Queen Mary
University of London

Main Examination period 2019/20

MTH6150: Numerical Computing in C and C++
Assignment date: Thursday 23/4/2020
Submission deadline: Monday 8/6/2020 at 23:55

The coursework is due by **Monday, 8th June 2020 at 23:55 BST**. Please submit a report (in `pdf` format) containing answers to **all** questions, complete with written explanations and printed tables or figures. Tables should contain a label for each column. Figures should contain a title, axis labels and legend. Please provide a brief explanation of any original algorithm used in the solution of the questions (if not discussed in lectures). Code comments should be used to briefly explain what your code is doing. You need to show that your program works using suitable examples. Build and run your code with any of the free IDEs discussed in class (such as Visual Studio, Xcode, CLion, or the online compiler `https://coliru.stacked-crooked.com/`). The code produced to answer each question should be submitted aside with the report, in a single `cpp` file, or multiple `cpp` files compressed into a single `zip` file. It is useful to organise the code in different directories, one for each question. The code for each question should also be copied into your report, so that it can be commented on when grading. This coursework should not take more than a total of 24 hours (assuming you have been engaging with lectures, labs and homework). **Only material submitted through QMPlus will be accepted. Late submissions will be treated in accordance with current College regulations. Plagiarism is an assessment offence and carries severe penalties. See the accompanying guidelines for additional information.**

Coursework [100 marks]

Question 1. [10 marks] *Self-consistent iteration.*
The geometry outside a spherical black hole can be expressed using a radial Schwarzschild coordinate $r \in [1, \infty)$ or a radial tortoise coordinate $x \in (-\infty, \infty)$. The transformation between the two coordinate systems, $x = x(r)$, is given by

$$x = r + \ln|r - 1|\,. \tag{1}$$

We seek the inverse transformation, $r = r(x)$. For a given coordinate x, we wish to obtain the coordinate r satisfying Eq. (1). Write a C++ function `rofx` that takes x as argument, solves Eq. (1) for r via self-consistent iteration, and returns the value of r, up to a specified tolerance ϵ. The iteration can be performed as a `for` or `while` loop that computes the sequence

$$r_{n+1} = x - \ln|r_n - 1| \quad \text{for} \quad n = 0, 1, 2, ... \tag{2}$$

and stops when $|r_{n+1} - r_n| < \epsilon$. The function `rofx` should take x as input and may use $r_0 = 0.8x$ as initial guess to perform the above iteration, in order to find $r_1, r_2, r_3, ...$ The value of x should be fixed throughout the iteration, but the value of r should be updated until it converges. The function `rofx` should return the converged value of r. You may use a tolerance parameter $\epsilon = 10^{-12}$. Run your code for a value of $x > 2$. In how many iterations did it converge? What is the final value of r? Substitute r into Eq. (1): is it satisfied? **[10]**

Question 2. [10 marks] *Inner products.*

(a) Write a function that takes as input two vectors $\vec{u} = \{u_0, u_1, ..., u_N\} \in \mathbb{R}^{N+1}$ and $\vec{v} = \{v_0, v_1, ..., v_N\} \in \mathbb{R}^{N+1}$ (of type `vector<double>`) and returns their inner product

$$\vec{u} \cdot \vec{v} = \sum_{i=0}^{N} u_i v_i \tag{3}$$

as a `double` number. Demonstrate that your program works by computing the inner product of two real Euclidean 3-vectors, `u={2,7,2}` and `v={3,1,4}`. [**5**]

(b) Write code for a function object that has a member variable m of type `int`, a suitable constructor, and a member function of the form
`double operator()(const vector<double> u, const vector<double> v) const {`
which returns the weighted norm

$$l_m(\vec{u}, \vec{v}) = \sqrt[m]{\sum_{i=0}^{N} |u_i v_i|^{m/2}} \tag{4}$$

Use this function object to calculate the norms $l_1(\vec{u}, \vec{v})$ and $l_2(\vec{u}, \vec{v})$ for the 3-vectors in Question 2a. Does the quantity $l_2(\vec{u}, \vec{v})^2$ equal the inner product $\vec{u} \cdot \vec{v}$ that you obtained above? [**5**]

Question 3. [15 marks] *Finite differences*.

