**School of Computing**
FACULTY OF ENGINEERING AND
PHYSICAL SCIENCES


# Final Report

## Learning Resources for Understanding and Using the Finite Element Method

**Kyle Antony Morris**

**Submitted in accordance with the requirements for the degree of Computer Science with Mathematics**

2022/23

COMP3931 Individual Project

The candidate confirms that the following have been submitted.

| Items | Format | Recipient(s) and Date |
|---|---|---|
| Final Report | PDF file | Uploaded to Minerva (16/05/23) |
| Main Code Repository: `Finite-Element-Method-Project` | URL | Sent to assessor and supervisor (16/05/23) |
| Secondary Code Repository: `FEMLearningResources` | URL | Sent to assessor and supervisor (16/05/23) |
| User testing / evaluation Questionnaire | URL | Sent to assessor and supervisor (16/05/23) |

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of Student) Kyle Antony Morris

**Summary**

The aim of this project is to address the issue of the mathematical complexity of the finite element method and create a set of learning resources that can be used to bridge the gap in understanding required to use FEM. The goal is to guide students through the mathematics as well as instruct them on how to implement the method themselves. In this report I will describe the background research done to influence my design decisions, detail the writing process, and discuss the results obtained from user feedback.
Key objectives for the project include:

- To gain a good enough understanding of finite element analysis for me to write effective learning material on the subject.
- To research good educational practice. I.e. How to most effectively engage students and maximise retention.
- To use this knowledge to develop an effective and comprehensive tool for understanding and using the finite element method.
- To evaluate the success of the resource analysing user feedback.

**Acknowledgements**

Thank you to my friends and family for supporting me throughout the journey of this project. Your kindness and love mean everything to me. Many thanks also go to my supervisor for their direction and their inspiration in helping me choose this fascinating topic.

# Contents

# Chapter 1

# Introduction and Background Research

## 1.1 Introduction

Partial differential equations (PDEs) are some of the most important types of formulae in mathematics. They arise in every field of mathematically inclined science, such as physics, engineering, chemistry and more. PDEs are used to describe physical systems. Some PDEs describe dynamical systems; that is, systems that change with time. Others describe static systems, that only vary in space, or sometimes other quantities. This might include the movement of fluids, heat and waves, or the deformation of certain structures acting under a force. They are used to model population dynamics, chemical reactions, electromagnetic fields and are also central to quantum mechanics. PDEs are fundamental to our understanding of science and the world[1].

Since PDEs are so important, the natural question is, how are they solved? Solving a PDE simply means obtaining an equation that describes the key variable explicitly, e.g. velocity of fluid particles given their coordinates in space, number of individuals in a population at any time, or amount of heat at a certain point on a metal rod. There are methods for solving certain types of PDE analytically, but generally PDEs are either incredibly difficult to solve, or in many cases, impossible.

Fortunately, methods have been developed to provide approximate solutions to PDEs, called numerical methods. These methods always have some degree of error, but by increasing the amount of computation performed, this error can be reduced. Numerical methods can involve an enormous amount of computation, making them the ideal candidates for modern computing, where processors perform billions of operations per second.

One such method is called the Finite Element Method (FEM), where the domain being solved over is discretised, and linear algebra techniques are used to construct a linear system of equations that can be numerically solved to obtain our approximate solution[2]. The method has a strong mathematical foundation and it can be shown that the approximation does converge as the number of discrete elements increases[3]. The method is fantastically versatile, as it can be applied to any PDE, whether linear or non-linear, the only problem is that it requires a good mathematical understanding to implement.

This finite element method has its origins in a numerical algorithm called the Rayleigh-Ritz method from the late nineteenth/early twentieth century, which was used for predicting stress and displacement on solid structures[4]. Later an alternative to this was developed called the Galerkin method, equivalence between these two was later proven in 1962 [5]. In 1956, the first academic paper was published which coined the term 'finite element' by Turner et al. [6], the main advantage of this over the previous methods was its application to complex domains. After this the use of the method began to spread and it was mathematically proven in a government report in 1972 [7]. Since then, the method has been ubiquitous throughout physics, engineering and science as a whole.

## 1.2 The Finite Element Method

### 1.2.1 How the Finite Element Method Works

As previously mentioned, FEM is a numerical method for solving differential equations. Most often, it is applied to partial differential equations. The general form of a PDE is shown in Equation 1.1. Here, the variables are denoted $t, x_1, \cdots, x_n$, of which there are $n+1$. The order of the equation is the order of the highest derivative of $u$, denoted $m$. $u$ is the unknown function that being solved for. Note that $t$ may not always be present here.

$$\begin{aligned} F\Big(u, t, x_1, x_2, \cdots, x_n \\ \frac{\partial u}{\partial x_1}, \frac{\partial u}{\partial x_2}, \cdots, \frac{\partial u}{\partial x_n}, \\ \frac{\partial^2 u}{\partial^2 x_1^2}, \frac{\partial^2 u}{\partial x_2^2}, \cdots, \frac{\partial^2 u}{\partial x_n^2}, \\ \cdots \\ \frac{\partial^m u}{\partial x_1^m}, \frac{\partial^m u}{\partial x_2^m}, \cdots, \frac{\partial^m u}{\partial x_n^m}\Big) = 0 \end{aligned} \tag{1.1}$$

Clearly, Equation 1.1 is a complicated equation, but most PDEs do not have this many variables. Also, it is rare that a PDE will have an order greater than two, although this can occur. Along side the equation itself, there will also be a set of conditions that constrain the variables to some domain $\Omega$. There will also be a set of conditions that constrain $u$ on the boundary of this domain $\delta\Omega$. These are called the boundary conditions. If one of the variables involved is time this must also be constrained by an initial condition. Together these make a initial value boundary problem (IVBP). A full instance can be seen in Equation 1.2.

$$\begin{cases} F = 0 & \text{on } \Omega \\ u = u_D(x_1, \cdots, x_n, t) & \text{on } \delta\Omega \\ u = u_{\text{init}}(x_1, \cdots, x_n) & \text{at } t = 0 \end{cases} \tag{1.2}$$

$$\text{where } \Omega \subset \mathbb{C}^n$$

The method of solving such an equation using the finite element method first starts by converting Equation 1.2 into a variational formulation. Specific methods of constructing this formulation vary from problem to problem but the general approach is as follows:

- Take the equation defined over $\Omega$ and multiply both sides by a test function $v$.
- Integrate both sides over $\Omega$.
- Integrate any second order derivatives by parts.
- Rearrange so all terms involving $u$ are on one side. This expression is called the bilinear form, denoted $a(u, v)$ and the other side is called the linear form, denoted $L(v)$.

After doing this, a few facts about the functions spaces $u$ and $v$ belong to need to be acknowledged. The class of functions that can solve Equation 1.2 belong to the function space,

$$V = \{v : v \in H^1(\Omega), v = u_D \text{ on } \delta\Omega, v = u_{\text{init}} \text{ at } t = 0\}$$

Clearly, in addition to this, they must also satisfy the main equation. $H_1(\Omega)$ is the Sobloev space, which is a space where all functions must be square integrable and continuous and their derivatives must simply be square integrable. $V$ is called the trial space, and $u$ is called the trial function. In order for the variational formulation to be valid, the test function must also come from $H_1(\Omega)$, but instead include the condition that it must vanish on the boundary of our domain. This yields the test space,

$$V_0 = \{v : v \in H^1(\Omega), v = 0 \text{ on } \delta\Omega, \}$$

This trick allows us to remove any parts of our integration that include $v$ on $\delta\Omega$. More importantly however, subspaces of $V$ and $V_0$, denoted $V_h$ and $V_{h,0}$ can be constructed, where the elements are piecewise polynomial functions. This can only be done because these spaces permit functions with discontinuous derivatives. So the transition from a continuous space to a discontinuous one is made, replacing $u$ with $u_h \in V_0$ and assuming $v \in V_{h,0}$. This is the finite element approximated problem.

$$\begin{aligned} &\text{Solve for } u_h \in V_h \text{ where,} \\ &a(u_h, v) = L(v) \quad \forall v \in V_{h,0} \end{aligned} \tag{1.3}$$

To solve this problem, our domain $\Omega$ must be split into discrete segments, called a mesh. In one dimension, this is amounts to splitting the number line into sub-intervals, in two dimensions, the plane is split into cells, or finite elements. These cells are primitive shapes like rectangles or triangles and depending on the domain, it can be simpler to use one or the other. The points where these cells meet each other are called nodes.

Now to solve, techniques from linear algebra are used. A basis of $V_h$ and $V_{h,0}$ must be constructed to rewrite $u$ and $v$ as a linear combination of basis functions. The one used in the finite element method is called the nodal basis, because the coefficients of a linear combination of each of these basis vectors are equal to the function at the nodes of the mesh. The usefulness of this, is that any function can be defined using only these nodal values, meaning if they can be obtained, the approximate solution can be found.

The way to obtain these coefficients, called degrees of freedom, is by substituting in $u$ and $v$ as a linear combination of these basis functions, called hat functions. This produces $n + 1$ independent equations, where $n$ is the number of nodes in the mesh. Then, the resulting system of equations can be solved using known numerical methods. One can use direct methods like Gaussian elimination and LU-Factorisation, or where appropriate, iterative methods like Jacobi, or Gauss-Seidel.

If time is a variable in the equation, one can use the numerical methods used for ordinary differential equations to discretise the time domain and obtain an approximation that way. The Runge-Kutta methods are a popular choice.

### 1.2.2 Convergence and Error Estimates

FEM is an extremely powerful technique because it can be applied to any problem of the form 1.1 if a variational form can be defined, and it can be shown that all problems have such a formulation [8]. There may be some practical difficulties along the way, but the sophistication of simultaneous equation solvers means that the most computationally intense problems can be scaled to work on high-performance clusters.
One natural question that arises is that of convergence. Convergence of the method can be proved by showing that the error of our approximated solution tends to 0 as the mesh size increases. A natural way to measure error is using the $L^2$-norm. This is because the Sobloev space $H^1$ is actually a Hilbert space, defined with the $L^2$-norm as its inner product. Thus, the $L^2$-norm is defined for all elements of $V_h$. The $L^2$ inner product of two functions $f$ and $g$ is defined as

$$\langle f, g \rangle_{L^2(\Omega)} = \int_\Omega f \bar{g} \, \mathrm{d}x$$

Where $\bar{g}$ is the complex conjugate of $g$. Then, the $L^2$-norm is the square root of the inner product of a function with itself, denoted

$$\|v\|_{L^2(\Omega)} = \sqrt{\int_\Omega |v^2| \, \mathrm{d}x}$$

An enlightening example would be $\mathbb{R}^n$, which is also a Hilbert space when equipped with the familiar "dot" product as its inner product. The norm can then be though of as the distance between two points if the following is calculated,

$$\|x - y\| = \sqrt{(x-y) \cdot (x-y)} = \sqrt{\sum_{i=0}^{n} (x_i - y_i)^2}$$

This is precisely the definition of distance between two points. $H^1$ works the same way, only with a different notion of "distance", the $L^2$-norm $\|u - v\|_{L^2(\Omega)}$. Then, the error of our approximation $u_h$ from the actual solution $u$ can be measured by $\|u - u_h\|_{L^2(\Omega)}$. In Bengzon and Larson's textbook on the finite element method[3], they find the best estimate of error to be

$$\|u - u_h\|_{L^2(\Omega)} \leq Ch^2 \|D^2 u\|_{L^2(\Omega)} \tag{1.4}$$

Where $C$ is a constant, $D^2 u$ is the second order total derivative of $u$ and $h$ is defined to be the longest edge of any cell in the mesh, which is a way of measuring the mesh size. It is clear from 1.4 that as $h \to 0$, the error also tends to 0, which means the finite element method converges as the size of the cells in the mesh decreases.

### 1.2.3 Overview of Tools

Finite element analysis is ubiquitous throughout science because it is so powerful. However, performing the steps outlined in Section 1.2.1 can be quite involved, and if one doesn't have a strong background in mathematics and computer programming, an implementation would be difficult to produce. Therefore, many tools have been developed to allow scientists of all fields to use FEM. These range from software libraries and packages, to bespoke programs designed

for the most specific PDEs. Each have their benefits and advantages, which will be reviewed now.

**PolyFEM** [9] is a simple and lightweight implementation of FEM. It is written in C++, like many of libraries used today, but also has a Python binding. The power of PolyFEM is in its simplicity. It abstracts the method of solving completely, and all that is required of the user is the definition of a mesh and the parameters of the problem. Creating a mesh is straightforward, one can use an array manipulation library like Numpy for simple domains, and for more complicated domains a mesh generation package like Gmsh can be used. In addition to these options, custom-made meshes can be created using external software and stored in `.OBJ` files, which PolyFEM also accepts. The trade-off here is of course versatility. PolyFEM only currently supports 10 PDEs, and within that customisation is limited.

**Deal.II** [10] is a C++ library for solving partial differential equations using the finite element method. It is one of the most modern implementations and includes many advanced methods. It is used widely in academic and commercial applications due to its versatility. While this is one of the strongest FEM libraries in the public domain, its drawback lies in the complexity of use. Their extensive documentation is well written, but extremely long due to the number of available features. This module is recommended for more advanced users. The advantage to having a FEM module written in a C-like language is that C allows programmers to have very strict control over their memory management. This allows for greater efficiency when performing calculations, leading to an overall increase in speed for the generation of meshes, and the solving of systems of equations, for instance.

**FreeFEM** [11] is another open-source library used for the development of finite element codes. It is based in C++, and contains pre-built solvers for many popular PDEs. It interfaces with some of the most popular mesh generation softwares, visualisation programs, and linear algebra libraries including Gmsh, Mumps, PETSc and ParaView. FreeFEM also comes with it's own scripting language FreeFEM++, which is used for the creation of more specific finite element solvers. Once, again the documentation is extensive, but rigorous and contains many tutorials to help users get started.

**FEniCSx** [12] is another open-source computing platform written in C++, with a Python binding. The mission statement explains it aims to enable users to "quickly translate scientific models into efficient finite element code". FEniCSx is written in a way that keeps code very close to its mathematical notation, meaning it is the perfect tool for users with a mathematical background. It is a package composed of four main modules. `DOLFINx` is is the main computational backend of FEniCSx. It is the backbone that connects all of the other libraries together and contains the majority of classes and functions. It contains interfaces for input/output, mesh generation, equation solvers, manipulation of finite element function spaces, and much more. `UFL` stand for Unified Form Language and is the language used to express variational forms of PDEs. Its syntax is designed to be as close to writing the actual mathematics as possible. `FFCx` is the FEniCSx Form Compiler. It takes variational forms in UFL and compiles them into systems of equations. `Basix` deals with the finite elements themselves. It can generate the basis functions of spaces and their derivatives, as well as perform interpolation, projection and more.

Aside from the libraries listed above, there are many bespoke programs available as well. Specifically, there are programs made with a specific PDE in mind that they allow the user to solve. This may be for fluid dynamics, elastic deformation, geophysics, or any number of others. The benefit to these, is that it allows a user to solve problems by only inputting parameters to their equation, or specific solving options. Visualisation often comes built into these programs through a GUI. They significantly lower the barrier to entry for finite element analysis, but their main problem is that they provide little to no variability in terms of the equations themselves. However, for specific practical purposes, and for known problems, they are excellent. Another drawback is that many of these programs are proprietary, and some provide a financial barrier to their use.

