk rotating at high speed, with it also being pulled outwards in the axial direction. This generated some interesting patches of stress across the main body of the structure which could easily be specified as edge rules. Looking at figure 34 it is possible to see very high range of stress values for stress observable within the model as a consequence of stress concentrating at particular points.

Still need to add some more to this bit



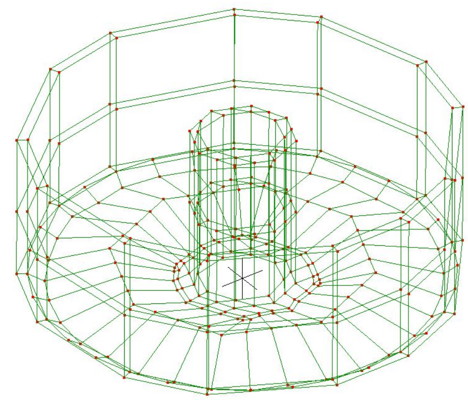
analyzed. The first example was the cylinder in Fig. 2 [4], the second example was the hook in Fig. 4 and the last example was the cross-section of a paper mill in Fig. 5 [5].

Golem needs three types of input files to build the rules:

- foreground examples,
- negative examples,
- background facts.

Fig. 5. Training example: the paper mill.

(a) Half of Cylinder structure described by dolsak in his papers for training ILP system



(b) Replication of mesh structure specified by Dolsak within his paper [3]

Figure 34: Execution time increase compared to the amount of information revealed for the different approaches

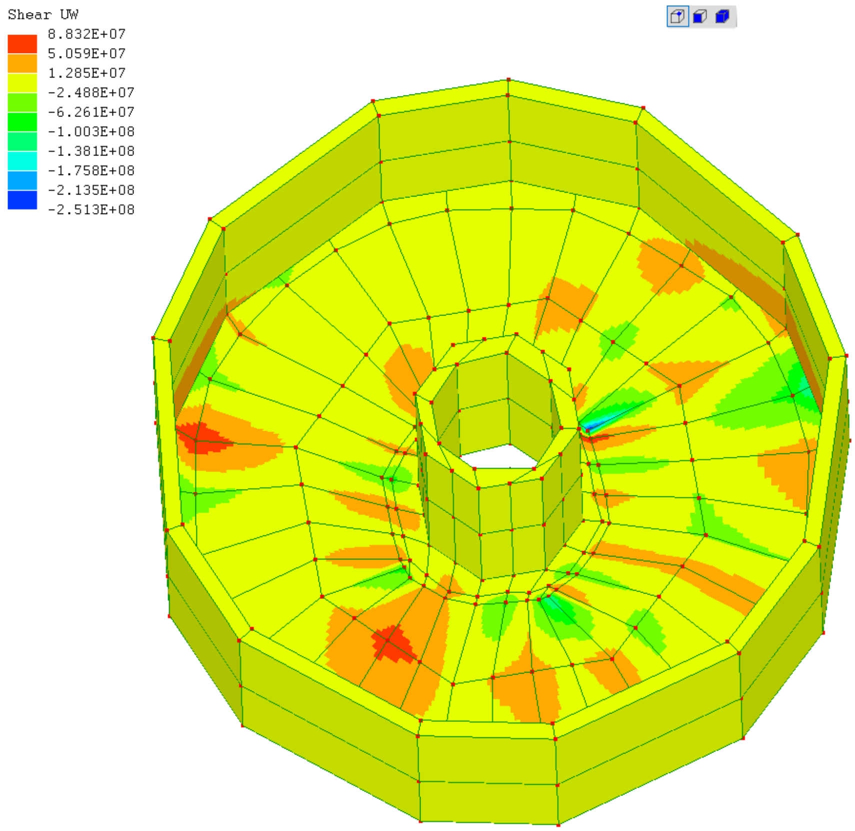
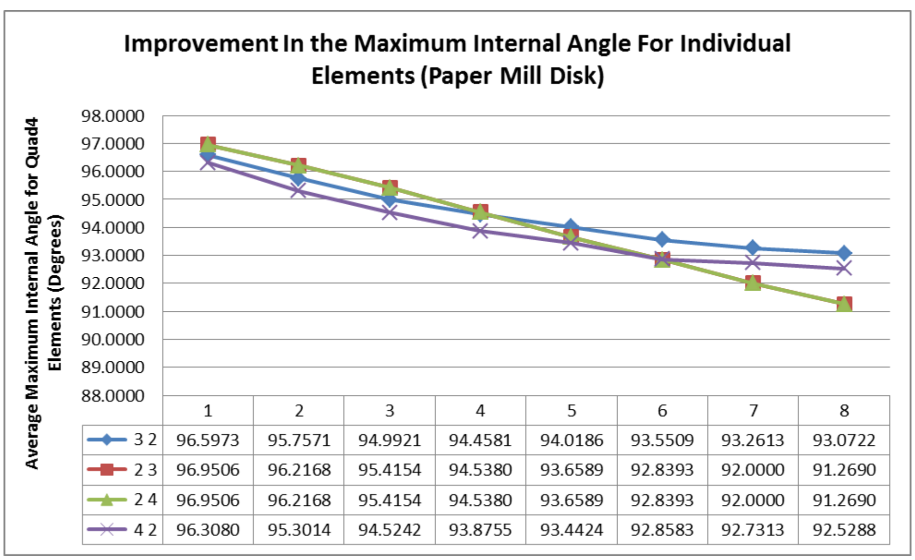
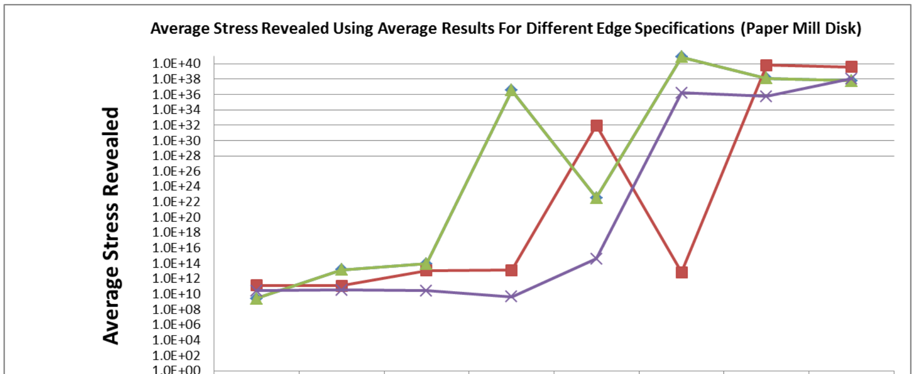