(a) Write a C++ program that uses finite difference methods to numerically evaluate the first derivative of a function $f(x)$ whose values on a fixed grid of points are specified $f(x_i),\ i = 0, 1, ..., N$. Your code should use three instances of a `vector<double>` to store the values of x_i, $f(x_i)$ and $f'(x_i)$. Assume the grid-points are located at $x_i = (2i - N)/N$ on the interval $[-1, 1]$ and use 2nd order finite differencing to compute an approximation for $f'(x_i)$:

$$\begin{aligned}
f'(x_0) &= \frac{-3f(x_0) + 4f(x_1) - f(x_2)}{2\Delta x} + O(\Delta x^2) \quad \text{for} \quad i = 0 \\
f'(x_i) &= \frac{f(x_{i+1}) - f(x_{i-1})}{2\Delta x} + O(\Delta x^2) \quad \text{for} \quad i = 1, 2, ..., N-1 \\
f'(x_N) &= \frac{f(x_{N-2}) - 4f(x_{N-1}) + 3f(x_N)}{2\Delta x} + O(\Delta x^2) \quad \text{for} \quad i = N
\end{aligned}$$

with $\Delta x = 2/N$. Demonstrate that your program works by evaluating the derivatives of a known function, $f(x) = e^{-x^2}$, with $N + 1 = 16$ points. Compute the difference between your numerical derivatives and the known analytical ones:

$$e_i = f'_{\text{numerical}}(x_i) - f'_{\text{analytical}}(x_i)$$

at each grid-point. Output the values e_i of this `vector<double>` on the screen and tabulate (or plot) them in your report. [**8**]

(b) For the same choice of $f(x)$, demonstrate 2nd-order convergence, by showing that, as N increases, the mean error $\langle e \rangle$ or the root mean square error $\langle e^2 \rangle^{1/2}$ decrease proportionally to $\Delta x^2 \propto N^{-2}$. You may do so by tabulating (or log-log plotting) the quantity $N^2 \langle e \rangle$ (or $N^2 \langle e^2 \rangle^{1/2}$) for different values of N (e.g. $N+1 = 16,\ 32,\ 64,\ 128$). Is the quantity $N^2 \langle e \rangle$ (or $N^2 \langle e^2 \rangle^{1/2}$) approximately constant?
Here, the mean error $\langle e \rangle$ is defined by

$$\langle e \rangle = \frac{1}{N+1} \sum_{i=0}^{N} |e_i|.$$

Alternatively, you may use the rms (root mean square) error

$$\langle e^2 \rangle^{1/2} = \sqrt{\frac{1}{N+1} \sum_{i=0}^{N} |e_i|^2}.$$

Hint: Given the vector $\vec{e}$, you may use your C++ implementation of Eq. (4) from Question 2 to compute $\langle e \rangle = l_1(\vec{e}, \vec{e})/(N+1)$ or $\langle e^2 \rangle^{1/2} = l_2(\vec{e}, \vec{e})/\sqrt{N+1}$.
The l_1 or l_2 error norms are given by Eq. (4) $\vec{u} = \vec{v} = \vec{e}$ for $m = 1$ or $m = 2$ respectively. **[7]**

Question 4. [20 marks] *Stellar equilibrium.*
Consider the Lane-Emden equation, in the form of the initial value problem

$$\begin{cases} h''(x) + \frac{2}{x} h'(x) + h(x) = 0 \\ h(0) = 1,\ h'(0) = 0 \end{cases}$$

This equation appears singular at $x = 0$, but one can use de l'Hôpital's rule or a Taylor expansion of $h(x)$ about $x = 0$ to show that $h''(0) = -1/3$. Setting $z(x) = h'(x)$, the above 2nd-order differential equation can be reduced to a system of two 1st-order differential equations for $h(x)$ and $z(x)$:

$$\begin{cases} z'(x) = \begin{cases} -\frac{1}{3} & x = 0 \\ -\frac{2}{x} z(x) - h(x) & x > 0 \end{cases} \\ h'(x) = z(x) \\ h(0) = 1,\ z(0) = 0 \end{cases}$$

(a) Solve the above 1st-order system numerically, with a 4th-order Runge-Kutta method, using $N + 1 = 101$ equidistant points in $x \in [0, \pi]$. Your code should use three instances of a `vector<double>` to store the values of x_i, $h(x_i)$, $z(x_i)$. Output the values $x_0, x_{10}, x_{20}, ..., x_{100}$ and $h(x_0), h(x_{10}), h(x_{20}), ..., h(x_{100})$ to the screen and tabulate them in your report. **[10]**

(b) Compute the difference $e(x) = h_{\text{numerical}}(x) - h_{\text{exact}}(x)$ between your numerical solution $h_{\text{numerical}}(x)$ and the exact solution $h_{\text{exact}}(x) = \frac{1}{x} \sin x$. Output the error values $e(x_0), e(x_{10}), e(x_{20}), ..., e(x_{100})$ to the screen and tabulate them in your report. **[5]**