## 1.3 Overview of Literature Regarding Mathematical Education

In order to write the learning resources effectively, a concrete understanding of the Finite Element Method is clearly needed. However an equally important part of writing a good learning resource involves having a good knowledge of educational practises and the utilisation of an approach that puts the student experience first. We will now discuss the different approaches in education and ultimately, how they apply to mathematical education.

### 1.3.1 Bloom's Taxonomy

In 1956, a group of researchers from the United States published arguably one of the most influential resources in the field of education. This resource was called Bloom's Taxonomy [13], named after Benjamin Bloom, the leader of the team. They devised a hierarchical classification of learning objectives. More specifically, it was a tiered structure of skills one must obtain in order to master a subject. The hierarchy reflects how difficult each skill is to obtain, and the progression a student would take to obtain this mastery.

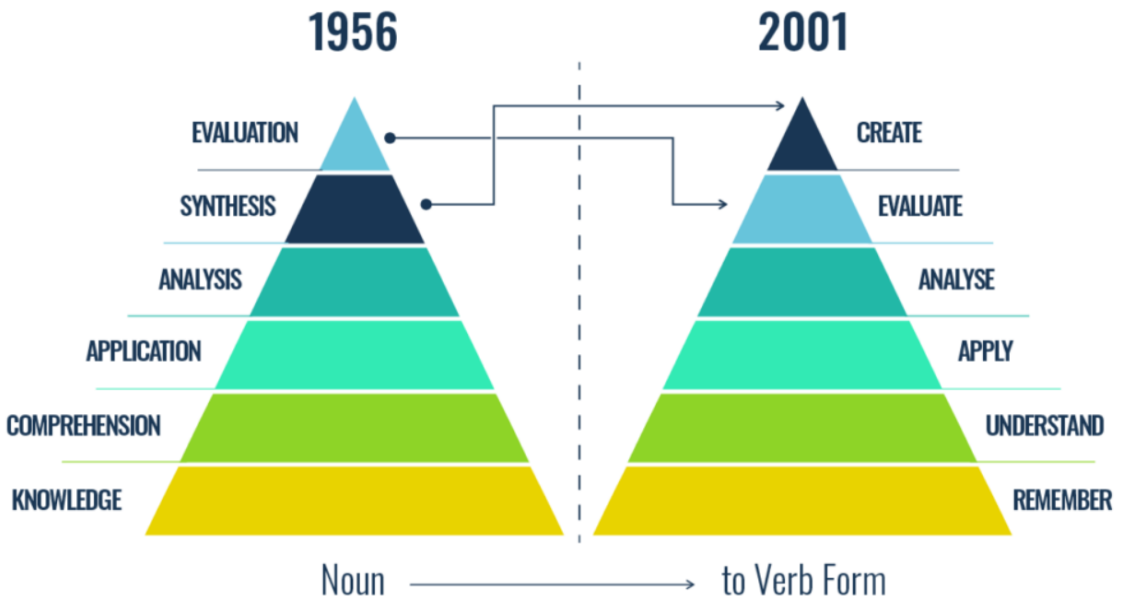



Figure 1.1: Comparison between Bloom's original Taxonomy and the revised version [14]

The original classification can be seen on the left hand side of Figure 1.1. Each layer dictates that certain tasks will be able to be completed once the layer has been mastered. These are the skills obtained.

**Knowledge** is the most basic and fundamental skill one must obtain. This objective is the backbone for all others, one needs to be able to remember requisite facts in order to obtain more complex understanding. Some skills gained from this level could be: definition, description, identification, and recollection.

**Comprehension** is the next level, it suggests a student needs more than to just blindly recall facts, they need to understand a certain logic behind them. They need to be able to describe the meaning behind statements and explain this to someone. Skills gained in this tier include: description, explanation, summarisation, and discussion.

**Application** is the execution of a certain procedure. This is the first part of the Taxonomy where students may not know the answer directly, instead they must use their prior knowledge to solve certain problems. Necessary skills to prove application has been obtained might include: execution, calculation, implementation and manipulation.

**Analysis** argues students should be able to break information into parts and draw conclusions from connections between these parts. This might be different topics in a module, or points in an argument. This is more advanced than application, since that skill was also on the recollection of steps of a procedure. Here students must generate their own conclusions, generating novel information. Skills gained here include: categorisation, comparison, deduction and criticism.

**Synthesis** is the next learning objective. This describes the process of creating new information. It involves organising knowledge in new ways to create a new piece of information, or a product. This skill may be required, in part, in the previous tiers, but truly creating something new is a skill all in its own. Skills that fall under this category might be: design, development, planning and invention.

**Evaluation** is the final learning objective in the 1956 version of Bloom's Taxonomy. In Bloom's handbook, evaluation is defined as "the making of judgments about the value of ideas, works, solutions, methods, material, etc." More plainly, it is the skill of critiquing things, something that can only be done effectively when your knowledge is at the most sophisticated level. This is why Bloom and his team put it at the peak of the taxonomy. Some of the skills one must be able to demonstrate in order to prove evaluation is satisfied, and hence a subject has been mastered are: justification, criticism, judgement and assessment.

This describes the full taxonomy when it was created. It can be clearly seen that each one flows into the next, but there is also some overlap between these levels. For instance, the skill of description could fall under knowledge or comprehension, since a student could remember a certain description which shows knowledge, but could also suggest understanding if it wasn't blindly quoted. This is why the taxonomy is not a strict rule-set for learning. Instead, it is a measurement of progress and any curriculum using it should be implemented with care, instead of blindly following the pyramid.

### 1.3.2 Revised Bloom's Taxonomy

Much later, in 2001, Lorin Anderson led a new team of educators to update the taxonomy, called the Revised Bloom's Taxonomy (RBT) [15]. The structure, and differences between the original taxonomy can be seen in Figure 1.1. They opted to simplify some of the language, breaking down one potential barrier to the taxonomy's use. They also opted to replace the abstract verbs with present tense versions of the same words. More importantly, it was decided that creation was a more complex skill than evaluation.
In addition to the rearrangement of learning objectives, a new dimension was added to make a so called "Taxonomy Table". This new dimension is called the knowledge dimension, while the original is called the cognitive process dimension.

| **Knowledge** | **Cognitive Process** | | | | | |
|---|---|---|---|---|---|---|
| | Remember | Understand | Apply | Analyse | Evaluate | Create |
| Factual | | | | | | |
| Conceptual | | | | | | |
| Procedural | | | | | | |
| Meta-Cognitive | | | | | | |

Figure 1.2: Taxonomy Table for Bloom's Revised Taxonomy

Figure 1.2 shows the taxonomy table, with the new knowledge dimension. The shading refers to the complexity of each skill, darker being more complex. The knowledge dimension refers to the *type* of knowledge used at that level, while the cognitive process dimension refers to *how* that knowledge should be used. For instance, for one to have mastered the apply column, they must be able to apply factual, conceptual, procedural, and meta-congitive knowledge. **Factual** knowledge refers to the most basic information a student must be familiar with. It includes simple definitions, terms and symbols. **Conceptual** knowledge is the kind of knowledge one gains by understanding relationships between definitions. A lot of mathematics lies here, laws and theorems are examples of this kind of knowledge, but scientific theories, and more advanced, composite definitions also belong here. **Procedural** knowledge refers to processes. This is the knowledge needed to perform algorithms and tasks. It also includes techniques, methods and the understanding of when such processes need to be performed. Finally, the most advanced type of knowledge is **meta-cognitive** knowledge. This refers to knowledge about one's own learning. It includes the understanding of a student's own strengths and weaknesses, how they learn best, and even how to teach others effectively. It also contains knowledge about knowledge itself, for instance the structure of material in a course or material that will appear on certain tests.
Another problem Anderson and his team set out to solve was the potential ambiguity within each of the definitions. They subdivided each item in the knowledge dimension into two or three "sub-types" and also divided each of the cognitive processes into categories, of which there were anywhere from two to seven. For instance, within analyse there are the categories of "differentiating", "organising", and "attributing". This would certainly help any potential educator planning to use the taxonomy table. Needless to say, it was a rigorous overhaul and

one of many overhauls Bloom's taxonomy has received since its original publication.

## 1.4 Learning Resource Requirements

With the aid of Bloom's taxonomy, and a few other sources it should be possible to outline the requirements of a good learning resource. These are the guidelines that will be attempted to be met, and will be evaluated against in the user testing. The definition of learning resource used here is specifically an article, textbook, or tutorial used for self-teaching. Not included here are things like videos, or spoken word, which have their own additional criteria. See Section 4.3 for more information on other possibilities that could be explored.
These are 9 aspects that contribute to the effectiveness of a learning resource under this definition:

- **Good readability**
  - Readability is a measure of how easy a text is to read. A highly readable text will have shorter sentences and simpler vocabulary. If a text is readable, it means that information can be conveyed more directly allowing otherwise difficult ideas to be explained simply. One should note however, that academic subjects do have a high level of complexity and often cannot be explained in a particularly readable way. Therefore, the goal is to maximise readability, while still conveying all the intended information.
- **Effective use of visual learning methods**
  - One way to aid a student in understanding difficult ideas is through visualisation. Color, graphs, flow charts, diagrams, and more, all fall under the umbrella of visual learning approaches. They provide a new way of thinking about certain ideas, allowing readers to more easily pick up the information [16].
- **Professional aesthetics**
  - While aesthetics are not all that important and the information and explanations are the key content, they are still relevant when listing important attributes of a good learning resource. This is because information on the page should not be cluttered or distracting, the focus should be on content such as text, diagrams, animations and code snippets.
- **Good pacing**
  - A good learning resource shouldn't move too quickly through content, otherwise students may feel unprepared for new concepts. A fair amount of time should be spent on examples and explanations in order for students to fully consolidate the information. Care should also be taken however, to not slow down the pace too much. Spending too much time on an individual idea might cause the student to lose interest or start skipping through sections.
- **Proof of usefulness**

  - The ultimate aim of a learning resource is for a student to take the knowledge they gain from it and hone a practical skill. Even some of the most pure mathematical fields have practical applications. Students want to know how the information presented actually links to real problems. [16]

- **Well-tuned difficulty**

  - This section certainly links to pacing, the content should not feel difficult because the student is unprepared. However, another point to note is that any exercises provided should not be too difficult, while providing a challenge at the same time. There should be a range of questions that test the student in the subject matter, having a level of difficulty that ranges as far as the student is willing to go.

- **Sensible structure**

  - The flow of the learning resource should make sense. Not only from paragraph to paragraph but from section to section. One should motivate the other, keeping the reader engaged and causing them to continue reading.

- **Clear objectives defined**

  - At the start of the resource, clear learning objectives should be set out. This should motivate the student to continue reading, since they will not only know what they will learn, but also how they will be able to apply it. Then, at the end, students will be able to reflect on these objectives and decide for themselves whether or not they have achieved them.

## 1.5 Analysis of Existing FEM Learning Resources

With a more precisely defined criteria of what makes a good learning resource, it is now possible to analyse some of the most popular learning resources in finite element analysis.

### 1.5.1 Bengzon and Larson's Finite Element Textbook

We start with a textbook written by Mats. G. Larson and Fredrik Bengzon called The Finite Element Method: Theory, Implementation, and Applications [3]. This textbook is meant to introduce students to the finite element method by explaining the necessary mathematics, while also showing how to implement it in MATLAB.
The textbook seems to meet most, if not all criteria listed above. It has decent readability and aesthetics. Visuals have been used well throughout, the diagrams of finite element basis functions are particularly useful. The difficulty is kept to a minimum. This is due to excellent pacing that begins by introducing the problem in a 1D example, then moves up to 2D, giving readers ample time to digest the methods presented. These first five chapters give enough information for a student to be able to implement most simple PDEs. Information after this gets more advanced, and several chapters are devoted to specific engineering problems. This also gives the reader some real world context to FEM, making the reading significantly more engaging.

At the end of each chapter a selection of exercises are presented. These are very well written, allowing the reader to consolidate information gained in the previous chapter. Bloom's Taxonomy seems to be mostly satisfied, there are questions relating to simple recollection and use of definitions (recall), and to applying methods learned to specific problems (comprehension and application). In questions that require the reader to construct a proof, analysis is met because, one must bring together several concepts and ideas to construct an argument. Synthesis could also be considered to be part of proof-writing, although the definition might be being stretched here. Evaluation however, is not so well explored. This could be one criticism against the book. Bengzon and Larson could have asked students which approach might be better for certain problems for instance, which would force the reader to weigh up the advantages and disadvantages of certain methods.

### 1.5.2 J. S. Dokken's FEniCSx Tutorial

The next FEM learning resource is a tutorial specifically written as a guide for FEniCSx, the previously mentioned FEM computing platform. It is written and maintained by Jørgen S. Dokken, a researcher in the field and member of the FEniCSx Steering Council. The tutorial was adapted from Langtangen and Logg's book *Solving PDEs in Python: The FEniCS Tutorial I* [17], which was written for the previous version of FEniCSx.

Rather than as a guide to the finite element method, this tutorial serves more as a guide on how to use FEniCSx. Having said that, mathematical explanations are provided, they are just simply not as detailed as Bengzon-Larson for instance. The tutorial is written using in a Notebook format, which is spoken about in Section 2.1.1, but means that code snippets feature heavily. Dokken uses a service called Binder to host the notebooks remotely, which allows users to run the code in their own browsers without downloading any dependencies. This removes a large barrier to the use of FEniCSx, and lets users tinker and get familiar with the library without having to download the libraries on their local machine. If one does not wish to use Binder, they are free to view the static version as well.

The notebook is very readable, jargon is kept to a minimum, although there is still quite a lot due to the subject matter. Many terms are hyperlinked to external sites where readers can find more detailed information. Visuals are used vary sparingly, which is one downside of the tutorial. It could certainly be improved with a few diagrams or illustrations. The pacing is excellent, and each chapter uses information from the previous to expand the reader's knowledge in an engaging way. Sometimes the pace can be slightly too fast, and important concepts don't have enough time spent on them, but in these cases external resources are often linked to, and Dokken recommends that readers unfamiliar with FEM read a textbook on the subject.

The usefulness is motivated fairly well, there is a section titled "Gallery of Finite Element Solvers" which provides plenty of examples. Most of the problems presented however seem to be idealised, but still provide good insight into the potential applications of FEniCSx. The sections flow well between each other, and difficulty is well managed. Learning objectives also feature at the start of every chapter, which is great to see.

# Chapter 2

# Methods

## 2.1 Software Design

With the necessary background research performed, the next step was to design the architecture of the online learning resource.

