CMP-6048A Advanced Programming

Project Report - 15 December 2021

Maths Interpreter Software

Group members:
James Mason and Nia Preston

School of Computing Sciences, University of East Anglia

Version 1.0

Abstract

An abstract is a brief summary (maximum 250 words) of your entire project. It should cover your objectives, your methodology used, how you implemented the methodology for your specific results and what your final results are, your final outcome or deliverable and conclusion. You do not cover literature reviews or background in an abstract nor should you use abbreviations or acronyms. In the remainder of the report the chapter titles are suggestions and can be changed (or you can add more chapters if you wish to do so). This template is designed to help you write a clear report but you are welcome to modify it (at your peril ...). Finally, a guideline in size is approximately 3,500 words (not including abstract, captions and references) but no real limit on figures, tables, etc.

Chapter 1

Introduction

1.1 Project statement

The purpose of this project is to create a piece of software that is an interpreter for mathematical functions. It will run both as an interactive prompt and a utility to interpret and execute pre-written files. It will additionally provide the ability to visualise mathematical functions such as on a graph.

1.2 MoSCoW

1.2.1 Musts

These elements largely comprise the minimum functional requirements of the program and are as such the most neceassary items.

- Expressions
- Statements
- Commands
- Custom syntax using a custom lexer and parser

1.2.2 Shoulds

- Function visualisation
- Intuitive UI
- Flexible and scalable architecture
- Parallel processing of intensive operations
- Simple trigonometric functions

1.2.3 Coulds

- Interactive visualisations
- Zero-crossings finder
- Differentiation and integration of functions
- Simple equation solver (for example finding a variable)

1.2.4 Won'ts

- Imaginary numbers
- 3D function graphing
- Multi-variable equation solver (beyond simple simultaneous equations)

1.3 Report structure

Breifly describe what you will cover in the remainder of the report, chapter by chapter.

Chapter 2

Background

Due to the nature of this project, most of the background research and literature review will focus on an analysis of similar systems. Some research of algorithms involved in interpreter software will also prove useful, however.

2.1 Similar Systems

2.1.1 MATLAB[1]

MATLAB is a standalone piece of mathematical software designed with a focus on manipulation of mathematical data types such as matrices as well as including generic programming. It is a turing complete language, so it is has a very wide range of possible uses, but it is very much geared towards easily running numeric calculations. It has its own syntax, which means that users will have to learn it, but the custom syntax it uses is very similar to many popular programming languages, and so users with that background may find transitioning easier.

2.1.2 Math Inspector[2]

Math Inspector is a package of addons for Python which allow for mathematical functions to be graphically visualised, both by displaying a flow-type diagrom of the function as well as graphing the numeric outputs across a range. Since it is an addon for Python, people who are familiar with the language already shouldn't have much difficulty using it. It may, however, mean that people have to install more than what they need in order to get it to run.

2.1.3 Wolfram Mathematica[3]

Wolfram Mathematica is an interactive tool used to define and visualise mathmatical functions. It's main foucs is using technology to visualise mathmatics and to allow the user to directly implement a vast variety of features. It offers over 5000 different in-built maths functions, meaning it contains very few restrictions. It runs on it's own in-built language which may take the user some time to learn and become familiar with. Its' main strength is definately its advanced visulisation features.

2.1.4 Maple[4]

Maple offers a number of different versions of its' software allowing the user to tailor their experience before they even download. It's main goal is solving equpations but it still offers an intuetive UI with function visulisation features. It allows the user to explore a range of mathmatical fields by offering a large variety of in-built functions. Similarly to the previous system, it operates using it's own programming language which may take users time to learn and understand. The different versions of the software (tailored based on who is using it) is this systems' main strength.

Depending on the nature of the project, this chapter may cover a literature, resource and/or (software) product review. For a literature review you will consult journal or conference papers outlining methodologies that you may (or may not) use but which are definitely relevant to your particular problem. Resource and/or product information will typcially be substantiated through internet links. You may use different sections if different subareas are part of your problem statement and/or solution. Since this chapter covers background resources, you should also update the corresponding bib file referred to in the bottom of this document and here it is called References.bib. You cite references like this: Taylor et al. [5] investigated non-linear FEA on the GPU. Morton [6] developed a file sequencing method in 1966. A website on OpenCL can be found here [7]. Etc.

