Hybrid Methods for Finite Element Meshing

Submitted April 2017, in partial fulfilment of the conditions of the award of the degree BSc (Hons) 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: 24/04/2017

Abstract

Background to Finite Element Analysis (FEA)
In simple terms Finite Element Analysis (FEA) is a technique that takes a structure and divides it into a number of small segments known as elements, the collection of elements form a mesh or more precisely a Finite Element Mesh (FEM). FEA then calculates a parameter (in this case stress) on each of the elements in the FEM from which it's possible to identify where failures are most likely to occur. To calculate high stress accurately and in as little time as possible it's necessary to create additional smaller elements in the areas most likely to experience high stress, this process is known as refinement. Traditionally those elements initially experiencing high stress are chosen for refinement then the stress calculations are repeated on the new FEM and process is repeated until the values of stress in each element cease to increase.

Objective of the project
The objective of the project was to develop a new approach to refining a Finite Element Mesh (FEM) which combines the traditional stress based approach, described above, with a machine learning approach based on a set of established engineering rules. The success of the project will be determined by whether the new hybrid approach delivers the same accuracy as the traditional approach but also by whether it's able to achieve this accuracy in less time than the traditional approach.

The Approach
Experienced engineers understand where high levels of stress and failures are most likely to occur in a structure, for example along edges where two surfaces are joined together. This information can be described in a set of user defined edge specifications. The proposed approach was to undertake the FEM refinement process in two stages, first applying the traditional stress based approach to a mesh and then further refining elements based on edge specifications. The result is a new mesh with additional elements created in areas where high stresses are most likely to occur. Having generated a new FEM using the combined approach it's necessary to check the quality of each element to ensure its shape has not become distorted before ordering the data to allow re-calculation of stresses. Finally, the stress data is output in a format for analysis and presentation after all the planned iterations of refinement are complete.

The Solution
The project used commercially available software (LISA) to calculate the stress in each element of the FEM. All other functions were implemented using bespoke software written as part of the project. In summary the bespoke software performed the following functions:

- Applied edge rules to a mesh to identify elements requiring refinement.
- Performed all mesh refinement (for both approaches)
- Arranged newly formed elements into an order acceptable to LISA.
- Performed quality checks on refined elements e.g. how distorted they were.
- Compared the results of various combinations of approaches through graphs

The solution was built using a modular and open architecture to allow for uncertainties during development and to accommodate future enhancements; all code was written using C#.

Evaluation of Findings
The solution was used to evaluate the effects of applying forces to three separate models, the analysis identified where stresses appeared on each model and how quickly they could be identified using each of the following approaches:

- FEA using a stress based refinement.
- FEA using a heuristic (rule) based refinement.
- FEA after using a hybrid approach i.e. a combination of stress and heuristic (rule) based refinement.

For each approach the project collected data on the following parameters:

- The time it took to undertake a defined number iterations of mesh refinement.

- The quality of the elements (how distorted they were).
- The number of elements generated by the approach.
- The level of stress revealed by the approach.

The collected data was plotted to allow an easy comparison of each of the approaches.

Conclusion
From the analysis undertaken it can be concluded that the hybrid approach located high stress points as accurately as the traditional stress based approach, these findings where consistent across all three models used. However, it was not possible to demonstrate conclusively whether the hybrid approach achieved the required results in less time than the stress based approach, this in part was due to having limited time available to test the approach on more complex models where run times are measured in hours rather than the tens of seconds which were the times taken for the most complex model used in this project. The developed system worked well and provides a valuable research tool for further work using more complex models of the type found in industry. The system allows further development of edge specifications without the need for additional coding and the ability to add additional refinement strategies with minimal effort as a consequence of a well defined data model and programming interface.

Contents

1 Introduction

The overall aim of this project is to perform the task of refining a Finite Element Mesh (FEM), (see section 2.1 for description of FEM) using a hybrid approach which combines stress based method as is traditionally used for FEM refinement with an approach that uses techniques from Artificial Intelligence and Machine Learning.

Finite Element Analysis (FEA) is a method widely used across different engineering domains to simulate structures under certain conditions. A typical step within 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 it is expected to perform under. Having defined the components geometry in continuous terms using CAD software a Finite Element System will break it down into a number of segments (known as elements) which together form a mesh (known as a Finite Element Mesh (FEM). The values of stress within each of the discrete elements is calculated and then the process of breaking the mesh down into further smaller elements in areas where high levels of stress are encountered, the process is then repeated in an iterative manner until the levels of stress cease to increase or after a defined number of iterations. FEA 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 of a quality comparable to that of a successful stress based method. Additionally there is benefit in achieving this quality in less time than the traditional stress based approach.

This Dissertation starts with a review of the FEA process followed by looking at some of approaches that have been taken to refine and evaluate the quality of meshes automatically 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

During a year long industrial placement at a major aerospace company I worked closely with mechanical engineers, who routinely ran computer programs to calculate the stresses through physical components used in aeronautical designs. The programs used Finite Element Analysis (FEA) to accurately calculate the stresses across the components. However, applying FEA to detailed structures of considerable size can take many hours in execution time to obtain a result. I discovered that it was not uncommon for experienced engineers to correctly predict the regions of high stress revealed by the analysis prior to its completion. This led me to consider how the knowledge of an experienced engineer could be used by an FEA process to reduce the time required to calculate the stress for mechanical components and was consequently the inspiration for the project.

Over the past forty years FEA has emerged as a prominent technology for simulating complex real world engineering problems [15, 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, 20]. 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, 35].

2.1 Properties of Finite Element Models

Although this is a computer science dissertation rather than a mechanical engineering dissertation it is still important to briefly outline the general principals and properties that underpin the FEA 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 (known as a Finite Element Mesh or FEM). 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 [20].

Elements within FE models can be different types each defined by 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 B shows the different element types supported by the FE solver (used to calculate stresses in elements) used for this project which is called 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 [20]. For this project attention is 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. Static structural analysis is one of the most common engineering applications of FEA and has a large body of research that is relevant to this project. [4][20].

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 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 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 [27].

- Poissons ratio - Amount of deformation that occurs perpendicular to the force that is applied to the material [12].

For the sake of simplicity all structures used to evaluate the final software solution have assumed steel as their material property. Steel is a 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 it takes for the solver to perform an analysis of the model. A major variable determining this trade-off is the model's mesh structure which discretizes the problem so that a 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 [10].

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 [35]. 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.

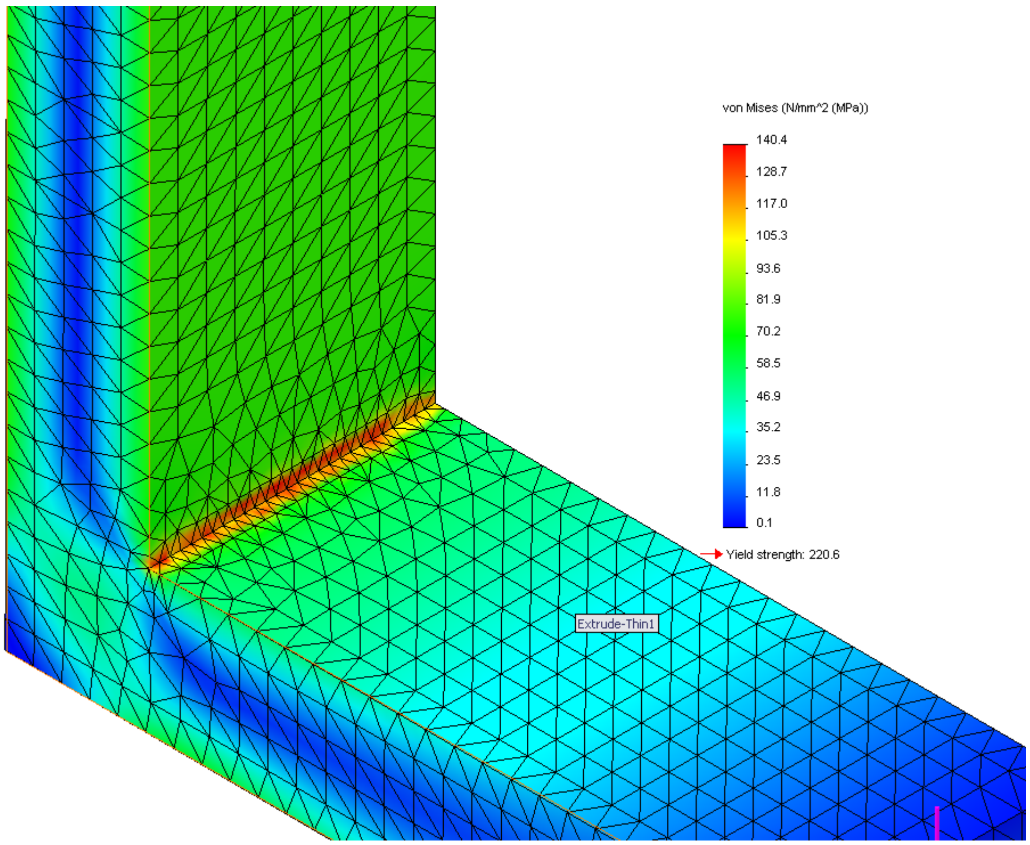


Figure 1: Mesh refinement in corner under high stress image source: ([32])

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 refinement also known as h-refinement and relocation refinement or r-refinement[14] [16]

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 triangular and quadrilateral shapes [14].

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 [16].

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 [31].