### 2.1.1 JupyterLab

JupyterLab is an interactive computing platform developed by Project Jupyter [18]. It allows users to run code inside their browsers, as well as displaying Markdown and LaTeX. Markdown is a simple language for formatting text. It includes things like headings, bullet points, tables, images etc. It is used widely in documentation, readme files and other educational coding resources. It is incredibly lightweight and easy to write, and it's philosophy is about putting readability first [19]. LaTeX is another so called "mark-up" language widely used in academic writing. It can be used to construct professional looking documents, and gives the user a massive amount of control in their construction. It also has an extensive set of mathematical symbols, and environments, making it one of the best tools for writing mathematics on a computer. JupyterLab uses a JavaScript library called MathJax [20] to implement the mathematical functionality of LaTeX.
The interface of JupyterLab can be seen in Figure 2.1, the main section on the right is where code and markdown is displayed, while the left shows other options. In this figure, it is showing the file tree, although this can also show kernels currently in use, a structured view of the notebook currently open and any extensions.

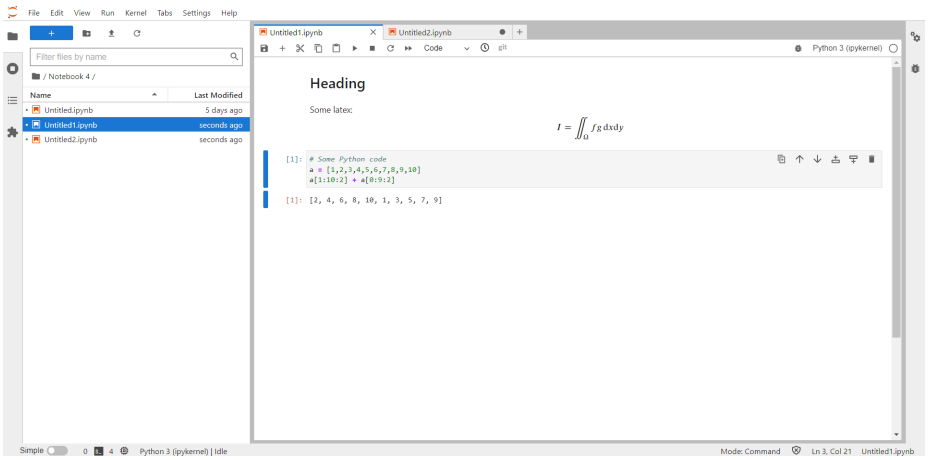



Figure 2.1: The JupyterLab Interface

Jupyter is an excellent way of running code for instructional or demonstrative purposes. The LaTeX allows for precise mathematical descriptions, while Markdown keeps everything readable and clean. It is also very quick to write in. Jupyter supports most modern languages by allowing the user to specify a kernel, although the main languages that are supported are Julia,

Python and R (Which are the origins of the name Jupyter). In this context, a kernel is the computational engine used to run code within the notebook, of which there are many official versions and community-maintained versions.

### 2.1.2 Python

The Finite Element library used is called FEniCSx, which was introduced in Section 1.2.3. The justification behind using this particular platform, is that it provides a good compromise between versatility and complexity. FEniCSx is very versatile because has a lot of features, meaning PDEs arising in any scientific discipline can be solved. It is also easier to pick up than something like FreeFEM++, since it is based solely on Python and there are some good already tutorials already created. In FEniCSx, the most basic PDE instances can be defined, solved and visualised in around fifty lines.
One of the downsides of using FEniCSx is that it has a lot of dependencies. The guide to installing all the source packages can be seen here. At least eleven separate libraries are required for the C++ core, after which one can install the Python interface. Needless to say, this is a time consuming and complicated process. It can be streamlined greatly by instead using Docker.

### 2.1.3 Docker

Docker is a virtualisation software that runs virtual machines at an operating system level. Each virtual machine is called a container, which is designated its own an area of memory. When creating a docker container, one must specify its contents. This is done by creating an image, a kind of template that containers copy from. Images contain a set of instructions that create directories, download and install programs, and generally set up a container for use. Docker containers are essentially separate, independent environments in which one can work. The point of having these separate environments is to allow for the development of independent software features. As well as it's use for software development, it can also be used to run complicated dependencies. An organisation, such as the Fenics Group, might host a Docker image from which any user can pull. Once the user has the Docker image locally, they are able to create their own Docker containers and work with the organisation's software. The FEniCSx group has such an image! It can be found on the DOLFINx Github page.
However, while these docker images are excellent and contain the whole of FEniCSx and more, they don't contain everything necessary for our learning resources. For instance the plotting library Matplotlib is not included in the main Docker image. Also certain timing and meshing libraries are not built in. It is possible to create a Docker image using a DockerFile. In said file, one can pull from an existing image and then add their own dependencies, which is exactly what would be needed in our case. However, the simpler option here was to use a pre-made docker image hosted by Dokken for use with his FEniCSx Tutorial [21].

### 2.1.4 Method of Delivery

Having created the chosen development environment, a docker container inheriting from Dokken's image, the next logical thought was how to allow users to access the notebooks. One

of the main features of a notebook that makes it appealing is the interactive nature of the blocks. Users can run code examples to see dynamically changing outputs. They can alter the author's code in any way they like, or use the dependencies to solve their own problems. As such, the interactive aspect needed to be kept, instead of converting the notebook files to static PDFs. The first method attempted, was to use a service called Binder [22]. Binder is a service that hosts Jupyter notebooks in an executable environment. This means anyone, anywhere can access the notebooks without setting up any dependencies. They only have to follow a link. Sounds like the perfect solution!

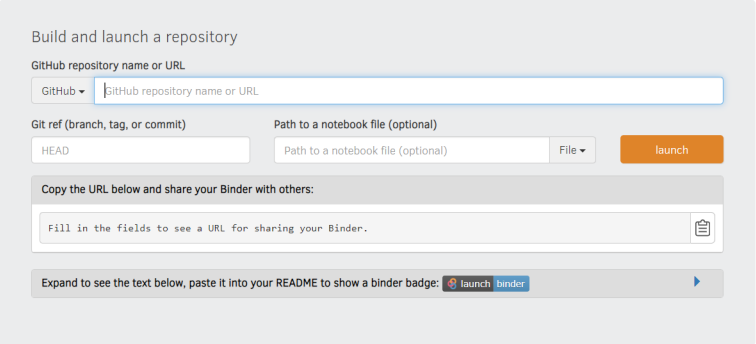



Figure 2.2: Hosting a Jupyter Notebook with Binder

All Binder needs to host a notebook is a link to the repository containing said notebooks, plus a few launch options. It's incredibly simple and would work excellently if it wasn't for one problem. The notebook's dependencies can normally be specified by adding a `requirements.txt` file listing the Python/Julia/R libraries needed and their versions. However, since we have many libraries that are actually built from C++, and require more care to install, this approach will not work. Instead a DockerFile must be provided. This makes sense, because Binder actually uses a Docker container themselves to host the notebooks being run. However, despite the feature being implemented by Binder, they actually discourage users from using it. They consider it an "advanced use case" and, crucially, "cannot guarantee that the DockerFile will work" [23].
Unfortunately, this was indeed the case upon integrating Dokken's DockerFile into the repository used to host the notebooks. Commits de7cec4 through to 53e565f were all attempts to get Binder working. It was all based off of Dokken's file, which already worked with Binder in his repository, but when the exact same files were in mine, it didn't work. This was unfortunate, but in the interest of time, the decision was made to not go with Binder and instead get users to run the notebooks themselves.
Getting users to run the notebooks themselves required them to install Docker on their own machines and create a container themselves. The installation of Docker can be fairly involved to someone who doesn't know the process, since it requires a separate virtualisation software at an OS-level. Linux based operating systems come with native virtualisation, but Windows does not. Therefore, one of the best ways to run docker on Windows is to install Linux on Windows and then perform the containerisation inside of Linux (inside of Windows). Figure 2.3 illustrates the hierarchy.
The process of installing WSL2 (Windows Subsystem for Linux) is not too complicated, although a guide is certainly required in order to explain the process, and also how to create

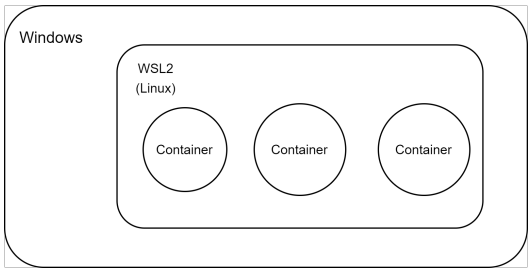



Figure 2.3: Virtualisation Hierarchy

the container itself. Therefore, since this seemed to be the only option, a guide was created and uploaded to the repository for users convenience. The resulting PDF can be found in Appendix C and describes multiple options depending on the user's system, with the goal being to allow as many students to access the material as possible.

## 2.2 Content Design

### 2.2.1 User Experience

A flow diagram of the user experience in the resource's current form is shown in Figure 2.4.

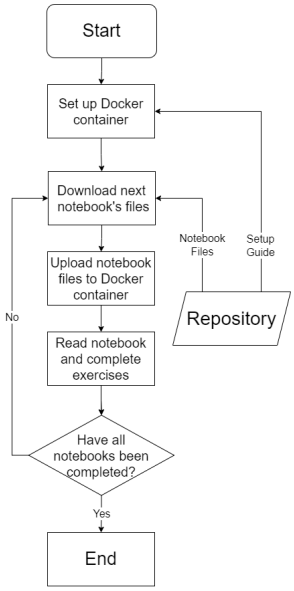



Figure 2.4: Flow Diagram of User Experience

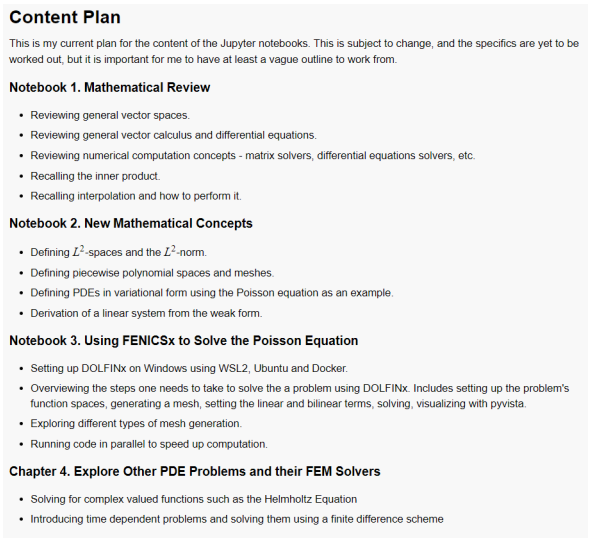
The user will begin by accessing the repository containing all the notebook files, called `FEMLearningResources`. From here, they must download and complete the setup guide to install Docker and create their Docker container. Once this is done they will have everything they need to proceed with running and viewing the notebooks in an interactive setting. Next, the user downloads the files for the notebook they want to read. This needs to be unzipped and then should be uploaded to the Docker container. Jupyter has functionality for uploading documents, so this can be used. Finally, the user can read the notebook, run code blocks and complete exercises.

While this experience is a little more involved than the Binder approach, it does teach students

how to use Docker, a very widely used tool in software development. However, it means that the barrier to entry is higher than it possibly needs to be.

### 2.2.2 Content Plan

With the learning structure defined, we will now discuss the design of the content plan. The initial goal was to teach students to understand and use the Finite Element Method. The belief during the design phase was that "understanding" took precedence over "using". This is because if one understands the theory behind a method, implementation is broadly the same over different libraries. Furthermore, FEniCSx is a library very closely related to the mathematical description of FEM, so a focus on the mathematics would lead more naturally into this. Another balance that was considered was between simplicity and complexity. Making the content too simple would not meet the goal of the project and would leave students feeling dissatisfied. Making the content too complex by increasing the scope would put pressure on the time constraints of the project, and may lead to students feeling rushed or unsure about certain topics due to the less considered development. This is how to original content plan was developed. It can be seen in Figure 2.5a.



**Content Plan**

This is my current plan for the content of the Jupyter notebooks. This is subject to change, and the specifics are yet to be worked out, but it is important for me to have at least a vague outline to work from.

**Notebook 1. Mathematical Review**

- Reviewing general vector spaces.
- Reviewing general vector calculus and differential equations.
- Reviewing numerical computation concepts - matrix solvers, differential equations solvers, etc.
- Recalling the inner product.
- Recalling interpolation and how to perform it.

**Notebook 2. New Mathematical Concepts**

- Defining $L^2$-spaces and the $L^2$-norm.
- Defining piecewise polynomial spaces and meshes.
- Defining PDEs in variational form using the Poisson equation as an example.
- Derivation of a linear system from the weak form.

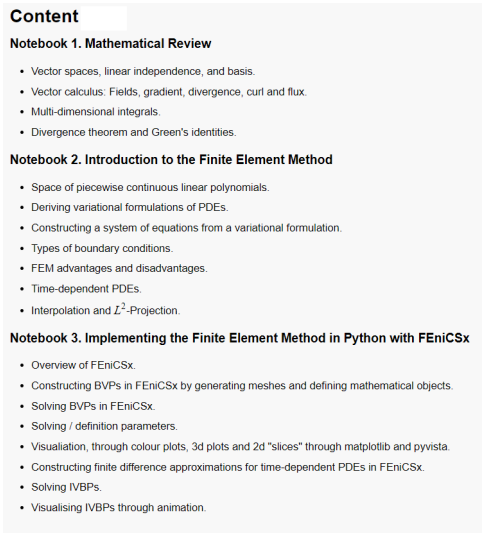
**Notebook 3. Using FENICSx to Solve the Poisson Equation**

- Setting up DOLFINx on Windows using WSL2, Ubuntu and Docker.
- Overviewing the steps one needs to take to solve the a problem using DOLFINx. Includes setting up the problem's function spaces, generating a mesh, setting the linear and bilinear terms, solving, visualizing with pyvista.
- Exploring different types of mesh generation.
- Running code in parallel to speed up computation.

**Chapter 4. Explore Other PDE Problems and their FEM Solvers**

- Solving for complex valued functions such as the Helmholtz Equation
- Introducing time dependent problems and solving them using a finite difference scheme

(a) Original Content Plan



**Content**

**Notebook 1. Mathematical Review**

- Vector spaces, linear independence, and basis.
- Vector calculus: Fields, gradient, divergence, curl and flux.
- Multi-dimensional integrals.
- Divergence theorem and Green's identities.

**Notebook 2. Introduction to the Finite Element Method**

- Space of piecewise continuous linear polynomials.
- Deriving variational formulations of PDEs.
- Constructing a system of equations from a variational formulation.
- Types of boundary conditions.
- FEM advantages and disadvantages.
- Time-dependent PDEs.
- Interpolation and $L^2$-Projection.

**Notebook 3. Implementing the Finite Element Method in Python with FEniCSx**

- Overview of FEniCSx.
- Constructing BVPs in FEniCSx by generating meshes and defining mathematical objects.
- Solving BVPs in FEniCSx.
- Solving / definition parameters.
- Visualiation, through colour plots, 3d plots and 2d "slices" through matplotlib and pyvista.
- Constructing finite difference approximations for time-dependent PDEs in FEniCSx.
- Solving IVBPs.
- Visualising IVBPs through animation.