Figure 35: The initially stressed paper mill part used to define edge sets for further meshing, stress concentrations can be observed in red with colour coding at the top indicating showing a rapidly exponential increase concentrated at those points



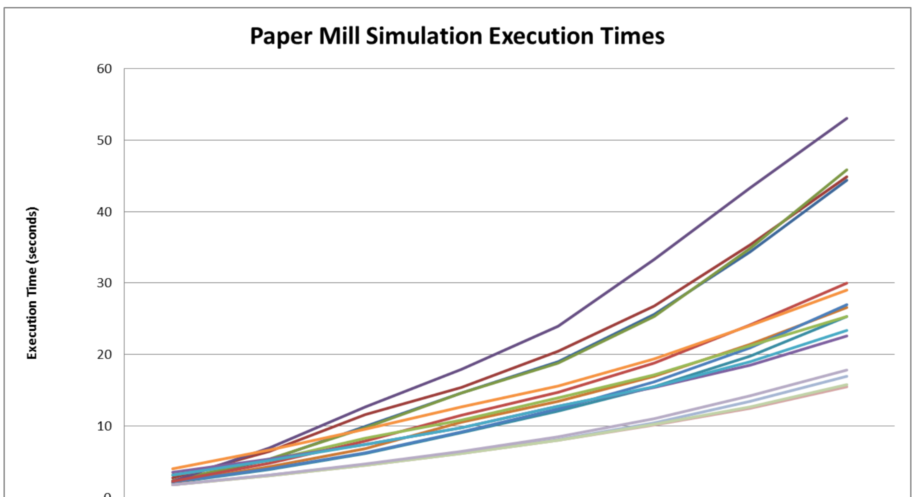
	1	2	3	4	5	6	7	8
3 2	96.5973	95.7571	94.9921	94.4581	94.0186	93.5509	93.2613	93.0722
2 3	96.9506	96.2168	95.4154	94.5380	93.6589	92.8393	92.0000	91.2690
2 4	96.9506	96.2168	95.4154	94.5380	93.6589	92.8393	92.0000	91.2690
4 2	96.3080	95.3014	94.5242	93.8755	93.4424	92.8583	92.7313	92.5288

