University of Dundee

MA40001

MA40001

Philip Murray

2024-10-09

Table of contents

Preface

Welcome to the module MA40001.

My name is Philip Murray and I am the module lead.

The main aims of the module are to develop:

- mathematical skills,
- independent investigation skills
- scientific/mathematical communication skills.

How to contact me?

- email: pmurray@dundee.ac.uk
- office: G11, Fulton Building
- Teams: PM me

Software

The taught component of the module will require you to use the following software:

- Quarto
- Visual Studio Code
- Github
- Latex
- Python

More details available here (see Chapter 1.1).

> **i** Why bother with all this?
>
> The aim is to provide you with *open-source* tools that enable you to generate modern scientific/mathematical documents.

Teaching materials

Lecture notes

You can find lecture notes for the module on this page. If you would like a pdf this can be easily generated by clicking on the pdf link of the webpage. I will occasionally edit/update the notes as we proceed through lectures. If you spot any errors, typos or omissions please let me know.

Template codes

I have provided template codes via the github page MA40001REsources.

In Chapter 6 you will learn how to *fork* this github repository so that you have template project/presentations.

Assessment

- Presentation
- Interim report
- Poster
- Final report
- Viva

Further details are available on MyDundee.

Teaching plan

Table 1: Semester 1

Week	Material	Assessment
1	Intro+Quarto	Formative
2-3	Quarto/Python	Formative
4-6	Python	Formative
7-10	Presentation practices	Formative
11	Assessed presentation	Summative
13	Interim report	Summative

Table 2: Semester 2

Week	Material	Assessment
11	Assessed poster presentation	Summative
13	Final report submission	Summative
Exam period	Viva (aural exam)	Summative

1 Software

1.1

A version of Visual Studio Code with Quarto + other software is available via Apps Anywhere via the app:

- Visual Studio Code 1.92.2 with Quarto 1.5.56

VS Code is an integrated development environment (IDE) - a platform from within which to develop and run codes.

The Apps Anywhere app has necessary codes pre-installed for you to immediately begin working on the module.

> **Note**
>
> The AppsAnywhere app is still being finalised by UoDIT. There is currently a bug that means:
>
> - you cannot use the commandline/terminal
> - simultaneuosly generate multiple output formats (e.g. html and pdf)
> - compile multi-file projects

1.2 On your personal computer

It is advised that you get VS Code + necessary software running on your personal computer.

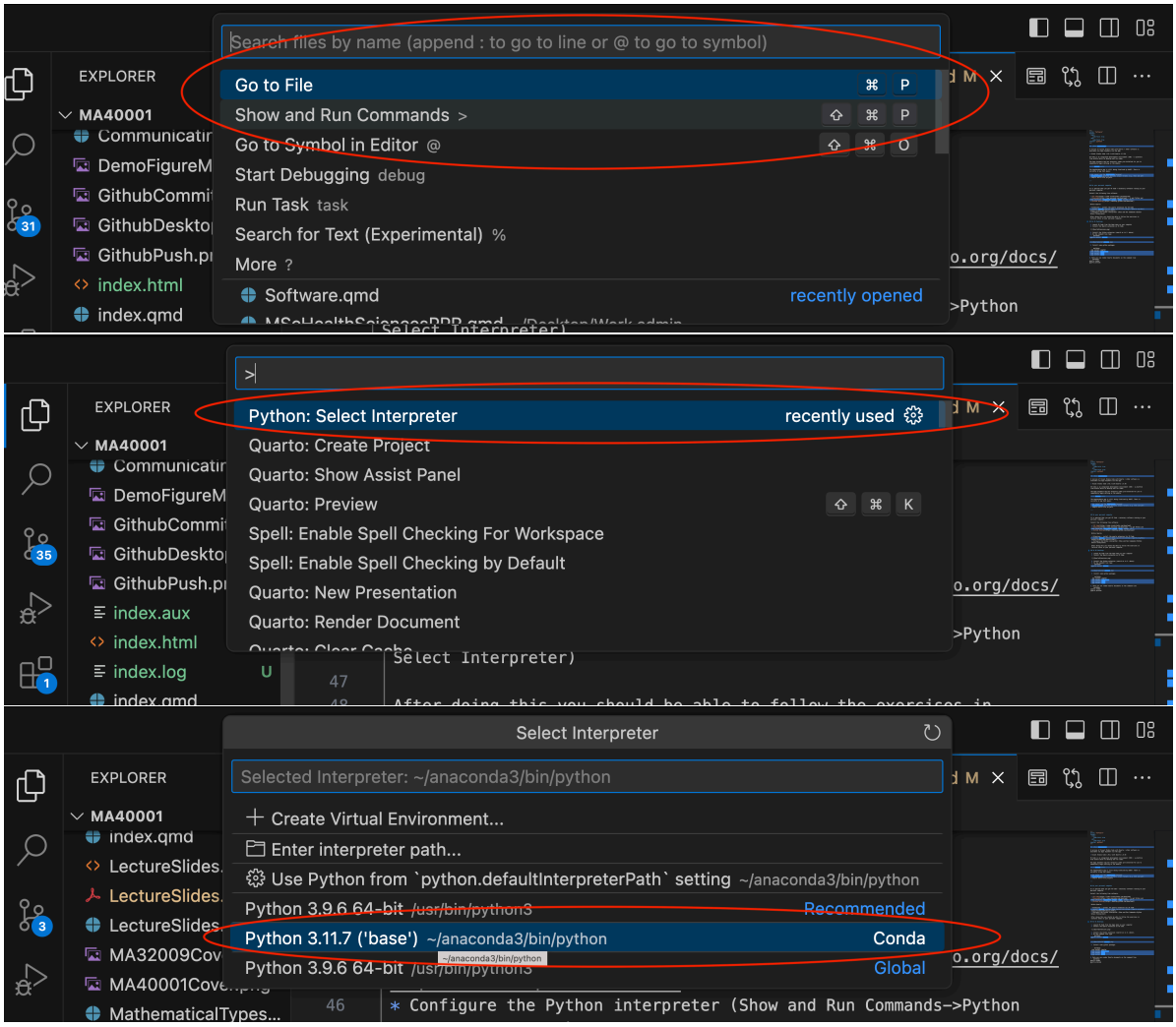
Install the following free software:

- VS Code
- Anaconda - to get Python and then libraries (matplotlib.pyplot, numpy, pandas, etc.)
- Github Desktop

Within Quarto:

- Extensions - install the quarto extension for VS Code

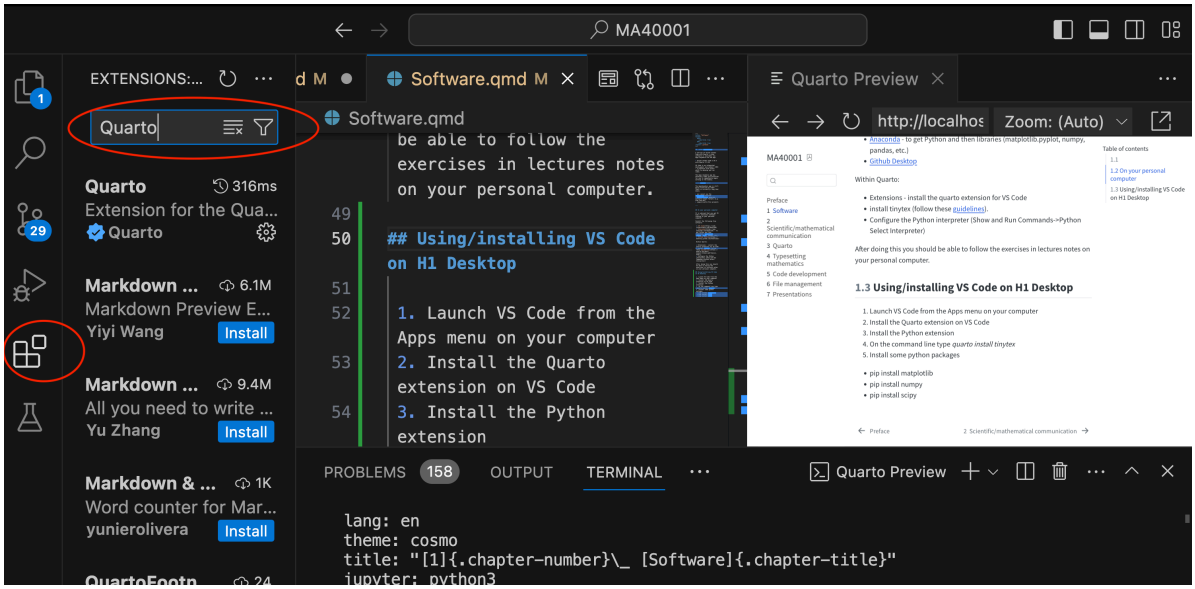
- install tinytex (follow these guidelines).
- Configure the Python interpreter (Show and Run Commands->Python Select Interpreter)



After doing this you should be able to follow the exercises in lectures notes on your personal computer.