(b) Updated Content Plan

Figure 2.5: Change in content between start and end of the project

The content originally spanned 4 Notebooks, each of increasing complexity and interest to the reader. The notebooks were to begin with a general mathematical review, preparing the reader for the new mathematics that was to follow. The reader was assumed to have a mathematical knowledge of a second or third year mathematics undergraduate student and basic knowledge of coding. This influenced the pace at which new material could be introduced as well as the topics that need to be included in the review. Topics like vector calculus, linear algebra and differential equations should be familiar to them, so could be covered in a less detailed way than the FEM content. It felt important to ease the readers in with a refresher, because rushing into new advanced mathematics could lead to confusion. They may have studied the concepts a while ago, so a reminder was important. As such, an entire notebook was devoted to revising these subjects.

Next was the first notebook of new content. The content can be seen in the figure, but it included all of the basic mathematical definitions and procedures for understanding the Finite Element Method. The third notebook was intended to give the readers a chance to get to grips with the implementation of the method. Originally, this is the point the setup guide was needed. The first two notebooks did not have interactive elements and were planned to be static resources. It was decided at a later point that even non-interactive parts of the resource would be created using the notebook format.
Finally, the fourth notebook was planned to give more complicated examples, such as the Helmholtz Equation, a PDE defined for complex-valued variables. Also the modelling of time-dependent systems was to be explored. This was left partially empty due to the idea that additional sections would be added later.
When development began, the content plan had to change. This was because a greater understanding of FEM had been obtained and certain aspects were deemed unnecessary to the goals of the project. Furthermore, a more refined idea of scope was developed, leading to the eventual cut of certain sections. This led to the refinement of four notebooks, to three. The revised syllabus can be seen in Figure 2.5b. A brief explanation of changes is listed below.

- Differential equations were removed from the review since enough knowledge could be gained from the vector calculus definitions to make this irrelevant.
- A precise description of inner products was removed since it wasn't all that necessary for understanding the function spaces involved.
- Emphasis was taken away from the $L^2$ spaces, they would still be mentioned, but it was not all that important to devote much time to them.
- Boundary conditions and FEM advantages and disadvantages were added.
- Some content was rearranged to increase flow and motivate future sections, specifically interpolation and time-dependent PDEs.
- Certain points were made much more specific.
- Descriptions of the parallel computation, different types of mesh generation and complex-valued PDEs were all dropped due to make the scope of the project more realistic.

### 2.2.3 Adapting the content

In order to maintain accuracy and correctness, the specific content of the notebooks had to be derived from well-respected and accurate sources. Notebook 1 spans the broadest spectrum of topics, and as such, was derived from the most sources. Knowledge gained from studying mathematics at university came in useful, but a set of invaluable resources in refreshing my memory was the lecture notes provided by module leaders at the University of Leeds. These modules included: *MATH2022 Groups and Vector Spaces*, *MATH1005 Core Mathematics*, *MATH2365 Vector Calculus*, and *MATH2375 Linear Differential Equations and Transforms*. They were useful not only for the correctness and accuracy of the mathematical statements, but also for their use in explaining concepts in a natural way. While these made up most of the

sources for Notebook 1, Wolfram MathWorld [24] also played a part in filling in some of the gaps where alternative or additional explanations were needed.
Notebook 2 was based primarily off Bengzon and Larson's textbook [3]. Inspiration was taken from the structure of the book. The authors decided the first two chapters should explain piecewise polynomial spaces and the finite element method in 1 dimension only. This means the domain being solved over was an interval on the number line. The simplicity enabled the reader to build upon the base knowledge presented and progression to higher dimensions followed very naturally.
Most of Notebook 3's material was taken from Dokken's tutorial [25]. His tutorial is one of the only learning resources on FEniCSx available and is by far the most accessible and well-written. Due to this fact and also due to the complexity of the library, a large portion of the code presented in Notebook 3 was written by Dokken. However, the explanations he gives in the tutorial can sometimes be a little shallow, and care has been taken to expand and enlighten.

### 2.2.4 Notebook Structure

The structure of the notebooks was laid out much like a book, with distinct sections and subsections of content separated with exercises. Each notebook was split into two or three sections each tackling a distinct area of study. Notebook 1 contains **Vector Spaces** and **Vector Calculus**. Notebook 2 contains **FEM in 1D**, **FEM in 2D**, and **Time Dependent Problems**. Notebook 3 contains **Introduction to FEniCSx** and **A Time Dependent Example**.
Development of the exercises was another important part of the design. Exercises serve to test understanding and cement newly ingested information. They can also be great at challenging possibly incorrect assumptions students may have picked up from the content. After each subsection in a notebook, there were a collection of "check" exercises. These ranged from simple calculations, to proving statements in the text, to explanations and sometimes more complicated derivations. For the computing side, exercises were harder to prescribe for each subsection, since an entire section spanned a single example. However, at the end of each section, a chance to apply the methods learned was given through way of practise question. There were even exercises that linked between each other in building a full PDE approximation, which was intended to leave a sense of learner satisfaction in creating their own approximation the whole way through.

## 2.3 Development Methodology

At the beginning of the project a plan was devised to ensure all tasks were completed on time, allowing for equal effort to be spent in all areas of research, development and testing. This plan took the form of a Gantt chart, a type of bar chart used for visualising schedules.
In Figure 2.6, most of the time spent was planned to be on the background research. This makes sense because in order to write about FEM one must ensure they have a very solid understanding. From early February, the development procedure was to begin.
Originally, the plan was to use an Agile development methodology, where a notebook would be created over a week and then testers would review the content and give immediate feedback on

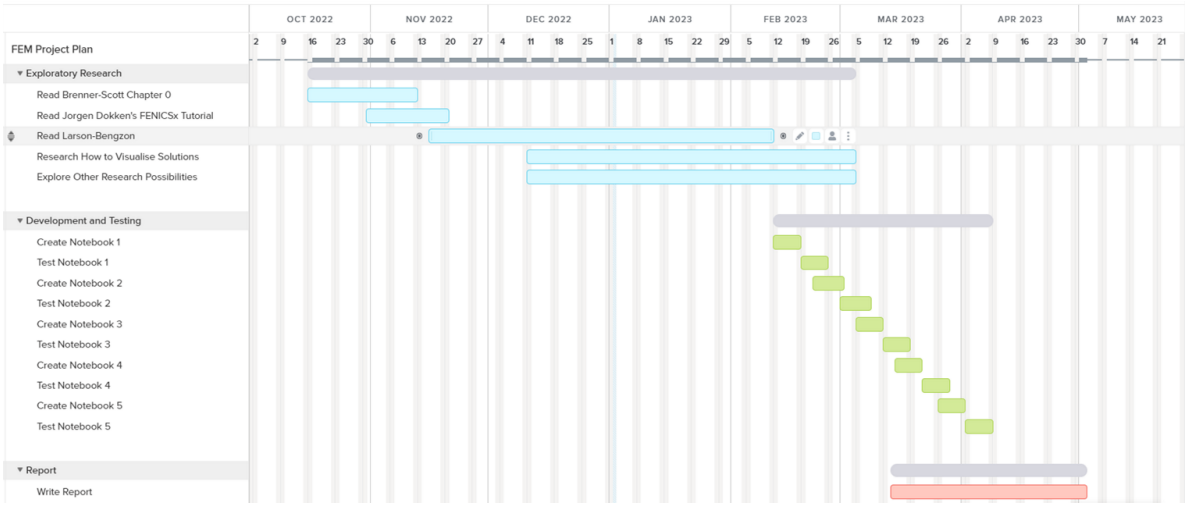



Figure 2.6: Gantt chart for the project

things they liked and didn't like about the product. This process would have involved a constant dialogue between myself and the tester continually improving the products until the next notebook was able to begin its development. This would have helped refine a product to the standards of typical users much quicker than other possible methodologies. There were a few problems however. Firstly, it required some testers who had in-depth knowledge of FEM, so that they could give feedback from an educated point of view, rather than being educated by the resource itself. There would also be a place for testers who had enough background to view the content for the first time. It was difficult to find testers who had the required knowledge and would be willing to submit themselves to such an involved procedure. Unfortunately, I could only obtain three testers who were willing and they were all students. This meant that they had commitments with their own final year projects and their studies and a constant dialogue was not possible.

This unfortunately meant that the project had to be taken in a different direction, using a strictly waterfall-based approach. The benefit of this was the burden of communication and waiting on feedback was not present. Work could commence immediately. The role of the testers would be to instead provide evaluation once the notebook had been completed, which would give an insight into the success of the notebooks.

Throughout the project, GitHub was used as the version control system. Two repositories were created, `Finite-Element-Method-Project` and `FEMLearningResources`. The first was made to host all files relating to the project, the one intended for submission and the one containing this report. The second was made for testers. They didn't need to see the whole project, they only needed access to the notebook files. Branching structures were not used for the repositories, there was only a main branch in each and commits were pushed directly to this. If the project had multiple contributors and there had been a possibility of a merge conflict, separate feature branches may have been necessary, but in this case it wasn't.

# Chapter 3

# Results

## 3.1 User Feedback

In order to evaluate the success of the solution produced, a method of collecting user feedback had to be developed. This was done by collecting a group of eligible testers who not only had relevant background to understand the material, but were also willing to devote time to the study and review the material. Then, a consistent and precise way of collecting the information had to be created so that a measurement of the success of the notebooks could be collected. The information assessing the notebooks would be gathered through a series of questionnaires, one for each notebook, which were hosted on Google Forms. A link to the questionnaire can be found here. In order to assess the success of the notebooks, the questions were tailored to points on the Learning Resource Requirements from Section 1.4. The survey consisted of a collection of questions aimed to assess how well these requirements were met and each question related to a specific requirement. Every question started with a description of the requirement in natural terms to give the tester appropriate background. Then they were asked to rate on a score of one to ten how well this objective was met. The final questionnaire can be seen in Appendix C at the end of the report.
The only difference to the requirements laid out earlier is the inclusion of a question asking how interesting the content was. This was not included in the requirements since it is a fairly personal question. A user might find the content interesting because they find the subject itself interesting, regardless of the quality of writing. The opposite is also true. As such, this metric is not as important, although interesting to measure since there could still be some correlation between quality and interest. A more engaging and excitingly written resource would lead to the user finding the resource more interesting.
The scoring system is a simple one to ten score, the interpretations vary however. For most questions a higher score indicates a better result in terms of quality. This is not true for all questions, for instance "How difficult was the content to understand?" is question that ideally would be placed at a five or six.

## 3.2 Notebook 1: Mathematical Review

We will now review the actual content produced. Each notebook starts with an introduction and a set of learning objectives that mirror those set out in the content plan. It is difficult here to give a complete description of each notebook, since it is a piece of written content and not code. The reader of this report is encouraged to view the notebooks themselves in the main repository.

### 3.2.1 Section 1: Vector Spaces

Notebook 1 Section 1 leads with vector spaces, a classic linear algebra topic. The definition of a field is introduced in order to further introduce the concept of a vector space. The reader is expected to recall the definition of a group and the axioms for a group are given in plain language, not given in mathematical notation. With these axioms and the additional axioms classifying a group as a vector space, the full definition is given. After this a few examples of vector spaces are given. In FEM, the only vector spaces we are concerned about are function spaces, but vector spaces are introduced more generally since the ideas of basis and linear independence are universal over all types of vector space. No check exercises were needed here.

Then a **Vector Space** over a field $F$ is a group $(V, +)$, satisfying all group axioms, together with a multiplication operation $\times$, defined between vectors and elements of $F$. A vector space must satisfy the following axioms for all $1, a, b \in F$ and $\underline{u}, \underline{v} \in V$,

$$1 \times \underline{v} = \underline{v} \quad \text{(Identity)}$$
$$a \times (\underline{u} + \underline{v}) = a \times \underline{u} + a \times \underline{v} \quad \text{(Distributing a scalar over vectors)}$$
$$(a + b) \times \underline{v} = a \times \underline{v} + b \times \underline{v} \quad \text{(Distributing a vector over scalars)}$$
$$a \times (b \times \underline{v}) = (a \times b) \times \underline{v} \quad \text{(Associativity of multiplication)}$$

Notice that commutativity is not necessary.

Linear independence is the next subsection. The ideas of span, linear combinations and linear dependence are reviewed. A derivation is made that motivates the idea that a matrix whose columns consist of linearly independent vectors has a non-zero determinant. The check exercises consist of fairly simple questions testing the reader has understood linear independence. The first is to prove a set is linearly independent, which can be done by constructing the matrix and showing it has non-zero determinant. The other is to prove a set in linearly dependent. This is done by finding a linear combination of vectors that produce another vector in the set. This could be done by inspection or trial and error, but the example was written to be too difficult to be done this way, so instead the reader is encouraged to find the linear combination by Gaussian Elimination.
The next subsection is about bases. The idea is introduced and then three examples are introduced. This is a fairly straightforward section. The check exercises here combine ideas from the previous sections, testing for analysis in Bloom's taxonomy. For instance, explaining why a set of vectors of size $n + 1$ from a vector space of dimension $n$ cannot be linearly independent uses the idea of basis, dimension and linear independence.

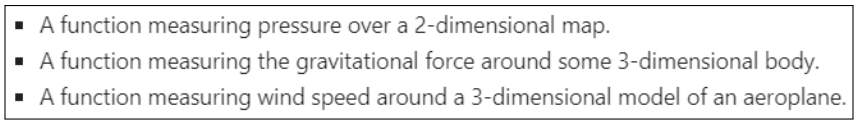
Check Excerises 2

- Prove that $\{(2, 0, 5), (1, -1, 1), (-3, 0, 0)\}$ forms a basis over $\mathbb{R}^3$.
- Prove that all bases of a vector space have the same size.
- Explain why a set of vectors of size $n + 1$ from a vector space of dimension $n$ cannot be linearly independent.
- Provide a basis for the vector space of *all* real-valued polynomials $P(\mathbb{R})$ and state it's dimension.

### 3.2.2 Section 2: Vector Calculus

This section aims to refresh students on vector calculus definitions, multiple integration and the divergence theorem. We start with a simple refresher on the idea of fields and mutlivariable functions, which leads directly into the idea of the partial derivative. The check exercises for this section ask the reader to state whether certain physical examples are vector or scalar fields.

A second question tests the reader's differentiation skills by asking them to evaluate a partial derivative.




The next subsection refreshes the reader on a few common differential operators. Gradient is the most important one, but divergence is also very important and is used in most of our weak formulations later on. A few paragraphs are devoted to explaining these and their interpretations in terms of fields. Curl and the Laplacian are also mentioned, since there are certain PDEs where these are useful. While explaining curl and its physical interpretation, a few diagrams are included. To finish the subsection, a question is asked about each of the differential operators introduced. The reader is asked to prove that gradient is a linear operator, to find the divergence of a field and ascertain whether it is incompressible, and to calculuate the curl of a field used in a previous exercise to verify it is irrotational.
The third subsection refers to multiple dimensional integration. It begins by discussing the idea of surface integrals over regions in the $xy$-plane. This is a precursor to the next point, which is on surface integrals over curved surfaces in three dimensional space. Volume integrals are mentioned, as a formula is presented that can be used to calculate a surface integral. This formula is not derived, but a derivation is not necessary at every stage since readers should still be familiar with these topics. Then an example of using this formula is run presented, where the function $f = 1$ is integrated over the surface of a sphere, which derives the surface area of a sphere. To finish out the subsection, the idea of flux integrals are presented, although an example was not given since it won't be strictly necessary to compute them. The check exercises consist of several questions asking readers to calculate certain surface integrals. They are also given the cylindrical coordinate transformation and asked to use this to calculate the surface area of a cylinder, a classic example.
The final subsection discusses the Divergence Theorem and Green's Identities. These theorems are very useful for deriving variational forms for use with FEM, and are the whole reason so much attention was given to surface and flux integrals. The divergence theorem is presented without proof, while Green's first identity is derived from this. The final exercise asks the reader to derive Green's second identity, with the appropriate substitution they need to perform being given.

