MATHEMATICAL INSTITUTE

UNIVERSITY OF OXFORD

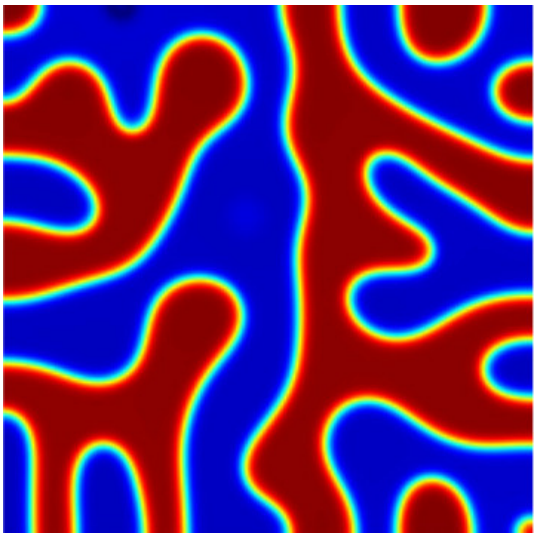
Computational Mathematics

Student Guide

Hilary Term 2023

by

Dr Kathryn Gillow



Acknowledgments: This course guide is based on previous editions by Andrew Thompson, Alberto Paganini, Vidit Nanda, Estelle Massart, Nick Trefethen and others. I am grateful to all previous authors.

Contents

Chapter 1

Introduction

The use of computers is widespread in all areas of life, and at universities they are used in both teaching and research. Computing fundamentally influences many areas of both applied and pure mathematics. MATLAB is one of several systems used at Oxford for doing mathematics by computer; others include Mathematica, Maple, Sage and SciPy/NumPy. These tools are sufficiently versatile to support many different branches of mathematical activity, and they may be used to construct complicated programs.

1.1 Objectives

The objective of this course is to help you learn about doing mathematics using MATLAB. Last term you were introduced to some basic techniques, by working through the Michaelmas Term Student Guide, which you are due to complete near the beginning of this term. After this, for the rest of this term, you will work alone on any two of the three projects presented in this booklet.

While MATLAB complements the traditional part of the degree course, we hope the projects may help you revise or understand topics which are related to your past or future studies. It is hoped that at the end of this course you will feel sufficiently confident to be able to use MATLAB (and/or other computer tools) throughout the rest of your undergraduate career.

1.2 Schedule

Lectures

There will be two lectures this term, at 11:00 on Tuesdays of Weeks 1 and 2 in lecture theatre L1.

Deadlines

- 12 noon, Monday, week 6: Online submission of first project.
- 12 noon, Monday, week 9: Online submission of second project.

You are free to **choose any two of the three projects described here**, and you don't have to do them in order. For instance, if you choose Projects A and C, you may submit Project C in Week 6 and Project A in Week 9.

Computer and demonstrator access

This term, the schedule for the practical sessions with demonstrators in Weeks 1 and 2 will be the same as for weeks 7 and 8, respectively, of Michaelmas Term. From Week 3 onwards, there are no fixed hours for each college, but drop-in sessions with demonstrators are scheduled for Mondays 3–4pm and Thursdays 3–4pm in Weeks 3–8 and also Wednesday 3–5pm in Week 5, Friday 3–5pm in Week 5, and Friday 3–5pm in Week 8. See the course website (`https://courses.maths.ox.ac.uk/course/view.php?id=604`) for updates and further information.

Demonstrators will be happy to help resolve general problems, but will not assist in the details of the actual project exercises.

1.3 Completing the projects

To carry out a project successfully, you need to master two ingredients: the mathematics, and the Matlab programming. Picking up the mathematics is a familiar activity that you practice when you attend a lecture or read your notes or mathematics texts. Building up a repertoire of Matlab commands and algorithmic ideas is a different skill that in some ways is more akin to learning a language. It is perfectly normal to do things somewhat inefficiently at first, and to achieve greater fluency as time goes by.