3.3 Uses of Artificial Intelligence and Machine Learning

Within the domains of AI and machine learning methods such as neural networks [13], case based reasoning [9] and inductive logic programming [4] have been adopted to facilitate the 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 types of 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 FEA 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" [19]. 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 [18]. 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 [18].

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 easy to obtain making ILP a highly practical solution. Along with his publication Muggleton also released his implementation of an ILP algorithm as a program titled “Golem" [17], 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, edges can be classified based on their significance to the design of the component. Dolsak recognises that edges act as an important intersections within the model and as such provide useful items of reference when designing heuristics with which to reason about the model [4] [5]. For example through use of ILP it was discovered that if it is know that an edge has a force applied close to it based on the initial model conditions then it is preferable for edges that intersect it to 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, 13] Metrics play an important role in assessing a mesh refinement process since the quality of a refinement process depends on its ability to refine the mesh without a deterioration in the quality of the mesh.

Metrics for assessing mesh quality 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 metrics exist for assessing a mesh on a global level such as element count score [2] 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 [20] 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 below in figure 2.

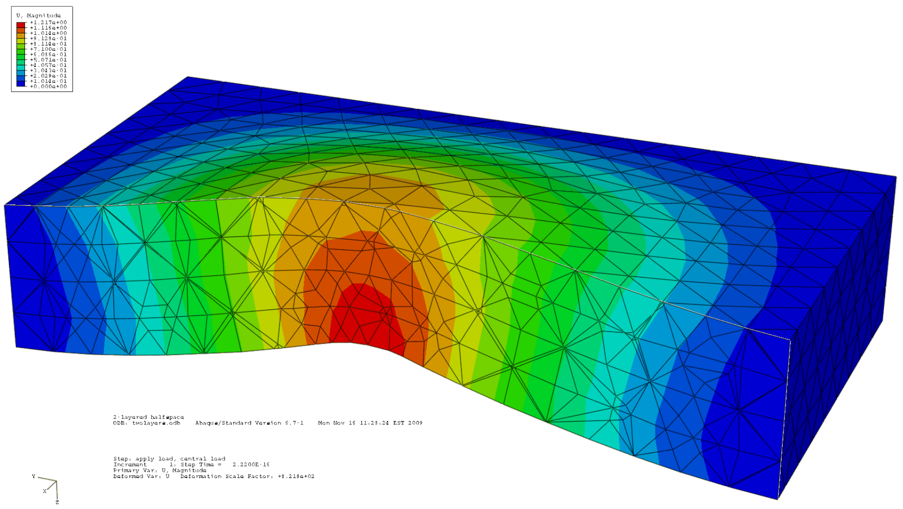


Figure 2: Example of how elements can be distorted in order to fit a geometry which will result in deterioration of gradient quality (image source: [21])

Some key shape metrics identified by Dittmer et al include (ideal values are for elements of quadrilateral type as used in the current prototypes), figure 3 shows how elements can be deformed when they do not score highly using these metrics:

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°

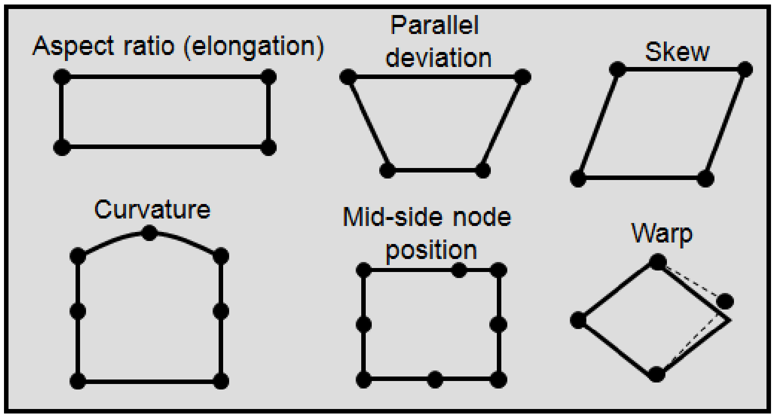


Figure 3: Types of distortion that can occur within elements that can affect accuracy of the results (image source: [8])

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 finite element 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.

An outline of the initial project plan along with a corresponding Gantt chart can be seen in appendix M.

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. Appendix A 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.

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 refine, evaluate and interface with the finite element solver (see section 5.4 on LISA, the selected solver), It was clear however that the overriding feature that most influenced the design was the iterative approach required in order to repeatedly refine and analyse the mesh. Within the main process loop the key focus was to ensure that the design remained modular with well defined relationships between classes so that the methods for refining and analysing the mesh could easily be added or modified independently of one another as the project progressed.

The system can be broken down into the following main tasks that form the overall process to be undertaken:

Generate Initial Model: Using an initial input file generated by LISA or other CAD package which contains the initial FEM model.

Solve Initial Mesh: To initially calculate the stress in each element of the FEM model (i.e. to solve it) the .liml file is provided to LISA before being input into the developed software so that refinement can occur.

Read Solver Output: Read the stress data computed across the mesh from the output file received from LISA and build a class model which can easily be manipulated by the refinement strategies.

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

Quality checks on Refined Mesh : Check the quality of the elements generated by refinement and prepare a new .liml file for next iteration of stress calculation.

The relationships between these stages of the process are further illustrated in figure 4 below which shows the overall solution.

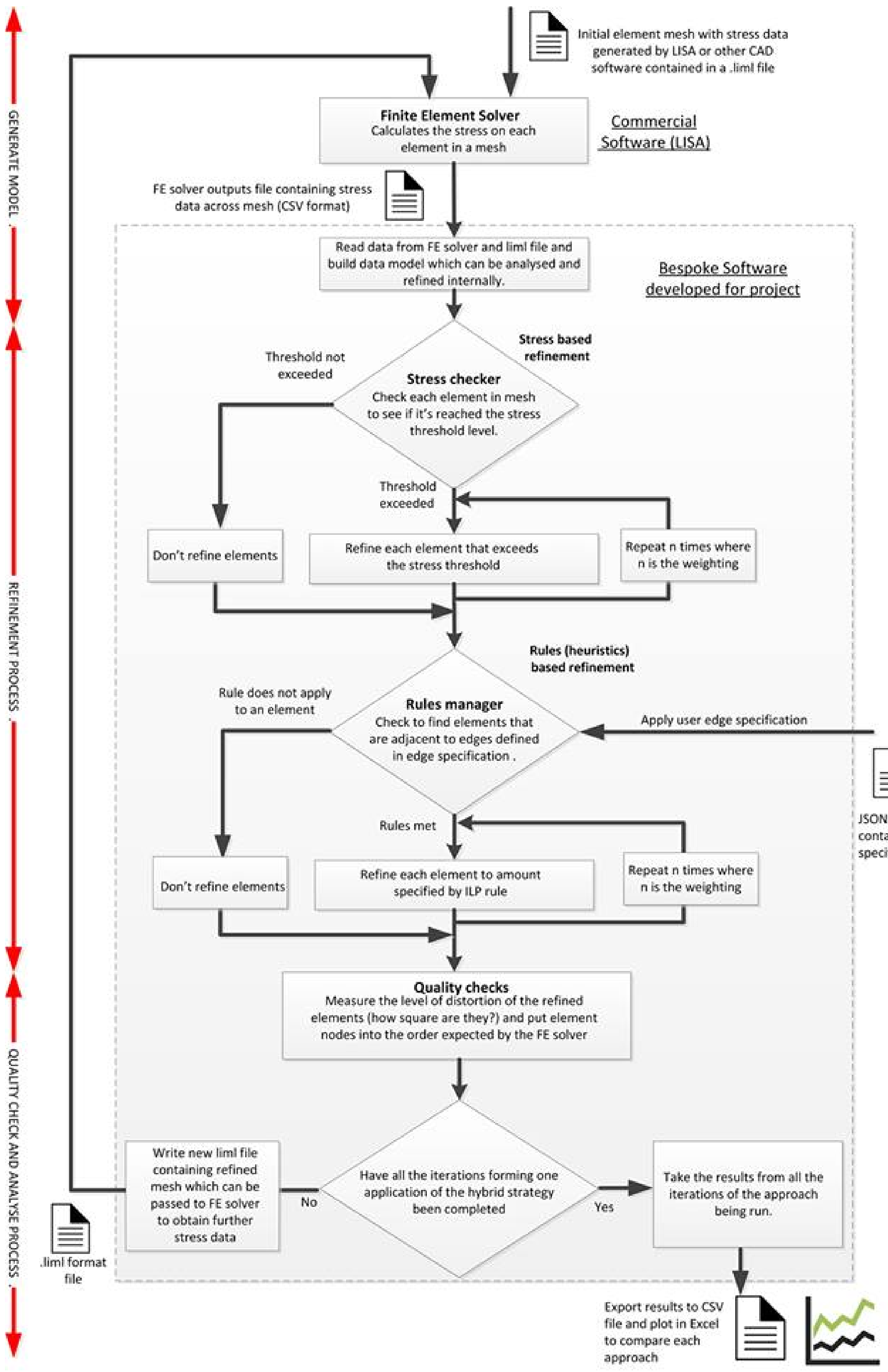

Initial element mesh with stress data generated by LISA or other CAD software contained in a .liml file
Finite Element Solver
Calculates the stress on each element in a mesh
Commercial Software (LISA)
FE solver outputs file containing stress data across mesh (CSV format)
Read data from FE solver and liml file and build data model which can be analysed and refined internally.
Bespoke Software developed for project
Stress based refinement
Threshold not exceeded
Stress checker
Check each element in mesh to see if it's reached the stress threshold level.
Threshold exceeded
Don't refine elements
Refine each element that exceeds the stress threshold
Repeat n times where n is the weighting
Rules (heuristics) based refinement
Rule does not apply to an element
Rules manager
Check to find elements that are adjacent to edges defined in edge specification .
Apply user edge specification
JSON file contains edge specification
Rules met
Don't refine elements
Refine each element to amount specified by ILP rule
Repeat n times where n is the weighting
Quality checks
Measure the level of distortion of the refined elements (how square are they?) and put element nodes into the order expected by the FE solver
Write new liml file containing refined mesh which can be passed to FE solver to obtain further stress data
No
Have all the iterations forming one application of the hybrid strategy been completed
Yes
Take the results from all the iterations of the approach being run.
.liml format file
Export results to CSV file and plot in Excel to compare each approach
GENERATE MODEL .
REFINEMENT PROCESS .
QUALITY CHECK AND ANALYSE PROCESS .

