Main Examination period 2021/22

MTH739U / MTH739P: Topics in Scientific Computing 2022/23 – Coursework 2

Assignment date: Monday 24/11/2022
Submission deadline: Monday 23/01/2023 at 23:55 BST

The coursework is due by **Monday 23rd January 2023 at 23:55 GMT**. Please submit a **report in pdf format** containing answers to all questions, complete with written explanations and printed figures. Figures muct contain a title, axis labels, and a legend (if more than one curve are shown in the same figure). Please provide an explanation of any original algorithms used for solving the problems (except for material explicitly discussed in lectures, e.g. the Runge-Kutta method). You normally must show that your program works using suitable examples. If using Wolfram language, the report and code can be submitted in a single Mathematica notebook (as a single .nb file accompanied by a .pdf copy of that notebook). If using another language, the report must be submitted as a pdf file, and the code produced to answer each question must be copied into the pdf report document and explained; the code must also be submitted in a separate file that can be compiled and executed. (It might be useful to organise the code in different files or folders, one for each question.) **Reports that do not include code or, conversely, code not accompanied by a report, cannot be marked. Only material submitted through QMPlus can be accepted.** Late submissions will be treated in accordance with College regulations. If you use online or other resources, please cite them appropriately. **QMPlus automatically screens for plagiarism. Plagiarism is an assessment offence and carries severe penalties.**

Part A: Coursework [55 marks]

Question 1. [10 marks]

(a) Write a program that numerically evaluates the first and second derivatives of a function $f(x)$ whose values on a fixed grid of points are specified $f(x_\imath),\ \imath = 0, 1, ..., N$, using the pseudo-spectral method described below. Assume that the points $x_\imath$ are located at the Chebyshev-Gauss-Lobatto nodes

$$x_\imath = \frac{b+a}{2} + \frac{b-a}{2} z_\imath, \quad z_\imath = \sin\left(\frac{2\imath - N}{2N}\pi\right), \quad \imath = 0, 1, \ldots, N \tag{1}$$

in the interval $x \in [a, b]$. These points are the extrema of the Chebyshev polynomial of the first kind $T_N(x)$. Evaluate the derivatives using the approximations

$$f'(x_\imath) \simeq \sum_{\jmath=0}^{N} D^{(1)}_{\imath\jmath} f(x_\jmath), \quad f''(x_\imath) \simeq \sum_{\jmath=0}^{N} D^{(2)}_{\imath\jmath} f(x_\jmath) \tag{2}$$

where $D^{(1)}_{\imath\jmath}$ are the elements of the $(N+1)\times(N+1)$ derivative matrices, given by

$$D^{(1)}_{\imath\jmath} = \frac{2}{b-a}\begin{cases} \frac{c_\imath(-1)^{\imath+\jmath}}{c_\jmath(z_\imath - z_\jmath)} & \imath \neq \jmath \\ -\frac{z_\jmath}{2(1-z_\jmath^2)} & \imath = \jmath \neq 0, N \\ -\frac{2N^2+1}{6} & \imath = \jmath = 0 \\ \frac{2N^2+1}{6} & \imath = \jmath = N \end{cases} \tag{3}$$

where $c_0 = c_N = 2$ and $c_1, \ldots, c_{N-1} = 1$. The second derivative matrix can be evaluated by $\mathbf{D}^{(2)} = (\mathbf{D}^{(1)})^2$ or, with lower round-off error, by

$$D^{(2)}_{\imath\jmath} = \left(\frac{2}{b-a}\right)^2 \begin{cases} \frac{(-1)^{\imath+\jmath}}{c_\jmath}\frac{z_\imath^2 + z_\imath z_\jmath - 2}{(1-z_\imath^2)(z_\imath - z_j)^2} & \imath \neq \jmath, \quad \imath \neq 0, \quad \imath \neq N \\ \frac{2}{3}\frac{(-1)^\jmath}{c_j}\frac{(2N^2+1)(1+z_j)-6}{(1+z_j)^2} & i \neq j, \quad \imath = 0 \\ \frac{2}{3}\frac{(-1)^{\jmath+N}}{c_\jmath}\frac{(2N^2+1)(1-z_\jmath)-6}{(1-z_\jmath)^2} & \imath \neq \jmath, \quad \imath = N \\ -\frac{(N^2-1)(1-z_\jmath^2)+3}{3(1-z_\jmath^2)^2} & \imath = \jmath, \quad \imath \neq 0, \quad \imath \neq N \\ \frac{N^4-1}{15} & \imath = \jmath = 0 \quad \text{or} \quad N \end{cases} \tag{4}$$