Figure 36: Improvement in the maximum internal Angle for Quad4 Elements



	1	2	3	4	5	6	7	8
Heur Ref: 2 Stress Ref: 3	2.6E+09	1.4E+13	9.1E+13	3.3E+36	3.4E+22	7.2E+40	1.2E+38	5.9E+37
Heur Ref: 3 Stress Ref: 2	1.3E+11	1.2E+11	1.2E+13	1.3E+13	8.8E+31	7.0E+12	6.5E+39	3.7E+39
Heur Ref: 2 Stress Ref: 4	2.6E+09	1.4E+13	9.1E+13	3.3E+36	3.4E+22	7.2E+40	1.2E+38	5.9E+37
Heur Ref: 4 Stress Ref: 2	2.7E+10	3.5E+10	2.6E+10	4.6E+09	3.7E+14	1.6E+36	5.8E+35	1.1E+38

Figure 37: Improvement on average for detecting across all nodes within the model over multiple iterations, results for each weighting with different edge heuristics also averaged



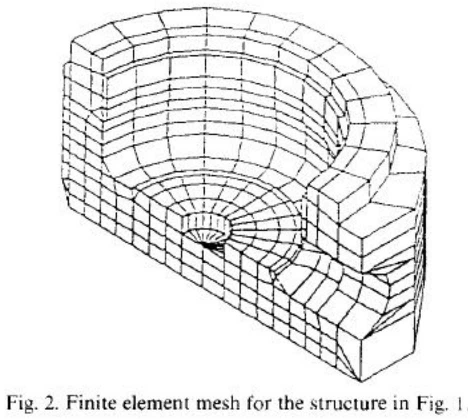
	1	2	3	4	5	6	7	8
Best 3 2	2.2480834	5.2980238	9.9278763	14.6413761	18.9525373	25.6362756	34.4172784	44.3590124
Best 2 3	2.6947666	6.4580226	11.5470852	15.3630073	20.4178244	26.810265	35.3091481	44.8931558
Best 2 4	2.3114277	5.3282764	9.71778	14.581265	18.8220534	25.322047	34.8874536	45.8797378
Best 4 2	2.2542435	6.8953451	12.7089335	17.9305188	23.9634844	33.3125908	43.2857606	53.0395733
Good 3 2	2.076859	3.9445738	6.1439582	9.0622778	12.1089675	15.4485565	19.7542634	25.2840629
Good 2 3	2.0717937	4.2480058	6.8725472	10.5305244	13.4817521	16.9947665	21.4580634	26.6223347
Good 2 4	2.1582927	3.9936058	6.2666088	9.1855752	12.4108067	16.1201035	20.955848	27.0023237
Good 4 2	2.2528445	4.8155588	7.8880126	11.4622627	14.7190347	18.8265281	24.1720304	29.9693025
Ok 3 2	3.1661625	5.189768	8.3273216	10.8483116	13.9243742	17.1495859	21.1905607	25.327526
Ok 2 3	3.5287104	5.3411082	7.4023018	9.7655707	12.7911976	15.3538062	18.5226808	22.6166594
Ok 2 4	3.1093936	5.1903325	7.3856129	9.7171008	12.7406506	15.5184714	19.018128	23.3591129
Ok 4 2	4.0268462	6.6101862	9.538086	12.6994566	15.5627201	19.3334752	24.0670395	29.039551
Poor 3 2	1.7880273	3.0816129	4.5597013	6.2189616	8.1110764	10.4183108	13.4025463	16.9503375
Poor 2 3	1.7992511	3.02152	4.4807479	6.1039625	7.9534522	10.0863081	12.5112035	15.4505198
Poor 2 4	1.7901696	3.0479097	4.4914188	6.1577408	7.9911611	10.1894514	12.6611883	15.823794
Poor 4 2	1.7664745	3.1016857	4.6446441	6.3941163	8.4486965	10.9680494	14.1788166	17.8675577

Iteration Number

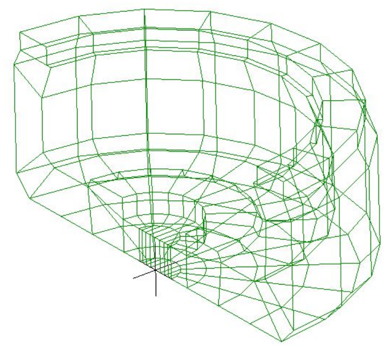
Figure 38: Time taken per iteration using the different hybrid weightings with varying edge quality specifications

J Half Cylinder Simulation Results

The half cylinder was the third model used to test the system.