Before you get started on a project, it is a good idea to look over all the exercises to understand what is being asked. To complete most of the exercises, you will have to find the relevant commands that make Matlab do what you want. There are clues and guidance given for this within each project, although it will often be necessary to consult the Matlab help system.

Each project is divided into several exercises, and earns a total of 20 marks, which are **split between mathematical content and clarity of presentation**. The projects must be completed and submitted electronically before the Monday deadlines in weeks 6 and 9, according to the instructions given below. The marks will count towards Prelims and will not be released until after the examinations.

Your answers will ideally display both your proficiency in Matlab and your appreciation of some of the underlying mathematics.

Try to make your Matlab code elegant and concise — with comments as necessary so that it readable by human beings. Many of the exercises require you to make plots. Please be sure your plots are legible, with all their components well labelled and their fonts not too tiny.

1.3.1 Getting help

You may discuss with the demonstrators and others the techniques described in the Michaelmas Term Student Guide, as well as those found in the Matlab help pages and other sources. You may also ask the Course Director, i.e., me, to clarify any unclear points in the projects.

All projects must be your own unaided work. You will be asked to make a declaration to this effect when you submit them.

1.3.2 Debugging and correcting errors

Debugging means eliminating errors in a program. When you write a program, do not be disheartened if it does not work when you first try to run it. In that case, before

attempting anything else, type `clear` at the command line and run it again. This has the effect of resetting all the variables, and may be successful at clearing the problem.

If the program still fails, a good strategy is to locate the line where the problem originates. Remove semicolons if necessary, so that intermediate calculations are printed out and you can spot the first place where things fail. You may also want to display additional output, for which the `disp` command can be useful. If the program runs but gives the wrong answer, try running it for simpler and simpler cases, until you reduce the problem to a minimal instance for which it gives the wrong answer. Remember that you can always remove code from execution that is not relevant to a particular calculation by inserting the comment character %, so that Matlab ignores everything that follows on that line. The command `return` can also be used to halt execution of a Matlab program at a particular point.

The more advanced way to debug is to use the Matlab debugger, which you may enjoy getting to know. Information can be found in a number of places including `https://www.mathworks.com/help/matlab/matlab_prog/debugging-process-and-features.html`.

Website

This manual can be found at `https://courses.maths.ox.ac.uk/course/view.php?id=604`. This site will also incorporate up-to-date information on the course, such as corrections of any errors, possible hints on the exercises, and instructions for the submission of projects.

Legal note

Both the University of Oxford and the Mathematical Institute have rules governing the use of computers, and these should be consulted at `https://www.maths.ox.ac.uk/members/it/it-notices-policies/rules`.

Chapter 2

Preparing your project

To start, say, project A, find the template `projAtemplate.m` on the course website `https://courses.maths.ox.ac.uk/course/view.php?id=604`. Save this file as `projectA.m`, in a folder/directory also called `projectA`. Do not use other names.

You will be submitting this entire folder, so please make sure it contains only files relevant to your project. You will probably end up creating several `.m` files within this folder as part of your project.

2.1 Matlab publish

Execution of the file `projectA.m` should produce your complete answer. We will use the Matlab `publish` system.

```
publish('projectA.m','pdf')
```

This will create a PDF report in `projectA/html/projectA.pdf`. The lecturer will give examples of `publish` in the lectures and post an example file on the course website. You should also read `help publish` and `doc publish`.

The assessors will read this published report in assessing your project. It is important that the report be well-presented. You will definitely lose points if your `projectA` just executes the maths without any discussion.

- Divide `projectA.m` into headings for each exercise.
- You can call other functions and scripts from within `projectA.m`. A good way to make this code appear in your published document is to write `type <name of function>` where appropriate in `projectA.m`.
- Make sure that your discussion answers all the questions.
- Include appropriate Matlab output: not pages and pages, but enough to make it clear you have understood and answered the question. This will require some judgment, but it's worth the effort. Clear presentation is a lifelong skill.