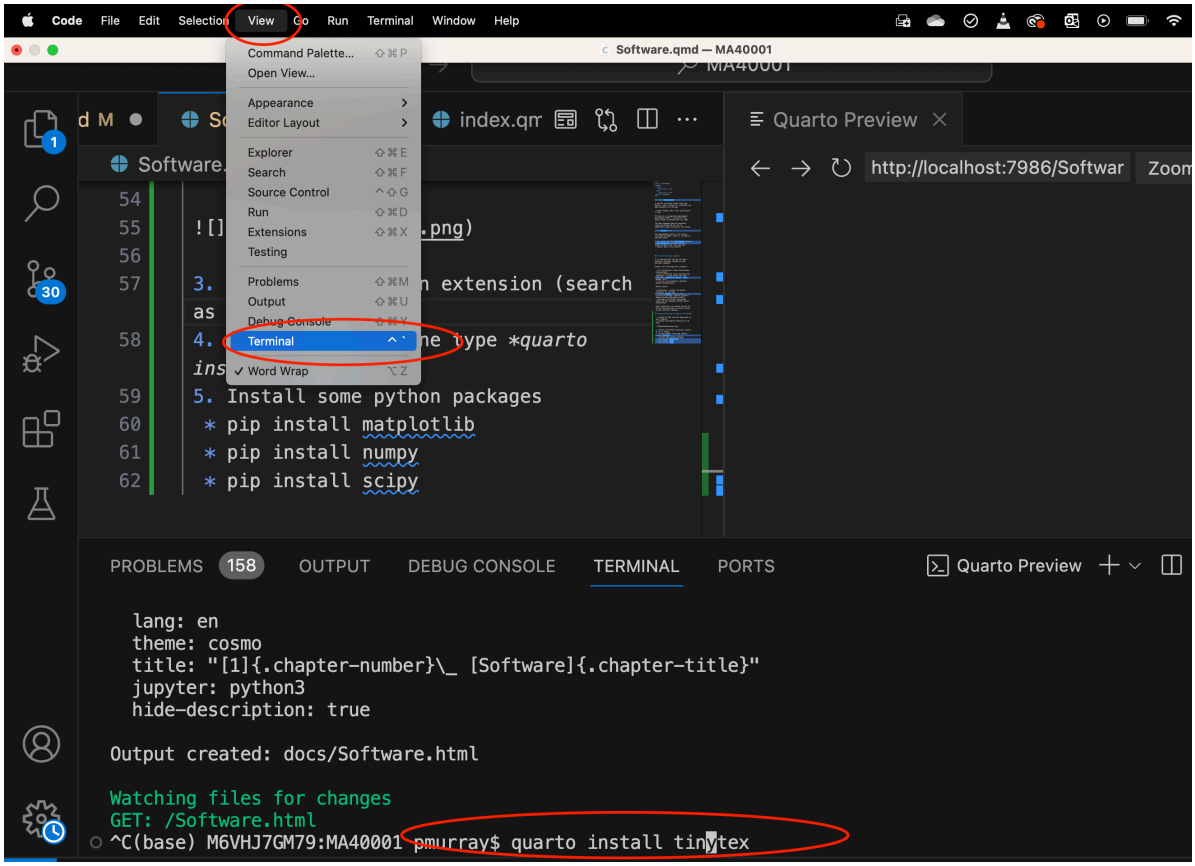
1.3 On H1 Desktops

1. Launch VS Code from the Apps menu on your computer
2. Install the Quarto extension on VS Code.



3. Install the Python extension (search as in 2. above).
4. On the command line type

```
quarto install tinytex
```



5. Install some python packages

```
pip install jupyter
pip install matplotlib
pip install numpy
pip install scipy
```

6. Note you can render Quarto documents on the command line

```
quarto render
quarto preview
```

2 Scientific/mathematical communication

The assessment of your project will requires you develop:

- presentation slide
- poster
- written reports (interim report and final report)

2.1 Common elements across different assessment points

2.1.1 Narrative

A common theme with the different assessment points is the need for a *narrative* around your project:

- can you describe the project in *one succinct sentence*?
- can you outline why the topic is important?
- what is the background to the project?
- what methods/results/techniques have you developed?
- can you summarise your findings?

Once you have settled on a narrative, the different assessment points can be thought of as variations on the presentation of your project narrative.

> ❗ Exercise - project narrative
>
> Define a narrative for your project:
>
> - One sentence description
> - Why is it important to study this topic?
> - Describe any relevant background
> - What are the project Aims?
> - What work have you done to address the Aims?
> - Outlook

2.1.2 Pitching the content appropriately

A key principle of good communication is to *know your audience*.

If you pitch at too high a level (e.g. assuming that your audience know more than they actually do), the audience will likely be confused and unable to follow your reasoning.

If you pitch at too low a level (e.g. by explaining concepts that your audience is already familiar with) they will likely be bored/feel condescended etc.

At all points of your assessment: assume that you are communicating with your peers, i.e. Level 4/5 of a undergraduate mathematics degree. This means that you should *not* assume that the audience have specific knowledge of the details/background of your project.

> ! Exercise - Pitching your project
>
> Practice pitching your project at different target audiences:
>
> - A high school student with Higher mathematics (*how much mathematical detail should you give?*)
> - A researcher in the field that you are working (*how much intro will they need?*)
> - A family member/flatmate.
> - At a job interview (*focus on parts of the project that are relevant*)

2.1.3 Equations

You will almost certainly use mathematical notation in your assessment.

Are equations presented accurately? Are mathematical objects accurately defined? Has sufficient background detail been presented so that the arguments can be reasonably followed?

2.1.3.1 Typesetting equations

To typeset formulae is actually quite difficult. Mathematics uses a variety of symbols and several different alphabets: Roman, Greek, Hebrew are the most common. In addition formulae are often more similar to graphics than to text. There are numerators and denominators which in turn can have fractions etc., e.g.:

$$f := \frac{1}{1+\frac{1}{1+\frac{1}{1+x}}}.$$

To make this formula look good requires either an advanced typesetting program or a lot of effort. Most common typesetting programmes come with some sort of equation editor, but

very few can handle such a problem. The most powerful mathematical typesetting program, which is also the format used for almost all mathematical literature is LaTeX. We will learn about LaTeX later.

As with grammar, for a language there exist also certain conventions about how to write formulae. Here is a (far from exhaustive) list of the most import conventions:

- Treat the formula like text. If the formula is at the end of a sentence there has to be a full stop at the end of the formula. If another formula follows use a comma or semicolon. This is how we count
$$1 + 0 = 1, 1 + 1 = 2.$$
- use Roman (typically lower case) letters in *italic* style for all variables: x, y, z, a, b, c, both if we refer to them in the text as well as in formulae. Note difference between x and x.
- use Greek letters for angles, and e.g. differential forms;
- typeset vectors in bold, $\mathbf{a}$, or using an arrow, $\vec{a}$;
- typeset functions in roman, $\sin(x)$ rather than $sin(x)$;
- typeset matrices using capital roman letters, e.g. M;
- represent number systems and also certain vector spaces in a style where certain lines are double: $\mathbb{R}$, $\mathbb{Q}$, $\mathbb{C}$, ...;
- use curly brackets for sets, e.g. $\{1, 2, 3\} = \{2, 1, 3\}$, and regular brackets for an ordered list $(1, 2, 3) \neq (2, 1, 3)$;
- denote a range by three dots: $i = 1, 2, ..., n$, (no bracket required);
- use brackets only where necessary, note that multiplication/division takes precedence over addition/subtraction; e.g
$$a + (b \cdot c)$$
does not need the brackets, but
$$(a + b) \cdot c$$
does. Any fraction replaces a bracket, so $\frac{5}{a+b}$ does not need the brackets.
- use a separate line for any formula that uses more than one line or complicated formula. Don't write $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$ in line. Instead write
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Common mistakes

Common mistakes are (in addition to violating the above conventions):

- there are typos/inconsistencies in the equations
- the presented system is not mathematically self-consistent (e.g. a system of ODEs is missing initial conditions)
- variables/parameters are not defined
- using different font styles and sizes to represent the same quantity;
- not using the same symbol (or font) in the text and separate formulae;
- not using spaces between symbols, and/or wrong symbols, e.g axb instead of $a \times b$;
- forgetting brackets or placing too many brackets.

2.1.4 Figures

A formula is often the best and most concise way to communicate a relation or function to a mathematically trained audience. However, there are many cases where even a mathematician might have difficulty understanding (e.g. in the short time available in a presentation) what a function represents. For functions of one and two variables there is always the option to show a plot of the function. Here is an example of a function in two variables:

$$f(x, y) = \cos(x) \sin(xy). \tag{2.1}$$

Although this is a comparatively simple function it takes some time to figure out what properties the function has, that is for instance:

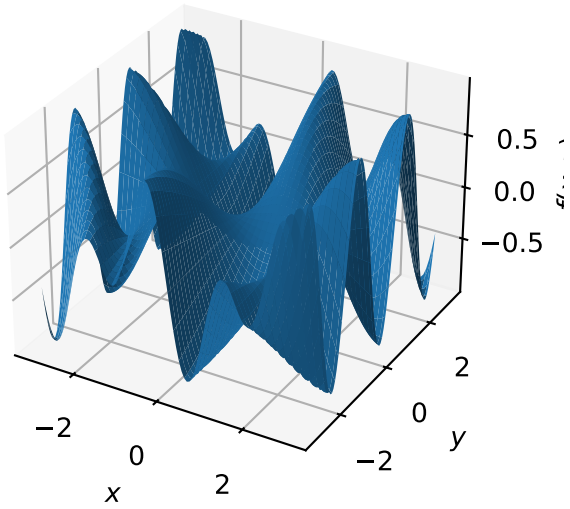
- Is the function periodic?
- How many maxima/minima does it have?
- How does it behave for $x, y \to \infty$?

To explain the behaviour of such a function, a plot will help! But which plot? For example, we could use a surface or contour plot (see Figure 2.1). The advantage of a contour plot is that it is often easier to see the locations of maxima and minima but it is not so easy to see how high the extrema are.

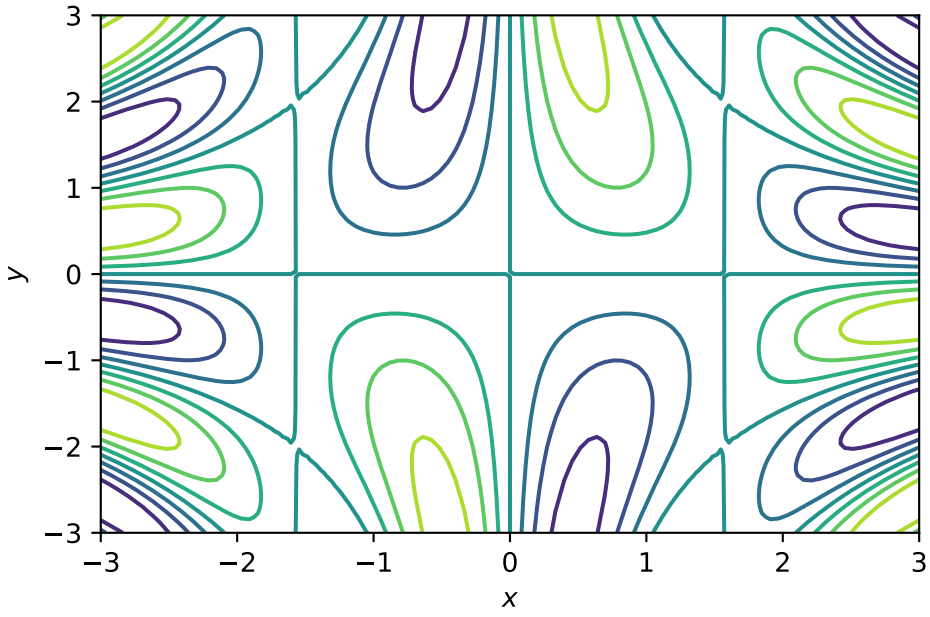
A good rule of thumb is: if figures and tables are removed from the text, does the text still read coherently? i.e. the figure is helping the reader to understant a point that is made within the text. It is *not* replacing the need for text.

Other types of plots are useful for different purposes:

- bar chart (Figure 2.2)
- pie chart (Figure 2.3)
- scatter plot (Figure 2.4)



(a) A 3D plot. $f(x, y)$ is plotted against x and y



(b) A contour plot. Contours of $f(x, y)$ are plotted against x and y.