## 3.3 Notebook 2: Introduction to the Finite Element Method

### 3.3.1 Section 3: Finite Element Method in 1D

Notebook 2 is the first with new content. It opens with a section talking about the purpose of FEM and presents the example PDE that is solved throughout the notebook. Then, the first concept of a mesh is introduced, the simple one-dimensional case. The space of linear polynomials $P_1(I)$ is defined after this, along with its basis. This is the same course taken by Bengzon and Larson. Since the space of continuous piecewise linear polynomials is just a combination of linear polynomials, it makes sense to start at the most fundamental part, and

build upwards. Then some visuals are presented illustrating the difference between continuous and discontinuous piecewise linear functions, which were created in Matplotlib. Finally, the space of piecewise continuous linear polynomials $V_h$ is presented, along with its basis of hat functions, which are key to understanding FEM. The exercises ask the reader to use knowledge from notebook 1 to prove the set of hat functions actually forms a basis on $V_h$. It then asks what its dimension would be on a mesh with $n+1$ nodes.

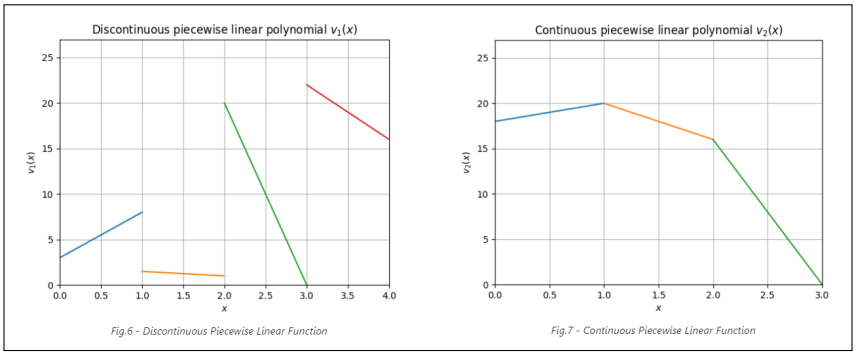



*Fig.6 - Discontinuous Piecewise Linear Function* *Fig.7 - Continuous Piecewise Linear Function*

The next chapter is a bit more dense with information, it introduces the idea of the weak formulation and derives the formulation for the example equation presented in the introduction, the Poisson equation. A brief amount of time is spent explaining the classes of the function spaces involved. This is more to satisfy reader curiosity and provide a sense of completeness, but the actual functional analysis involved with defining these spaces properly is beyond the scope of this project. This allows us to produce our finite element approximated problem, similarly to Section 1.2.1. The exercise here presents a simplified version of the one-dimensional heat equation called the steady state heat equation. The reader is asked to derive its variational form in the same way as the example.
The next subsection is short, it moves from the variational form to a system of linear equations through some derivations. The derivation is the same as the one used in Bengzon-Larson, with some extra lines added for additional clarity. The exercise asks the reader to derive the system of linear equations for the steady state heat equation.

Recall that a function $v \in V_{h,0}$ can be written $v(x) = \sum_{i=1}^{n-1} c_i \phi_i(x)$ (note the lack of half-hats). Then (5) becomes,

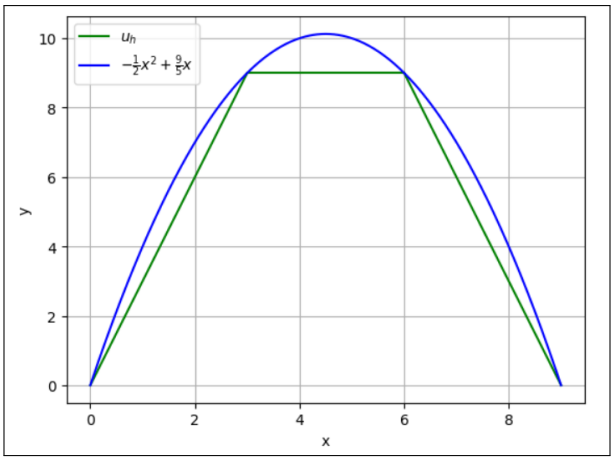
$$\int_I \left( f \sum_{i=1}^{n-1} c_i \phi_i(x) \right) \mathrm{d}x = \int_I \left( u_h' \sum_{i=1}^{n-1} c_i \phi_i'(x) \right) \mathrm{d}x$$

$$\sum_{i=1}^{n-1} c_i \int_I f \phi_i(x) \, \mathrm{d}x = \sum_{i=1}^{n-1} c_i \int_I u_h' \phi_i'(x) \, \mathrm{d}x$$

$$\sum_{i=1}^{n-1} \int_I f \phi_i(x) \, \mathrm{d}x = \sum_{i=1}^{n-1} \int_I u_h' \phi_i'(x) \, \mathrm{d}x$$

$$\int_I f \phi_i(x) \, \mathrm{d}x = \int_I u_h' \phi_i'(x) \, \mathrm{d}x, \quad i = 1, 2, \cdots, n-1$$

Finally, the section ends with a complete example, solving a very simple one dimensional differential equation. This example can be solved analytically by simply integrating twice and fixing boundary conditions. The reason for such simplicity is to provide a way of comparing the accuracy and showing that the method does, in fact, work. A mesh of three cells is used to calculate the finite element approximation by hand. This is then plotted against the analytical solution using Matplotlib. Then some additional points are mentioned about the convergence of FEM, types of quadrature, linear equation solvers and higher dimensional piecewise continuous function spaces.




### 3.3.2 Section 4: Finite Element Method in 2D

The subsections here mostly parallel Section 3. We start with an introduction to piecewise polynomial functions of two variables although we skip the more intricate definitions in favour of a blunt approach, since the method of understanding the two are very similar. The real idea of a mesh is introduced and some FEniCSx code is presented that generates a mesh over a circle and visualises it with Pyvista.
After this, the variational form is defined for the two dimensional Poisson equation, here using Green's first identity derived in the previous notebook. Then, as before, the linear system is constructed. The final subsection of the 2D portion discusses boundary conditions and modelling. The concepts of Dirichlet, Neumann, and Robin boundary conditions are introduced and explained. The strengths and weaknesses of FEM are presented as well as the final steps for performing it.

1. Define a mesh over your solution space. The greater the mesh size, the more accurate your solution will be.

2. Derive the variational form of your problem. This is done in 3 steps:

   - Multiply both sides of the PDE by a test function $v$.
   - Integrate over your domain.
   - Perform integration by parts and use the fact that $v = 0$ on the boundary to simplify in a linear and bilinear form.

3. Construct a linear system of equations by replacing your test and trial spaces, with a finite element subspaces and substituting in your test and trial functions as a linear combination of basis functions from these subspaces.

4. Solve the resulting linear system.

### 3.3.3 Section 5: Time Dependent Problems

This short section aims to introduce the idea of time dependent problems in theory, for practical implementation later. The example presented is the 2D heat equation with Dirichlet boundary conditions.

$$\begin{aligned} \frac{\partial u}{\partial t} &= \nabla^2 u + f && x \in \Omega,\ t \in [0,T] \\ u &= u_D && x \in \delta\Omega,\ t \in [0,T] \\ u &= u_0 && t = 0 \end{aligned}$$

The method presented is called the backwards Euler method, where time is discretised and a

difference quotient is used to approximate the derivative.

$$\left(\frac{\partial u}{\partial t}\right)^{n+1} \approx \frac{u^{n+1}-u^n}{\Delta t}$$

$$\begin{aligned}\frac{u^{n+1}-u^n}{\Delta t} &= \nabla^2 u^{n+1} + f^{n+1}\\ u^{n+1}-u^n &= \nabla^2 u^{n+1}\Delta t + f^{n+1}\Delta t\\ u^{n+1} - \nabla^2 u^{n+1}\Delta t &= u^n + f^{n+1}\Delta t\end{aligned}$$

The variational form is not derived, but the finite difference approximation is. Then finally, the processes of interpolation and $L^2$ projection are discussed. This is necessary because when solving a system with an initial condition using Euler's method and FEM, the initial condition has to be from the same space as the solution and so must be approximated in that space. This is done via these techniques. The final exercise asks the reader to complete the derivation of the variational form of the two-dimensional heat equation. A challenge that brings together all the work done so far, but definitely within the realm of understanding.

## 3.4 Notebook 3: Implementing FEM in Python with FEniCSx

### 3.4.1 Section 6: Introduction to FEniCSx

The final notebook intends to show readers how to implement some simple equations in the previously finite element computing platform FEniCSx. It begins with an introduction to FEniCSx and each of its main constituent libraries as in Section 1.2.3. The section will be devoted to solving the following instance of the Poisson equation.

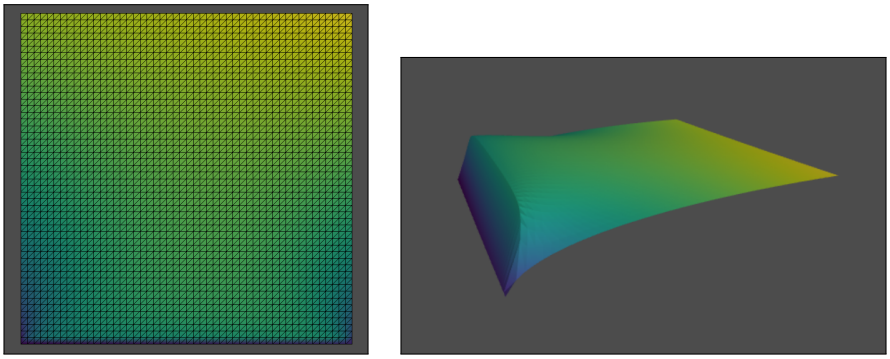
$$-\nabla^2 u(x,y) = f = 5y^{-\frac{3}{2}} - 1 \qquad \text{in } \Omega \tag{3.1}$$

$$u(x,y) = u_D = 20y^{\frac{1}{2}} + \frac{1}{2}x^2 + 2x \qquad \text{on } \delta\Omega, \tag{3.2}$$

$$\text{where } \delta\Omega = [0,1]\times[0,1] \tag{3.3}$$

This BVP was created using the method of manufactured solutions, a method that guarantees an analytical solution will exist. The solution was designed so that the boundary condition is the solution. As before, this is so we can compare our approximated solution to the one produced. The majority of the code is adapted from Dokken's tutorial but has been changed to fit the example presented. A further degree of complexity has been added to the problem from Dokken's example by the inclusion of fractional powers.
First, the boundary is defined as the unit square in Gmsh. Code performing this is shown, along with a block creating a visualisation in Pyvista. Next, all the problem parameters are defined, including the function spaces, boundary conditions, test and trial functions, linear and bilinear terms in UFL and the PETSc linear problem solver. Intuitive explanations are given every step of the way. A few paragraphs are also given to the numerical methods happening behind the scenes, with attention being given to LU Factorisation. With the problem wholly defined, it is then easy to solve using PETSc.
With the equation solved, the next thing to teach was how to visualise the solution obtained.

This was done over a few methods. Firstly, Pyvista was used. The approach was the exact same as visualising the mesh earlier, the difference being that the nodes are coloured based on the values of the degrees of freedom of the approximated solution. This plotter can then be extended to three dimensional space with the `warp_by_scalar()` method.



The other method of visualisation presented was by plotting the value of the function along a straight line through the domain, a so called "slice" of the function. This method is covered by Dokken in his elastic deformation chapter, but is used here in a different context with much more explanation. The process can be broken down into three steps: 1. Define all the points on the line $x = 0.5$ (the middle of the domain). 2. Evaluate the approximate solution at these points. 3. Plot these values against the points on the line.
Numpy's array manipulation techniques are first used to define the points on the line. DOLFINx's way of evaluating the function over a set of point efficiently involves the use of Axis Aligned Bounding Boxes. A detailed description of the algorithms and data structures involved is given, with this Azure From The Trenches post [26] being the main source.

```
Function Binary_Search( x, i ):
    if ( x is inside bounding box denoted by vetex i ):
        if ( i is a branch ):
            return [Binary_Search( x, i.L ), Binary_Search( x, i.R )]
        else if ( i is a leaf ):
            return node

Function Find_Cell_Containing( x ):
    potential_cells = []
    potential_cells.append(Binary_Search( x, root ))
    potential_cells.flatten()

    cells = []
    for c in potential_cells:
        if ( x is inside c ):
            return cells
```

Then, with $u_h$ evaluated over the points, the curve is plotted in Matplotlib. This process was repeated multiple times with varying mesh sizes to demonstrate the convergence to a final solution. This was then compared to the analytical solution to show readers that the solution is correct. Finally, the complete set of steps for solving a BVP in FEniCSX is summarised and an exercise is presented asking readers to solve an instance Poisson equation in FEniCSX.

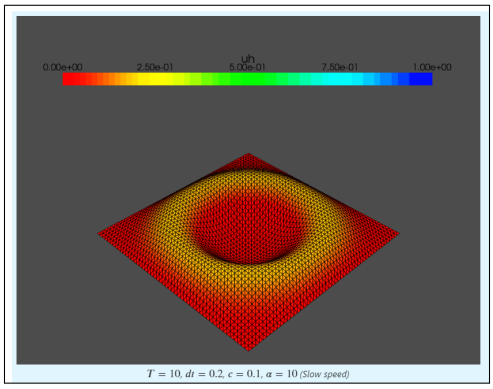
### 3.4.2 Section 7: A Time Dependent Example

The final section of the final notebook is devoted to solving a time dependent problem in FEniCSx. This chapter aims to illustrate the power and versatility of FEniCSx. The first

subsection introduces the problem to be solved, the wave equation. This is based on Dokken's tutorial on the heat equation, but has been adapted for a different example. The main difference between the heat equation and the wave equation is that the heat equation contains a first order derivative with respect to time, while the wave equation contains a second order derivative. This means in approximating with a difference quotient, a second order one must be used, which changes the structure of the problem. The first subsection introduces the instance to be solved, and derives the linear and bilinear terms from this.

$$
\begin{aligned}
L(v) = \iint_\Omega v\left(2u^n - u^{n-1}\right)\mathrm{d}S &= \iint_\Omega v\left(u^{n+1} - (\Delta t)^2 c^2 \nabla^2 u^{n+1}\right)\mathrm{d}S \\
&= \iint_\Omega v u^{n+1}\mathrm{d}S - (\Delta t)^2 c^2 \iint_\Omega v\nabla^2 u^{n+1}\mathrm{d}S \\
&= \iint_\Omega v u^{n+1}\mathrm{d}S - (\Delta t)^2 c^2 \left(\iint_\Omega \nabla v\cdot\nabla u^{n+1}\mathrm{d}S - \oint_{\delta\Omega} v\nabla u^{n+1}\cdot\underline{\mathrm{d}S}\right) \\
&= \iint_\Omega v u^{n+1}\mathrm{d}S - (\Delta t)^2 c^2 \iint_\Omega \nabla v\cdot\nabla u^{n+1}\mathrm{d}S \\
&= \iint_\Omega \left(v u^{n+1} - (\Delta t)^2 c^2 \nabla v\cdot\nabla u^{n+1}\right)\mathrm{d}S = a(u^{n+1}, v) \quad \forall v\in V_h, \quad n = 0, 1, \cdots
\end{aligned}
$$

The final subsection contains the implementation. The problem is defined in FEniCSx, along with all parameters such as time-step size, mesh size and some constants in the equation. Everything is mostly the same as the previous section until we reach the point of solving. Instead of constructing a linear problem like before, a linear problem needs to be created at every time step. If one were to create such a large number of new PETSc linear problems, the performance impact would be quite considerable. Instead a more clever approach is used by noticing that the mass matrix stays constant throughout all the linear problems. FEniCSx has functionality built in to allow the mass matrix and load vector to be built separately, so only the load vector is built at each time step.
The implementation of this is shown, and to visualise this time dependent solution, every frame is rendered into a `.gif` file. The result is an animation, of which one frame can be seen below.