The Chebyshev differentiation matrices can be constructed automatically (with slightly higher error) via the Wolfram Language commands:

```
D1=NDSolve‘FiniteDifferenceDerivative[Derivative[1],X,
"DifferenceOrder"->"Pseudospectral"]@"DifferentiationMatrix"

D2=NDSolve‘FiniteDifferenceDerivative[Derivative[2],X,
"DifferenceOrder"->"Pseudospectral"]@"DifferentiationMatrix"
```

which respectively return the matrices $\mathbf{D}^{(1)}, \mathbf{D}^{(2)}$ for a list X of nodes given by Eq. (1). **[3]**

(b) Demonstrate that your program works by evaluating the first and second derivatives of a known function, e^{-16x^2}. Graph or tabulate the difference between the numerical and analytical derivatives, $e_i = f'_{\text{numerical}}(x_i) - f'_{\text{analytical}}(x_i)$ and $r_i = f''_{\text{numerical}}(x_i) - f''_{\text{analytical}}(x_i)$, as a function of x_i. **[3]**

(c) Show that the difference (measured with an error norm such as l_1) between your numerical derivatives and the known analytical ones approaches zero as e^{-N}. You might do this by tabulating or graphing $e^N l_1 \sim$ constant or $\ln l_1 \sim$ constant $- N$ for several different values of N, where $l_1 = \sum_{i=0}^{N} |e_i|$ for the first derivative error norm, and similarly for the second derivative. **[4]**

Remark: For stability and accuracy purposes, it is often beneficial to replace each element D_{ii} of the main diagonal with the negative sum of all matrix elements in the same row, $D_{ii} = -\sum_{j\neq i} D_{ij}$. This ensures that the differentiation matrix returns zero when multiplying a constant vector.

Question 2. [10 marks] *Implicit and explicit integration of ODEs.*
Consider the 2D anisotropic harmonic oscillator, described by the first order system:

$$\begin{cases} \dfrac{dx}{dt} = p_x \\ \dfrac{dy}{dt} = p_y \\ \dfrac{dp_x}{dt} = -\dfrac{\partial V}{\partial x} \\ \dfrac{dp_y}{dt} = -\dfrac{\partial V}{\partial y} \end{cases}$$

where $V(x,y) = \sin x + \cos y$ is the potential.

(a) Write the system in the vector form $d\vec{u}/dt = \vec{f}(t,\vec{u})$, where $\vec{u} := \{x, y, p_x, p_y\}$. What is the vector-valued function $\vec{f}(t,\vec{u})$? Define this function in Wolfram Language (or a language of your choice). **[1]**

(b) Use a 2nd order Runge-Kutta (RK2) method that increments the vector $\vec{u}$ in each time step to evolve the system. Use $x(0) = 0, y(0) = 0, p_x(0) = 1, p_y(0) = 1$ as initial conditions and $\Delta t = 0.1$ as the time-step. Stop the evolution at time $t = 100$. Describe how your code works. Plot the position components $x(t), y(t)$ as functions of time t. Plot the energy $E(t) = \frac{1}{2}(p_x^2 + p_y^2) + V(x,y)$ and the difference $\delta E(t) = E(t) - E(0)$ as functions of time. Does the RK2 method conserve energy? Why or why not? **[4]**

(c) Repeat (b) using the trapezium rule

$$\vec{u}_{n+1} - \vec{u}_n = \int_{t_n}^{t_{n+1}} \vec{f}(t,\vec{u}(t))dt \simeq \frac{\Delta t}{2}(\vec{f}_{n+1} + \vec{f}_n) \tag{5}$$

to evolve the system, where $\vec{u}_n = \vec{u}(t_n), \vec{f}_n = \vec{f}(t_n,\vec{u}_n)$, and $\Delta t = t_{n+1} - t_n$ is the (constant) time step. Does the trapezium rule conserve energy? Why or why not?