Figure 2.1: 3D plot of Equation 2.1

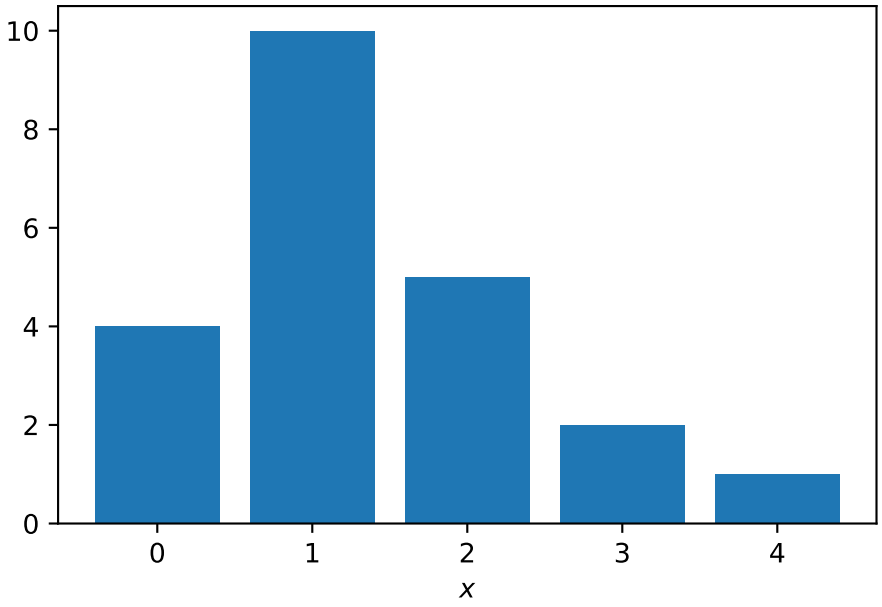


Figure 2.2: A bar chart.

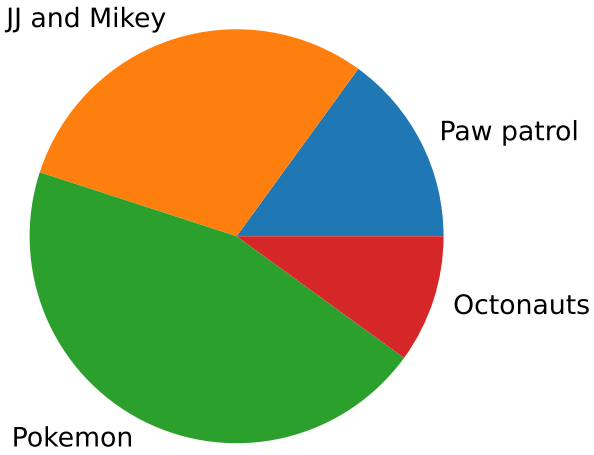


Figure 2.3: A pie chart.

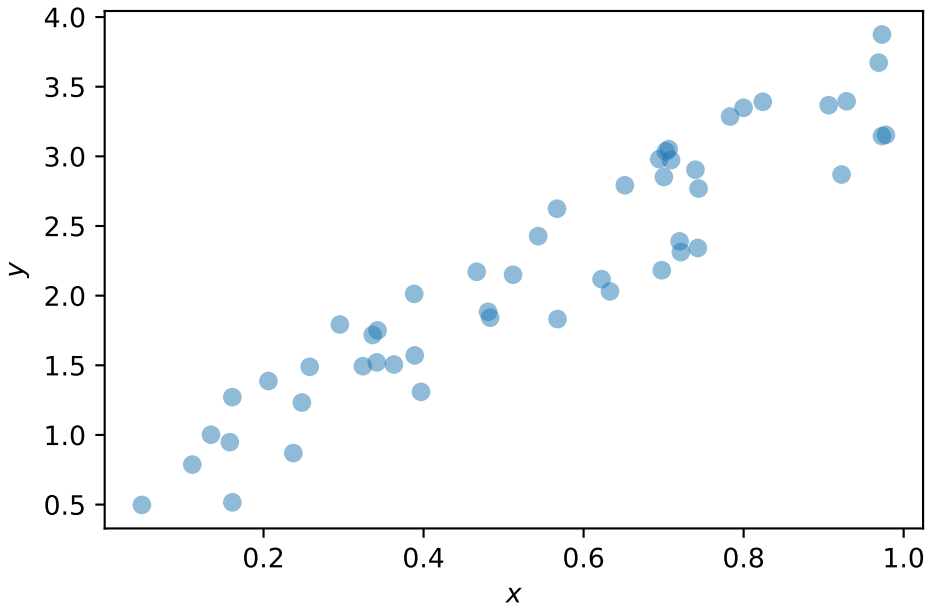


Figure 2.4: A scatter plot.

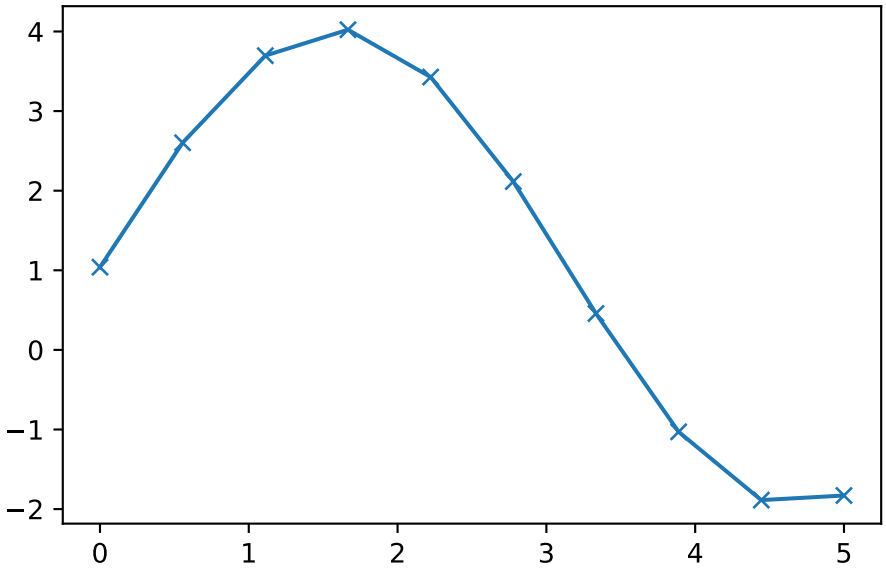


Figure 2.5: y is plotted against x.

common mistakes

- hanging figures (i.e. figures that are not referenced/placed in context in the text (e.g. context for the information in Figure 2.3 is not described in the text. Figure 2.5 is not dven referenced)
- figures without axis labels (Figure 2.5 has no axis labels)
- coloured graphs without colour scale/legend
- labels/ tick marks are too small
- type of graph unsuitable for the data shown
- the figure was not adequately connected to the text (it is not clear, for example, what equations were solved, what method was used.)

Some rules -of-thumb:

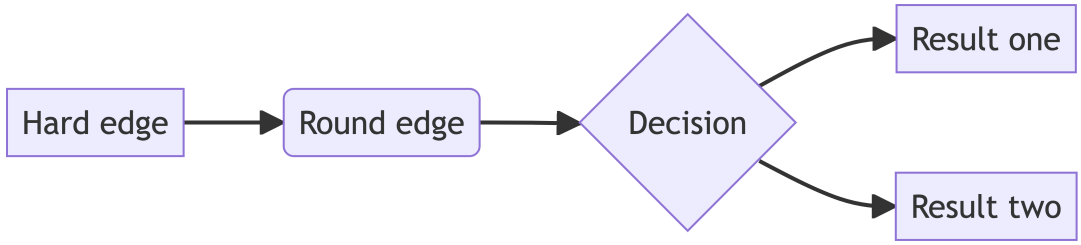
- check that any text in axis labels/legends is approximately the same size as font in the main text

2.1.5 Schematic diagrams

Schematic diagrams are a useful way to help to try and introduce a new concept/summarise a finding.

There are many ways to generate schematic diagrams (e.g. generating in software such as Photoshop or Illustrator and saving as an image).

Alternatively Quarto (Chapter 3)) provides an interface to a number of graph-based tools that generate schematic diagrams, e.g.



2.2 References

All content in your report must be either original or attributed, via references, to the originating author(s). The majority of the content of a report should be in the author's original words. It should be the author's original logical argument, with citations used to document and justify key assertions and facts.

2.2.1 Plagiarism

Passing off someone else's work as your own is *plagiarism*. It undermines the integrity of science, and has serious consequences for the authors as well. (e.g., failing grades, expulsion from a degree program, loss of an academic position, being shunned by the scientific community, etc.)

Some tips:

- You should try and try and cite primary sources (e.g. research papers or textbooks) as these have permanent bibliographic identifiers (dois).
- Wikipedia is an excellent resource for learning new topics. However, it is not an acceptable source for citation in a report or publication as:
  - the content of wikipedia changes continuously;
  - it is generally not written by experts and often contains errors.
- A word-for-word quotation should be indicated with quotation symbols. Such as: The whatever effect causes "extremely aggressive phenotypes" to dominate (Smith et al. 2010). Extensive quotes of more than one sentence should generally be set apart as a separate paragraph.
- limit quotations to no more than 1-3 sentences. Instead, one should paraphrase and/or summarise cited work. Hence, quotations should be kept to a minimum.

You must take great care when quoting or paraphrasing text to cite the original source.

2.3 Use of AI

Large language models (LLMs), such as ChatGPT, Dall-E, are being used to generate increasingly complex outputs (e.g. images, text). AI is also increasingly used in the development of codes (e.g. CodePilot).

LLMs use statistical patterns in datasets upon which they have been trained to generate responses to natural language queries. To the untrained eye, the output can seem very convincing and, depending on the query, it can be accurate. But the output can also be completely incorrect.

Using generative AI materials without attributing the source is a form of plagiarism. Hence if you use generative AI in your project it must be clearly declared in your *Use of AI* statement (see template project). You can find the University policy on this topic here.

The referencing styles are still developing their guidance on how to reference GAI tools such as ChatGPT. As content generated by GAI is non-recoverable, it cannot be retrieved or linked

to in the way that other resources that may be referenced in your work. University policy recommends that AI-generated content should be cited as a personal communication as an in-text citation.

i University of Dundee definition of plagiarism

Plagiarism is the unacknowledged use of another's work as if it were one's own. Examples are:

- the inclusion of more than a single phrase from another's work without the use of quotation marks and acknowledgement of the source;
- summarising another's work by changing a few words or altering the order of presentation without acknowledgement;
- copying another's work;
- the use of another's ideas without acknowledgement.

