Assignment

Key Topics: Floating point variables, matrix methods, interpolation, Fourier transforms and ordinary differential equations

The basic idea of this assignment is to work through a series of coding problems designed to familiarise you with the introductory topics of the course.

1 Floating point variables

a) Write a short program to find the nearest representable real numbers higher and lower than a given floating point value. Hence, find the upper and lower difference within which real numbers are rounded to the given floating point value. Return this range expressed as fractions of the value. Use your program to find the fractional rounding range of 0.25.

b) Again, for a floating point value of 0.25, take the nearest two values found using your program and then find the two nearest values to these in turn. (In each case, one of these latter two should be 0.25 if your program is working correctly.) Find the fractional ranges for both these numbers. Comment on your results from parts a) and b).

2 Matrix methods

a) Write code to carry out LU decomposition of an arbitrary $N \times N$ matrix using Crout's algorithm. Return the result in the form of an $N \times N$ matrix containing all the elements of $\mathbf{U}$, and the non-diagonal elements of $\mathbf{L}$. Your routine may either return the result as a new matrix, or just overwrite the input matrix.

b) Use your routine to express the matrix

$$\mathbf{A} = \begin{bmatrix} 3 & 1 & 0 & 0 & 0 \\ 3 & 9 & 4 & 0 & 0 \\ 0 & 8 & 20 & 10 & 0 \\ 0 & 0 & -22 & 31 & -25 \\ 0 & 0 & 0 & -35 & 61 \end{bmatrix} \tag{1}$$

as a product of upper and lower diagonal matrices, and compute $\det(\mathbf{A})$ using your answer.

c) Write a short function that solves the matrix equation $\mathbf{L} \cdot \mathbf{U} \cdot \vec{x} = \vec{b}$ for $\vec{x}$ using forward and back substitution, where $\mathbf{L}$ and $\mathbf{U}$ are $N \times N$ upper and lower-triangular matrices, respectively. Here $\mathbf{L}$, $\mathbf{U}$ and the vector $\vec{b}$ should be input parameters of your routine.

d) Use your routine together with that of part a) to solve

$$\mathbf{A} \cdot \vec{x} = \begin{bmatrix} 2 \\ 5 \\ -4 \\ 8 \\ 9 \end{bmatrix} \tag{2}$$

for $\vec{x}$, where $\mathbf{A}$ is the matrix given in part b).

e) Use your routine together with that of part a) to calculate $\mathbf{A}^{-1}$.

3 Interpolation

a) Write a simple routine to perform Lagrange polynomial interpolation on a tabulated set of x–y data.

b) Write a routine to perform cubic spline interpolation on a tabulated set of x–y data, using the natural spline boundary condition. The matrix solver should be called as an external routine rather than coded inline into the interpolator like in Numerical Recipes. Use your own matrix solver from Q2 for this.

c) Consider the following table of data

x	y
-0.75	0.10
-0.5	0.30
-0.35	0.47
-0.1	0.66
0.05	0.60
0.1	0.54
0.23	0.30
0.29	0.15
0.48	-0.32
0.6	-0.54
0.92	-0.60
1.05	-0.47
1.5	-0.08

Using both Lagrange polynomial interpolation and natural cubic splines, interpolate the tabulated function everywhere between its first and last entries, and plot the results on the same set of axes. Use some high enough density of points that your curves look continuous and smooth (where relevant). Briefly comment on the result.

4 Fourier transforms

a) Write a program that uses either numpy.fft or the external FFT library FFTW (www.fftw.org, available in Python as package pyFFTW) to convolve the signal function

$$h(t) = 4 \qquad \text{for } 5 \le t \le 7 \tag{3}$$

$$h(t) = 0 \qquad \text{for } t < 5 \text{ and } t > 7 \tag{4}$$

with the response function

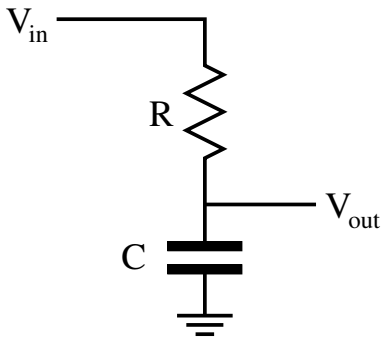
$$g(t) = \frac{1}{\sqrt{2\pi}} e^{-t^2/4}. \tag{5}$$

Explain your design choices regarding sampling, aliasing and padding.

b) Plot $h(t)$, $g(t)$ and $(g * h)(t)$ over appropriate ranges in t.