Remark: this method is implicit, because t_{n+1} appears on both sides of the evolution equations. To obtain the future values of the momentum and position, one may use a `Do` loop at each time step (i.e. a nested `Do` loop). The loop at each time step can use a RK1 or RK2 step as initial guess, and then iterate the above system of equations 10 times, repeatedly substituting the last value $\vec{u}_{n+1}$ into the function $\vec{f}_{n+1} = \vec{f}(t_{n+1},\vec{u}_{n+1})$ on right-hand side, to compute the new value of the left-hand side. That is, at each time step, one can perform the self-consistent iteration

$$\vec{u}_{n+1}^{\,\text{new}} = \vec{u}_n + \frac{\Delta t}{2}[\vec{f}(t_{n+1},\vec{u}_{n+1}^{\,\text{old}}) + \vec{f}(t_n,\vec{u}_n)] \tag{6}$$

and stop after 10 iterations. **[5]**

Question 3. [15 marks] *Explicit and implicit integration of PDEs.*

(a) Consider the hyperboloidal advection equation

$$\partial_t u + \frac{x^2}{1+x}\,\partial_x u = 0 \tag{7}$$

for the scalar function $u(t,x)$, with initial data $u(0,x) = f(x)$ for some chosen function $f(x)$, in the domain $x \in [0,1]$. You may assume that $f(x) = e^{-\alpha[\beta - \ln(1-x)]^2}$, with $\alpha = 220$ and $\beta = -2/3$. Show that $u_{\text{analytical}}(t,x) = e^{-\alpha[\beta - t - \ln(1-x)]^2}$ satisfies Eq. (7). (You may use computer algebra to show this analytically). [**1**]

(b) Write a program that uses the method of lines to evolve Eq. (7). Use a pseudo-spectral method for spatial discretization (for a Chebyshev collocation method, the 1st order differentiation matrix is given in Question 1). Use an explicit Runge-Kutta method (or equivalent Taylor method) for time integration. Explain how your programme works. Plot the solution $u_{\text{numerical}}(t,x)$ and the error $\delta u(t) = u_{\text{numerical}}(t,x) - u_{\text{analytical}}(t,x)$ as a function of x at different instants of time t. What happens if the time-step exceeds the Courant limit? [**5**]

(c) Plot the energy error $\delta E(t) = E(t) - E(0)$ as a function of time t, where

$$E = \int_0^1 dx\, \frac{x^2}{1+x} (\partial_x u)^2. \tag{8}$$

is the energy. Is energy conserved numerically? Why?
Hint: if using a Chebyshev collocation method with the grid-points of Question 1, you may use the Clenshaw-Curtis quadrature rule from Coursework 1 to calculate the energy integral.

Hint: stop evaluating the energy before the wave reaches the boundaries, otherwise you will observe energy loss as the pulse exits the domain. [**4**]

(d) Repeat (b) and (c) using an implicit method (such as the trapezium or Hermite rule) for time integration. What happens if you increase the time-step? Is energy conserved numerically? Why? [**5**]

Hint: the trapezium rule applied to the discretized advection equation should yield a system of the form:

$$\vec{u}_{n+1} - \vec{u}_n \;=\; \int_{t_n}^{t_{n+1}} \vec{f}(t, \vec{u}(t))dt \simeq \frac{\Delta t}{2}(\vec{f}_{n+1} + \vec{f}_n) \tag{9}$$

where the vectors $\vec{u}_{n+1}$ and $\vec{f}_{n+1}$ represent $u_i^{n+1} = u(t_{n+1}, x_i)$ and $f_i^{n+1} = f(t_{n+1}, u_i^{n+1})$ on the gridpoints $\{x_i\}$. This system is implicit, as t_{n+1} appears on both sides of the equations. One may use self-consistent iteration to solve this system in each time-step, as done in Eq. (6) earlier. Alternatively, because the advection equation is linear, Eq. (9) is linear in the unkown quantities $\vec{u}_{n+1}$. Thus, you may analytically solve this system (or use a Padé method) to express $\vec{u}_{n+1}$ in terms of the known quantities $\vec{u}_n$.

Question 4. [10 marks] *Generating random numbers*.
For the probability distributions detailed below, construct functions to obtain random numbers from these distributions, and test these functions by creating histograms from a sufficient number of samples and plotting the histograms together with the corresponding distribution functions. You are allowed to use any *uniform* number generator. Use of off-the-shelf generators for non-uniform distributions will lead to half marks.

