Assignment

Key Topics: Floating point variables, matrix methods, interpolation, Fourier transforms and random numbers

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 calculate the floating-point machine accuracy on your system.

b) What is the machine accuracy of your chosen hardware and programming language for single, double and extended precision floating point variables? What would you expect them to be, theoretically?

2 Matrix methods

a) Write a code to carry out LU decomposition of an arbitrary $N \times N$ matrix using Crout's method. Return the result in the form of an $N \times N$ matrix containing all the elements of U, and the non-diagonal elements of L. Your routine may either return the result as a new matrix, or just overwrite the input matrix. You needn't implement pivoting (although you can if you like).

b) Use your routine to express the matrix

$$A = \begin{bmatrix} 3 & 1 & 0 & 0 & 0 \\ 3 & 9 & 4 & 0 & 0 \\ 0 & 9 & 20 & 10 & 0 \\ 0 & 0 & -22 & 31 & -25 \\ 0 & 0 & 0 & -55 & 60 \end{bmatrix} \tag{1}$$

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

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

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

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

for $\mathbf{x}$, where A is as per Q1.b).

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

3 Interpolation

a) Write a simple routine to perform linear 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
−2.1	0.012155
−1.45	0.122151
−1.3	0.184520
−0.2	0.960789
0.1	0.990050
0.15	0.977751
0.8	0.527292
1.1	0.298197
1.5	0.105399
2.8	3.936690×10^{-4}
3.8	5.355348×10^{-7}

Using both linear 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).

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

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

$$h(t) = 0 \qquad \text{for } t < 3 \text{ and } t > 5 \tag{4}$$

with the response function

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

using the spectral techniques described in class. 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 Random numbers

a) Using a decent built-in uniform deviate generator (or one from an external library – either way, justify your choice), write a short program to compute 10^5 uniformly-distributed random numbers over the interval $x \in [0, 1]$, based on a single seed. Plot the resultant distribution as a binned histogram, with the value of x on the x axis and the number of samples in the bin on the y axis. Choose an appropriate number of bins.

b) Now compute random numbers distributed over the interval $x \in [0, \pi]$ as

$$\text{pdf}(x) = \frac{1}{2} \sin(x). \tag{6}$$

Use the transformation method, and show your preparatory working out. Plot the resultant distribution for 10^5 points based on a single seed.

c) Write another short program to do the same thing instead using the rejection method, for the distribution

$$\text{pdf}(x) = \frac{2}{\pi} \sin^2(x), \tag{7}$$

again over $x \in [0, \pi]$. Include a similar plot, with 10^5 samples based on a single seed. (*Hint*: the previous question should be helpful in finding a good comparison function.) What is the ratio of time taken to produce the 10^5 samples in this question, to the time required in the previous question? What would you expect the ratio to be be, theoretically, and why?

Computational Physics Assignment: Guidelines & Submission

Write-up

As a guide, your written set of answers to the questions should not need to be more than about 2000 words long. We will not set a hard upper limit on the length for this assignment, but recognise that *if your write-up is unnecessarily long it will cost you some marks, as getting information across in a concise manner is important.*

Your answers to the questions should show all of your working out, a brief explanation of why you chose the methods that you did for solving the problem, and any noteworthy programming techniques you may have used (do not feel compelled to use any special techniques beyond those covered in the lectures, notes and/or Numerical Recipes, however). 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.

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 400 words on average 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 achieved
- understanding shown of numerical methods employed
- organisation, presentation and general quality of the write-up (including graphs), and
- quality of programming.

Code

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 that you present in your write-up—without the reader having to comment in or out sections of code or otherwise edit any files. You must include a `README` text file describing how to invoke your code to produce the submitted results. We also request that in your `README` file, you provide a short commentary on your choice of language and coding style, to help the reader appreciate your work—100 words should be sufficient for this. 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 even if that is the case, for the purposes of this assessed assignment, we require you to write your own comments in English to make it easier to follow and understand 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 2018–19" on Blackboard Learn no later than **12 noon** on **Monday 12th November (2018)**. Please submit your answers 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.