NB: if you wish to reference your own work, it is important to acknowledge yourself as the source and provide the appropriate reference.

2.4 Written reports

The overall structure of a report is (generally) as follows:

1. Title page: This should give the title of the report, the author list, and the date.
2. Abstract: provide a short synopsis (~10 sentence) of the project.
3. Introduction: This introduces the overall context and importance of the problem you are addressing. It should give:
    - A basic paragraph or two on background of the problem, its significance, and motivation for the paper. It should make us want to continue reading.
    - In a research-grade paper, you would include information as to why preceding work by others has been insufficient. What work, findings, or improved methodology are required?
    - A summary of any hypotheses that you will develop and justify throughout the paper
    - A basic outline of the remainder of the paper, including a note on the methodology (in mathematics, these would be your modelling, analytical, and numerical techniques).
4. Actual content: There should be one or more sections that logically progress your argument and analysis. For example:

- Formulation of the problem
- Methods used to solve the problem
- Results

5. Discussion and Conclusions: Wrap up with a summary of your major results, the significance of your conclusions (including any additional analysis of the results to lay out this significance), and an outlook what else could be done/ or where improvements are required.

> **i** Some common errors in the stucture
>
> - Insufficient development of background (the author assumes that the reader knows more than they do)
> - The aims of the project are not clearly stated
> - The importance of the topic is insufficiently described (e.g. why study this problem?)
> - the discussion is too short

7. References: This section contains the full list of publications that you cited in your report.
8. Appendices

- Use of AI Statement
- (optional): Lengthy and/or tedious calculations or details that are too distracting to the overall flow of your paper, and yet are necessary to fully document your work, should be placed in appendices. You must also present any codes that you have developed.

A template project report is available via the MA40001Resources github repository.

2.5 Verbal presentation

At the end of Semester 1 you will be asked to give an approx. 10 minute verbal presentation of your project. More information will be provided in Weeks 7-10.

A template presentation is available via the MA40001Resources github repository.

3 Quarto

Quarto is an open source scientific and technical publishing system. It can be used to make a range of publishable outputs (reports, posters, slides, blogs, webpages dashboards etc.). In this section you will learn to use Quarto to write reports and make slide decks. However, one of the advantages of learning Quarto is that it is straightforward to write and publish websites, blogs, dashboards and books.

It is assumed that you are using VSCode with Quarto from Apps Anywhere on the Uni machines. However, you are also encouraged to install Quarto/VSCode on your personal computer.

Quarto documents are written using Markdown (you may have previously used RMarkdown). In the background, Quarto uses an open source document convertor called Pandoc.

3.1 Getting started

Visual Studio code is a popular integrated development environment (IDE) that is used to write software.

On Apps Anywhere you can find an iteration of VS Code that has Quarto, Python and Latex preinstalled. The app is called

Visual Studio Code 1.92.2 with Quarto 1.5.56 extension

3.2 A first markdown document

To begin with, let's consider a simple markdown document.

```
<!-- Configuration information -->
---
title: "Hello, Quarto"
format: html
---
```

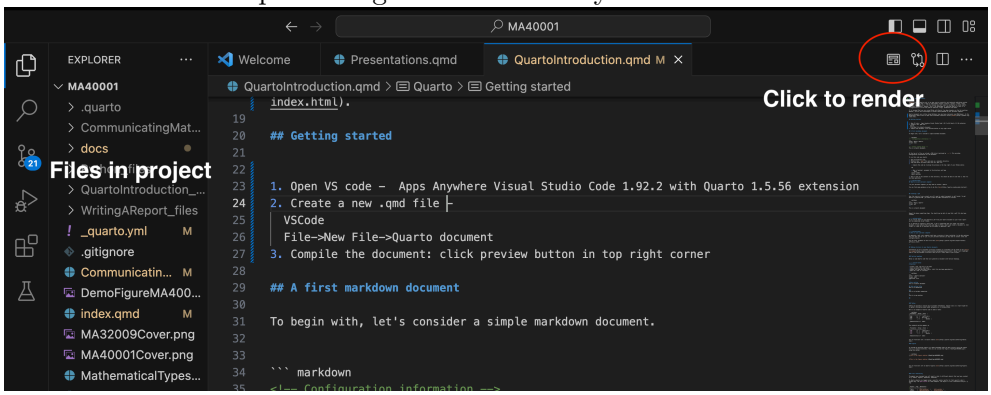
```
<!-- Insert content below -->
This is a Quarto document.
```

At the top of a file we include a YAML block (enclosed by `---`). This provides configuration information for the document.

! Exercise

To run this code you should:

1. Open the VisualStudio Code app.
2. Create a new `.qmd` file and save in a sensible directory.
3. Copy and paste the above code into the .qmd file.
4. - Compile the code by clicking the preview on the top right of your VSCode editor.



   OR

   - Open a terminal, navigate to the directory and type

```
quarto render
quarto preview
```

5. Have a look at the contents of the directory. You should be able to see that a .html file has been generated.
6. Open the .html file in a web browser.

3.3 Creating a pdf

Over the course of your project you will need to submit documents in pdf format. To get Quarto to generate a PDF, you can edit the YAML information as follows.

```
---
title: "Hello, Quarto"
format: pdf
---

This is a Quarto document
```

> **!** Exercise
>
> Repeat the above compiling steps. You should now be able to see that a pdf file has been created in the directory alongside your .qmd file.
> It is crucial that you can generate a pdf from your Quarto document as your final report will be submitted in pdf format.
> It is advised to regularly check that a pdf is generated when you render your Quarto document. I have noticed that with some bugs in Latex (more on this later), Quarto can render a document in .html format (it looks ok in preview) but be unable to generate a pdf.

3.4 Adding structure to your Quarto documents

Information on how to document structural elements to a document can be found on the Quarto help pages. Below I will highlight some of the key document structures that you will likely need in your project.

3.4.1 Section headings

Below is some Quarto code that will generate a document with Section headings.

```
---
title: " Quarto sections"
format: html
code-fold: True
---

# Main section
```

```
This is a Quarto document

## Sub section title
This is a subsection

###
This is a further subsection

#
This is a new section

##
```

! Exercise

- Create a new .qmd file in VS Code.
- Copy and paste the above code.
- Render the code and check that a .html file has been generated.
- Add new section headings.
- Generate a table of contents.

3.4.2 Tables

Tables can provide a concise way to present information. Typical uses in a report might be to gather information about model parameters or to display data.

Here is an example of Quarto code to make a table:

```
| Parameter | Value | Unit  |
|---------|:-----|------:|
| $a$      | 1  |    ms$^{-1}$ |
| $b$      | 2  |   s$^{-1}$  |
| $c$        | 3    |     Nondim  |

: Demonstration of  table
```

The rendered version appears as

Table 3.1: Demonstration of table

Parameter	Value	Unit
a	1	ms^{-1}
b	2	s^{-1}
c	3	Nondim

You can find more info. on Quarto tables here.

! Exercise

- Copy and paste the above table into your .qmd file.
- Check that it renders.
- Add another column to the table and render.
- Add another row to the table and render.

3.4.3 Figures

To include an image file in a Quarto document, you must firstly store the image file in a sensible directory. Suppose you store the image in the same directory as your .qmd file. Then you could include the figure ('DemoFigureMA40001.png') using the syntax:

```
![This is the figure caption.](DemoFigureMA40001.png)
```

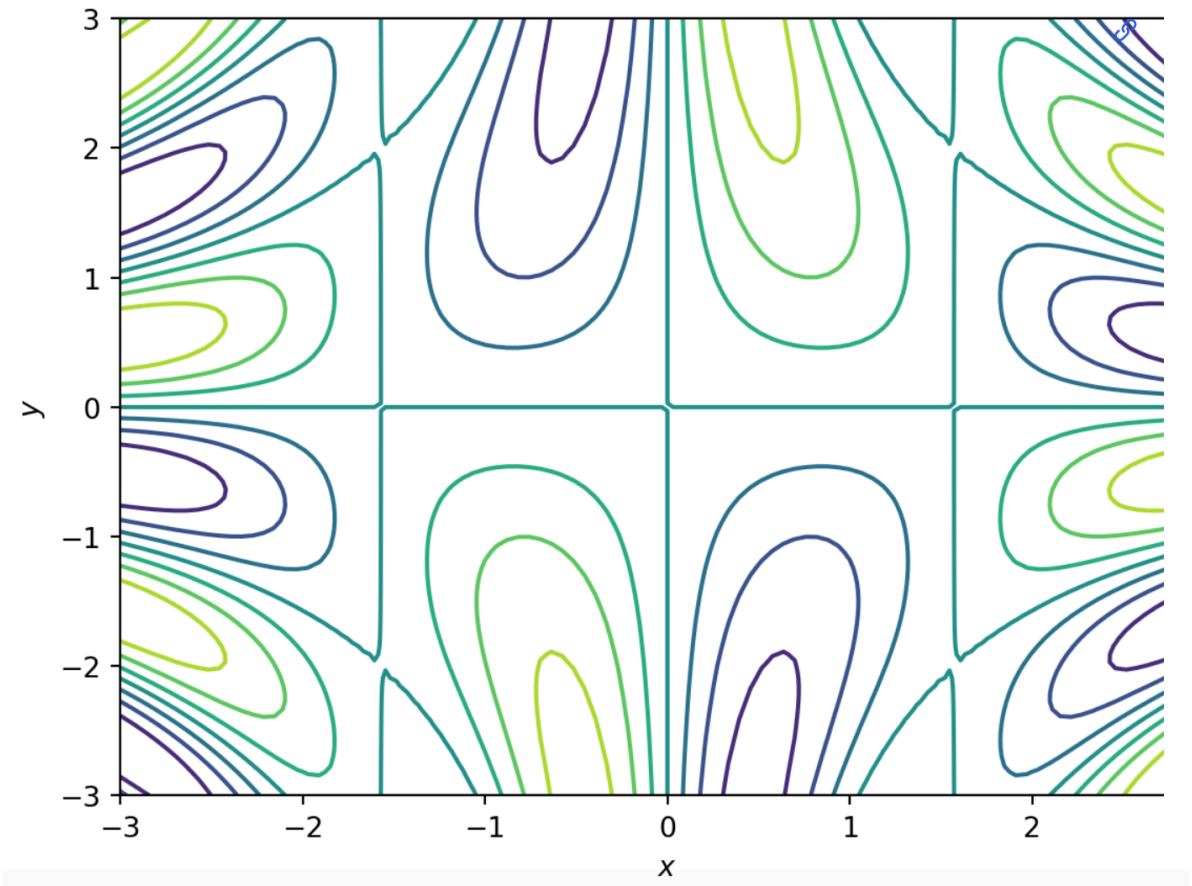


Figure 3.1: This is the figure caption.