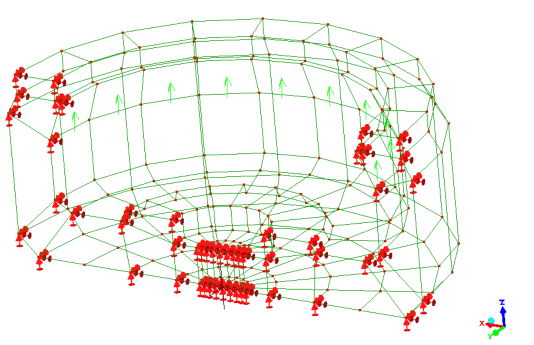


(a) Half of Cylinder structure described by dolsak in his papers for training ILP system

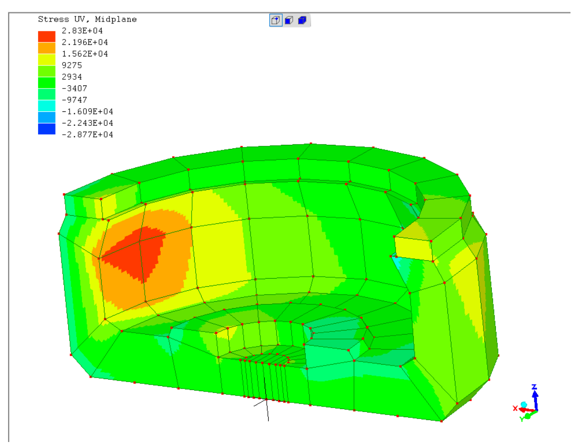


(b) Replication of mesh structure specified by Dolsak within his paper [3]

Figure 39: Execution time increase compared to the amount of information revealed for the different approaches



(a) Cylinder Constrained by its adjacent half with forces applied up and outwards on its inner rim



(b) Half cylinder with an initial stress concentration having performed a simple execution of the model.

Figure 40: Initial configuration for the half cylinder and some stresses revealed on the structure

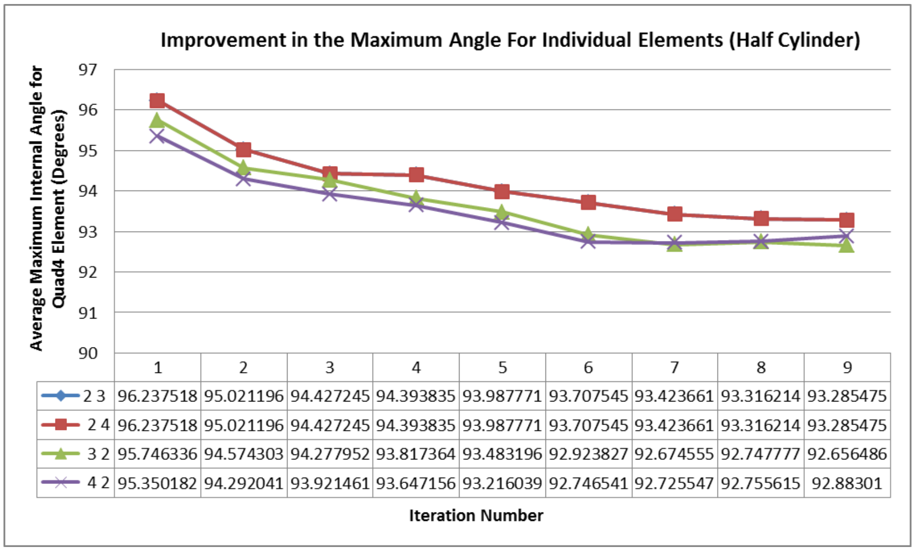


Figure 41: Improvement in corner angles for Quad4 elements using the different hybrid methods on the cylinder