(a) Uniform distribution over the interval $[-2\pi, \pi]$. **[2]**

(b) Uniform distribution over the union of the three intervals $[1,2] \cup [3,4] \cup [5,6]$. **[2]**

(c) Gaussian distribution with a given mean value μ and variance σ^2. Use the inverse CDF method, or the Box-Muller method, or the Marsaglia polar method, or another method of your choice, and test your function by sampling from a Gaussian with $\mu = 12$ and $\sigma^2 = 3$. [Half marks awarded if you use an off-the-shelf function like the Mathematica built-in function $\mathrm{NormalDistribution}[\mu, \sigma]$.] **[2]**

(d) Continuous distribution with probability density function:

$$f(x; \lambda) = \lambda e^{-\lambda x}$$

for $x \geq 0$, where λ is a parameter provided by the user. Test your function by sampling from the three distributions obtained by setting λ respectively equal to 0.7, 1.5, and 3.5. **[2]**

(e) Continuous distribution with cumulative density function:

$$F(x) = \frac{1}{6}\left(x^2 + x\right)$$

for $x \in [0, 2]$. **[2]**

Question 5. [10 marks] *Simple sampling*.
The Monte-Carlo estimate of an integral is a random variable, whose mean value approaches the real value of the integral as the number of samples tends to infinity. Write a module that implements a mean-value Monte-Carlo estimate of the integral

$$I = \int_0^1 x^3(1-x)^{1/2}\mathrm{d}x = \frac{32}{315}$$

which is the value of the Euler Beta function:

$$B(a,b) = \int_0^1 x^{a-1}(1-x)^{b-1}\mathrm{d}x$$

for $a = 4$ and $b = 3/2$.

(a) Run the code for $N = 10^n$, $n = 1, 2, 3, ...$ uniform random numbers in $[0, 1]$, and obtain the sequence of Monte-Carlo estimates $I_1, I_2, I_3, \ldots$. **[3]**

(b) Compute the sequence of absolute errors $E_n = |I - I_n|$ and show that the estimate I_n of I improves when the number N of samples increases. **[3]**

(c) Compute and plot the variance σ^2 of the Monte-Carlo estimates as a function of n. How does the variance change when your estimate is based on more samples? **[4]**

Part B: Programming Project and Report [45 marks]

Answer ***one*** of the following questions.

Question 6. [45 marks] *Hyperboloidal wave equation.*

(a) Consider the hyperboloidal wave equation

$$-(1+x)\partial_t^2\psi + (1-2x^2)\partial_x\partial_t\psi + (1-x)x^2\partial_x^2\psi - 2x\,\partial_t\psi + x(2-3x)\partial_x\psi = 0, \quad (10)$$

for the scalar function $\psi(t,x)$, with initial data $\psi(0,x) = f(x)$, $\dot\psi(0,x) = -\frac{x^2}{1+x}f'(x)$ for some chosen function $f(x)$, in the domain $x \in [0,1]$. You may assume that $f(x) = e^{-\alpha[\beta - \ln(1-x)]^2}$, with $\alpha = 220$ and $\beta = -2/3$. Show that $\psi_{\text{analytical}}(t,x) = e^{-\alpha[\beta - t - \ln(1-x)]^2}$ satisfies Eq. (11). (You may use computer algebra to show this analytically). The above wave equation is second order in space and time. Define an auxiliary variable $\pi = \partial_t\psi + b(x)\partial_x\psi$ and use it to reduce the wave equation to a system fully *first order in time and space*, of the form $\partial_t u = \hat{L}\,u$, where $u = \begin{pmatrix}\psi\\ \pi\end{pmatrix}$ and $\hat{L} = \begin{pmatrix}\hat{L}_1 & \hat{L}_2\\ \hat{L}_3 & \hat{L}_4\end{pmatrix}$ is a matrix of spatial differentiation operators. What are the values of $b(x)$ that eliminate $\partial_x^2\psi$ terms from the equation? What are the operators $\hat{L}_1, \ldots, \hat{L}_4$? [**2**]

(b) Write a program that uses the method of lines to evolve the first order reduced wave equation. Use a pseudo-spectral method for spatial discretization (for a Chebyshev collocation method, the differentiation matrices are given in Question 1). Use an explicit Runge-Kutta method (or equivalent Taylor method) for time integration. Explain how your programme works. Plot the solution $u_{\text{numerical}}(t,x)$ and the error $\delta u(t) = u_{\text{numerical}}(t,x) - u_{\text{analytical}}(t,x)$ as a function of x at different instants of time t. What happens if the time-step exceeds the Courant limit? [**10**]

(c) Plot the energy error $\delta E(t) = E(t) - E(0)$ as a function of time t, where

$$E = \int_0^1 dx\left[(1+x)(\partial_t\psi)^2 + x^2(1-x)(\partial_x\psi)^2\right]. \quad (11)$$

is the energy. Is energy conserved numerically? Why?
Hint: if using a Chebyshev collocation method with the grid-points of Question 1, you may use the Clenshaw-Curtis quadrature rule from Coursework 1 to calculate the energy integral.