You can find more info on Quarto figures here.

! Exercise

- Choose a .png file to include in your document.
- Save the .png file in the same directory as your .qmd file.
- Include the image and render your Quarto document.
- Try to modify the image height and width.
- Experiment with subfigures.

3.4.4 Cross referencing

Throughout your document you will need to refer to different objects that you have created (e.g. Tables, Figures, equations, theorems).

In Quarto, objects are tagged using a specific syntax (prefix for that specific object + unique tag). See Table 3.2 for some examples. Such objects can then be cross referenced using an '@' tag.

Table 3.2: A table with syntax for referencing some Quarto objects.

Object	Tag	Reference
Table	`{#tbl-mytable}`	`@tbl-mytable`
Figure	`{#fig-myfigure}`	`@fig-myfigure`
Equation	`{#eq-myequation}`	`@eq-myequation`

The table could be tagged as follows

```
| Parameter | Value | Unit  |
|---------|:-----|------:|
| $a$      | 1  |    ms$^{-1}$ |
| $b$      | 2  |   s$^{-1}$  |
| $c$        | 3    |     Nondim  |

: Demonstration of  table {#tbl-parameters}
```

The tag can now be cross referenced using the tag (see Table 3.3).

Table 3.3: Demonstration of table

Parameter	Value	Unit
a	1	ms^{-1}
b	2	s^{-1}
c	3	Nondim

In a similar manner we can cross reference an equation (see tag `#eq-emc2`). This is cross-referenced using the handle `@eq-emc2`.

```
$$
E=mc^2
$$ {#eq-emc2}
```

So if I defined an equation

$$E = mc^2, \tag{3.1}$$

I can refer to it in the text via Equation 3.1.

Quarto allows for definition and cross referencing of a range of mathematical objects (e.g. theorems, corollaries)

You can find out more about cross referencing here.

! Exercise

- Tag the table and figure that you made above.
- Cross-reference them in your document.
- Write down a theorem and cross reference it.

3.4.5 Citations and references

In your final report will need to provide a list of references that are cited at relevant points in your document.

This can be achieved relatively straightforwardly in Quarto.

You need to 1. create a `.bib` file (e.g. `mybibliography.bib`) and save it in a sensible directory (e.g. alongside your .qmd files).

2. to populate the .bib file with bibliographic entries, each of which will have a unique tag (e.g. `my_bib_tag`)
3. in your `.qmd` file you can cite a reference using the `@' handle (e.g.`@my_bib_tag').

You can find out more about citations in Quarto here.

i Creating a .bib file

You could use a reference manager such as Mendeley or Jabref.
Alternatively, you

- go to google scholar.
- go to settings tab
- In the Bibliography manager, select -> *Show links to import citations into* Bibtex
- Now when you search for a paper/textbook in Google scholar, there should be an additional link: 'import into Bibtex'
- Copy and paste the contents in the link into your .bib file
- Cite the source in your Quarto document.

3.4.6 Schematic diagrams

You can learn how to make diagrams here. Here is an example:

````
```{mermaid}
flowchart LR
  A[Hard edge] --> B(Round edge)
  B --> C{Decision}
  C --> D[Result one]
  C --> E[Result two]
```
````

3.5 Appendices

3.5.1 Submitting code

In your final report you should include codes that you have developed in the appendix. You can do this using a *code block*.

````
``` markdown
Copy and paste code here
```
````

3.6 Websites and blogs in Quarto

Given what you have achieved thus far, it is not a very big step to generate a Quarto project (e.g. books, blogs and dashboards).

You can create a project template on VSCode using *File->NewFile-> Quarto Project*.

The main difference with what we have previously done is that the configuration information is now defined in a *.yaml* file. See image below for an example.

Users > pmurray > Desktop > Teaching > MA40001Resources > pmurray > Final Report > ! _qu

```yaml
project:
  type: book
  output-dir: docs



book:
  title: "Project template"
  author: "J Smith"
  date: "12/22/2023"
  downloads: pdf

  abstract: this is an abstract. Can you write an approx. ten sentence su

  chapters:
    - index.qmd
    - intro.qmd
    - ResultsChapter1.qmd
    - summary.qmd
  appendices:
    - AIStatement.qmd
    - Appendix1.qmd

```

To render a Quarto project you need to use the terminal:

```
quarto render
```

3.7 Quarto Publishing

Given what you have achieved thus far, it is straightforward to publish Quarto output on the web.

There are some tutorials available on how to do this on the Quarto pages.

For guidance on publishing material see here.

4 Typesetting mathematics

4.1 The math environment in latex

Latex is a programming language used for mathematical typesetting. In its original form a latex file is compiled to generate a .pdf file. Mathematical notation is written in the 'math environment'. You can find a detailed introduction to latex here.

Within Quarto we can access the latex math environment by enclosing text within dollar symbols.

To typeset mathematics inline (e.g. $x+y=2$) we write ...

```
To typeset mathematics in line (e.g. $x+y=2$) we write ...
```

To typeset mathematics in a new line we use double dollar symbols. To obtain the expression

$$\frac{x+y}{2}=4,$$

we write

```
$$
\frac{x+y}{2}=4.
$$
```

It is worth spending some time familiarising yourself with basic latex commands.

Here is the same equation with a cross reference tag, i.e.

$$\frac{x+y}{2}=4. \tag{4.1}$$

Now I can cross reference Equation 4.1.

This has been achieved using

```
$$
\frac{x+y}{2}=4.
$${#eq-myequation}
```

It is worth knowing how to:

- write a system of aligned equations

$$\begin{aligned}\sum_{1}^{3} n &= 1+2+3\\ &= 6\\ \sum_{1}^{4} n &= 1+2+3+4\\ &= 10\end{aligned}$$

  Note that the equations are aligned such that the equal signs within the ampersands are at the same place;

- Use limits and sums, i.e.

$$\lim_{n\to\infty} \sum_{k=1}^{n} \frac{1}{k^2} = \frac{\pi^2}{6};$$

- define sets of numbers, i.e.

$$x^2 \geq 0 \qquad \text{for all} x \in \mathbb{R}$$

- have several expressions separated by some space

$$\sqrt{x^2+\sqrt{y}} \quad \overline{m+n} \quad \underbrace{a+b+\cdots+z}_{26};$$

- write a matrix

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \ldots \\ x_{21} & x_{22} & \ldots \\ \vdots & \vdots & \ddots \end{pmatrix};$$

- use conditional statements

$$y = \begin{cases} a, & \text{if } d > c, \\ b+x, & \text{in the morning,} \\ l, & \text{all day long;} \end{cases}$$

- adjust the size of brackets

$$(4)$$
$$(\frac{4}{3}) \leftarrow \text{this is bad}$$
$$\left(\frac{4}{3}\right) \leftarrow \text{much better}$$

4.2 Exercise: an example worksheet

Try to typeset the questions below in latex:

1. Denote the roots of the equation $x^2 + 5x + 1 = 0$ by $x_1 = \alpha$ and $x_2 = \beta$.
2. $$\sin\left(3\theta + \frac{\pi}{2}\right) = \frac{1}{2} \qquad 0 \leq \theta \leq \pi.$$
3. Express $\dfrac{2x - 26}{x^2 - 2x - 8}$ in partial fractions.
4. $$\text{(a) } \sum_{k=1}^{500}(2k - 21), \qquad \text{(b) } \sum_{k=1}^{20}\frac{(-2)^k}{5}, \qquad \text{(c) } \sum_{k=1}^{\infty} 5\left(\frac{1}{3}\right)^k.$$
5. $$\int_0^{1/2} x\sqrt{1 - 2x}dx.$$
6. $$\frac{\partial v}{\partial t} = \frac{\partial^2 v}{\partial x^2} + v^2(1 - v) \qquad x \in \mathbb{R}, \;\; t > 0.$$
7. $$\frac{\partial u}{\partial t} = (a - u + u^2 v) + \frac{\partial^2 u}{\partial x^2},$$
$$\frac{\partial v}{\partial t} = (b - u^2 v) + d\frac{\partial^2 v}{\partial x^2},$$
8. $$n_t = -(nu_t)_x + rn(1 - n),$$
$$Nu_{xx} + (\tau n\rho)_x = s\rho u,$$
$$\rho_t + (\rho u_t)_x = 0,$$

9.

$$I(t) = \begin{cases} I_0(t), & 0 \leq t < \tau, \\ I_0(t) + S(0) - S(t-\tau), & \tau \leq t < \tau + \sigma, \\ S(t-\tau-\sigma) - S(t-\tau), & \tau + \sigma \leq t, \end{cases}$$

10.

$$A = \begin{bmatrix} 0 & 5 & -2 \\ 5 & -7 & 5 \\ -2 & 5 & 0 \end{bmatrix}.$$

11.

$$f(x) = \begin{cases} x^2 & x < 0 \\ \sqrt{x+1} & x \geq 0 \end{cases}$$

12.

$$x^2 + 4x + 4 = 0. \tag{4.2}$$

Equation Equation 4.2 is a quadratic.

5 Code development

5.1 Introduction

It is strongly encouraged that you make use of opportunities to use and develop computer programs over the course of your project. You should discuss programming with your project supervisor.

In this section we will develop some Python codes. Python has been chosen because it:

- is versatile (there are lots of existing libraries to explore);
- is open source (you will have free access to it after you graduate);
- it can be easily embedded within Quarto;
- is used across industry and academia.

It is, of course, fine if you wish to use other programming languages in your project.

> **Note:**
>
> On the Uni machines Quarto has been set up so that it interfaces to an installation of Python with some pre-installed libraries. To set up on your own machine you will need to install Python, create a Python environment and interface with Quarto (Show and Run Commands -> Select Python Interpreter).

5.2 Some key ideas

5.2.1 Python in Quarto

To run Python within Quarto, create a new .qmd file. Within the file create a python environment as follows:

```
title: " Code development"
format: html
```

```
# Python code
```{python}

# Insert Python code here.

```
```

Note that a similar syntax is used to embed codes in other open source languages (R and Julia).

5.2.2 Calculator

We can use Python as a simple calculator.

```
3
```

> **!** Exercise
>
> Use Python to compute the product of 157 and 213.

5.2.3 Datatypes

It is worth having a concept of different 'datatypes' before you start progamming. In Python, some of the key datatypes are:

- float (decimals)
- int integers
- str (string of text)
- boolean (logical)

In Python, we do not explicitly declare a datatype (the interpreter figures this out based upon the variable that has been defined). See code below for an example.

```
<class 'int'>
<class 'float'>
<class 'str'>
```

> **!** Exercise
>
> Define a float and integer (e.g. $x = 2.0$ and $y = 7$).
> What datatype is the product?