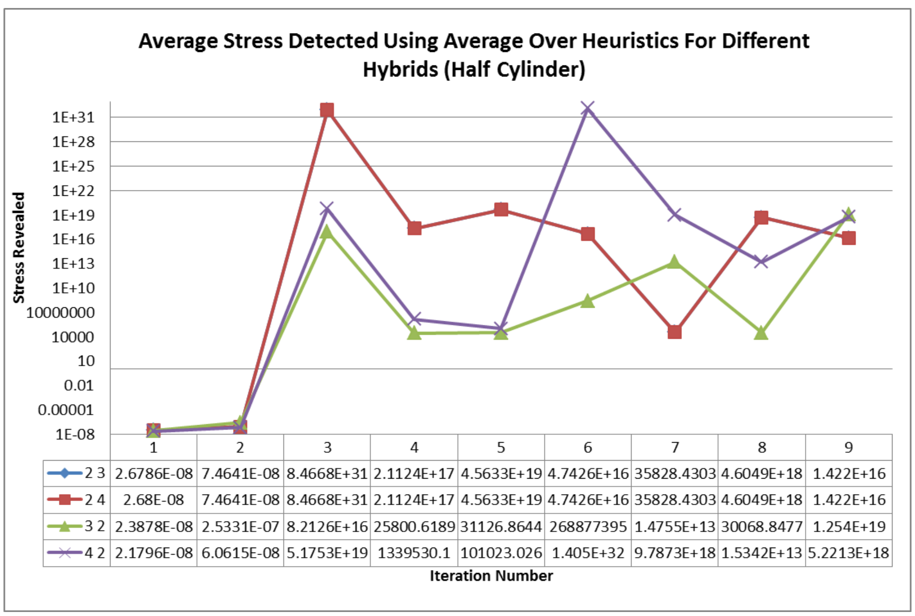
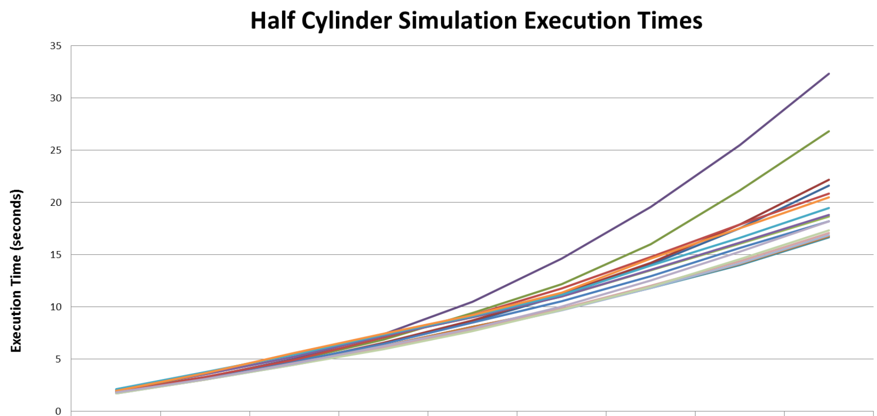