Chapter 3

Methodology

Describe here various methods that will be used in your project. Use different sections for distinctly different subjects and use subsections for specific details on the same subject. Only use subsubsections or paragraphs (which are not numbered) if you believe this is really necessary. Since implementation will happen in sprints, this section may need several updates with parts being added and deleted across the project.

3.1 Method 1

3.1.1 Method 1 specific detail 1

In case you need maths, here is an example to write an equation:

$$\int_{\Omega_0} \delta u \frac{\partial \mathbf{P}}{\partial X} d\Omega_0 + \int_{\Omega_0} \delta u \mathbf{b} d\Omega_0 + \int_{\Omega_0} \delta u \rho_0 \ddot{\mathbf{u}} d\Omega_0 = 0 \tag{3.1}$$

And here we show how to write a matrix equation:

$$\mathbf{X} \frac{\partial N}{\partial e_c} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -1 & -1 \end{bmatrix} \tag{3.2}$$

3.1.2 Method 1 specific detail 2

blablabla

b lablabla

3.2 Method 2

3.2.1 Method 2 specific detail 1

3.3 Etc.

3.3.1 Etc.

Chapter 4

Implementation

In this chapter you cover the actual implementation of your project. Implementation will be done in sprints so you may wish to use different sub-sections for each sprint and then dedicate a section to your final deliverable. Section 4.3 with figures should not remain in the final report.

4.1 Early sprints

4.1.1 Sprint 1

4.1.2 Sprint n

4.2 Final implementation

4.3 Figures, tables, etc.

The purpose of this section is to just show you how to integrate figures, tables, etc. and should disappear in the final report. Figures and tables should be distributed across the document wherever they are needed but you should use an appendix if they are many figures/tables of the same kind. Fig. 4.1 shows a bony pelvis.

Fig. 4.2 shows a UML class diagram (class, sequence and state diagrams are the most frequently used UML diagrams):

Algorithms can be either used in this chapter or alternatively in Chapter 3 if it is a more generic algorithm:

Algorithm 1 The projection based contact method algorithm

1: Retrieve current node displacement u
```
float3 u = m_U_new[nodeIndex].xyz;
```
2: Retrieve constraint plane equation
```
float4 plane = m_constraintMagnitude[nodeIndex];
```
3: Calculate dot product with plane normal
```
float d = dot(u, plane.xyz);
```
4: Find node penetration into the plane's negative half-space
```
float penetration = plane.w - d;
```
5: **if** penetration is greater than zero **then**
6: Find projection onto the plane surface
```
float3 proj = u + plane.xyz * penetration;
```
7: Prescribe new nodal position to be on the surface
```
m_U_new[nodeIndex] = (float4)(proj, 0.0f);
```
8: **end if**

Tables such as Table 4.1 can also be useful here or in Chapter 3.