5.2.4 Containers

Python comes with some default containers:

- Lists
- Tuples
- Sets
- Dictionaries

These have different uses.

5.2.4.1 Lists

Lists are mutable, i.e. they can be modified within the code. Note that Python indexing starts at zero!

```
[1, 2, 3]
1
['a', 'b', 'c']
a
['a', 'd', 'c']
```

> **!** Exercise
>
> 1. Create a list of five integers.
> 2. Sort the list in order of increasing size.
> 3. What happens if you try to access the '6th' entry in the list?
> 4. Append another entry to the list.

5.2.4.2 Tuples

Tuples are immutable container types, i.e.

```
(1, 2, 3)
1
('a', 'b', 'c')
a
```

5.2.4.3 Sets

Sets are denoted with curly brackets and behave like mathematical sets.

They are not indexed and there are no repeated entries.

```
{1, 2, 3}
{'a', 'c', 'b'}
```

> **!** Exercise
>
> 1. Create a set of floating point numbers.
> 2. Add another entry to the set.
> 3. Try to add an entry that already exists in the set.
> 4. Compute the number of items in the set.

5.2.4.4 Dictionaries

Dictionaries are used to interface datatypes that are connected. The first entry is known as the *key*.

Suppose John, Helen and Carol have been assigned projects 1,2 and 3, respectively. I could use a dictionary to connect these pieces of data as follows:

```
{'John': 1, 'Helen': 2, 'Carol': 3}
Helen's project is
2
```

! Exercise

1. Create a dictionary that connects your connects your modules codes for this semester with the module title ().
2. Access the module title using the module code as per the example.

5.2.5 Logical statements and control loops

It is essential to be competent using logical statements statements in any programming language. Here we will consider for loops and if statements.

Note the positioning of the colon and the indentation in the code example below:

```
0
1
2
3
4
```

! Exercise

Write a for loop that loops over one of the lists that you defined above and prints each item to screen.

Suppose we only wish to print out i when it is greater than three in the above code. We could introduce an if statement as follows (note colon and indentation again):

```
4
```

Note nested indentation of the for loop and if statement!

! Exercise

Modify the above for loop as follows. Write an if statement within the for loop so that only certain entries in the list get printed out.

Python has nicer/more efficient ways to loop over containers (lists, arrays etc.) but this is enough for now.

5.2.6 Writing functions

A good rule of thumb is that if you find yourself using the same piece of code three or more times, you should write a function (module). This avoids duplication of code.

Suppose we find ourselves manually computing the sum of positive integers many times, i.e.

$$s_2 = 0 + 1 + 2 = 3$$

and

$$s_3 = 0 + 1 + 2 + 3 = 6.$$

It makes sense to write a function that computes the sum for arbitrary n. Then we call that function when needed. This way the logic of the function is only written out in one place.

```
3
7140
```

> **!** Exercise
>
> Write a function that takes a list as an agrument and prints out the entries using a for loop.

5.2.7 Code debugging

You will have bugs in your code! And it can be incredibly frustrating trying to find them!

There are three main types of bugs:

- syntax errors (e.g. not calling a function correctly)
- runtime errors (e.g. dividing by zero)
- logic errors (more fundamental problems with the algorithm)

Copy and past the code snippets below into the python environment in Quarto. Can you find the bugs?

> **!** Exercise

```
sum=0.0
for i in range(5)
  sum=sum+i
```

! Exercise

```
for i in range(5):
sum=sum+i
```

! Exercise

What about here?

```
sum=0.0
for i in range(5):
  sum=(sum+i)/i
```

5.2.8 Good code hygiene

- Plan your code
- Keep code clean (e.g. use variables, write expressions that are easy to interpret)
- Test code often
- Develop code on simple cases where you know what the answer should be
- Comment code

5.2.9 Some tips for dealing with bugs

1. read the error message in terminal. Try to find the earliest sign of a problem in readout. This will tell you what line of your code is causing the first problem.
2. print variables to screen - do they have the expected datatypes/values?
3. check datatypes - are the objects you have defined doing what you think they are (e.g. if it is a matrix, does it have the expected shape)
4. if you are calling a function, is the syntax correct (hover the pointer over the function in VSCode or google 'python + name of the function'). Use the working example, usually found at the bottom of the help pages, to ensure syntax is correct. Do you understand what kind of objects the functions is going to return?
5. Test your code on a problem where you know the answer.
6. Python has a debugger that lets you follow the program as it executes.

5.3 Python libraries

5.3.1 Essential

5.3.1.1 Matrix computation (numpy)

Numpy is a widely used Python libary. It is a standard way to use arrays in Python. Numpy also contains lots of algorithm (e.g. linear algebra, calculus, mathematical functions, integration, random number generation etc.). You can find a beginner's guide here.

Numpy provides tools for calculating many mathematical operations.

```
sin (3.14) is:
0.0015926529164868282
1.2246467991473532e-16
```

We can also use numpy to define and manipulate arrays. In the example below we use python lists to define two 1D arrays.

```
[1 2 3 4 5 6]
The sum of a and b is:
[ 8 10 12 14 16 18]
The first entry in a is

1
```

We can use numpy for higher dimensional arrays

Calculate the determinant of the 2x2 matrix

$$A = \begin{pmatrix} 4 & 3 \\ 2 & 1 \end{pmatrix}.$$

```
The matrix is:
[[4. 3.]
 [2. 1.]]
The determinant is:
-2.0
```

! Exercise

1. Compute the trace of the matrix A.
2. Compute the determinant of the 3×3 matrix

$$B = \begin{pmatrix} 4 & 3 & 2 \\ 2 & 1 & 4 \\ 3 & 2 & 1 \end{pmatrix}.$$

3. Compute the eigenvalues of B.

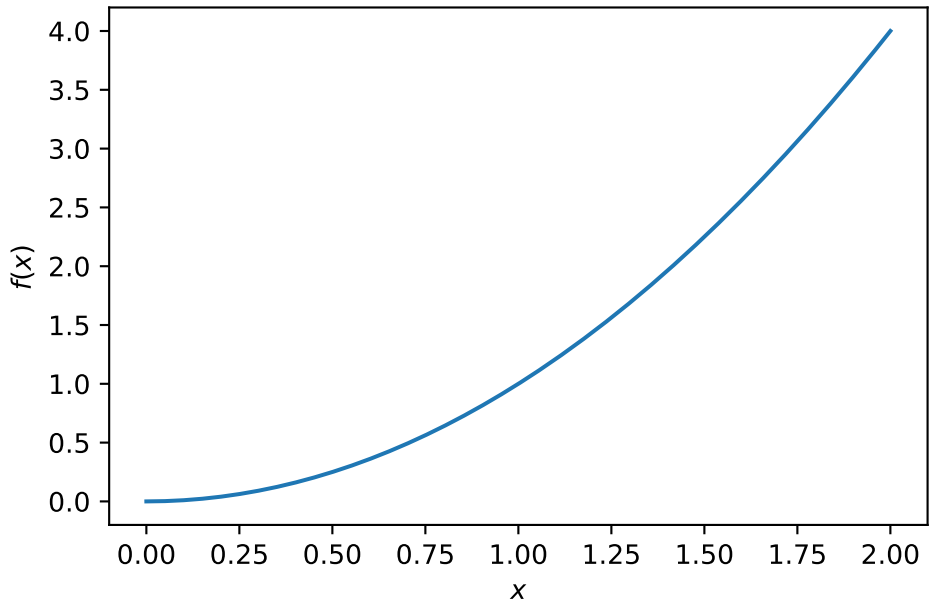
5.3.1.2 Plotting (matplotlib)

Matplotlib is a pyton library for plotting.

Let's plot some well-known functions.

Suppose we wish to plot the function

$$f(x) = x^2, \quad x \in [0, 2].$$



! Exercise

1. Change the domain to $x \in [0, 4]$.
2. Plot the function $f(x) = \sin(x)$.
3. Create two subplots.
4. Make the font on the labels larger
5. Add a legend to the figures.

5.3.1.3 Symbolic computation in Python (sympy)

sympy provides a symbolic calculator in Python. For example, suppose I want to differentiate or integrate the function

$$f(x) = x^3.$$

```
The derivative is
3*x**2
The derivative evaluated at x=1 is
3
```

! Exercise

Use sympy to compute the integral

$$\int_0^1 x^3 dx.$$

5.3.1.4 scipy

Scipy is library of useful mathematical algorithms. It builds upon numpy.

For example, suppose you want to numerically calculate the integral

$$\int_0^1 x^3 dx.$$

Using scipy

```
0.3333333333333333
```

! Exercise

Use scipy to numerically compute the integral

$$\int_0^1 \sin(x)dx.$$

5.3.2 Python libraries of interest to particular projects

5.3.3 Numerically solving differential equations (sci-py)

5.3.3.1 ODEs

Suppose we wish to solve the Lottka Volterra equations

$$\frac{dn}{dt} = n(1-p)$$
$$\frac{dp}{dt} = \alpha(n-1)$$

with initial conditions

$$n(0) = 2 \quad p(0) = 1.$$

! Exercise

1. Plot the solution in the np phase plane.
2. Annotate the phase plane by plotting the steady state and nullclines.

5.3.4 Optimisation (scipy.optimise)

Suppose we want to numerically estimate the local minima of the function

$$f(x) = x^4 - 2x^2.$$

We can do this using the scipy-optimize function minimise.

```
0.9999994698341323
```

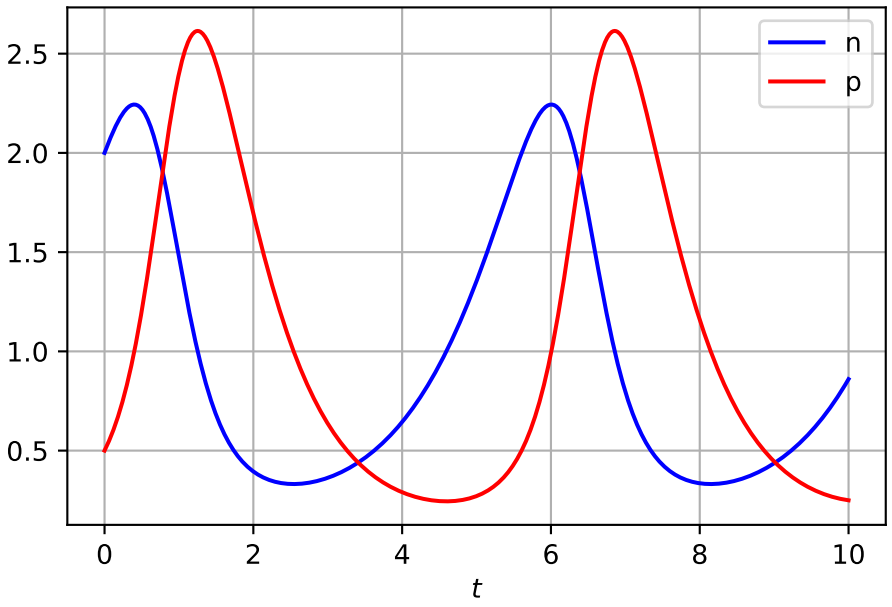


Figure 5.1: Numerically solving the Lotka Volterra ODEs.