The examiners may also run your codes.

Make sure you run publish one last time before submitting your project. Then double-check the results.

2.2 Zip up your files

Make a `projectA.zip` or `projectA.tar.gz` file of your projectA folder or directory including all files and subfolders or subdirectories. No `.rar` files please. It is recommended you make sure you know how to do this well before the deadline.

Double-check that you have all files for your project and only those files for your project.

2.3 Submitting the projects

Projects will be submitted online via Inspera; see `https://www.ox.ac.uk/students/academic/exams/open-book/online-assessments`. Further information about the submission process will be emailed to you later on if necessary.

Submission deadlines were given in Section 1.2, and these deadlines are strict. Penalties for late submission are specified in the Prelims Examination Conventions, which you can find at `https://www.maths.ox.ac.uk/members/students/undergraduate-courses/examinations-assessments/examination-conventions`. You should give yourself plenty of time to submit your projects, preferably at least a day or two in advance of the deadline. You will need your University Single Sign On username and password in order to submit each project, and also your examination candidate number (available from Student Self-Service). If you have lost track of your Single Sign On details, you will need to sort this out with IT Services well before the first deadline.

Chapter 3

Project A: Infectious Disease Modelling

The SIR model is a simple model of the spread of a disease in which, at any given time t, a population is split into three groups: the susceptible proportion, $S(t)$, who have not been infected; the infected proportion, $I(t)$ who are currently infected; and the recovered proportion, $R(t)$, who have recovered from the disease and are considered immune. The evolution of these proportions of the population is governed by the ordinary differential equations

$$\frac{\mathrm{d}S}{\mathrm{d}t} = -rS(t)I(t) \tag{3.1}$$

$$\frac{\mathrm{d}I}{\mathrm{d}t} = rS(t)I(t) - aI(t) \tag{3.2}$$

$$\frac{\mathrm{d}R}{\mathrm{d}t} = aI(t) \tag{3.3}$$

and the initial conditions are given by $S(0) = S_0$, $I(0) = I_0$ and $R(0) = 0$ where $S_0 + I_0 = 1$. Note that adding Equations (3.1), (3.2), and (3.3) gives

$$\frac{\mathrm{d}(S+I+R)}{\mathrm{d}t} = 0 \tag{3.4}$$

with $(S+I+R)(0) = 1$ so that $S+I+R = 1$ for all t, as expected. In Equations (3.1), (3.2), and (3.3), the parameter r is proportional to the number of people each infected person passes the disease to. The parameter a is inversely proportional to the recovery time for the disease.

3.1 Exercise A1

First use the explicit Euler scheme to find an approximate solution to the system of differential equations. In the explicit Euler scheme we use a small value of h, set $t_n = nh$ for $n = 0, 1, 2, \ldots, N$ and approximate $S(t_n)$ by S_n etc. The explicit Euler scheme then

approximates the derivatives by "finite differences" using the formulae

$$\frac{S_{n+1} - S_n}{h} = -rS_nI_n \tag{3.5}$$

$$\frac{I_{n+1} - I_n}{h} = rS_nI_n - aI_n \tag{3.6}$$

$$\frac{R_{n+1} - R_n}{h} = aI_n \tag{3.7}$$

for $n = 0, 1, 2, \ldots, N-1$.

Use $h = 0.1$, $N = 1500$ (so that the final time is $t_N = 150$), $r = 1/2$, $a = 1/3$ and $I_0 = 10^{-6}$ and calculate approximate solutions using the formulae (3.5), (3.6), and (3.7). Plot the resulting solutions on the same figure. Verify that $S_n + I_n + R_n = 1$ for all n.