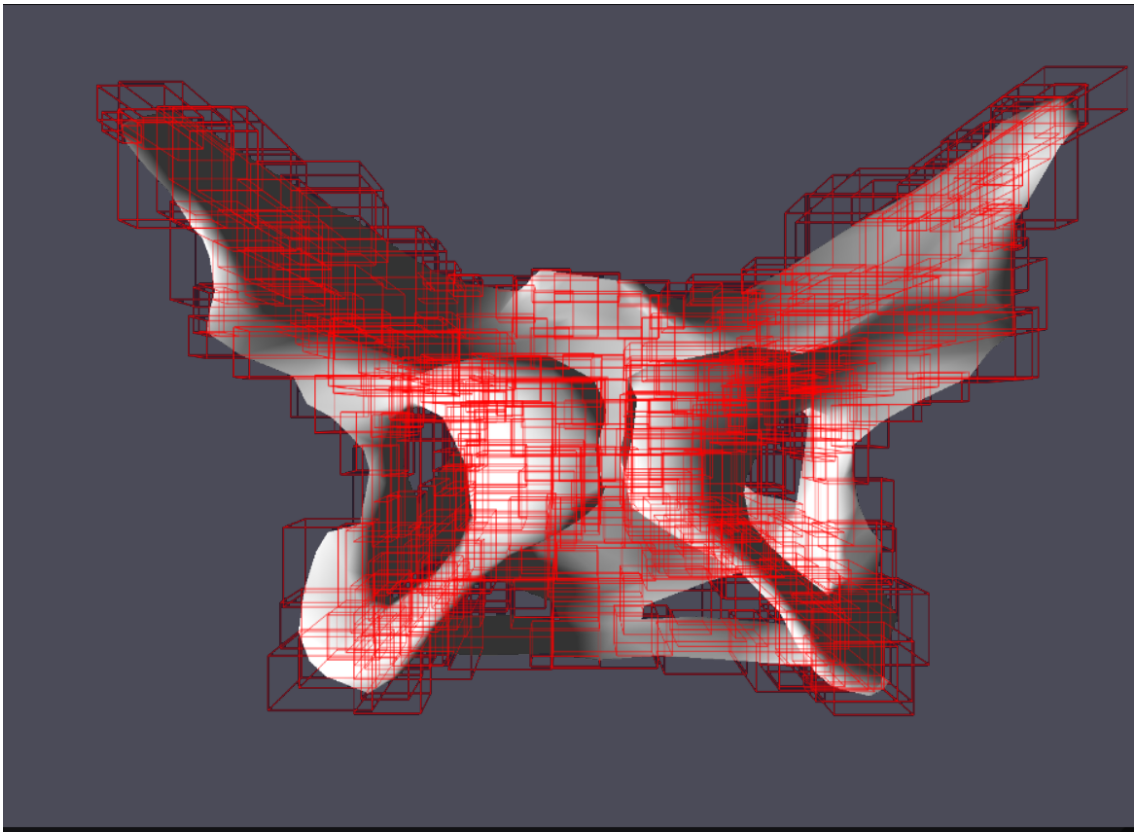

Figure 4.1: The bony pelvis model with octree based AABBs (Axis Aligned Bounding Boxes).

Note that code snippets or lists of crucial programming code or large UML diagrams should go in the Appendix/Appendices.