Four example animations are presented with different parameters, but the reader is encouraged to experiment to see what other kinds of interesting instances they can produce. The final exercise asks the user to create a finite element approximation to an instance of the heat equation of their choosing. A fitting end since if they can complete this exercise, they will have successfully create their own finite element approximation from start to finish!

# Chapter 4

# Discussion

## 4.1 User Feedback and Analysis

We will now discuss the results of user feedback and what they mean for the success of the project.

### 4.1.1 Quantitative Results

| Name | Notebook | Q1 | Q2 | Q3 | Q4 | Q5 | Q6' | Q7 | Q8 | Q9 | Average | Standard Deviation |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rebecca Potts | 1 | 6 | 7 | 9 | 10 | 8 | 4 | 10 | 6 | 10 | 7.78 | 2.17 |
| Rebecca Potts | 2 | 9 | 9 | 10 | 9 | 7 | 6 | 8 | 6 | 9 | 8.11 | 1.45 |
| Rebecca Potts | 3 | 9 | 10 | 10 | 9 | 8 | 4 | 9 | 8 | 9 | 8.44 | 1.81 |
| Anthony Moran | 1 | 8 | 8 | 10 | 10 | 8 | 10 | 10 | 6 | 8 | 8.67 | 1.41 |
| Anthony Moran | 2 | 8 | 10 | 10 | 10 | 6 | 6 | 10 | 5 | 7 | 8 | 2.06 |
| Anthony Moran | 3 | 7 | 9 | 8 | 10 | 6 | 10 | 8 | 6 | 7 | 7.89 | 1.54 |
| Callum Anderson | 1 | 8 | 7 | 8 | 10 | 10 | 8 | 8 | 6 | 8 | 8.11 | 1.27 |
| Callum Anderson | 2 | 6 | 9 | 10 | 8 | 6 | 6 | 8 | 8 | 8 | 7.67 | 1.41 |
| Callum Anderson | 3 | 7 | 7 | 8 | 9 | 8 | 6 | 7 | 8 | 8 | 7.56 | 0.88 |
| Average | ALL | 7.56 | 8.44 | 9.22 | 9.44 | 7.44 | 6.67 | 8.67 | 6.56 | 8.22 | 8.03 | 1.56 |

Figure 4.1: Quantitative results of user feedback

The numerical results have been compiled into a table. Recall that Question 6 asked the user how difficult the notebooks were, with a more balanced result being more preferable. In hindsight, this was probably not a good idea. The question should have been phrased "How balanced was the notebook?". This is because in order to compute averages and perform other statistics correctly, all questions need to be on the same scale. Since the questionnaire has already been completed, the questions cannot be changed, so instead a transformation is applied to get the result in the same scale as the others. The closer to the middle, the higher the output. So 5 or 6 should be a 10, 5 or 7 should be an 8 and so on. The problem with this approach is that it increases the standard deviation of answers to this question, but it is the best we can do.

A few important statistics have been calculated in Figure 4.1. Firstly, averages have been calculated for each submission along with the standard deviation. Then along the bottom row, averages have been calculated for each question and then finally the complete average and average standard deviation. This is a fairly crude metric of measuring success, since the sample size is quite small. Another problem with these statistics is that in order for the average score to equate with quality, we need to make the assumption that each question has equal impact on the quality of the notebook. This is just not true. It is hard to say what aspects are more important, and certain requirements might not even have a linear impact on the quality, for instance some might have diminishing returns.

However, if we do make that assumption, an average score of 8.03 quite good. The average standard deviation is 1.56, which is also acceptable. The lowest score presented is a 5 with all

others being much higher than this. The lowest scoring question on average is Q8, asking how interesting the content was. The reason for such a low score here is probably due to sample size. This question is largely based on personal opinion and background. If one is not disposed to enjoying numerical methods and the mathematics involved, they may not find personal satisfaction in reading about a fairly complex method in that field. Based on the result it seems this cohort didn't have a particular love for the subject. The highest scoring question on average was Q4 which asked how prepared the readers were for the content presented to them. This is no surprise, all of the testers have a very similar background and the content was designed specifically to cater to this. It also means that content flowed well and the structure helped in preparing readers for upcoming content.

### 4.1.2 Qualitative Results

The testers were also encouraged to provide qualitative feedback through the form of comments boxes for each question. Some very useful feedback was provided, but it is too much to entirely cover here. However, some recurring points and potential problems will be highlighted.
Readability was overall rated well. Each tester had their own views, but largely it was said to be easy to read. One of their main concerns was with typos and grammatical mistakes.
Another potential addition that sounded like a good idea was more underlining/emboldening of key terms, as this was already done to a small extent. For graphs and other visuals, the response was largely positive. All said they appreciated their inclusion and one said they "made it easier to understand the content". There were suggestions for more animated content and one tester found the curl diagrams a bit confusing.
Aesthetics had few comments, which makes sense since it is a very small part of what makes an effective notebook. This scored well and comments largely said that LaTeX looks very professional. There were a few comments about scroll bars appearing around long sections of text, which wasn't a problem on my system but I can understand how this might break up the flow of reading. Preparedness was good and its responses correlate with the fact it was the highest scoring question. One said that Notebook 3 "built well on their knowledge of numerical methods".
Usefulness is next and it seems some users expected to see more real-world application and motivations. The main user asking for this has a background in physics, which explains why their interest lies in the practical applications. Despite this potential bias, it is true that there could have been slightly more links to physical problems, since this is the motivation for studying FEM in the first place. Difficulty was an interesting one, it seemed by far that notebook 2 was the most difficult for people to digest, scoring 8 from all three testers. The other two notebooks were quite well balanced and people found them easier than hard. I think the perceived difficulty of notebook 2 can be explained by the density of mathematical material. Some said "it became difficult to remember what [the symbols] meant" and "I thought it was pretty tough content".
Structure was also rated quite well, tester said the material "flowed naturally" and was "well broken down in the correct order". The interest comments were another particularly personal one, as discussed before this did not score as well as the others and the comments reflect this. One tester said "it's not boring or anything, its just pure maths" for notebook 2. One even said

"for the most part I didn't understand a lot, which also affected my lack of interest". This is unfortunate to hear, since the burden is on the author to make a subject accessible and easy to understand, not the student. However, with a larger sample size one would hope to see that students who have these kinds of experiences become the outliers. There were still positive comments, people seemed to find the gifs at the end of notebook 3 quite engaging. One said it "made learning all the theory seem worth it".
The final comments were instructive, people recommended that summaries be added to the end of the notebook, which seems like a good idea that was not thought of at the time. Some expressed an interest in more colour, more exercises and some specific changes, but overall the feedback was positive.

## 4.2 Conclusions

Overall, the project goals were met. The result is not fully polished, and could possibly be more extensive in its scope, but it succeeded in teaching students that the theory and implementation of the Finite Element Method is a fascinating subject. On the whole, testers walked away satisfied that they had learned something, even if they wouldn't personally use it for anything themselves.
With the notebooks designed in a scientific manner using Bloom's taxonomy and other sources to devise a criteria for successful learning resources, it can be said that the result is backed by existing research in the field of education.

## 4.3 Ideas for future work

There are many ways this project could be expanded. Firstly, and most obviously, a more complete resource could be created. This would span many more types of PDE, including the Navier-Stokes fluid flow equations, Maxwell's electromagnetic equations, elastic deformation equations and more. Different solvers could be covered, other than LU Factorisation. Advanced meshing techniques like adaptive mesh refinement, complex valued functions, different types of finite element, a more in-depth look at function spaces. The list is endless because the field is so deep.
Other than expanding the syllabus, one could try using more modern technologies. Services like Brilliant and Khan Academy use sleek web design and videos to provide a more complete and smooth experience. Inspiration could be taken from them to build a web interface that hosts all the content. At the very least, getting Binder to work would help make the experience a bit smoother for students.

# References


[1] Michael Renardy and Robert C. Rogers. *An Introduction to Partial Differential Equations*. Number 13 in Texts in Applied Mathematics. Springer=Verlag, New York, 2 edition, 2004.

[2] Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*. Number 15 in Texts in Applied Mathematics. Springer Science + Business Media, New York, 2008.

[3] Mats G. Larson and Fredrik Bengzon. *The Finite Element Method: Theory, Implementation, and Applications*. Number 10 in Texts in Computational Science and Engineering. Springer-Verlag, New York, 2013.

[4] W. Ritz. Über eine neue Methode zur Lösung gewisser Variationsprobleme der mathematischen Physik. *Journal für Reine und Angewandte Mathematik*, 135, 1908.

[5] J. Singer. On Equivalence of the Galerkin and Reyleigh-Ritz Methods. *Journal of the Royal Aeronautical Society*, 66(621):592, 1962.

[6] M.J. Turner, R. W. Clough, H.C. Martin, and L.J. Topp. Stiffness and Deflection Analysis of Complex Structures. *Journal of the Aeronautical Sciences*, 23(9), 1956.

[7] I. Babuska and A.K. Aziz. Lectures on mathematical foundations of the finite element method. Technical report, United States, 1972.

[8] E. Toni. Variational formulation for every nonlinear problem. *International Journal of Engineering Science*, 22(11/12):1343–1371, 1984.

[9] Geometric Computing Lab. PolyFEM. https://polyfem.github.io, NYU Courant Institute of Mathematical Sciences and the University of Victoria, Canada, 2023.

[10] Daniel Arndt, Wolfgang Bangerth, Marco Feder, Marc Fehling, Rene Gassmöller, Timo Heister, Luca Heltai, Martin Kronbichler, Matthias Maier, Peter Munch, Jean-Paul Pelteret, Simon Sticko, Bruno Turcksin, and David Wells. The `deal.II` Library, Version 9.4. *Journal of Numerical Mathematics*, 30(3):231–246, 2022.

[11] F. Hecht. New development in FreeFem++. *Journal of Numerical Mathematics*, 20(3-4):251–265, 2012.

[12] FEniCS Project. FEniCSx. https://fenicsproject.org/, 2023.

[13] Benjamin S. Bloom (Ed.), Max D. Englehart, Edward J. Furst, Walker H. Hill, and David R. Krathwohl. *Handbook 1: Cognitive Domain*. Taxonomy of Educational Objectives - The Classification of Educational Goals. David McKay Company, Inc., 1956.

[14] Growth Engineering. Bloom's Taxonomy : Master Your Learning Objectives. https://www.growthengineering.co.uk/what-can-blooms-taxonomy-tell-us-about-online-learning/, 2022.

[15] Lorin W. Anderson, David R. Krathwohl, Peter W. Airasian, Kathleen A. Cruikshank, Richard E. Mayer, Paul R. Rintrich, James Raths, and Merlin C. Wittrock. *A Taxonomy for Learning, Teaching, and Assessing - Revision of Bloom's Taxonomy of Educational Objectives*. Addison Wesley Longman, Inc., 2001.

[16] Layal Hakim. Top tips to improve the teaching of mathematics in universities. https://www.timeshighereducation.com/campus/top-tips-improve-teaching-mathematics-universities, Times Higher Education, University of Exeter, 2022.

[17] Hans Petter Langtangen and Anders Logg. *Solving PDEs in Python: The FEniCS Tutorial I*. Springer International Publishing, 2016.

[18] Thomas Kluyver, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[19] John Gruber. Daring Fireball | Markdown: Syntax | Philosophy. https://daringfireball.net/projects/markdown/syntax#philosophy, 2013.

[20] Davide Cervone and Volker Sorge. MathJax. https://www.mathjax.org/, 2013.

[21] J. S. Dokken. DOLFINx Tutorial Docker Image. https://github.com/jorgensd/dolfinx-tutorial#docker-images, GitHub, 2023.

[22] Jupyter et al. Binder 2.0 - Reproducible, Interactive, Sharable Environments for Science at Scale. https://conference.scipy.org/proceedings/scipy2018/project_jupyter, 2018.

[23] Jupyter et al. Binder User Guide | Use a Dockerfile for your Binder repository. https://mybinder.readthedocs.io/en/latest/tutorials/dockerfile.html#preparing-your-dockerfile, 2022.

[24] Wolfram. Wolfram MathWorld. https://mathworld.wolfram.com/.

[25] Jørgen S. Dokken. The FEniCSx tutorial, 2022.

[26] James. Introductory Guide to AABB Tree Collision Detection. https://www.azurefromthetrenches.com/introductory-guide-to-aabb-tree-collision-detection/, Azure From The Trenches.

# Appendix A

# Self-appraisal

## A.1 Critical self-evaluation

Broadly speaking I would say the project was a success, although there were certainly things I would do differently if I were to do it again. The quality of the notebooks are quite high, and I feel the content I decided to cover was covered well. This being said, there is also room for a lot more and I wish I had managed my time better in order to be able to cover these topics. One of the main reasons I chose this area of research was due to my interest in fluid flow problems and the Naiver-Stokes equations. In order to learn about these, I had to first obtain a strong base knowledge in the finite element method, which is a fairly dense and sometimes inaccessible topic. As such, it took a long time to get to a place where I was comfortable with the majority of the terms and ideas which meant I didn't have enough time to research the fluid flow problems I chose the topic for.

The first text I read was *The Mathematical Theory of Finite Element Methods* by Brenner and Scott. This is a high-level mathematical text, suited for mathematicians with knowledge of functional analysis, which I lacked. This text put me off due to its complexity and made me feel that the topic was too complex for an undergraduate final year project. It gave me good initial insight however, and after finding the Bengzon-Larson textbook, I felt much more confident in the way forward, but much time had passed until I got to this point.

Another thing I would do differently is try to get more testers. Having three testers definitely impacted my ability to evaluate the success of the project effectively. The small sample size meant that individual biases stood out much more than they should have. Also none of the testers had a background in FEM, meaning potential inaccuracies in my explanations couldn't be identified as well. Cross-referencing against my sources, and taking great care was my attempt to minimise the inaccuracies, but there will undoubtedly be some present.

However, despite the drawbacks of the evaluation and problems with time management, I am personally very proud of the project. A lot of research was performed in a field I hadn't even heard of before starting. Sifting through complicated texts and tutorials developed my skills in critical reading and research.