Figure 4: High level Design of system/ refinement process

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 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 is highly modular It is 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 not 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 F) 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 important to first 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 a wide range in both complexity and cost across all of them them. Finite element software is typically very expensive however due to it having both a high development cost due to its complexity along with a relatively small and specialist user 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 [34]. 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 the full version for 299.99 Canadian dollars or £174.07. This purchase was only made however once at a stage in the project where each sub problem had been solved for small models containing less than 1300 elements and further work required testing with larger models.

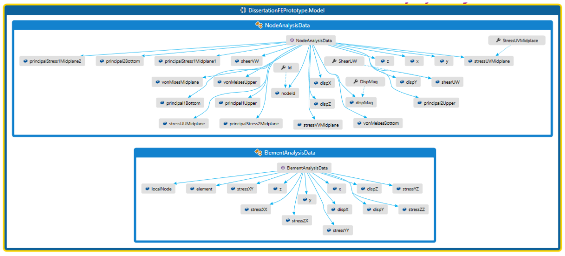
LISA also provides a GUI which allows visual inspection of the model and its mesh; this is particularly useful for observing the output of the meshing algorithms which can often provide a human with a much better understanding of how the method has performed and whether or not there are obvious bugs. 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 [26]. 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 and manipulate its content.

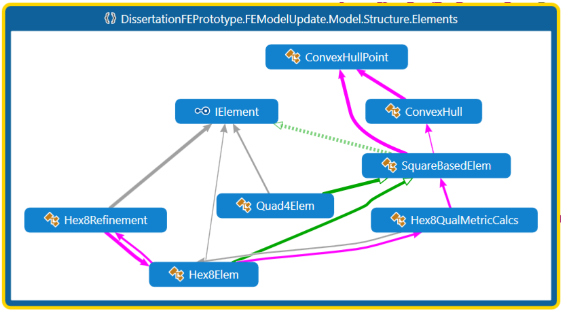
5.5 Simulation Data Model

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 to avoid having to regularly perform direct string manipulation of the input files when manipulating the model.

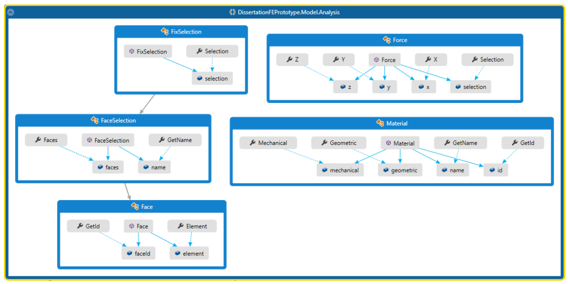
When the first re-meshing iteration occurs the system needs to read the input .liml file into an equivalent class model which closely resembles the files schema. The model also contains the stress data from the previous iteration of the model. Diagrams for the model can be seen in figure 5 below. Each class in this model contains corresponding data and methods used to represent and manipulate the FE 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 refined 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.



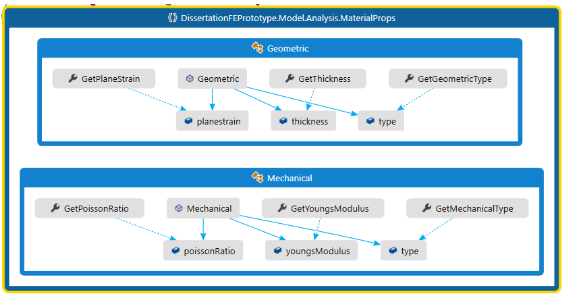
(a) Model classes



(b) Element Classes



(c) Model Analysis classes



(d) Material Property classes

Figure 5: 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. 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 types shown in appendix B.