Hint: stop evaluating the energy before the wave reaches the boundaries, otherwise you will observe energy loss as the pulse exits the domain. [**8**]

(d) Repeat (b) and (c) using an implicit method (such as the trapezium or Hermite rule) for time integration. What happens if you increase the time-step? Is energy conserved numerically? Why? [**10**]

Hint: the trapezium rule applied to the discretized advection equation should yield a system of the form:

$$\vec{u}_{n+1} - \vec{u}_n \quad = \quad \int_{t_n}^{t_{n+1}} \vec{f}(t, \vec{u}(t))dt \simeq \frac{\Delta t}{2}(\vec{f}_{n+1} + \vec{f}_n) \qquad (12)$$

where the vectors $\vec{u}_{n+1}$ and $\vec{f}_{n+1}$ represent $u_i^{n+1} = u(t_{n+1}, x_i)$ and $f_i^{n+1} = f(t_{n+1}, u_i^{n+1})$ on the gridpoints $\{x_i\}$. This system is implicit, as t_{n+1} appears on both sides of the equations. One may use self-consistent iteration to solve this system in each time-step, as done in Eqs. (6) and (9) earlier. Alternatively, because the wave equation is linear, Eq. (13) is linear in the unkown quantities $\vec{u}_{n+1}$. Thus, you may analytically solve this system (or use a Padé method) to express $\vec{u}_{n+1}$ in terms of the known quantities $\vec{u}_n$.

(e) Write a report describing the two different time evolution methods and details of your implementation. Compute the Courant limit for the method in parts (b) and (d). What happens if the time step is too large? Reflect on the relative strengths and weaknesses of these methods. Back up your conclusions with evidence from specific data such as number of time-steps needed and total run times. [**15**]

Question 7. [45 marks] *Black-Scholes equation.*
Consider the Black-Scholes equation

$$\frac{\partial c(t,s)}{\partial t} + \frac{1}{2}s^2\sigma^2\frac{\partial^2 c(t,s)}{\partial s^2} + rs\frac{\partial c(t,s)}{\partial s} - rc(t,s) = 0$$

with terminal and boundary conditions

$$\begin{aligned} c(T,s) &= \max(s-k,0) \\ c(t,0) &= 0 \end{aligned}$$

where c is the price of the option as a function of stock price $s > 0$ and time $t \in [0,T]$, r is the risk-free interest rate, σ is the volatility of the stock and k is the strike price of the call. In their Nobel-prize winning paper, Black and Scholes noted that, by a transformation of variables $c = ue^{rt}$, $t = T - \tau$, $s = e^{x-(r-\sigma/2)\tau}$, their equation can be reduced to a linear diffusion equation

$$\frac{\partial u(\tau,x)}{\partial \tau} = \frac{1}{2}\sigma^2\frac{\partial^2 u(\tau,x)}{\partial x^2}$$

with initial condition $u(0,x) = e^{-rT}\max(0,e^x - k)$ and boundary condition $u(\tau,x) \approx 0$ for $x \to \infty$. [**Hint**: you may use the Wolfram Language command given in Q7a below to check and implement exact boundary conditions if needed].

For this programming project, you will solve the Black-Scholes equation numerically and determine the price of an option.

(a) Use finite differencing and an explicit time integration method (such as Runge-Kutta) to solve the Black-Scholes equation *or* the diffusion equation via the numerical method of lines. Plot the solution $c(t,s)$ as a function of $s \in [10,50]$ for $k = 50, \sigma = 0.1, T = 1, r = 0.04$ for different times $t \in [0,T]$. Evaluate the solution for $t = 0, s = 50$. How does your result compare to that returned via the following Wolfram Language command?

```
FinancialDerivative[{"European", "Call"},
{"StrikePrice" -> 50.00, "Expiration" -> 1},
{"InterestRate" -> 0.04, "Volatility" -> 0.1,
"CurrentPrice" -> 50}]
```
[15]

(b) Repeat (a) using an implicit time integration method, such as the trapezium or Hermite rule. **[15]**

(c) Write a report describing the two different time evolution methods and details of your implementation. Compute the Courant limit for the method in part (a) and (b). What happens if the time step is too large? Reflect on the relative strengths and weaknesses of these methods. Back up your conclusions with evidence from specific data such as number of time-steps needed and total run times. **[15]**

End of Paper.