3.2 Exercise A2

Since S, I, and R are positive, Equation (3.1) tells us that S is a decreasing function of time and Equation (3.3) tells us that R is an increasing function of time. We can write Equation (3.2) as

$$\frac{\mathrm{d}I}{\mathrm{d}t} = I(t)(rS(t) - a) \tag{3.8}$$

so that I is increasing if $rS - a > 0$ and decreasing if $rS - a < 0$. In particular, if $rS_0 - a < 0$ then I is decreasing for all time and the disease dies out, whereas if $rS_0 - a > 0$ then I increases to begin with, reaches a maximum when $S(t) = a/r$ and then decreases to zero.

Demonstrate this behaviour by running your code with $h = 0.1$, $N = 1000$, $I_0 = 0.1$ and

1. $r = 1/2$, $a = 1/3$
2. $r = 1/3$, $a = 1/3$

3.3 Exercise A3

From Equations (3.1) and (3.2) we get

$$\frac{\mathrm{d}I}{\mathrm{d}S} = -1 + \frac{\rho}{S} \tag{3.9}$$

where $\rho = a/r$. We also have the condition that $I = I_0$ when $S = S_0$. Use `dsolve` to solve this differential equation with $I_0 = 10^{-6}$, $r = 1/2$ and $a = 1/3$.

3.4 Exercise A4

It is of interest to know the proportion of people who are infected at the peak of an epidemic. In the case where $rS_0 - a > 0$, Equation (3.8) (or Equation (3.9)) tells us that the maximum value of I is obtained when $S = \rho$. Use the parameters and exact solution calculated in Exercise A3 and evaluate I at $S = \rho$ to find the exact maximum value of I. Now use your explicit Euler code (with the same parameters) and take in turn $N = 100$, 1000, and 10000. Find the approximate maximum value of I for each value of N (i.e. find the solutions I_n using the explicit Euler code and then take the maximum of that vector). What do you notice about the error in your numerical approximation to the maximum?

Chapter 4

Project B: Quadrature

In this project we will look at integrals of the form

$$I(f) \quad = \quad \int_{-\infty}^{\infty} \mathrm{e}^{-x^2} f(x) \mathrm{d}x \tag{4.1}$$

where $f(x)$ is a function that decays appropriately as $x \to \pm\infty$ so that the integral in (4.1) is finite. Such integrals occur in fields as seemingly diverse as computing moments of a normal probability density function and computing reaction rates in chemistry. In all but the simplest cases, (4.1) cannot be evaluated analytically, but we can use numerical approximations to accurately approximate its value. A numerical approximation, known as a quadrature rule, takes the form

$$I_n(f) \quad = \quad \sum_{j=1}^{n} w_j f(x_j) \tag{4.2}$$

where the $\{x_j\}$ are the real and distinct nodes, and the $\{w_j\}$ are the positive weights.

4.1 Exercise B1

First consider the Gauss-Hermite quadrature rule. This is designed so that (4.2) is exact whenever $f(x)$ is a polynomial of degree at most $2n-1$. It can be shown that when $n = 3$ the nodes are $-\sqrt{3/2}$, 0, and $\sqrt{3/2}$ and the weights are $\sqrt{\pi}/6$, $2\sqrt{\pi}/3$, and $\sqrt{\pi}/6$. Verify that this quadrature rule is exact for $f(x) = x^k$ for $k = 0, 1, \ldots, 5$ but not for $f(x) = x^6$. You may use the fact that

$$\int_{-\infty}^{\infty} \mathrm{e}^{-x^2} x^k \mathrm{d}x \quad = \quad \begin{cases} \frac{k!\sqrt{\pi}}{2^k (k/2)!} & \text{if } k \text{ is even} \\ 0 & \text{if } k \text{ is odd} \end{cases} \tag{4.3}$$

to evaluate the integral exactly.