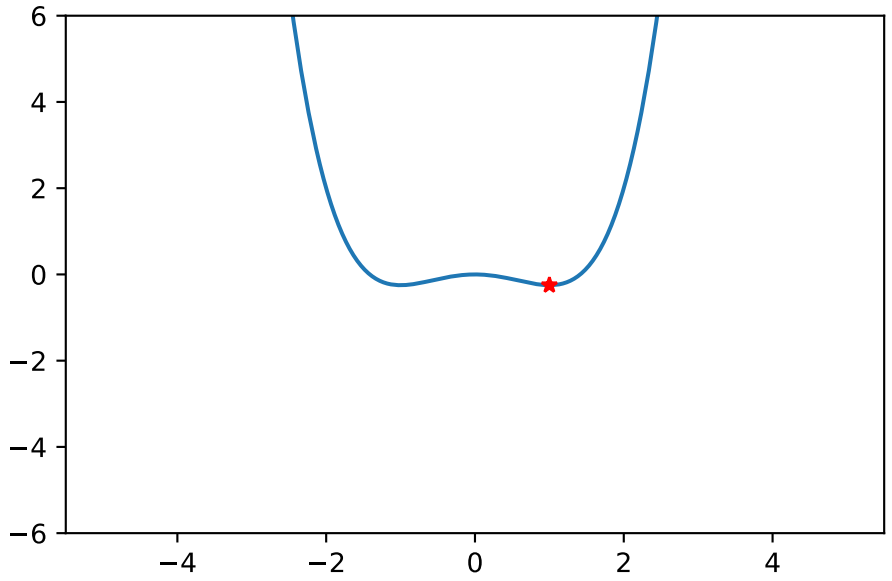


Figure 5.2: Finding local minima using scipy optimize.

5.3.5 Data analysis (pandas)

Pandas is the standard library for data analysis in Python.

The main datastructure is known as a *Dataframe*.

To view the first few rows of a dataframe use

	A	B	C	D
0	1	2	3	4
1	2	4	6	8
2	3	6	9	12
3	4	8	12	16
4	5	10	15	20

To generate a quick summary use

	A	B	C	D
count	5.000000	5.000000	5.000000	5.000000
mean	3.000000	6.000000	9.000000	12.000000
std	1.581139	3.162278	4.743416	6.324555
min	1.000000	2.000000	3.000000	4.000000
25%	2.000000	4.000000	6.000000	8.000000
50%	3.000000	6.000000	9.000000	12.000000
75%	4.000000	8.000000	12.000000	16.000000
max	5.000000	10.000000	15.000000	20.000000

To sort the data based upon values in a column

	A	B	C	D
0	1	2	3	4
1	2	4	6	8
2	3	6	9	12
3	4	8	12	16
4	5	10	15	20

To view a single column, e.g. the C column use

	C
0	3
1	6
2	9
3	12
4	15

To use logical operators to filter data, e.g. to identify only rows where the A column is positive

	A	B	C	D
4	5	10	15	20

To compute some basic statistics, e.g. the mean of each column

	0
A	3.0
B	6.0
C	9.0
D	12.0

5.4 Writing your own scripts

At some point (either over the course of your project or later) you will likely encounter a problem that cannot be solved using existing code libraries, i.e. you will need to write your own programmes. To prepare for this day, it is a good idea to practice your algorithm/code development skills on problems where the solutions are already known. For example, could you write an algorithm that:

- numerically solves the ODEs in Figure 5.1.
- finds the local minima in Figure 5.2.

Over the course of the project assessment you will be asked about the methods that you have used in your project. You should be able to describe how the method works.

It is much easier to defend the use of a method if you have a clear idea how to programme it; then the limitations of the method are much clearer.

6 File management

6.1 Introduction

Github is a developer platform that allows developers to create, store, manage and share their code. It is version-controlled, meaning that you have access to all previously *pushed* version of that file.

- Github is used across academia and industry.
- It provides you with a back-up for your project.

6.1.1 Getting started on Github

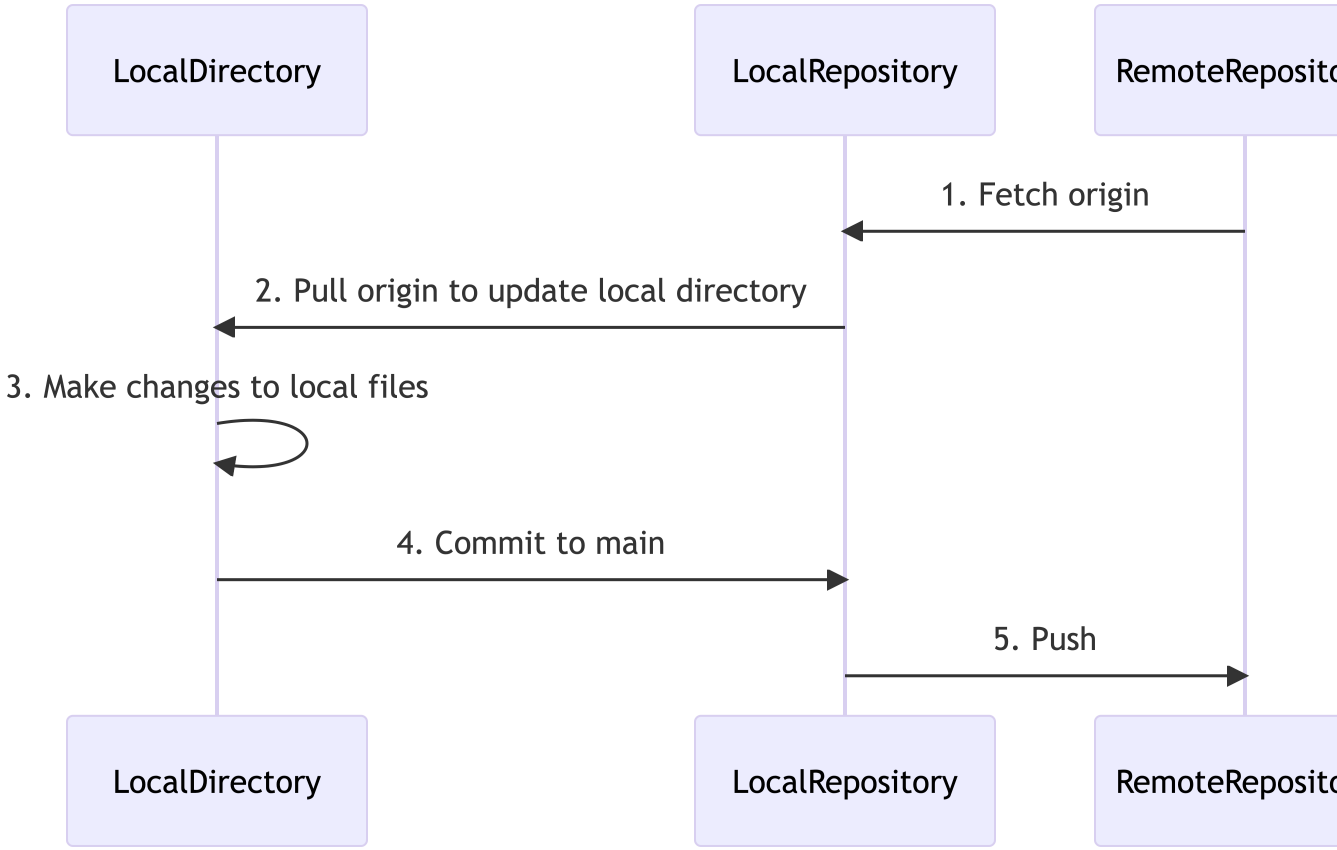
1. Launch the Github Desktop app on Apps Anywhere. To deal with a bug in AppsAnywhere:
    - Open CloudpagingPlayer (little blue icon on the toolbar on the bottom right of your screen):
    - *Stop* Github Desktop.
    - *Remove* Github Desktop.
    - Relaunch Github Desktop in AppsAnywhere.
2. You will need to create a github account if you do not currently have one.

6.1.2 Creating a repository

There are three important features of repository management:

1. Remote repository (i.e. the cloud)
2. Local repository (i.e. your local version of the files in the cloud)
3. Local directory

The workflow for managing your repository is as follows:



6.1.2.1 Create in the cloud and clone onto your machine

1. Go to your online github account.

- Create a new repository (why not MA40001Sandbox?).

2. On GithubDesktop

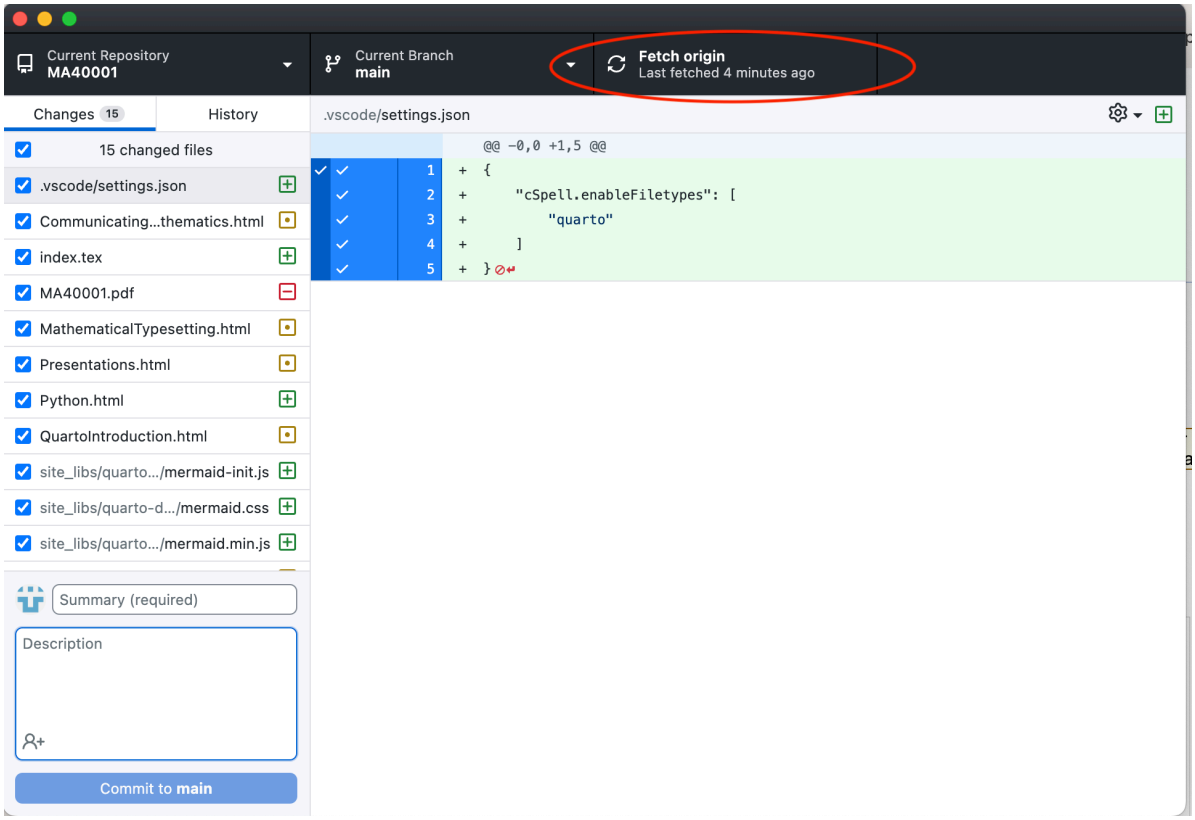
- Clone the repository onto your local machine (it now exists on your local machine).