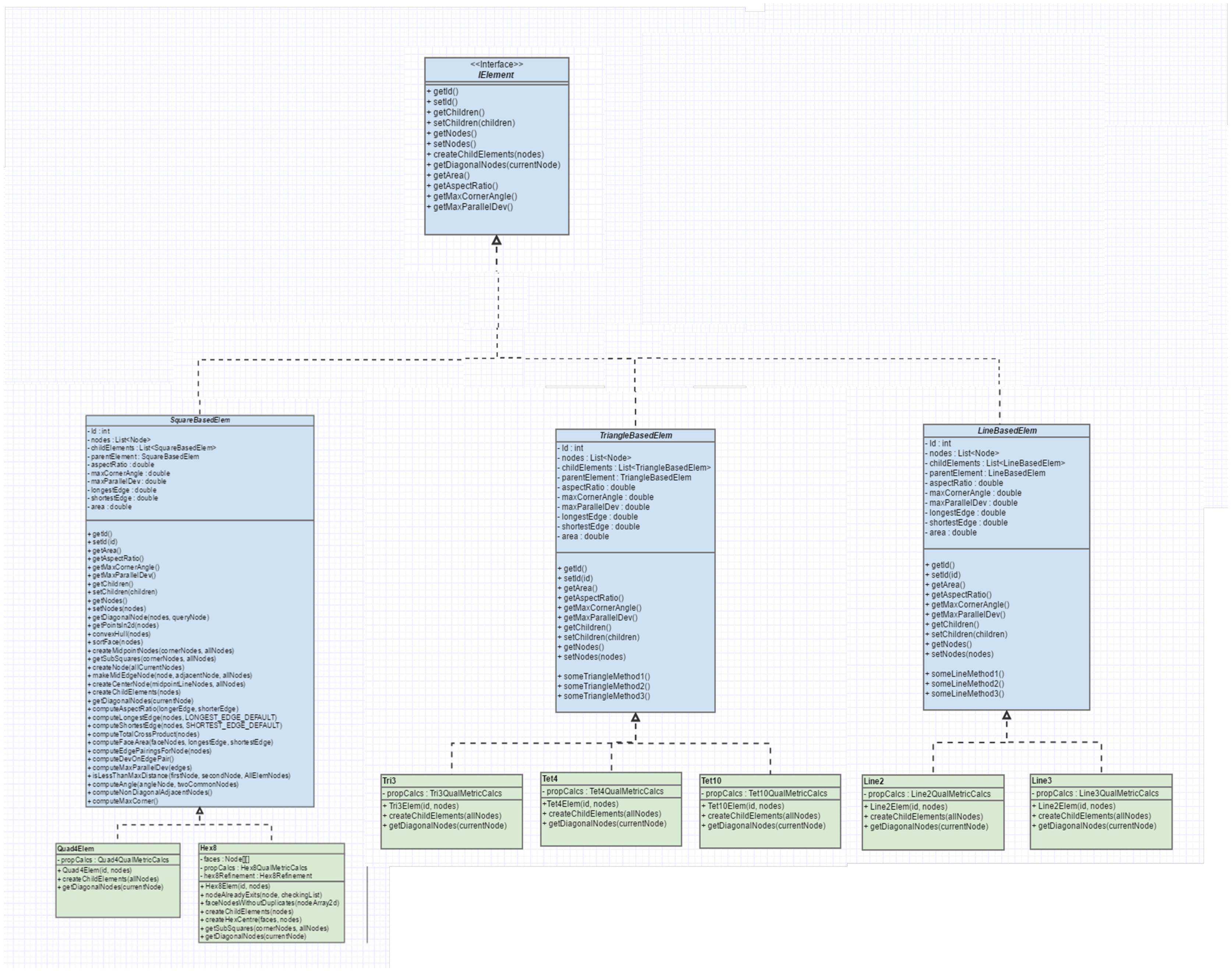

<<Interface>>
IElement
+ getId()
+ setId()
+ getChildren()
+ setChildren(children)
+ getNodes()
+ setNodes()
+ createChildElements(nodes)
+ getDiagonalNodes(currentNode)
+ getArea()
+ getAspectRatio()
+ getMaxCornerAngle()
+ getMaxParallelDev()
SquareBasedElem
- Id : int
- nodes : List<Node>
- childElements : List<SquareBasedElem>
- parentElement : SquareBasedElem
- aspectRatio : double
- maxCornerAngle : double
- maxParallelDev : double
- longestEdge : double
- shortestEdge : double
- area : double
+ getId()
+ setId(id)
+ getArea()
+ getAspectRatio()
+ getMaxCornerAngle()
+ getMaxParallelDev()
+ getChildren()
+ setChildren(children)
+ getNodes()
+ setNodes(nodes)
+ getDiagonalNode(nodes, queryNode)
+ getPointsIn2d(nodes)
+ convexHull(nodes)
+ sortFace(nodes)
+ createMidpointNodes(cornerNodes, allNodes)
+ getSubSquares(cornerNodes, allNodes)
+ createNode(allCurrentNodes)
+ makeMidEdgeNode(node, adjacentNode, allNodes)
+ createCenterNode(midpointLineNodes, allNodes)
+ createChildElements(nodes)
+ getDiagonalNodes(currentNode)
+ computeAspectRatio(longerEdge, shorterEdge)
+ computeLongestEdge(nodes, LONGEST_EDGE_DEFAULT)
+ computeShortestEdge(nodes, SHORTEST_EDGE_DEFAULT)
+ computeTotalCrossProduct(nodes)
+ computeFaceArea(faceNodes, longestEdge, shortestEdge)
+ computeEdgePairingsForNode(nodes)
+ computeDevOnEdgePair()
+ computeMaxParallelDev(edges)
+ isLessThanMaxDistance(firstNode, secondNode, AllElemNodes)
+ computeAngle(angleNode, twoCommonNodes)
+ computeNonDiagonalAdjacentNodes()
+ computeMaxCorner()
TriangleBasedElem
- Id : int
- nodes : List<Node>
- childElements : List<TriangleBasedElem>
- parentElement : TriangleBasedElem
- aspectRatio : double
- maxCornerAngle : double
- maxParallelDev : double
- longestEdge : double
- shortestEdge : double
- area : double
+ getId()
+ setId(id)
+ getArea()
+ getAspectRatio()
+ getMaxCornerAngle()
+ getMaxParallelDev()
+ getChildren()
+ setChildren(children)
+ getNodes()
+ setNodes(nodes)
+ someTriangleMethod1()
+ someTriangleMethod2()
+ someTriangleMethod3()
LineBasedElem
- Id : int
- nodes : List<Node>
- childElements : List<LineBasedElem>
- parentElement : LineBasedElem
- aspectRatio : double
- maxCornerAngle : double
- maxParallelDev : double
- longestEdge : double
- shortestEdge : double
- area : double
+ getId()
+ setId(id)
+ getArea()
+ getAspectRatio()
+ getMaxCornerAngle()
+ getMaxParallelDev()
+ getChildren()
+ setChildren(children)
+ getNodes()
+ setNodes(nodes)
+ someLineMethod1()
+ someLineMethod2()
+ someLineMethod3()
Quad4Elem
- propCalcs : Quad4QualMetricCalcs
+ Quad4Elem(id, nodes)
+ createChildElements(allNodes)
+ getDiagonalNodes(currentNode)
Hex8
- faces : Node[][]
- propCalcs : Hex8QualMetricCalcs
- hex8Refinement : Hex8Refinement
+ Hex8Elem(id, nodes)
+ nodeAlreadyExits(node, checkingList)
+ faceNodesWithoutDuplicates(nodeArray2d)
+ createChildElements(nodes)
+ createHexCentre(faces, nodes)
+ getSubSquares(cornerNodes, allNodes)
+ getDiagonalNodes(currentNode)
Tri3
- propCalcs : Tri3QualMetricCalcs
+ Tri3Elem(id, nodes)
+ createChildElements(allNodes)
+ getDiagonalNodes(currentNode)
Tet4
- propCalcs : Tet4QualMetricCalcs
+Tet4Elem(id, nodes)
+ createChildElements(allNodes)
+ getDiagonalNodes(currentNode)
Tet10
- propCalcs : Tet10QualMetricCalcs
+ Tet10Elem(id, nodes)
+ createChildElements(allNodes)
+ getDiagonalNodes(currentNode)
Line2
- propCalcs : Line2QualMetricCalcs
+ Line2Elem(id, nodes)
+ createChildElements(allNodes)
+ getDiagonalNodes(currentNode)
Line3
- propCalcs : Line3QualMetricCalcs
+ Line2Elem(id, nodes)
+ createChildElements(allNodes)
+ getDiagonalNodes(currentNode)

Figure 6: Class diagram showing the hierarchy of element classification within the data model

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 new meshing procedures that may be added later. An FE mesh refinement system can be thought of as performing two distinct but related tasks:

Element Refinement: How are elements going to be refined when it is known where it's best to refine within the mesh?

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

Both h and r refinement (described in section 3.1) 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 of knowing the level of stress. Having computed the stress this also acts as the means by which to assess the quality of the strategy, that is being assessed by 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 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 B for element types.

The Design solution relating to this aspect of 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 chosen area. Using the interface the strategy can refine all elements it has selected as being beneficial for refinement by simply calling the elements createChildElements() method through the interface. This allowed any strategy to refine any type of element.

Selecting a Subdivision Approach: Having reviewed both h-refinement [14] and r-refinement [16] 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 [14, 16]. 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 complicates an otherwise simple process of delegating 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 there are lots of elements in order to further refine that area.

After deciding 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 7.

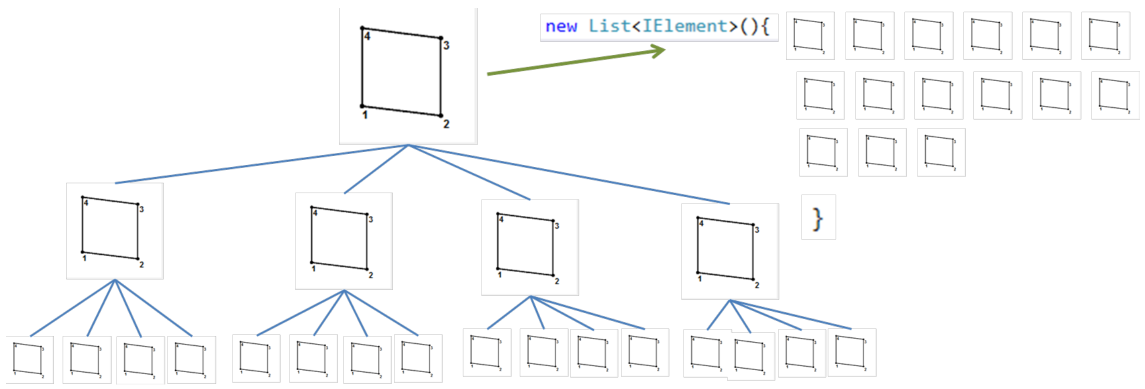

```
new List<IElement>(){
}
```

Figure 7: Process of flattening a refined element tree into a single list which can be handed back to LISA for processing

Stress Refinement: Designing a stress based refinement process involved three main steps, the specifics of which for my implementation are described under section 6.3. Firstly data needs to be obtained about the model stresses from the previous iteration, this is generated through execution of the mesh that is about to be refined. Once stress data is obtained along with a reference to the mesh it is possible to refine the mesh by using some evaluation function which takes both the mesh and the stress data and for every element in the mesh determines whether or not that element should be refined. selecting the threshold used by the evaluation function is perhaps the most important aspect of the design since since a high threshold, figure 8 shows stress refinement being applied across a simple four element mesh where the threshold is set as 7

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 it was realised that given the inherent complexity of the problem of developing a sophisticated AI method was going to be too time consuming. Dolsaks expert system [3, 4], again reviewed in section 3.3 was therefore selected. Unlike some of the other approaches the literature also offered a straightforward implementation with clearly defined concepts and rules that directly utilise the data already available within the LISA models. It's also important to note that the Dolsak used a machine learning technique and has published results that support the claims made of its capability, [3, 4, 5]. Use of this method as the second part of a hybrid approach ensures that the strengths of AI are Incorporated into the overall approach.

Unlike the stress refinement process which meshes purely based upon properties present in the current state of the model. The heuristic refinement process combines two sources of independent information when determining where best to mesh, these are the derived ILP rules and user defined edges.

Rules: The rules produced by Golem, see section 3.3 are derived from training the ILP method using a series of meshes that were provided by by Dolsak [3, 5]. The rules are designed to take edges within the model as input and can decide how much meshing should be performed around those edges based on experience embedded within the rules by Golem through analysis of the training meshes. Ten of these rules which are provided within the research literature [3, 5] have been programmed into the system inside the "RuleManager" class within my system.

Edges: Edges are the data used by the rules and are defined by engineers using the system. These engineers are likely to have some background knowledge of the component they are designing and so have the ability to assign some meta data to the edges, see section 6.4 and appendix E.

Having been provided with a set of rules which can be applied to sets of edges it was important to design a mapping between the concept of an edge as described by Dolsak and the data model representing the FE mesh so the rules could drive the refinement. It was possible to approximate the concept of a smooth edge as a series of discrete node pairs within the mesh in order to form a path as a chain of nodes. This approach allows edges to be arbitrarily specified within the mesh structure using as much detail as was possible given the mesh's fidelity. Using this approach for defining edges in terms of the FE model meant that elements running along an edge that refinement needed to be made for could quickly be accessed by cross referencing the node Ids along the edge with those forming particular elements.