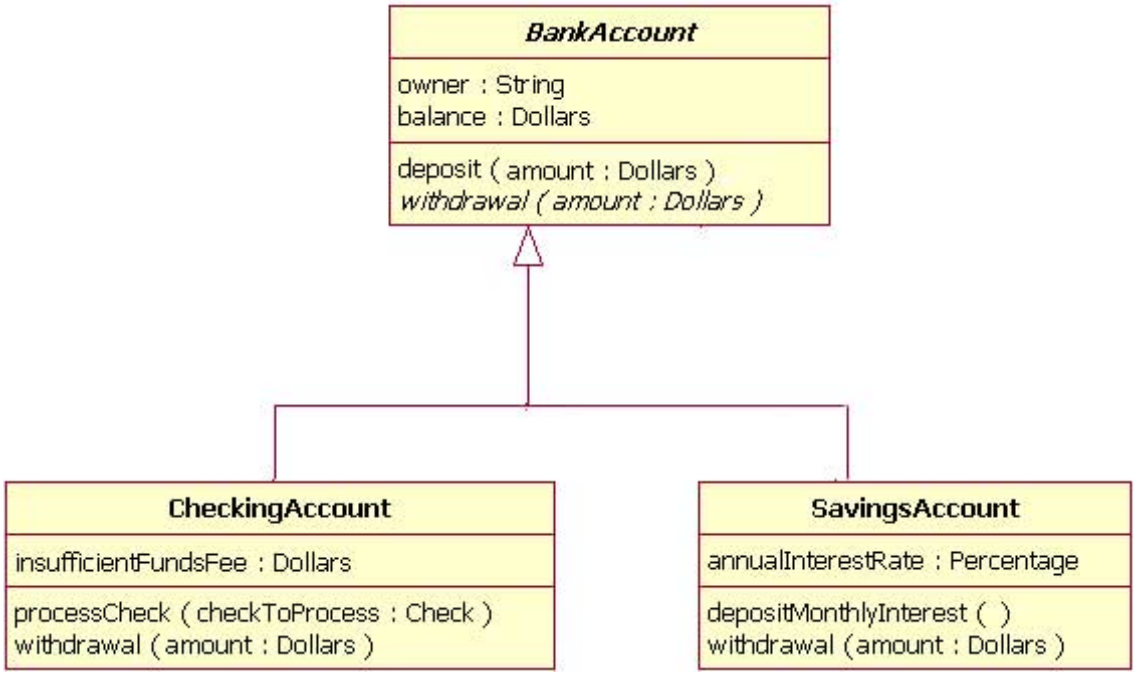

Figure 4.2: A UML class diagram.

Table 4.1: Original diameters and diametral strains as reported by Sorbe and Dahlgren [8] (columns 1-2), from a previous experiment by Lapeer and Prager and reported in [9] (columns 3-4), and from the current experiment (columns 5-6).

	S-D		L-P old		L-P new	
Diameter	length	strain	length	strain	length	strain
MaVD	140.5	+1.90	129.3	+0.30	129.3	+1.43
OrOD	131.4	+0.10	-	-	119.9	+1.85
OrVD	126.9	+2.20	119.3	+0.25	119.3	+1.24
OFD	134.0	+0.40	-	-	119.7	+1.82
SOFD	-	-	-	-	113.2	-0.85
SOBD	117.1	-1.70	88.7	-1.07	88.7	-2.52
BPD	105.0	0.00	89.7	-0.21	89.7	-0.83

Chapter 5

Testing

Describe various experiments you designed to test your software product. This could be subdivided to be in line with the Sprints in Chapter 4. In case you have protocols which cover various pages, please put them in an appendix (e.g. Appendix A) instead.

Chapter 6

Discussion, conclusion and future work

Briefly discuss and conclude your achievements and put them in perspective with the MoSCoW analysis you did early on. Be honest by declaring for example 'S' categorised objectives which did not make it to the final deliverable rather than reversely modifying your MoSCoW in Chapter 1! Also discuss future developments and how you see the deliverable improving if more time could be spent. Note that this section should not be used as a medium to vent frustrations on whatever did not work out (pandemic, group partners, etc.) as there are other means for this (labs, e-mail MO, ...) that should be used well before any such problems become an issue.

Bibliography

[1] MathWorks, Natick, Massachusetts, United States. *MATLAB*, 2017.

[2] *Math Inspector*, 2021.

[3] Wolfram, The Wolfram Centre, Lower Road, Long Hanborough, Oxfordshire OX29 8FD, United Kingdom. *Wolfram Mathematica*, 2021.

[4] Maplesoft, 615 Kumpf Drive, Waterloo, ON, Canada, N2V 1K8. *Maple*, 2021.

[5] Zeike A. Taylor, Mario Cheng, and Sébastien Ourselin. Real-Time Nonlinear Finite Element Analysis for Surgical Simulation Using Graphics Processing Units. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2007*, pages 701–708. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[6] G.M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. Technical Report Ottawa, Ontario, Canada, 1966.

[7] Mate Soos. AMD's OpenCL heaven and hell. https://www.msoos.org/2012/01/amds-opencl-heaven-and-hell/, 2012.

[8] B. Sorbe and S. Dahlgren. Some important factors in the molding of the fetal head during vaginal delivery - a photographic study. *International Journal of Gynecology & Obstetrics*, 21(3):205–212, 1983.

[9] R. J. Lapeer and R. W. Prager. Fetal head moulding: finite element analysis of a fetal skull subjected to uterine pressures during the first stage of labour. *Journal of Biomechanics*, 34(9):1125–1133, September 2001.

Contributions

State here the % contribution to the project of each individual member of the group and describe in brief what each member has done (if this corresponds to particular sections in the report then please specify these).

Appendix A

Put in tables of data or protocols (e.g. for testing) or code listings or UML diagrams which may take up several pages and do not sit well in the main body text.