4.2 Exercise B2

An algorithm for computing the nodes and weights for Gauss-Hermite rules was introduced by Golub and Welsch in 1969. The idea is to construct a symmetric tridiagonal matrix $T \in \mathbb{R}^{n \times n}$ with entries $T_{j,j+1} = T_{j+1,j} = \sqrt{j/2}$ for $j = 1, 2, \ldots, n-1$. The Matlab command

`eig` can be used to compute the eigenvalues $\{\lambda_j\}$ and eigenvectors $\{v_j\}$. Then the nodes for Gauss-Hermite quadrature are given by $x_j = \lambda_j$ and the weights are the squares of the first entries of the corresponding eigenvectors multiplied by $\sqrt{\pi}$. Note that the eigenvectors must be normalised so that the squares of their entries sum to one (which the `eig` command will do). Use this to write a Matlab function of the form `[x,w]=gausshermite(n)` which returns the nodes and weights for an n-point Gauss-Hermite quadrature rule and check the results with $n = 3$ agree with those given above.

4.3 Exercise B3

Consider the function $f(x) = \exp(\sin(x^2))$. Approximate $I(f)$ by $I_n(f)$ with $n = 1000$. This should be a very accurate approximation. Now make a semilogy plot of $|I_n(f) - I_{1000}(f)|$ against n for $n = 1, 2, \ldots, 900$, and comment on the shape of this plot.

4.4 Exercise B4

Take $n = 50$ and make a semilogy plot of the nodes weights produced by the function `gausshermite` (the nodes should be on the x-axis and the weights on the y-axis). You should see that when $|x| > 5$ the weights are less than 10^{-10} and so the majority of the terms contribute very little to the sum in Equation (4.2). This motivates the use of the composite trapezium rule on the truncated interval $[a, b]$, namely

$$T_n(f) = \frac{b-a}{2(n-1)} \left(\mathrm{e}^{-\hat{x}_1^2} f(\hat{x}_1) + 2 \sum_{j=2}^{n-1} \mathrm{e}^{-\hat{x}_j^2} f(\hat{x}_j) + \mathrm{e}^{-\hat{x}_n^2} f(\hat{x}_n) \right) \tag{4.4}$$

$$= \sum_{j=1}^{n} \hat{w}_j \mathrm{e}^{-\hat{x}_j^2} f(\hat{x}_j) \tag{4.5}$$

where $\hat{x}_j = a + (j-1)(b-a)/(n-1)$ for $j = 1, 2, \ldots, n$ and

$$\hat{w}_j = \begin{cases} \frac{b-a}{2(n-1)} & j = 1, n \\ \frac{b-a}{(n-1)} & j = 2, \ldots, n-1 \end{cases} \tag{4.6}$$

Write a Matlab function of the form `[x,w]=trappts(n,a,b)` which returns the nodes and weights for an n-point composite trapezium rule on the interval $[a, b]$. Use this to repeat Exercise B3 but with the composite trapezium rule on the interval $[-5, 5]$. Comment on your results.

Chapter 5

Project C: Continued Fractions

Every real number x can be written as a *continued fraction* in the form

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \ldots}}}, \tag{5.1}$$

where the a_k are all integers. Here a_0 may be negative or zero, but all other coefficients are positive. We will abbreviate Equation (5.1) by writing $x = [a_0; a_1, a_2, a_3, \ldots]$. If x is irrational, there is a unique continued fraction representation of x, whereas if x is rational there are two equivalent forms, $x = [a_0; a_1, a_2, \ldots, a_n]$ and $x = [a_0; a_1, a_2, \ldots, a_n - 1, 1]$. For example, we may write

$$\frac{15}{8} = 1 + \cfrac{1}{1 + \cfrac{1}{7}} = 1 + \cfrac{1}{1 + \cfrac{1}{6 + \cfrac{1}{1}}} \tag{5.2}$$

so that $15/8 = [1; 1, 7] = [1; 1, 6, 1]$. It can also be shown that x has a finite representation as a continued fraction if and only if x is rational.

In order to compute a continued fraction representation of x, define $\lfloor x \rfloor$ to be the *floor* of x (or the integer part of x, namely the closest integer to x when rounding down), and define $\{x\} = x - \lfloor x \rfloor$ to be the fractional part of x. Note that $0 \leq \{x\} < 1$. The continued fraction representation of x is $[\lfloor x \rfloor; a_1, a_2, a_3, \ldots]$ where $[a_1; a_2, a_3, \ldots]$ is the continued fraction representation of $1/\{x\}$.