Figure 42: Stress revealed for each iteration using the different hybrid methods

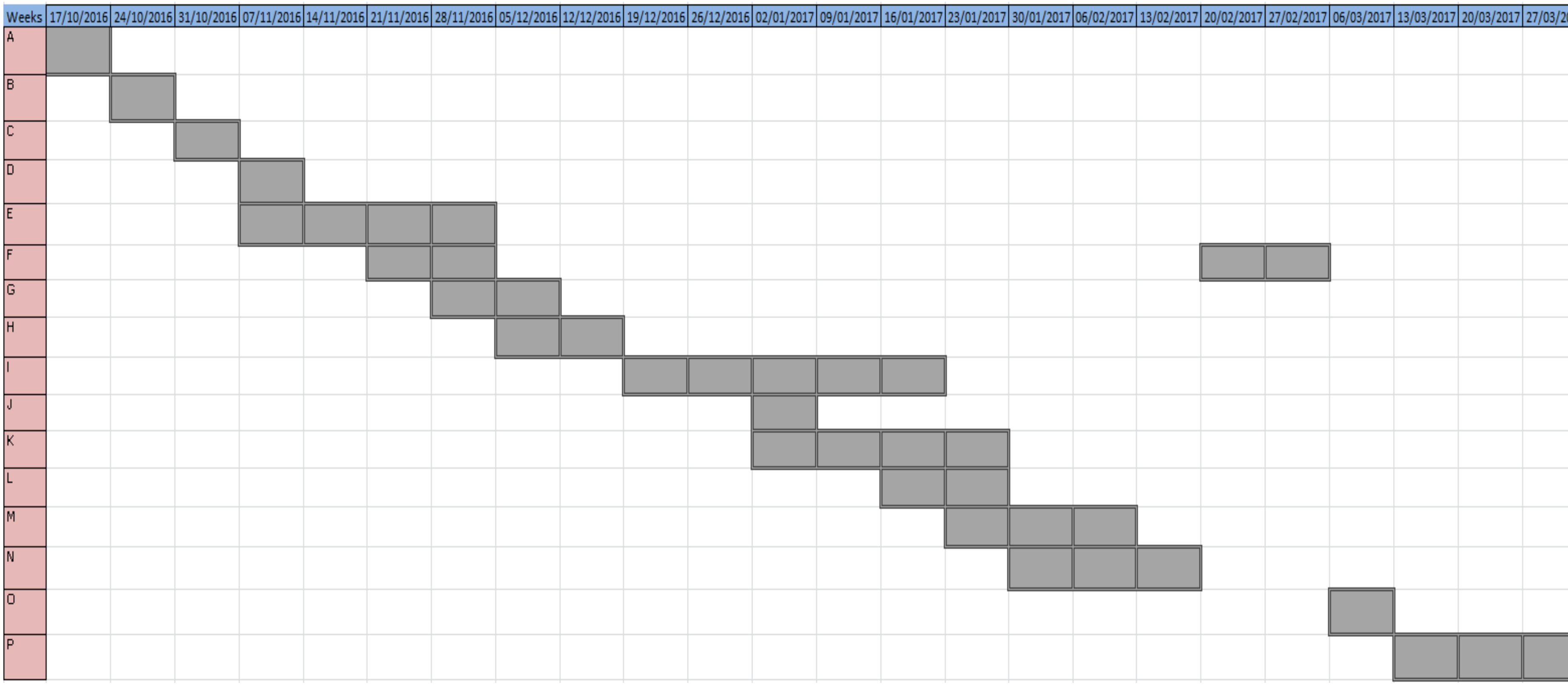


	1	2	3	4	5	6	7	8	9
Best 2 3	1.7796214	3.1146889	4.6861514	6.507011	8.6277327	11.1308824	14.0441411	17.5459759	21.6086869
Best 2 4	1.7714943	3.1279125	4.696951	6.5534305	8.7332297	11.2602181	14.2232932	17.8808227	22.1667537
Best 3 2	1.8646285	3.2570549	4.8737412	6.8810577	9.4044212	12.1656023	16.0221925	21.1508489	26.8507046
Best 4 2	1.7592208	3.2409788	5.021166	7.3999081	10.4846543	14.6367394	19.596518	25.5232238	32.349154
Good 2 3	1.8597843	3.2010833	4.6035048	6.2357525	8.1064808	9.7939765	11.812564	14.0242655	16.6813242
Good 2 4	1.8479953	3.2093708	4.6637149	6.2659178	8.1005984	9.8378459	11.8606859	14.0958345	16.7853164
Good 3 2	1.8615058	3.238007	4.7817504	6.4638284	8.5000891	10.5425836	12.9361904	15.6719938	18.1937456
Good 4 2	1.9461849	3.3055916	4.9453004	7.0760994	9.230467	11.761233	14.7737408	17.9003281	20.8317977
Ok 2 3	2.0262619	3.5613508	5.2485797	7.20605	8.9957797	10.9995476	13.482133	16.0389695	18.6761522
Ok 2 4	2.0854873	3.5713539	5.2494259	7.2215862	9.0173402	11.0263401	13.5415401	16.1558542	18.8114346
Ok 3 2	2.1480329	3.7834975	5.4561403	7.353233	9.0969068	11.1403439	13.94624	16.6112364	19.472947
Ok 4 2	1.9825304	3.6451697	5.6026499	7.4330267	9.1964132	11.3393298	14.646817	17.533913	20.4980566
Poor 2 3	1.7363632	3.0682604	4.4566294	6.0655103	7.8116782	9.7030631	11.8466321	14.1910517	16.8827906
Poor 2 4	1.7582856	3.0657518	4.6211484	6.1455792	7.8944	9.8101157	12.0212102	14.3877435	17.0669146
Poor 3 2	1.7256743	3.0283878	4.4490978	5.9507029	7.6748704	9.7157166	11.9402994	14.5644364	17.3337012
Poor 4 2	1.8297664	3.092236	4.5859376	6.23386	7.9586173	10.027494	12.547741	15.3155929	18.1967242

Iteration Number

Figure 43: Time taken per iteration using the different hybrid weightings with varying edge quality specifications

K Gantt Chart for Project Time Management


Weeks
17/10/2016
24/10/2016
31/10/2016
07/11/2016
14/11/2016
21/11/2016
28/11/2016
05/12/2016
12/12/2016
19/12/2016
26/12/2016
02/01/2017
09/01/2017
16/01/2017
23/01/2017
30/01/2017
06/02/2017
13/02/2017
20/02/2017
27/02/2017
06/03/2017
13/03/2017
20/03/2017
27/03/20
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P