6.1.2.2 Create a repository on your local drive and push to the cloud

1. In GithubDesktop … 'Create a new repository on your local drive'
2. Publish your repository to Github.
3. Check https://github.com to ensure that your repository has been created.

6.1.3 Add a file to your MA40001 repository

In Github Desktop refresh your local version of the repository: 1. 'Fetch Origin'

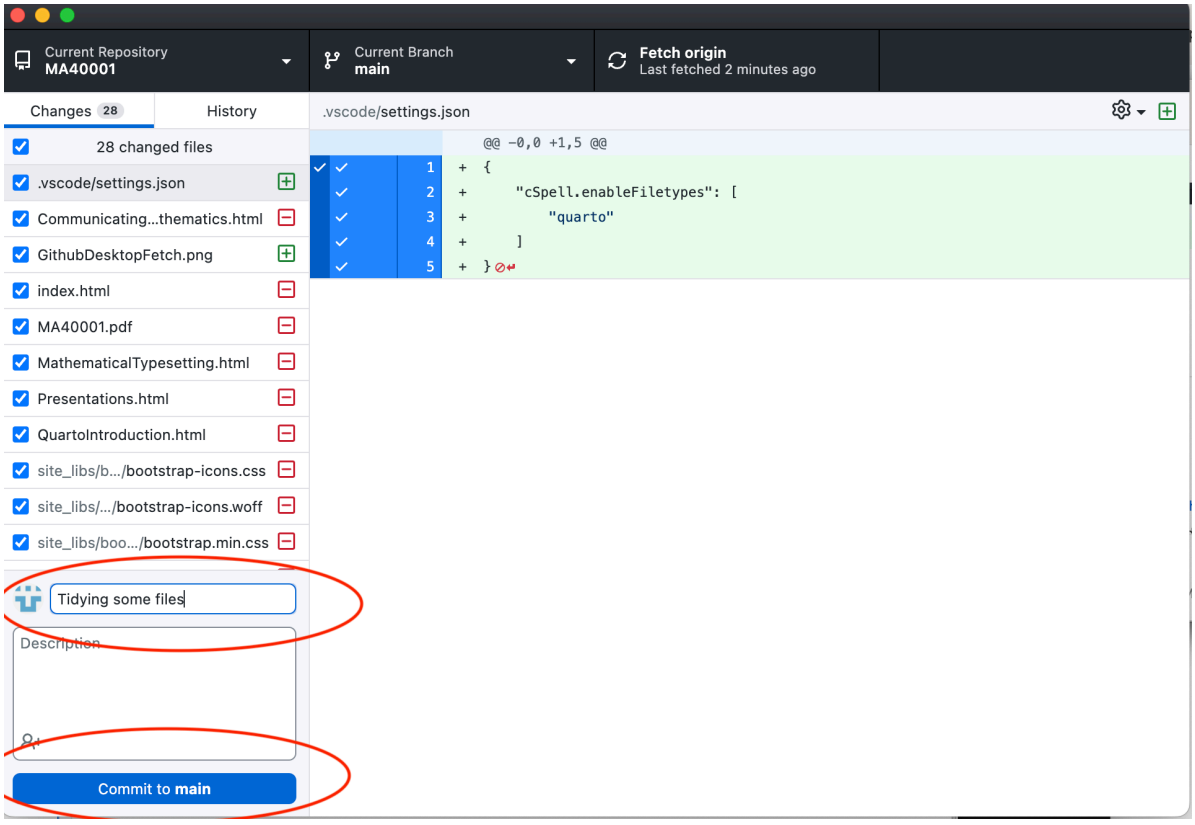


In VSCode

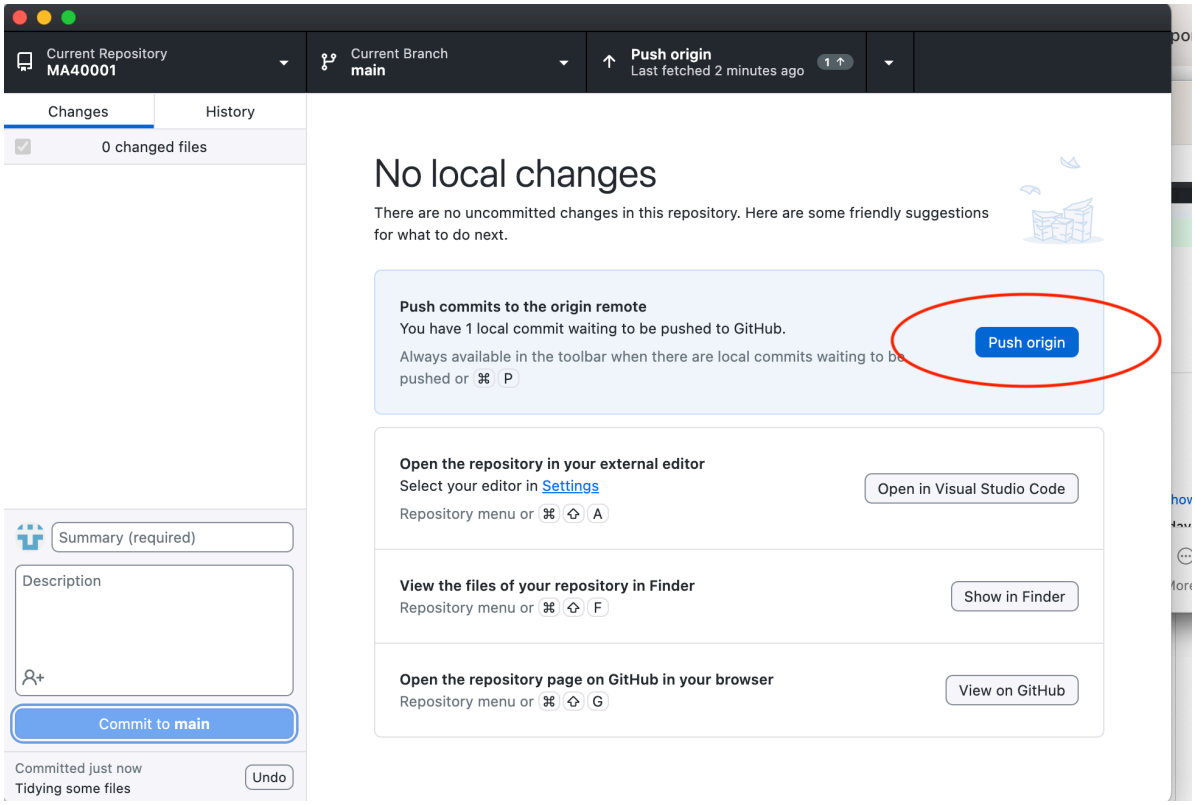
1. Open your MA40001 repository (File->Open Folder)
2. Create a new file (e.g. *testgithub.qmd*?)

In Github Desktop, push your changes to the remote repository:

1. 'Commit to Main' (you'll need to include a summary comment)


Current Repository
MA40001
Current Branch
main
Fetch origin
Last fetched 2 minutes ago
Changes 28
History
28 changed files
.vscode/settings.json
Communicating...thematics.html
GithubDesktopFetch.png
index.html
MA40001.pdf
MathematicalTypesetting.html
Presentations.html
QuartoIntroduction.html
site_libs/b.../bootstrap-icons.css
site_libs/.../bootstrap-icons.woff
site_libs/boo.../bootstrap.min.css
Tidying some files
Description
Commit to main
.vscode/settings.json
@@ -0,0 +1,5 @@
+ {
+ "cSpell.enableFiletypes": [
+ "quarto"
+]
+ }

2. 'Push origin'



6.1.4 Edit the existing file and push the modified version to github

- Open the file *testgithub.qmd* in VSCode
- Edit the text.
- Commit the file and push it to github.
- Check that the version of the file in the remote repository has changed to reflect your edit.

6.1.5 Revert to a previous version of a file using version control

Suppose that you have made a mistake and want to revert to a previous version of *testgithub.qmd*

- Find the file in the github repository
- Look back at previous versions of the file (click history).
- You could now download the copy that you want to revert to.

6.2 Forking MA40001 resources to obtain a template project and talk

Go to the MA40001Resources github page.

- Follow the instructions here for *forking* the repository. This creates a fork of the repository in your github account.
- Follow the instructions for cloning the repository. This creates a copy of the repository on your local machine.
- You should now have a local copy of the resources on your computer and own a new (forked) repository.
- create a new folder and copy template folders into it.
- commit and push back to the cloud.

Blog

A quarto blog is a nice way to record progress in your project.

1. Copy the blog folder in MA40001Resources
2. Each .qmd in the posts folder will be a new blog post
3. Render index.qmd and the blog will get refreshed.

Why do this?

1. You have a record of any text that you write - later you can copy and paste for assessment purposes
2. If you embed Python in the .qmd, you have a record of data associated with any numerical results (e.g parameter values, initial conditions).
3. You can revisit blog posts as your project develops.
4. Use github to store your files and you have a version controlled back up of your project.

7 Presentations

As part of your project you will give an approx. 10 minute presentation. To do this, you will have to develop some slides to present.

Quarto has a number of different options for producing slide decks. Below we will use Revealjs.