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 to Form a Hybrid Approach

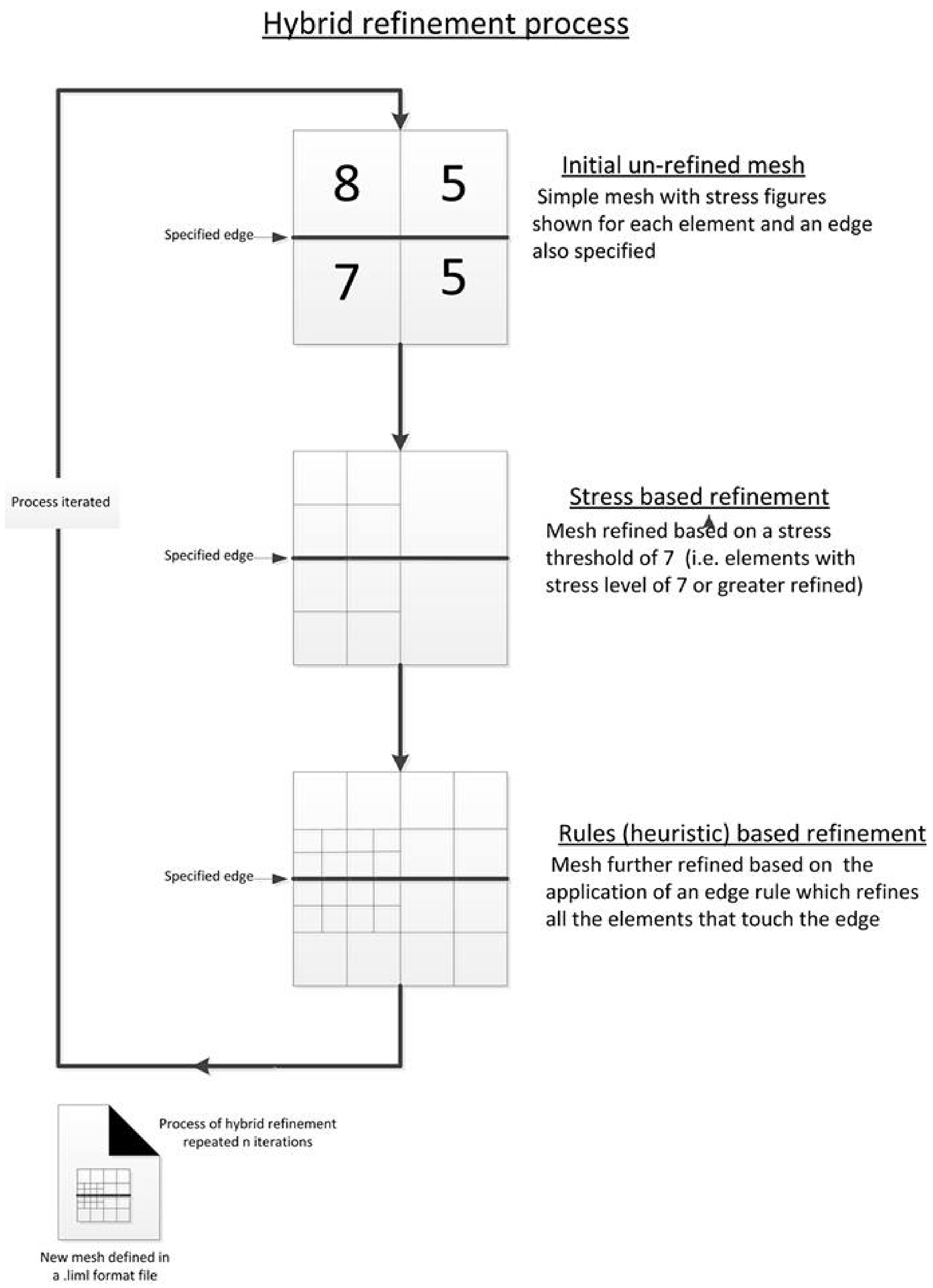
Since each refinement method performed a discrete amount of subdivision every time it was called it made sense when developing a hybrid approach based on the two methods to define each potential hybrid as some weighted combination of the two methods. The simplest way to represent this appeared to be 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 the trees of those elements that are affected by the rule as illustrated above 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. With the average increase in element count per iteration being calculated as 6% of the model total in the case of the best sets of edge specifications (see section 7.6) 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.

An example of how both the stress and rule based refinement methods can be applied to the same mesh for a single solve iteration can be seen below in figure 8. Visual examples of how each of the meshing procedures affects the mesh can be seen in appendix J. Further detailed design details can be found in the section 6 covering the Software implementation.



Hybrid refinement process
8
5
7
5
Specified edge
Initial un-refined mesh
Simple mesh with stress figures shown for each element and an edge also specified
Process iterated
Specified edge
Stress based refinement
Mesh refined based on a stress threshold of 7 (i.e. elements with stress level of 7 or greater refined)
Specified edge
Rules (heuristic) based refinement
Mesh further refined based on the application of an edge rule which refines all the elements that touch the edge
Process of hybrid refinement repeated n iterations
New mesh defined in a .liml format file

Figure 8: Applying both the stress and rule based refinement as part of one overall refinement iteration for a hybrid

6 Software Implementation

The following subsections detail the implementation of the final software solution that has been written to meet the objectives of the project. The final software implementation consists of 35 classes containing approximately 6000 lines of code. Implementation began early in the project with the basic data model and external LISA interfaces being the first tasks to be completed before focus was subsequently directed towards the more challenging aspects such as the refinement processes.

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 operations 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 are known and 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 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. This process is shown in figure 10 where elements along edge A are meshed three times.

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:

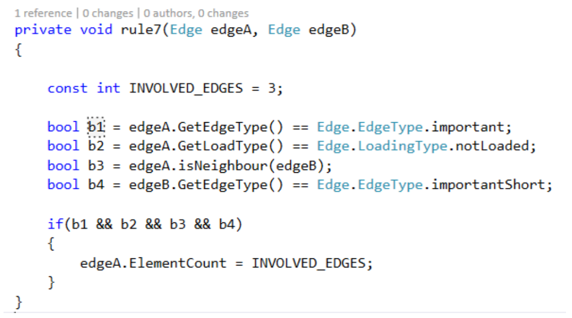
- Id number - used to identify edge uniquely within the RuleManager
- Edge type - How would the engineer describe the edge, does it form a circuit, is it an important aspect of the models design or is it along the edge of some hole?
- Load type - Is the edge between two areas with forces applied to them, 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 represented as enumerable type within the system as defined by Dolsak within his paper.The accepted values for these enumerable types have been listed in appendix D. In addition to these properties being used by the refinement system for deciding where to mesh. Figure 9 below shows a representation of an ILP rule as described by Dolsak with an equivalent implementation which checks the edge relationships using the enumerable types.

Since the 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.

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

(a) Rule 7 as stated by dolsak in his papers [5]



```
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;
    }
}
```

(b) Code implementation of rule 7 provided by Dolsak within the RuleManger class

Figure 9: Example of rule implementation within software implementation

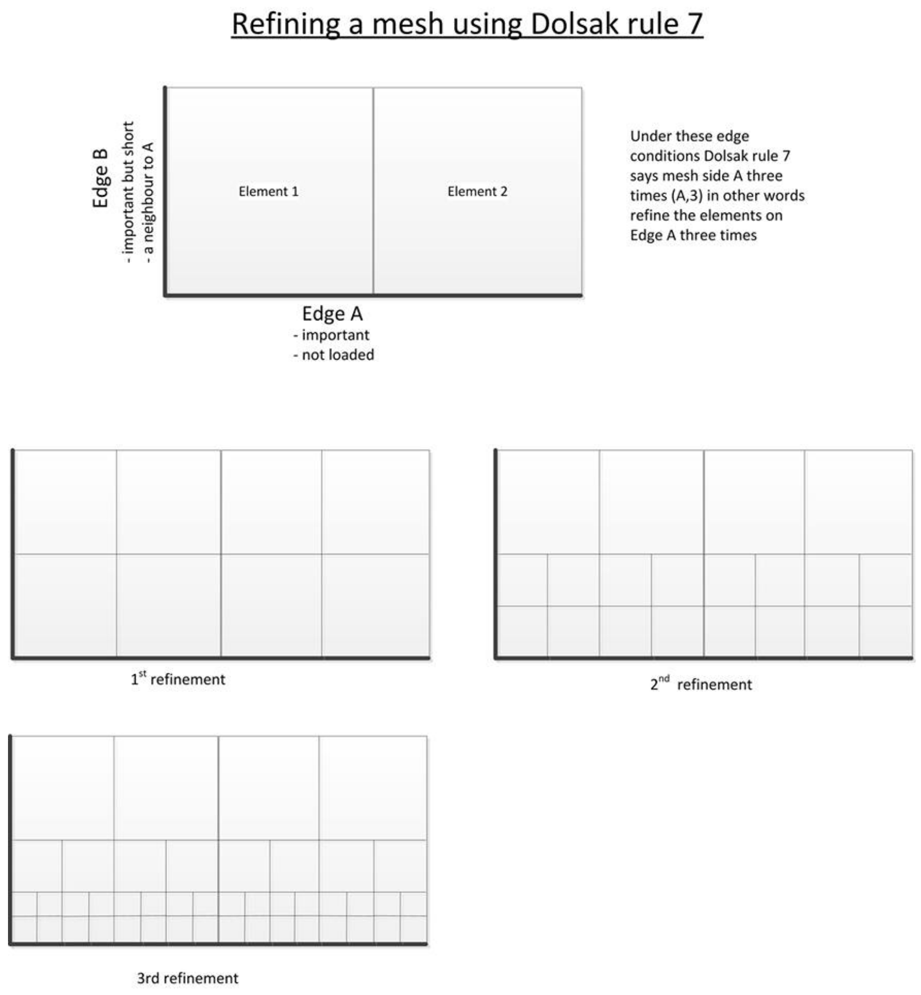

Refining a mesh using Dolsak rule 7
Edge B
- important but short
- a neighbour to A
Element 1
Element 2
Under these edge conditions Dolsak rule 7 says mesh side A three times (A,3) in other words refine the elements on Edge A three times
Edge A
- important
- not loaded
1st refinement
2nd refinement
3rd refinement

Figure 10: The process of refining elements along an edge within the mesh through application of an ILP generated rule

Having refined a targeted area using the heuristics edges needed to be redefined 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 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. An edge is therefore redefined in terms of the nodes that form the closest to the shortest distance path between each of the node pairs.