It was interesting writing a project with a minimal amount of programming. Most projects develop a software product, and while this is still a software product, the majority of the implementation was just natural English text. This meant that past projects, while still very helpful, had many fundamental differences to mine. I think this project blurs the lines between a maths research dissertation and a computer science project, which made me doubt the validity of the idea. There were times where I thought about scrapping the idea and simply building a variety of solvers in FEniCSx, but so much time had been put into research it probably wasn't a good idea.

Also, if I had more time I would have produced a document containing the solutions to all exercises. Due to time constraints, I felt it was more important to get the content produced

and out to testers instead of focusing on the "quality of life" aspects. If this were a real product, the solutions would be of great importance, but since I was producing a minimum viable product, they were left as an additional consideration that I unfortunately never got round to completing. It should still be noted that I did complete all exercises to ensure they had the right degree of challenge and were, in fact, soundly constructed.

## A.2 Personal reflection and lessons learned

One of the main lessons learned was in time-keeping and maintaining a consistent workflow. Unfortunately, the Gantt chart proposed in Section 2.3 was not stuck to and the idea of Agile development had to be scrapped. This probably effected the quality of the final product slightly and it meant that one of the initial goals of the project had to be changed. I failed to stick to the timeline I proposed for a few reasons, one of which being an unfortunate set of circumstances at home, outside of my control. However, I should still take some personal responsibility in the matter and I know I could have devoted more time at the beginning. The problem was, I chose my modules for this academic year by choosing three quarters of my modules to be in the first semester and the other quarter in my second semester. This meant, over the course of the first term, almost no work was done towards completing the project aside from a minimal amount of reading. If I were to do this again, I would try to maintain a much more consistent workflow. I also feel I could have used the knowledge and advice of my supervisor better. Being independent is a useful skill, but knowing when to ask for help and letting others know if you are struggling is infinitely more so.

## A.3 Legal, social, ethical and professional issues

### A.3.1 Legal issues

For all sources used, I had to check if licensing permitted my use, replication, and modification of it. All the software sources used were open source and fell under various public licenses. All softwares that specify credit should be given have been.

### A.3.2 Social issues

The only social issue that might relate to the project is the idea of education and open-source content. Education is a right that is often taken for granted, and people in developing countries may not always have access to the same resources that we do. As such, if this were a real product being released, I would make it open-source and free for everyone. With many of the proprietary and advanced FEM softwares and books on the subject being kept behind paywalls, open source tools like FEniCSx and the tutorials surrounding it are crucial to allowing people to learn all around the world. Whether they go to university or not, education should be accessible and free for all (in my opinion).

### A.3.3 Ethical issues

The ethical issues of the project mainly surrounded user testing. In order to get the testers assistance, I created some user testing questionnaires explaining the project, some background information, their role within it and how their data would be stored. I specified they retain the right to leave the project at any time and that we could discuss options if they were unable to complete certain tasks. Luckily, this did not occur. They can be seen in Appendix C. While they are not legally binding in any sense, they do provide a good-will basis of communication. They serve more as a consent for use of their data than a binding agreement to participate. Outside of the consent forms, all communication regarding the project happened over Microsoft Teams and email, where I announced when notebooks were available for testing and provided links to the repository and questionnaire. Inside the repository there was a friendly but professional readme file that explained the goals of the project and what was expected of the testers.
I believe the project was conducted in an ethical and responsible way.

### A.3.4 Professional issues

There were no professional issues regarding this project. No external businesses or corporations were contacted all testing happened with University Of Leeds students.

# Appendix B

# External Material

This code is from J.S. Dokken's FEniCSx Tutorial [25] and was adapted to fit the examples presented.

Listing B.1: Solving the Poisson Equation

```
from mpi4py import MPI
from dolfinx import mesh
domain = mesh.create_unit_square(MPI.COMM_WORLD, 8, 8,
                                 mesh.CellType.quadrilateral)
from dolfinx.fem import FunctionSpace
V = FunctionSpace(domain, ("CG", 1))
from dolfinx import fem
uD = fem.Function(V)
uD.interpolate(lambda x: 1 + x[0]**2 + 2 * x[1]**2)
import numpy
# Create facet to cell connectivity required to determine boundary facets
tdim = domain.topology.dim
fdim = tdim - 1
domain.topology.create_connectivity(fdim, tdim)
boundary_facets = mesh.exterior_facet_indices(domain.topology)
boundary_dofs = fem.locate_dofs_topological(V, fdim, boundary_facets)
bc = fem.dirichletbc(uD, boundary_dofs)
import ufl
u = ufl.TrialFunction(V)
v = ufl.TestFunction(V)
from petsc4py.PETSc import ScalarType
f = fem.Constant(domain, ScalarType(-6))
a = ufl.dot(ufl.grad(u), ufl.grad(v)) * ufl.dx
L = f * v * ufl.dx
problem = fem.petsc.LinearProblem(a, L, bcs=[bc],
          petsc_options={"ksp_type": "preonly", "pc_type": "lu"})
uh = problem.solve()
import pyvista
from dolfinx import plot
pyvista.start_xvfb()
topology, cell_types, geometry = plot.create_vtk_mesh(domain, tdim)
grid = pyvista.UnstructuredGrid(topology, cell_types, geometry)
plotter = pyvista.Plotter()
plotter.add_mesh(grid, show_edges=True)
plotter.view_xy()
```

```
if not pyvista.OFF_SCREEN:
    plotter.show()
else:
    figure = plotter.screenshot("fundamentals_mesh.png")
u_topology, u_cell_types, u_geometry = plot.create_vtk_mesh(V)
u_grid = pyvista.UnstructuredGrid(u_topology, u_cell_types, u_geometry)
u_grid.point_data["u"] = uh.x.array.real
u_grid.set_active_scalars("u")
u_plotter = pyvista.Plotter()
u_plotter.add_mesh(u_grid, show_edges=True)
u_plotter.view_xy()
if not pyvista.OFF_SCREEN:
    u_plotter.show()
warped = u_grid.warp_by_scalar()
plotter2 = pyvista.Plotter()
plotter2.add_mesh(warped, show_edges=True, show_scalar_bar=True)
if not pyvista.OFF_SCREEN:
    plotter2.show()
```

Listing B.2: Extract from Deflection of a Membrane

```
from dolfinx import geometry
bb_tree = geometry.BoundingBoxTree(domain, domain.topology.dim)
cells = []
points_on_proc = []
# Find cells whose bounding-box collide with the the points
cell_candidates = geometry.compute_collisions(bb_tree, points.T)
# Choose one of the cells that contains the point
colliding_cells = geometry.compute_colliding_cells(domain, cell_candidates,
                                                   points.T)
for i, point in enumerate(points.T):
    if len(colliding_cells.links(i))>0:
        points_on_proc.append(point)
        cells.append(colliding_cells.links(i)[0])
points_on_proc = np.array(points_on_proc, dtype=np.float64)
u_values = uh.eval(points_on_proc, cells)
p_values = pressure.eval(points_on_proc, cells)
import matplotlib.pyplot as plt
fig = plt.figure()
plt.plot(points_on_proc[:,1], 50*u_values, "k", linewidth=2,
         label="Deflection ($\\times 50$)")
plt.plot(points_on_proc[:, 1], p_values, "b--", linewidth = 2,
         label="Load")
plt.grid(True)
```

```
plt.xlabel("y")
plt.legend()
# If run in parallel as a python file, we save a plot per processor
plt.savefig(f"membrane_rank{MPI.COMM_WORLD.rank:d}.png")
```

Listing B.3: The Heat Equation - Diffusion of a Gaussian Function

```
import numpy as np
from mpi4py import MPI
from petsc4py import PETSc
from dolfinx import fem, mesh, io, plot
# Define temporal parameters
t = 0 # Start time
T = 1.0 # Final time
num_steps = 50
dt = T / num_steps # time step size
# Define mesh
nx, ny = 50, 50
domain = mesh.create_rectangle(MPI.COMM_WORLD, [np.array([-2, -2]),
            np.array([2, 2])], [nx, ny], mesh.CellType.triangle)
V = fem.FunctionSpace(domain, ("CG", 1))
# Create initial condition
def initial_condition(x, a=5):
    return np.exp(-a*(x[0]**2+x[1]**2))
u_n = fem.Function(V)
u_n.name = "u_n"
u_n.interpolate(initial_condition)
# Create boundary condition
fdim = domain.topology.dim - 1
boundary_facets = mesh.locate_entities_boundary( domain, fdim,
                  lambda x: np.full(x.shape[1], True, dtype=bool))
bc = fem.dirichletbc(PETSc.ScalarType(0),
                     fem.locate_dofs_topological(V, fdim, boundary_facets),
                     V)
xdmf = io.XDMFFile(domain.comm, "diffusion.xdmf", "w")
xdmf.write_mesh(domain)
# Define solution variable, and interpolate initial solution for
# visualization in Paraview
uh = fem.Function(V)
uh.name = "uh"
uh.interpolate(initial_condition)
xdmf.write_function(uh, t)
import ufl
u, v = ufl.TrialFunction(V), ufl.TestFunction(V)
```

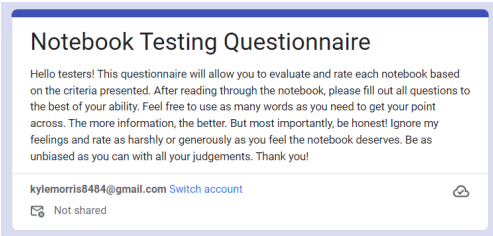
```
f = fem.Constant(domain, PETSc.ScalarType(0))
a = u * v * ufl.dx + dt*ufl.dot(ufl.grad(u), ufl.grad(v)) * ufl.dx
L = (u_n + dt * f) * v * ufl.dx
bilinear_form = fem.form(a)
linear_form = fem.form(L)
A = fem.petsc.assemble_matrix(bilinear_form, bcs=[bc])
A.assemble()
b = fem.petsc.create_vector(linear_form)
solver = PETSc.KSP().create(domain.comm)
solver.setOperators(A)
solver.setType(PETSc.KSP.Type.PREONLY)
solver.getPC().setType(PETSc.PC.Type.LU)
import pyvista
pyvista.start_xvfb()
import matplotlib.pyplot as plt

grid = pyvista.UnstructuredGrid(*plot.create_vtk_mesh(V))

plotter = pyvista.Plotter()
plotter.open_gif("u_time.gif")

grid.point_data["uh"] = uh.x.array
warped = grid.warp_by_scalar("uh", factor=1)

viridis = plt.cm.get_cmap("viridis", 25)
sargs = dict(title_font_size=25, label_font_size=20, fmt="%.2e", color="black",
             position_x=0.1, position_y=0.8, width=0.8, height=0.1)

renderer = plotter.add_mesh(warped, show_edges=True, lighting=False,
                            cmap=viridis, scalar_bar_args=sargs,
                            clim=[0, max(uh.x.array)])
for i in range(num_steps):
    t += dt

    # Update the right hand side reusing the initial vector
    with b.localForm() as loc_b:
        loc_b.set(0)
    fem.petsc.assemble_vector(b, linear_form)

    # Apply Dirichlet boundary condition to the vector
    fem.petsc.apply_lifting(b, [bilinear_form], [[bc]])
    b.ghostUpdate(addv=PETSc.InsertMode.ADD_VALUES,
                  mode=PETSc.ScatterMode.REVERSE)
```

```
    fem.petsc.set_bc(b, [bc])

    # Solve linear problem
    solver.solve(b, uh.vector)
    uh.x.scatter_forward()

    # Update solution at previous time step (u_n)
    u_n.x.array[:] = uh.x.array

    # Write solution to file
    xdmf.write_function(uh, t)
    # Update plot
    warped = grid.warp_by_scalar("uh", factor=1)
    plotter.update_coordinates(warped.points.copy(), render=False)
    plotter.update_scalars(uh.x.array, render=False)
    plotter.write_frame()
plotter.close()
xdmf.close()
```

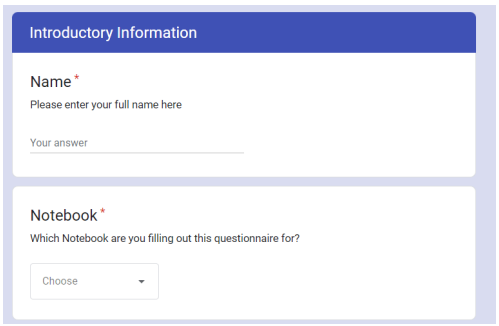
# Appendix C

## Additional Documents

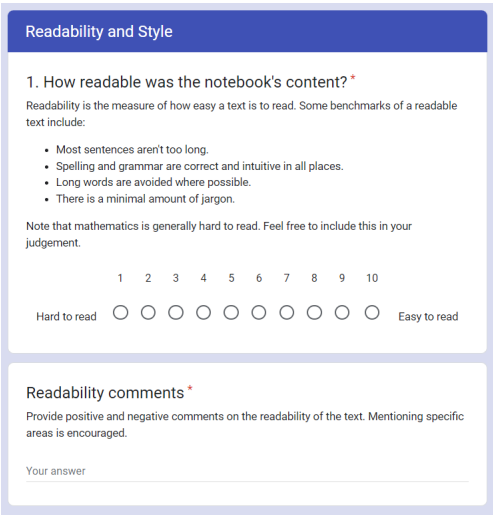
The following pages contain some additional PDFs created by me that might be of interest to the reader. All of these can also be found in the main code repository `Finite-Element-Method-Project`. The notebooks themselves have not been included here and can be found by accessing the repository.



