PX425 Assignment 1

Introduction

This task will help you establish if your level of programming skill in C is sufficient to complete the module without spending a disproportionate amount of time on the assessed work. It should not take more than two to three hours, hopefully less. If you struggle to complete it, or find yourself taking longer than three hours, you should brush up on your programming skills before continuing with this module.

You should by now have logged into the SCRTP server `godzilla` to confirm that your account is working correctly. This assignment will require you to be familiar with using the SCRTP desktop system. If you have not used the SCRTP system before then you are encouraged to read through the workshop script first, and assistance with logging on to `godzilla` and editing/compiling code will be available during the live event. There should be more than enough time to complete the assignment before the deadline, but if you have other commitments during this time I can extend the deadline case-by-case if required.

You can alternatively complete this assignment without participating in the live event, in which case the deadline is the same. If you haven't finished everything perfectly by the deadline, just submit whatever you have at that time. The assignment only constitutes 7.5 % of the final module mark, so do not over-stress about it, and remember a half-finished version is better than nothing.

However, please note that we will be strict in marking these assignments. If you do not follow the instructions carefully, e.g. pay attention to what the program does, how it is to be compiled and run, the exact output it should generate, and how you are to submit the work, then you are liable to drop marks.

Read *all* of the following instructions before writing any code. Be sure to follow the specification carefully.

Background

This task relates to wavefunctions of the Quantum Harmonic Oscillator, but do not worry if you are unfamiliar with these as the physics is not relevant here apart from as motivation for choosing specific functions to integrate. If we work in 'natural units' appropriate to a harmonic oscillator with angular frequency ω and mass m, the Hamiltonian is $H = -\frac{1}{2}\frac{d^2}{dx^2}$, which has solutions indexed by n where

$$\psi_n(x) = A_n \exp(-x^2/2) H_n(x) \tag{1}$$

with energy $E_n = n + \frac{1}{2}$, where $H_n(x)$ is the Hermite polynomial of order n and $A_n = \pi^{-1/4}/\sqrt{2^n n!}$ is a normalisation factor.

We are going to calculate by numerical integration the integral of $|\psi_3(x)|^2$ (for which $H_3(x) = 8x^3 - 12x$). We will omit its normalisation factor A_n, and compare the result to the formula for $1/|A_n|^2$. To make things more interesting we will integrate on a non-uniform grid! The i^{th} point of this grid is at $x_i = ai^2$, hence the spacings $h_i = x_{i+1} - x_i$ are variable, given by $h_i = a(2i+1)$. The integral you are finding is:

$$\int_{-\infty}^{\infty} |H_3(x)\exp(-x^2/2)|^2 \mathrm{d}x = 2\int_0^{\infty} |H_3(x)|^2 \exp(-x^2)\,. \tag{2}$$

As discussed in the Wikipedia page on Simpson's Rule, an appropriate quadrature rule here is the Composite Simpson's rule for irregularly spaced data. This should be implemented by coding up the following:

$$\int_a^b f(x)dx = \sum_{i=0}^{N/2-1} \frac{h_{2i}+h_{2i+1}}{6}\left[\left(2-\frac{h_{2i+1}}{h_{2i}}\right)f(x_{2i}) + \frac{(h_{2i}+h_{2i+1})^2}{h_{2i}h_{2i+1}}f(x_{2i+1}) + \left(2-\frac{h_{2i}}{h_{2i+1}}\right)f(x_{2i+2})\right] \tag{3}$$

where h_i are irregular intervals spanning the range $[a,b]$. The upper limit is ∞ but we do not need to sum to infinity, since the exponential term ensures the value of the integrand becomes small very quickly for large x. Additionally, the fact that the spacings get larger for large i should mean that you can accurately integrate the tail without particularly extreme grid sizes. You should find that $a = 0.00001$ and $i_{\max} = 1000$ are more than enough to generate agreement to the limit of floating point accuracy with the analytic result.

Code specification

All your code implementing this should be in one source file `qho.c`. Suitable text editors available on the SCRTP system include `gedit`, `geany`, `emacs`, and `vim`, or you could work remotely on Visual Studio Code or similar.

Your program should contain a function `double f(double x)`, which returns the integrand, a function `double x(int i, double a)` which returns the value of the i'th point x_i, and a function `double h(int i, double a)` which returns the spacing h_i. It is probably a good idea to write and test these first to check they work. On startup your code should read two values from the standard input (eg with `scanf`) which are the values of the spacing parameter a and the maximum number of intervals $i_{\max}$. You should sanity check these and return error messages if they contain values for which the code will not work correctly.

You should not show messages prompting the user to supply input values as these may interfere with automatic testing of your code. You can test the input by ensuring the code can be compiled and run using the following, which uses the `echo -e` command to send a particular string to the standard input (note that `\n` produces a newline (enter) in the input):

```
gcc -o qho qho.c -lm
echo -e "0.00001\n1000\n" | ./qho
```

You should start by implementing the functions described above for $f(x)$, $x(i)$ and $h(i)$ and then handle the inputs. Once you have that working, make a loop from 0 to $i_{\max}$ and implement the formula for the integral. Ideally, your code should exit its main loop early if the contribution to the integral has fallen below a tolerance set to ensure the final result is unaffected (and is staying there). At the end calculate and print $1/|A_3|^2$, and calculate and print the magnitude of the difference between this and your result.

If things are not working, I suggest debugging with `printf()` statements, but please remove any debug prints before submitting. Output at the end of the code should be printed to screen in a specific format, with the last two lines of output replicating the following lines (the results should be the 3rd entry on the line, and choices relating to whitespace and leading zeros are not significant):

```
Numerical integral:     ⊔⊔⊔⊔⊔.⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔
Analytic integral:      ⊔⊔⊔⊔⊔.⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔
Difference (error):     ⊔⊔⊔⊔⊔.⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔⊔
```

The symbol ⊔ represents a single numeric digit (or sign), with at least 15 after the decimal point. The tests run for grading purposes will use an appropriate level of precision suitable to check for correct implementation of the algorithm.

For sensible inputs, your program should run very quickly on `godzilla` or any other modern computer. The code should be well commented (a comment for each non-trivial block of code). If compiled with the `-Wall` and `-Wextra` options to `gcc`, no warnings should be produced. Marks will be lost if the code prints a final result which differs from the correct value by more than a calculated tolerance or is not formatted in a way that the grading script recognises, as specified above. You might like to investigate what are the most optimal values of a and $i_{\max}$ to obtain a correct result to a given precision.

Submission

You will need to upload the single source file for your code via the Moodle page for Assignment 1.

Submissions will be marked and feedback posted on Moodle before the Assignment 2 deadline.