Several factors influenced the choice to adopt a basic a greedy search algorithm for solving this problem. Firstly there was no need for a completely optimal solution with time being a more important consideration due to repeated application of the method for multiple edges after each solve iteration. Additionally the problem is highly localised to the area of the mesh containing the particular edge allowing the rest of the mesh to be excluded from the search.

The greedy search starts by computing the distance between the start and end node points for all nearby nodes. The method then determines the nodes directly adjacent to the start and then of those nodes which 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 without creating noticeable increase in runtime. A more detailed pseudo code description of this process can be seen below:

```
forall $E \in refinedEdges$ 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 $edgeNodePairsN \in E$ 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 $nearbyElemNode \in K$ 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 $n2Node \in n2$ do
                if $n2Node.Ref \in F$ 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 1: 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 primary evaluation on was one which described the maximum corner angles, this 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 section 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.

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 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 11.

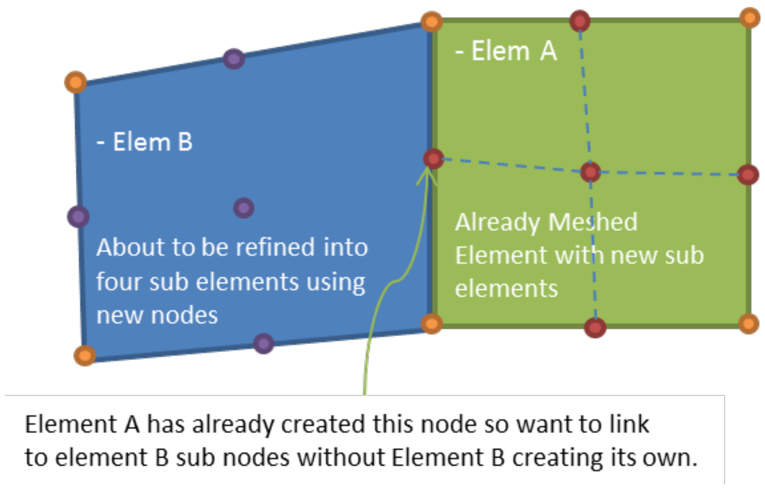

- Elem A
- Elem B
About to be refined into four sub elements using new nodes
Already Meshed Element with new sub elements
Element A has already created this node so want to link to element B sub nodes without Element B creating its own.

Figure 11: The need for an element to check for existing adjacent nodes when subdividing itself during refinement,

Orange Nodes - An original node for one or more elements
Red Nodes - new nodes made by Elem A
Purple Nodes - new nodes made by Elem B

The solution to this problem was to store all the nodes in the mesh model within a C# dictionary structure a reference to which is passed to each element within the model. The dictionary can be indexed using a Tuple of the x, y and z coordinates for the new potential element which will either return a node already at that location or indicate that no such node exists, in which case that element is then responsible for creating the node as its first instance. Dictionaries in C# represent a generalised instance of a hash table ensuring that lookup and insert are both constant time on average.

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

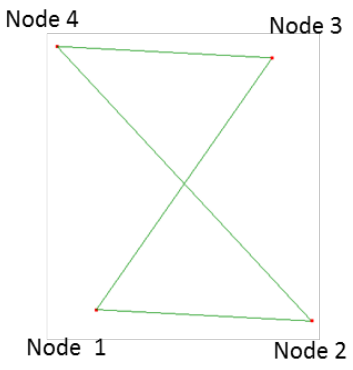


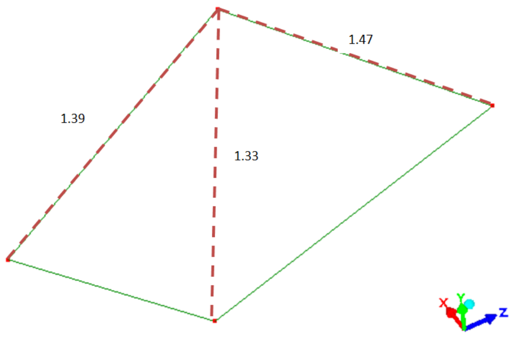
Figure 12: Element with 3D skew resulting in edges between diagonals being shortest by a small amount.

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:

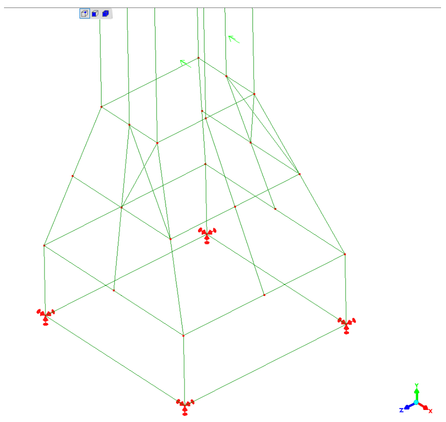
```
make a list to contain sorted nodes called SN **while** $\exists node \in unstortedNodes$ **do**
    Get distance between current node and the next two nodes in unstortedNodes;
    **if** $sortedNodes.Count == 1$ **then**
        Calculate the distance between the origin node and the penultimately 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 2: 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 13 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 elements.



(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 13: Incorrectly sorted elements arising from failure of traversal routine for skewed elements

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 alternatives 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 [23] 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 shown below in figure 14. The approach is directly applicable in the case of quad4 elements where the LISA sort order is the convex hull of the points.

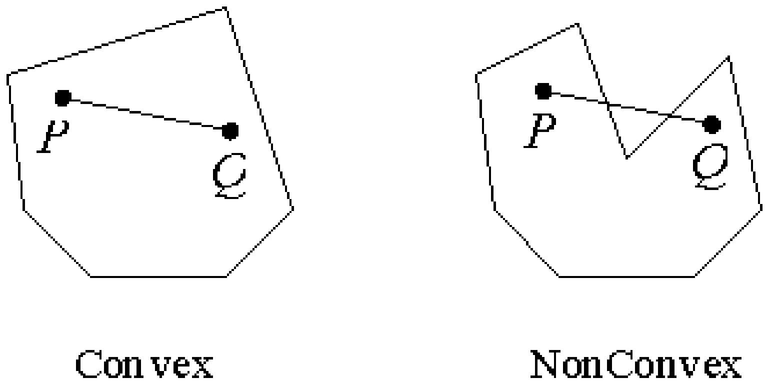


Figure 14: Illustration of convex hull definition, imace source: [23]

After reviewing the different convex hull algorithms including Grapham scan [30] $O(n\ log\ n)$ and brute force scan $O(n^4)$ [23] [29] I decided to trial the following C# implementation of the Monotone Chain algorithm, also $O(n\ log\ n)$ [22].

The Monotone Chain 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 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 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 [24].

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 3: Monotone Chain algorithm for generating convex hull, pseudocode description credit: [33]

The additional complexity of implementing a 3D convex hull algorithm meant it was much easier to experiment with the approach as a potential solution to the problem using a 2D implementation by simply creating a representation of the element within 2D space. This was done 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 higher elements would lie on a perfectly diagonal plane resulting in two axis of elimination using this method. This problem was avoided however by using the basic traversal to sort these elements.

Although these methods appear to work well for 3D elements that can easily be flattened into a 2D representation for sorting this is clearly problematic for more complex elements such as Hex8s and Tet4s, see appendix B. One solution to this which was briefly explored and was used when running the system for evaluation was to split elements using x, y and z planes as described in appendix I.

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 had to be run on a number of basic models with different input configurations. Having done this the results clearly demonstrate the system's ability to evaluate the quality of meshes using multiple refinement processes based on the stresses present within the structure and the specification of edges by a user.

7.2 Validation of Software Quality

In the case of quality for the system's design and documentation evidence is provided in, (see appendices C, G and H) to the requirements specified with the project submission containing detailed documentation in the form of a Deoxygen guide which shows the 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 models and conditions when performing simulations.

7.3 Unit Testing

Holistic evaluation provided evidence of the overall system's effectiveness. Verification of individual components created trust in the accuracy of the results produced. Unit testing was also conducted from within Visual Studio (VS) 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 functionality 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 software was developed using industry best practice including the use of appropriate variable naming, loose coupling of classes, use of abstractions and descriptive error messages which make the software easier to read and debug for any potential future developers. An effective version control strategy was also adopted by backing up all software to Github daily.

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. Upon completion of the project the average maintainability index [28] across all modules was 75 with the lowest score for any high level module being 60 and the highest 92. According to the Microsoft Developer Network (MSDN) website code with an index of between 0 and 9 indicates low maintainability, 10 to 19 indicates moderately maintainable and 20 to 100 high maintainability.

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 of meeting the goals of the projects 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 Hybrids and Individual Methods of Refinement

In order to validate the different methods of refinement it was important to carefully design tests for the system which would fairly measure its ability to compare 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 performs as expected, this is a 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 the systems ability to work for a range of different model inputs.

Hybrid performance evaluation should consider both good and potentially bad user input.
Three models were created in total for evaluation, with each of the models being a general simplification of a more complex model that could be expected within an industrial engineering setting. They model a suspension bridge, a part of a paper mill and a section of a generic cylinder. These models were developed specifically for the project although the paper mill and cylinder were based on two models used by Dolsak in order to train the ILP method when generating the rule set. 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. The results for the two models not analysed in this section (the cylinder and paper mill disk) can be found in appendices K and L.

Different Quality Edge Specifications: Validation of edge specifications to reflect real world conditions was necessary, 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 15 and the mesh structure in appendix J.

Since the three models were very different it was not possible to objectively compare the edge specifications for each of them. a basic criteria was therefore developed so that comparisons could be drawn. For each model four sets 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.