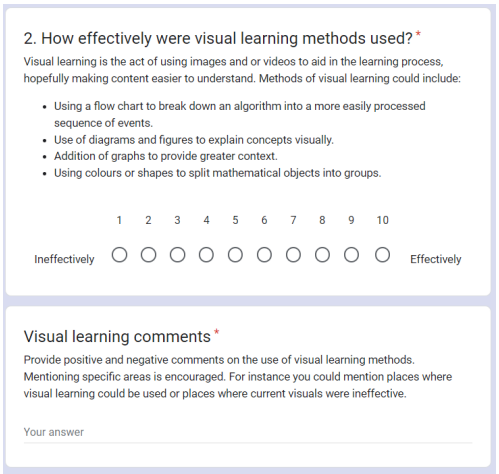

Preamble



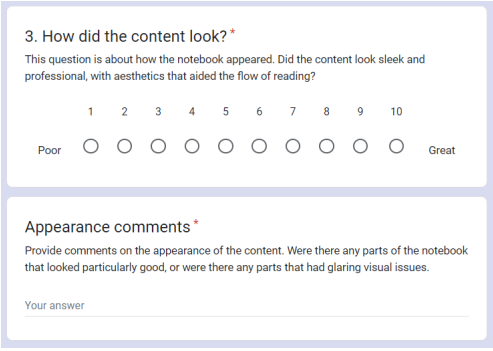

Admin Questions



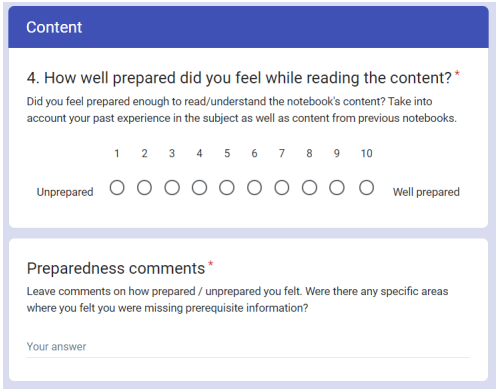

1. Readability



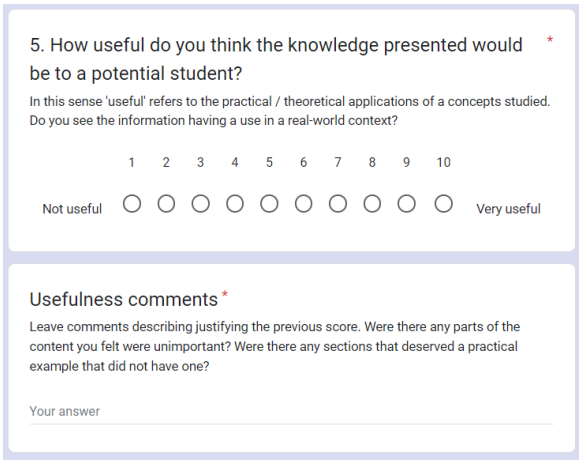

2. Visual Learning



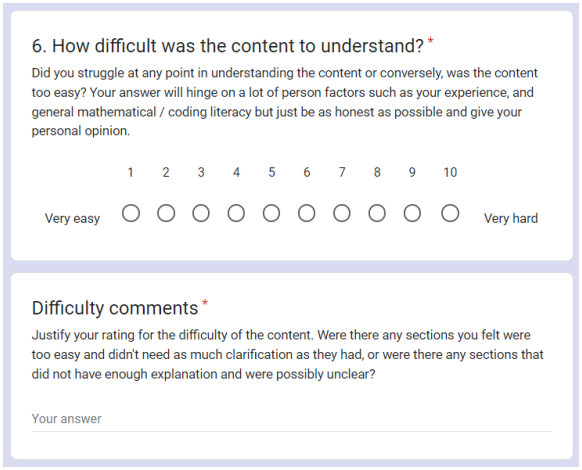

3. Aesthetics



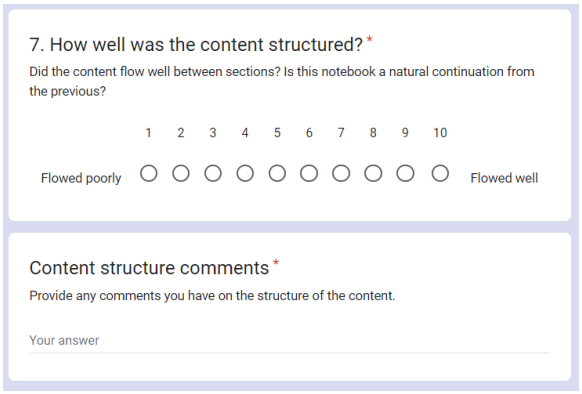

4. Preparedness (Pace)



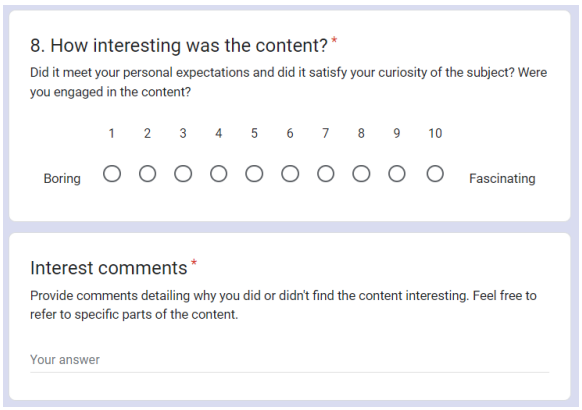

5. Usefulness



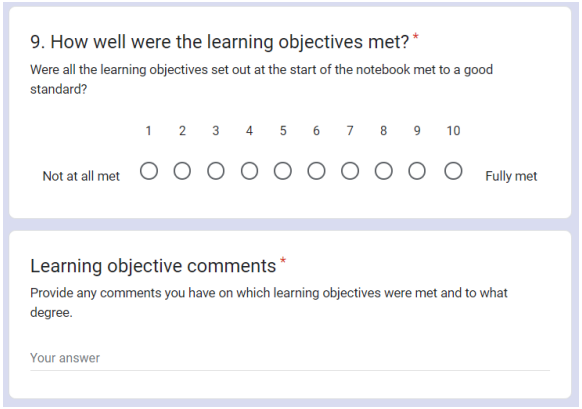

6. Difficulty



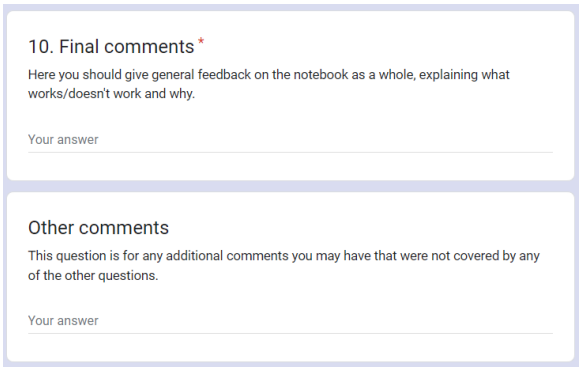

7. Structure




8. Interest




9. Learning objectives




10. Final/Other Comments

# Tester Consent Form

## Learning Resources for Understanding and Using the Finite Element Method

### Background Information

My name is Kyle Morris, and I am a third year Computer Science with Mathematics student at the University of Leeds. This is my final year, and my project involves creating a series of learning resources which would be used to teach the mathematical theory and practical implementation of the Finite Element Method. The Finite Element Method (FEM) is a technique used to numerically solve partial differential equations. The learning resources will primarily take the form of Jupyter Notebooks and the code will all be written in Python, specifically using an open-source scientific computing platform called FEniCSx.

### Your role

Development will take place over the course of roughly 4 weeks. Over that time, I will routinely create the notebooks and it will be your role to test them. I will provide testing rubrics in the form of questionnaires, and for each notebook, you will be tasked with reading through it and attempting any 'check' exercises. Then you should fill out the questionnaire to provide me with immediate feedback on how I can improve the product.

### Data

By giving your consent to participate as a tester in my project, you consent to the collection, storage, and processing of your data. Data will be collected via Google Forms which stores its responses in Google Drive. Google maintains that the information is protected in "world-class security". Information regarding security and privacy can be found here.

### Conditions

You are free to leave the project at any time for any reason whatsoever. Do not feel under pressure to complete this work, if you are struggling for any reason, feel free to speak to me privately and we can discuss options. Finally, thank you for choosing to participate in the project! It means a lot to me and will be an invaluable addition to the work. Your contributions will allow the project to reach a level of refinement that would not have otherwise been possible.

---

### Consent

I consent to participating as a tester in the "Learning Resources for Understanding and Using the Finite Element Method" Project, which includes agreeing to the conditions stated above.

**Name: Anthony Moran** **Date: 12/03/2023**

# Tester Consent Form

## Learning Resources for Understanding and Using the Finite Element Method

### Background Information

My name is Kyle Morris, and I am a third year Computer Science with Mathematics student at the University of Leeds. This is my final year, and my project involves creating a series of learning resources which would be used to teach the mathematical theory and practical implementation of the Finite Element Method. The Finite Element Method (FEM) is a technique used to numerically solve partial differential equations. The learning resources will primarily take the form of Jupyter Notebooks and the code will all be written in Python, specifically using an open-source scientific computing platform called FEniCSx.

### Your role

Development will take place over the course of roughly 4 weeks. Over that time, I will routinely create the notebooks and it will be your role to test them. I will provide testing rubrics in the form of questionnaires, and for each notebook, you will be tasked with reading through it and attempting any 'check' exercises. Then you should fill out the questionnaire to provide me with immediate feedback on how I can improve the product.

### Data

By giving your consent to participate as a tester in my project, you consent to the collection, storage, and processing of your data. Data will be collected via Google Forms which stores its responses in Google Drive. Google maintains that the information is protected in "world-class security". Information regarding security and privacy can be found here.

### Conditions

You are free to leave the project at any time for any reason whatsoever. Do not feel under pressure to complete this work, if you are struggling for any reason, feel free to speak to me privately and we can discuss options. Finally, thank you for choosing to participate in the project! It means a lot to me and will be an invaluable addition to the work. Your contributions will allow the project to reach a level of refinement that would not have otherwise been possible.

---

### Consent

I consent to participating as a tester in the "Learning Resources for Understanding and Using the Finite Element Method" Project, which includes agreeing to the conditions stated above.

**Name: Rebecca Potts_____** **Date: _____13/03/2023_____**

# Tester Consent Form

## Learning Resources for Understanding and Using the Finite Element Method

### Background Information

My name is Kyle Morris, and I am a third year Computer Science with Mathematics student at the University of Leeds. This is my final year, and my project involves creating a series of learning resources which would be used to teach the mathematical theory and practical implementation of the Finite Element Method. The Finite Element Method (FEM) is a technique used to numerically solve partial differential equations. The learning resources will primarily take the form of Jupyter Notebooks and the code will all be written in Python, specifically using an open-source scientific computing platform called FEniCSx.

### Your role

Development will take place over the course of roughly 4 weeks. Over that time, I will routinely create the notebooks and it will be your role to test them. I will provide testing rubrics in the form of questionnaires, and for each notebook, you will be tasked with reading through it and attempting any 'check' exercises. Then you should fill out the questionnaire to provide me with immediate feedback on how I can improve the product.

### Data

By giving your consent to participate as a tester in my project, you consent to the collection, storage, and processing of your data. Data will be collected via Google Forms which stores its responses in Google Drive. Google maintains that the information is protected in "world-class security". Information regarding security and privacy can be found here.

### Conditions

You are free to leave the project at any time for any reason whatsoever. Do not feel under pressure to complete this work, if you are struggling for any reason, feel free to speak to me privately and we can discuss options. Finally, thank you for choosing to participate in the project! It means a lot to me and will be an invaluable addition to the work. Your contributions will allow the project to reach a level of refinement that would not have otherwise been possible.

---

### Consent

I consent to participating as a tester in the "Learning Resources for Understanding and Using the Finite Element Method" Project, which includes agreeing to the conditions stated above.

**Name:** _____Callum Anderson_____ **Date: ___09/03/2023_____**

# Setup Guide for "Understanding and Using the Finite Element Method"

Kyle Morris

March 16, 2023

# 1 Introduction

Welcome to the setup guide for my project "Understanding and Using the Finite Element Method". This guide should get you ready to use the notebooks provided. This guide is intended for Windows users only. If you are on Mac or Linux, the overarching process will be similar, but the individual steps will be different. You will not need WSL2 or Hyper-V, you only need to install Docker itself. This guide in the Docker documentation should help you complete the installation https://docs.docker.com/engine/install/ if you must use one of these operating systems. From there running the container should be fairly similar - use the same link to Jørgen's image. The rest of this guide is intended for Windows users.

# 2 Virtualisation Environment

Docker is a software that allows us to run applications in a virtual environment called a **container**. It allows us to obtain the many dependencies necessary to work with FEniCSx (Our PDE solving software), without having to set them up ourselves. Instead of installing all the necessary modules ourselves, we can use someone else's Docker **image** that installs them on a container for us. Then we can run Python inside the container, and run our Notebooks hassle free.
Working with Docker is the recommended way to use FEniCSx; it is widely-used in running development environments with complicated dependencies. The method of installing Docker depends on your Windows version. This is because Docker needs a way of emulating the containers in a virtual environment and different versions of Windows have different virtualisation environments. You can check your Windows version by pressing Win+R and typing `winver`, or you can right-click on This PC and choose Properties. If you are on Windows Home, see the WSL2 section. Otherwise, see the Hyper-V section.

### 2.1 WSL2

WSL stands for Windows Subsystem for Linux. It allows us to run a Linux environment inside of Windows, which Docker will use to run the containers. To install WSL2, go to the command line and execute the command `wsl.exe --install` [1]. This will install Ubuntu by default. After this, restart your computer. When you reboot you will be prompted to set up a user on your new Ubuntu system. After doing this, WSL2 will be installed. It can be used by searching for Ubuntu in the search bar or with the `wsl` command in the command line.

### 2.2 Hyper-V

If you use Windows 10/11 Pro, Enterprise, or Education, you can use Hyper-V instead (although WSL2 will still work). Hyper-V stands for Hyper-Virtualisation and is a virtualisation software built into Windows. It is another software that allows you to create virtual machines inside of Windows. To use it, you will need to ensure Hyper-V is enabled on your system. To do this search Hyper-V in the search bar, and it should come up with the result "Turn Windows features on or off". Click this, find Hyper-V in the list and enable it by ticking the box and pressing Ok. [2]

## 3 Docker Desktop

Now we are free to download Docker Desktop. You can get it from `https://www.docker.com/products/docker-desktop/`. Run the executable file. If prompted, choose the backend you downloaded in 2 (either WSL or Hyper-V). You should now have Docker Desktop installed. You can find it by searching for Docker Desktop in the search bar.

# 4 Creating The Container

Now we can create our container. The docker container will inherit from a prebuilt image, created by Jørgen S. Dokken, creator of The FEniCSx tutorial [3]. His image includes many dependencies, some of which we will use and others we will not. The list is as follows:

- Basic dependencies: Python, JupyterLab, Pip, Setuptools
- PDE solving: DOLFINx, UFL, Basix, FFCX
- Data visualisation: Matplotlib, Seaborn, Pyvista, Pythreejs and Ipygany
- Numerical computation: PETSc, Pandas, H5py, Meshio, Tqdm, MPI

To create and run the container, type `docker run -p 8888:8888 ghcr.io/jorgensd/dolfinx-tutorial:v0.5.1` in the command line. This should launch a JupyterLab session. You can access the JupyterLab by copying the bottom URL in the command line and pasting it into your browser. Alternatively, you can go to `http:localhost:8888/lab` and copy the token from the console. (The token is the long string after "token=" in the url).
You may see a file called `requirements.txt`. This can be removed. To check everything has worked correctly, you can open the terminal and run `python3`. If this works, check dolfinx is installed by running `import dolfinx`. If there are no errors congratulations! You are now ready to begin using FEniCSx.

## 4.1 Future use

To stop running the container, the easiest way is to open Docker Desktop. Then go to containers and find the one which inherits from Jørgen's image. It will have a random name. Press the stop button located to the right of its entry. In future, launching the container is as simple as pressing the play button where the stop button used to be.

## References


[1] Joey Sneddon (2023) *How to Install WSL 2 on Windows 10 (Updated)*, Omg!Ubuntu!
https://www.omgubuntu.co.uk/how-to-install-wsl2-on-windows-10

[2] Scoley et. al (2022) *Install Hyper-V on Windows 10*, Microsoft Learn, Windows Virtualisation Guides
https://learn.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/enable-hyper-v

[3] Jørgen S. Dokken (2023) *The FEniCSx tutorial*, `jsdokken.com`,
https://jsdokken.com/dolfinx-tutorial/index.html