5 Ordinary differential equations

Consider the simple "impedance-divider" circuit with a resistor R and capacitor C as shown in the diagram. The input voltage $V_{\text{in}}(t)$ is a controllable voltage source and the circuit gives an output voltage $V_{\text{out}}(t)$.



a) You should remember from electronics courses that $Q = CV_{\text{out}}$ gives charge Q stored on the capacitor. The current through the resistor is dQ/dt, so $V_{\text{in}} - V_{\text{out}} = R\,dQ/dt$. Express this problem as a first-order ordinary differential equation for $V_{\text{out}}(t)$ given a known time-dependent input voltage $V_{\text{in}}(t)$, using scaled variables as appropriate.

b) Write a program which finds V_{out} at a number time steps given an input function $V_{\text{in}}(t)$. The program should have a switch enabling it to solve the problem using the fourth-order Runge-Kutta method or the fourth-order Adams-Bashforth method.

c) Take the specific case of constant $V_{\text{in}} = V_0$ for all $t < 0$, such that the circuit has settled into time-independent state. At $t = 0$, V_{in} changes instantaneously (compared to any other timescales in the problem) to $V_{\text{in}} = 0$ and remains at this value for all $t \geq 0$. Use your program to find V_{out}. Justify your choice of time period and step size. Compare the two methods to the analytic solution and comment on your results.

d) Using the fourth-order Runge-Kutta method, rerun your program over the same time period but with the step size changed up and down by a factor of two. Comment on whether your solution error scales as you expect.

e) Now use your program for the case where again $V_{\text{in}} = V_0$ for $t < 0$, but it is then a square wave with a period T, i.e. $V_{\text{in}} = 0$ for $0 \leq t < T/2$ and $V_{\text{in}} = V_0$ for $T/2 \leq t < T$, with this pattern then repeating. Using the fourth-order Runge-Kutta method, find two solutions using two different values of the period, namely $T = 2RC$ and $T = RC/2$. Comment on your results.

Computational Physics Assignment: Guidelines & Submission

Write-up

The Assignment write-up should provide the answers to each question including any requested comments on the results. It should give enough explanation for the marker to understand your answer. As a guide, your written set of answers to the questions should not need to be more than about 1000 words long. *Long report-style documents will not gain you any marks.*
You must explicitly answer all questions – do not rely on us to find the answers by running your code. You should also briefly describe any validations that you performed on the code, to help the reader (and you!) gain confidence that it was implemented correctly. As in real life, code that has not been validated is often worse than no code at all – *presenting validations of your algorithms **will** gain you marks.*

This assignment takes the form of short exercises rather than a project or a lab, so your write-up should reflect this. With five major questions, you should not really be looking at more than about 200 words for each question. This should indicate to you the level of detail that you need to go into. You should answer each question directly; abstracts, aims, conclusions, a detailed review of the techniques covered in class, etc., are not necessary. There will not be space to repeat the premise of a question in depth.
Remember that visualisation is very important; a plot can tell the reader a huge amount, and each one should be accompanied by a full and substantial caption. However, do not overburden the reader with too many plots; be selective, combine related curves and data points into single plots etc., so that every plot that the reader does spend time digesting is worth their while!

Marks will be awarded for:

- Results and validation - 75%
- Quality of programming - 25%

Code

You must implement the parts of the code related to the question topic yourself. The general principle is that you need to demonstrate your abilities to understand, and hence be able to implement, the algorithms needed for the assignment questions. If you use built-in functions for major parts of these, then this will not show the assessor that you understand and so you will lose marks. Hence, the default should be that anything directly related to the topic being studied must be implemented by you.
For example, in general you can use simple built-in functions (for things like multiplying matrices and vectors, sorting lists, etc.) in all cases. If you are being tested on matrix albegra, you cannot use higher level built-in functions such an inversion or LU decomposition function. However, if the problem concerns solving partial differential equations and, as an internal step, this requires a matrix inversion, then it would be acceptable to use a built-in function for inversion at this point.
It will be clear from the assignment questions what you are being tested on for each question and hence what needs to be coded by you. (The projects later in the term will similarly contain specific guidance about what should be done in each case.) If you are unsure if you should use a particular built-in function, please discuss this with your demonstrator during the consultation sessions.

All source code that you write to solve these problems should be included in your submission. The code you submit should compile/execute and be able to reproduce all the results and plots that you present in your write-up—without the reader having to comment in or out sections of code or otherwise edit any files. Be mindful that we will reproduce the results in your report using the code you submit - an efficient process that produces everything we need with a minimal number of commands is preferable to having to run twenty different executables and plotting scripts over many different inputs.
You must include a `README` text file describing how to invoke your code to produce the submitted results. If your code needs to be compiled, please include a `Makefile` that works with common (**free**) compilers, and give instructions on its use.

Your code needs to be well commented and easy to read! We need to be able to see that you understand what you've done. As a good rule of thumb, you should aim to have *at least* one line of comments for every two or three lines of code. That doesn't mean write a paragraph at the start of a function and then nothing in the function – try to tell us why you're doing what you're doing in basically every line. We are aware that good code should be "self-commenting", but for the purposes of this assessed assignment, we require you to write your own comments in English to make it easier to follow your code. Make extra sure to comment on any tricks or strategies you have used to make the code better or more efficient. Commenting **will** impact your marks!

Submission

Your answers to the questions and source code must be submitted to "Computational Physics 2020–21" on Blackboard Learn no later than **12 noon** on **Monday 16$^{\mathbf{th}}$ November (2020)**. Please submit your writeup, including plots, in a single file in PDF format only. Please put multiple source code files into a ZIP file before uploading. There will be separate places to submit the answers and the code. Please remember that both your answers and code will be run through plagiarism detection software.