Not being a mechanical engineer it was necessary to find a way of defining what 'Best' and 'Good' looked like. To do this the traditional stress based refinement approach was run on each model to establish where high stress areas exist before using this to define a series of edge specifications of varying quality. Consequently it is only possible to objectively judge each of the different sets on the basis of the results they produce. These results would have had greater validity if time had allowed input from a range of practicing mechanical engineers.

7.6.1 Metrics Selections

Due to the complex nature of finite element models there are a number of potential methods that can be used to evaluate the effectiveness of my approach. 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 is detailed below.

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 ensure accuracy of subsequent results. Figure 3 show types of distortion, such as skewing, which can be detected by measuring the internal corner angle.

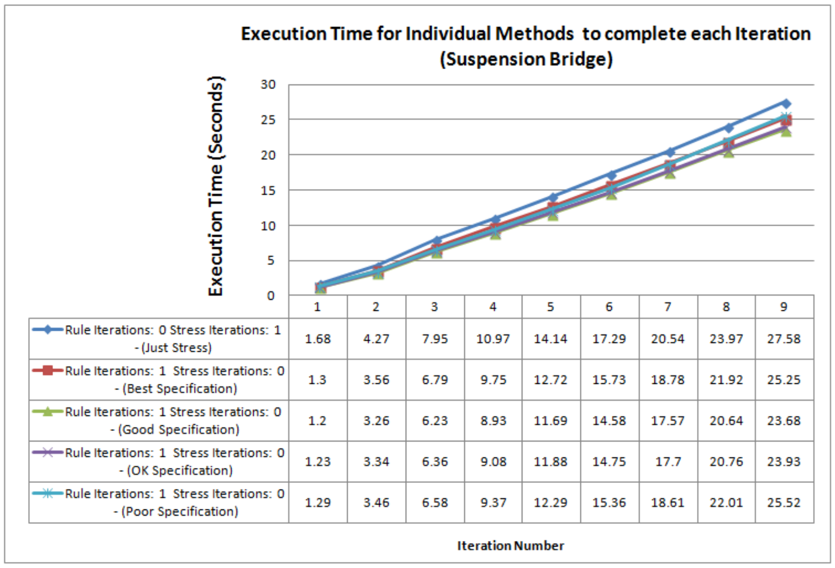
Execution Times: Since it is important for all methods to run in a reasonable amount of time, measuring the increase in runtime with additional iterations provided a good indication of how costly each approach became and whether there were any points at which refined meshing became significantly more expensive. Comparing the execution times for the different approaches also indicates how much work each method is doing per iteration allowing the hybrids to be balanced, see section 5.8.

Average Stress Revealed for Each Iteration: In order to measure the different methods ability to reveal stress over time the average stress across all the elements throughout the model is an effective metric to use. An increase in the average occurs 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. To simplify evaluation so as to take into account the different edge specifications the hybrid data below is based on the average values obtained for the heuristics when running all of the edge sets for the different models.

7.6.2 Evaluation of Bridge Structure

The suspension bridge model (see figure 19) 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 were applied across the structure along the negative x axis.

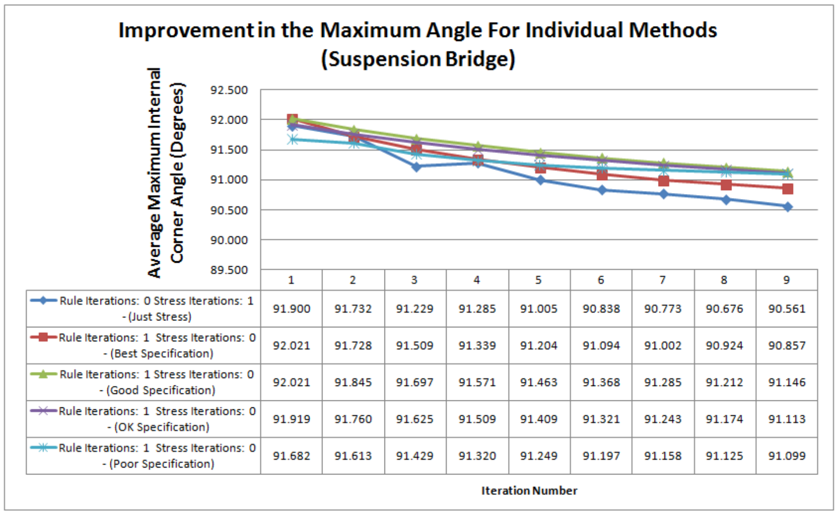
Evaluating Individual Refinement Methods: An important part of validating the proposed hybrid approach was to first evaluate each of the refinement methods individually and observe their performance. This then informs the weighting applied to each method to avoid either method dominating the results of the hybrid approach. To test the individual approaches effectiveness they were evaluated against the performance metrics defined in section 7.6.1 (i.e. time required to refine per iteration and average maximum internal element angle). The results of these validation tests are shown in figures 15 and 16. From these graphs it can be concluded that no individual method reduces the mesh quality and that both the individual methods spend approximately the same amount of time performing refinement per solve iteration meaning neither is overly favoured by a hybrid approach.



	1	2	3	4	5	6	7	8	9
◆ Rule Iterations: 0 Stress Iterations: 1 - (Just Stress)	1.68	4.27	7.95	10.97	14.14	17.29	20.54	23.97	27.58
■ Rule Iterations: 1 Stress Iterations: 0 - (Best Specification)	1.3	3.56	6.79	9.75	12.72	15.73	18.78	21.92	25.25
▲ Rule Iterations: 1 Stress Iterations: 0 - (Good Specification)	1.2	3.26	6.23	8.93	11.69	14.58	17.57	20.64	23.68
× Rule Iterations: 1 Stress Iterations: 0 - (OK Specification)	1.23	3.34	6.36	9.08	11.88	14.75	17.7	20.76	23.93
✳ Rule Iterations: 1 Stress Iterations: 0 - (Poor Specification)	1.29	3.46	6.58	9.37	12.29	15.36	18.61	22.01	25.52

Figure 15: Time taken to complete each iteration using the different methods

The maximum internal corner angles can be seen as improving linearly over time although with 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 retains its accuracy.

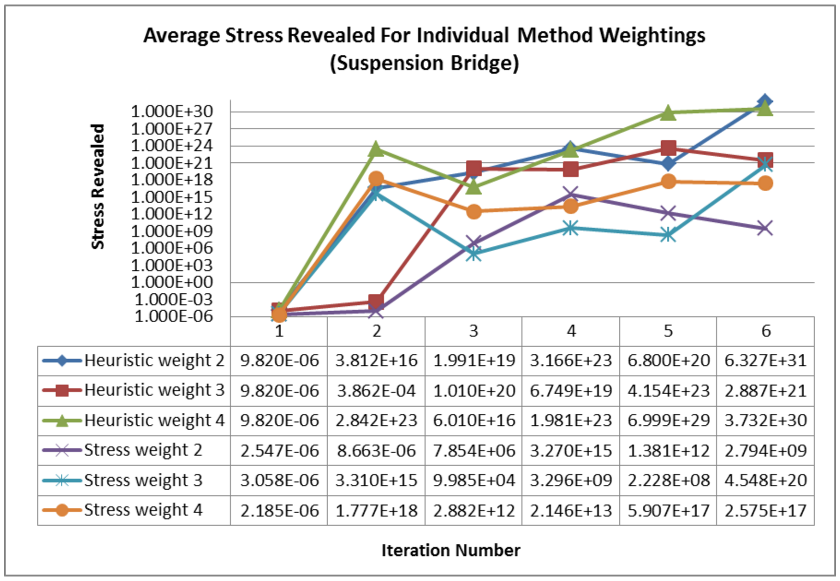


	1	2	3	4	5	6	7	8	9
Rule Iterations: 0 Stress Iterations: 1 - (Just Stress)	91.900	91.732	91.229	91.285	91.005	90.838	90.773	90.676	90.561
Rule Iterations: 1 Stress Iterations: 0 - (Best Specification)	92.021	91.728	91.509	91.339	91.204	91.094	91.002	90.924	90.857
Rule Iterations: 1 Stress Iterations: 0 - (Good Specification)	92.021	91.845	91.697	91.571	91.463	91.368	91.285	91.212	91.146
Rule Iterations: 1 Stress Iterations: 0 - (OK Specification)	91.919	91.760	91.625	91.509	91.409	91.321	91.243	91.174	91.113
Rule Iterations: 1 Stress Iterations: 0 - (Poor Specification)	91.682	91.613	91.429	91.320	91.249	91.197	91.158	91.125	91.099

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

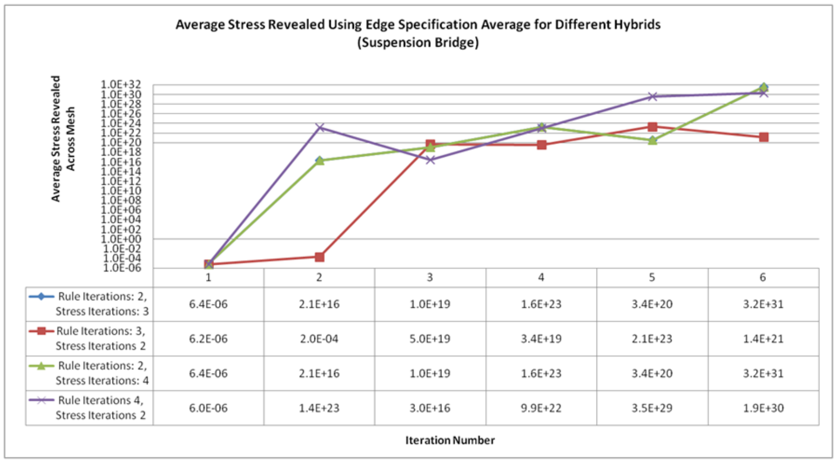
Evaluating Hybrid Refinement Methods: When used on their own the heuristics (rules) based approach to FEM refinement revealed greater stresses than the traditional stress based approach. This is evidenced by figure 17 where after 6 iterations a heuristic approach had revealed stress levels up to the value of 6.3E+31 pascals whilst the traditional stress based approach revealed up to 4.5E+20 pascals. It can be concluded that there is demonstrative value in using a heuristic (rules) based approach regardless of whether used on its own or as part of a hybrid refinement strategy.