For the example above with $x = 15/8$ we have $\lfloor 15/8 \rfloor = 1$ and $\{15/8\} = 7/8$. So $a_0 = 1$ and we need to find the continued fraction representation of $1/\{x\} = 8/7$. Then $a_1 = \lfloor 8/7 \rfloor = 1$ and $\{8/7\} = 1/7$. Then $1/\{8/7\} = 7$ so $a_2 = 7$ and the continued fraction terminates.

5.1 Exercise C1

Write a Matlab function that takes as arguments an integer, n, and a real number, x, and computes a continued fraction representation of a real number in the form $x = [a_0; a_1, a_2, \ldots, a_n]$. If x has a finite continued fraction representation, $x = [a_0; a_1, a_2, \ldots, a_m]$ with $m < n$, the

function should return this finite representation. Use this function to calculate the coefficients in the continued fractions with $n = 14$ for the numbers

1. exp(1)
2. π
3. the golden ratio, $\phi = (1 + \sqrt{5})/2$
4. $\sqrt{2}$
5. 3.0578

You may wish to experiment with the use of `sym` for the values of x.

5.2 Exercise C2

The *convergents* of a continued fraction are the initial terms in the continued fraction, i.e.

$$a_0, \quad a_0 + \frac{1}{a_1}, \quad a_0 + \frac{1}{a_1 + \dfrac{1}{a_2}}, \quad a_0 + \frac{1}{a_1 + \dfrac{1}{a_2 + \dfrac{1}{a_3}}} . \tag{5.3}$$

It can be shown that these can be written as rational approximations to x as p_n/q_n where $p_0 = a_0$, $p_1 = a_0 a_1 + 1$, $q_0 = 1$, $q_1 = a_1$ and

$$p_n = a_n p_{n-1} + p_{n-2} \tag{5.4}$$

$$q_n = a_n q_{n-1} + q_{n-2} \tag{5.5}$$

Compute the convergents for each of the numbers in Exercise C1 (i.e. compute p_k/q_k for $k = 0, 1, \ldots, 14$) and make a plot of the error $|p_n/q_n - x|$ as a function of n for each. (The plot may be clearer if you use a `semilogy` plot.)

5.3 Exercise C3

Hurwitz's Theorem states that if x is irrational there are infinitely many rationals p/q with

$$\left| x - \frac{p}{q} \right| < \frac{1}{q^2\sqrt{5}} . \tag{5.6}$$

By plotting $|\phi - p_n/q_n|$ as a function of n and $1/(q_n^2\sqrt{5})$ as a function of n, show that the convergents of ϕ are almost exactly $1/(q_n^2\sqrt{5})$ away from ϕ.

5.4 Exercise C4

It can be shown that the coefficients of a continued fraction of the square root of a non-square number are periodic in the sense that

$$\sqrt{D} = \left[a_0; \overline{a_1, a_2, \ldots, a_{m-1}, 2a_0}\right] \tag{5.7}$$

where the sequence of numbers under the bar is repeated infinitely many times. Convince yourself that this is true by looking at a suitable number of coefficients in the continued fraction form of $\sqrt{19}$.

5.5 Exercise C5

One application of continued fractions is in finding integer solutions of Pell's equation, $x^2 - Dy^2 = 1$, where $D \in \mathbb{N}$. Here we compute a continued fraction representation of $\sqrt{D}$ as in Equation (5.7), along with the corresponding convergents. If the length, m, of the period is even then an integer solution of Pell's equation is $(x, y) = (p_{m-1}, q_{m-1})$, and if m is odd an integer solution is $(x, y) = (p_{2m-1}, q_{2m-1})$. Use this to find solutions to Pell's equation for $D = 19$ and $D = 17$.