(c) Compute the error norms:

$$l_1(\vec{e},\vec{e}) = \sum_{i=0}^{N} |e_i|, \quad l_2(\vec{e},\vec{e}) = \sqrt{\sum_{i=0}^{N} |e_i|^2}$$

where $e_i = e(x_i)$ is the error at each grid-point. [Hint: you may use your C++ implementation of Eq. (4) from Question 2b to compute the norms.] **[5]**

Question 5. [30 marks] *Numerical integration.*
We wish to compute the definite integral

$$I = \int_{-1}^{1} \frac{1}{1+25x^2}\, dx$$

numerically and compare to the exact result, $I_{\text{exact}} = \frac{2}{5}\tan^{-1}(5)$.

(a) Use the composite trapezium rule

$$\int_a^b f(x)dx \simeq \sum_{i=0}^{N} w_i f_i, \quad w_i = \begin{cases} \Delta x/2, & i = 0 \text{ or } i = N \\ \Delta x & 1 \le i \le N-1 \end{cases}, \quad \Delta x = \frac{b-a}{N},$$

to compute the integral I, using $N+1 = 64$ equidistant points in $x \in [-1, 1]$. Use three instances of a `vector<double>` to store the values of the gridpoints x_i, function values $f_i = f(x_i)$ and weights w_i. [Hint: you may use the function from Question 2a to compute the dot product of the `vectors` w_i and f_i.] Output to the screen (and list in your report) your numerical result $I_{\text{trapezium}}$ and the difference $I_{\text{trapezium}} - I_{\text{exact}}$. **[5]**

(b) Use the composite Hermite integration rule

$$\int_a^b f(x)dx \simeq \sum_{i=0}^{N} w_i f_i + \frac{\Delta x^2}{12}[f'(a) - f'(b)]$$

with the derivatives $f'(x)$ at $x = a$ and $x = b$ evaluated analytically (and the weights w_i identical to those given above for the trapezium rule), to compute the integral I, using $N+1 = 64$ equidistant points in $x \in [-1, 1]$. Output to the screen (and list in your report) your numerical result I_{Hermite} and the difference $I_{\text{Hermite}} - I_{\text{exact}}$. **[5]**

(c) Use the Clenshaw-Curtis quadrature rule

$$\int_a^b f(x)dx \simeq \sum_{i=0}^{N} w_i f_i, \quad w_i = \begin{cases} \frac{1}{N^2}, & i = 0 \text{ or } i = N \\ \frac{2}{N}\left(1 - \sum_{k=1}^{(N-1)/2} \frac{2\cos(2k\theta_i)}{4k^2-1}\right) & 1 \le i \le N-1 \end{cases},$$

on a grid of $N+1 = 64$ points $x_i = -\cos\theta_i$, where $\theta_i = i\pi/N$, $i = 0, 1, ..., N$ to compute the integral I. [Hint: First compute the values of θ_i, x_i, $f_i = f(x_i)$ and w_i and store them as `vectors`. Then, you may use the function from Question 2a to compute the dot product of the `vectors` w_i and f_i.] Output to the screen (and list in your report) your numerical result $I_{\text{ClenshawCurtis}}$ and the difference $I_{\text{ClenshawCurtis}} - I_{\text{exact}}$. **[10]**

(d) Compute the integral I using a hit and miss Monte Carlo method with $N = 10000$ samples. Output to the screen (and list in your report) your numerical result $I_{\text{MonteCarlo}}$ and the difference $I_{\text{MonteCarlo}} - I_{\text{exact}}$. **[10]**

Question 6. [15 marks] *Harmonic oscillator.*

Consider the first-order system:

$$\begin{cases} \dfrac{dq}{dt} = p \\ \dfrac{dp}{dt} = -q \end{cases}$$

(a) Use a 2nd-order Runge-Kutta (RK2) midpoint method to evolve the system, with initial conditions $q(0) = 0, p(0) = \sqrt{2}$ and time-step $\Delta t = 0.1$. Stop the evolution at time $t = 100$. Describe how your code works. **[8]**

(b) Output to the screen (and tabulate in your report) the values of the position $q(t)$, momentum $p(t)$, energy $E(t) = \frac{1}{2}(p^2 + q^2)$ and the difference $e(t) = E(t) - E(0)$ for $t = 0$, $t = 1$, $t = 10$ and $t = 100$. Is $E(t)$ constant numerically? **[7]**

End of Paper.