Having completed analysis for each of the individual methods tests were run using hybrid strategies for each of the models. This produced results that indicate rapid overall improvement with regards to finding stress as can be seen below in figure 17 below. The improvement can be seen during the first few iterations in figure 18 before achieving a plateau. The accuracy of the results were initially questioned with the values being extremely high, however Re-execution of the model with varying configurations including varying force and alternative constraints resulted in minimal differences. Conducting additional research revealed that this is not in fact an uncommon property of stress gradients within FE models [11] as the calculated values for stress stress will trend towards infinity at points of serious weakness within a design when a high force is exerted and the stress is able to concentrate at particular points. A real structure would not be able to enduring stresses of these magnitudes and would break at these points of stress concentration before the stress calculated by the solver could actually occur. Figure 19 below shows the bridge model undergoing relatively high stress at various points across the model but with rapid increase at specific points where structures join one another (highly focused red patches in 19b).



	1	2	3	4	5	6
Heuristic weight 2	9.820E-06	3.812E+16	1.991E+19	3.166E+23	6.800E+20	6.327E+31
Heuristic weight 3	9.820E-06	3.862E-04	1.010E+20	6.749E+19	4.154E+23	2.887E+21
Heuristic weight 4	9.820E-06	2.842E+23	6.010E+16	1.981E+23	6.999E+29	3.732E+30
Stress weight 2	2.547E-06	8.663E-06	7.854E+06	3.270E+15	1.381E+12	2.794E+09
Stress weight 3	3.058E-06	3.310E+15	9.985E+04	3.296E+09	2.228E+08	4.548E+20
Stress weight 4	2.185E-06	1.777E+18	2.882E+12	2.146E+13	5.907E+17	2.575E+17

Figure 17: Average stress revealed with different weightings for the individual methods



	1	2	3	4	5	6
Rule Iterations: 2, Stress Iterations: 3	6.4E-06	2.1E+16	1.0E+19	1.6E+23	3.4E+20	3.2E+31
Rule Iterations: 3, Stress Iterations 2	6.2E-06	2.0E-04	5.0E+19	3.4E+19	2.1E+23	1.4E+21
Rule Iterations: 2, Stress Iterations: 4	6.4E-06	2.1E+16	1.0E+19	1.6E+23	3.4E+20	3.2E+31
Rule Iterations 4, Stress Iterations 2	6.0E-06	1.4E+23	3.0E+16	9.9E+22	3.5E+29	1.9E+30

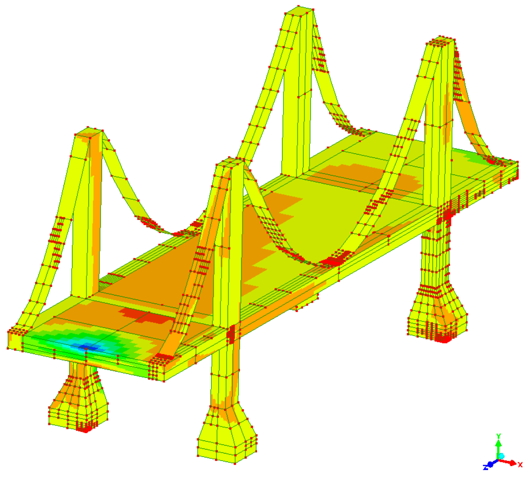
Figure 18: Average stress revealed using multiple iterations of the different hybrid methods over time

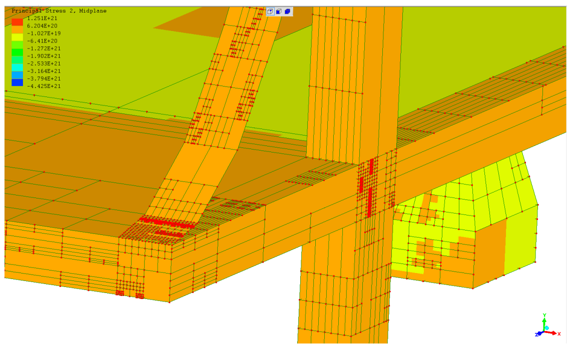
Figure 19 indicates some unpredictability in the hybrid results i.e. when combining the individual methods. For example hybrid (3, 2) shows poor performance until iteration three while the other hybrid approaches, show similar accuracy to the individual methods. All the hybrid methods show greater consistency of results after iteration 3.

When combining the heuristic and stress based approaches together to form a hybrid approach the level of stress found and the speed at which it was found was no worse than using the heuristics approach on its own. This is evidenced by comparing data from figure 19 (hybrid plots) against figure 17 (individual plots) i.e. after 6 iterations the hybrid approach had found stresses of value 3.163E+31 pascals against a figure of 6.32E+31 pascals for individual plots.

From this observation it can be concluded that the influence of the heuristics on the overall hybrid approach is much greater than anticipated. Even when this affect is compensated for by using a hybrid strategy which heavily weights in favour of the stress approach (e.g. where heuristic rules weighting = 2, stress weighting = 4) the average stress after 6 iterations remains similar to a rules only approach. This suggests the need for further work to identify why this is occurring and then feed this back into the weightings for future development.

Having run the model with the hybrids the coloured stress gradients across each structure can be inspected by an engineer, see figure 19 and appendices J below. LISA assigns colours to different ranges of stress based on the range of values within the model. In the majority of cases red areas represent those areas of highest stress followed by orange, yellow, green, light blue and eventually dark blue.


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



(b) Meshing highly concentrated in those areas of the structure where the stress converges

Figure 19: 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

In addition to the data summarised in this section of the report additional data collected for the various hybrid approaches applied to two other models is summarised in Appendices K and L. These results show how the hybrid approaches perform against the performance metrics identified in section 7.6.1 and provide reassurance that whilst the hybrid method cannot conclusively be demonstrated to calculate stress quicker than traditional methods the quality of the output (as measured by the performance metric) is as good as any other method used.

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, see appendix M. Quality was also maintained throughout the project by the application of good software engineering practice.

The adoption of a modular architecture was a great great strength of the system allowing for a huge amount amount of potential extendability in the future and simplifying the ease with which both the individual methods could be combined to form new hybrid approaches. 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 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 to inform the refinement of the mesh.

A downside of the current design is the need for the user to manually specify the edges by manually entering data 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 large scale FE analysis. To address 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 in order to simplify the design and reduce unnecessary state, see appendix A. This was largely adhered to with significant use of lambda and higher order functions throughout the codebase. However, in the later stages of the project it became apparent that in many cases reliance on these features over more traditional programming constructs such as loops and conditions resulted in reduced readability of the code and a potential reduction in speed for implementations of the algorithms described in this dissertation.

8 Further Work

This section details some areas which given additional time would be investigated, if not implemented. Each of these areas would hopefully provide some benefit in assisting to further 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. Feedback from engineers with applied industrial experience along with that of academics would have allowed for a more conclusive analysis of the system and its ability to work across a greater number of general case scenarios. No user feedback was 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

Lack of straightforward portability for the system was one factor that would have made it difficult to get feedback from industry, an alternative approach would be to develop a web interface so as to allow users to easily interact with the system externally. This approach would allow engineers to submit feedback digitally which could then be aggregated from a much greater range of users separated geographically.

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)) [26]. 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 models 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 fixed combinations of different methods each weighted in a predefined manner. This is a simple approach to demonstrating the working concept, however 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 along with hybrid composites of the two it has been shown that the project has been delivered to meet each of the three main objectives outlined under “Description Of The Work”, see section 4. The delivered system has been demonstrated capable of effectively evaluating 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 K and L) 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 useful as an opportunity to learn more about the underlying processes through both research and practical experimentation. As a means of facilitating my personal learning 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 the 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.

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 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 was the rapid increase in stress at particular points which took me by surprise having not deliberately stressed models to breaking point before. 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. Given the chance to repeat the project having now learnt a lot about finite element systems I feel I would be better placed to evaluate the potential refinement approaches in less time and so would be able to focus more improving the systems ability to combine methods of refinement.

As the project progressed the increase in scope also presented problems for me as the sole researcher and developer of the system. With there being 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 my limited time and experience.

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 took considerably longer than initially predicted and thus reduced the amount of time that could be directed towards the other more theoretical components.

Repeating the project I would also like to implement the refinement processes for a wider variety of different element types such as Tri3 and Tet4 as shown in appendix B. 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.

Perhaps the most successful technical finding of the project was the success of Dolsaks ILP knowledge base which proved effective as a second means of refinement 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] S. Hale. Will poorly-shaped elements really affect my solution? https://caeai.com/blog/will-poorly-shaped-elements-really-affect-my-solution.

[9] 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.

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

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

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

[13] 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.

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

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

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

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

[18] S. Muggleton. Logic based and probabilistic symbolic learning. https://www.youtube.com/watch?v=4Cwd05dWW98.

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

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

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

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

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

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

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

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

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

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

[29] 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.

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

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

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

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

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

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