Imperial College London

DEPARTMENT OF MATHEMATICS
IMPERIAL COLLEGE LONDON
Academic Year 2023-2024

Partial Differential Equations in Action **MATH50008**

Problem Sheet 7

This problem sheet is not like other problem sheets. You can think about it as a learning companion to the short introduction to numerical integration of PDEs found in the lecture notes (Chapter 4). This problem sheet has been designed as a mini-project which will lead you to implement most of the numerical techniques which we have introduced in the accompanying lectures.
Throughout this problem sheet, you will be implementing numerical methods. You can obviously implement them in whichever programming language you feel most comfortable with but know that the solutions will be provided in Python.

1. ✦ Consider the function $u(x) = \sin x$, $0 \le x \le 2\pi$. In this problem, we wish to approximate its first and second derivatives at the nodes $x_i = i\Delta x$ with $\Delta x = 2\pi/N$, where N is the number of discretization points.

   (a) Approximate these derivatives using (i) forward differences, (ii) backward difference, (iii) second-order central differences and (iv) fourth-order central difference. You may want to do this by defining differentiation matrices, i.e. matrices D such that $D\mathbf{u}$ (where $\mathbf{u}$ is a vector containing the values that $u(x)$ takes at the grid points) gives you the appropriate approximate derivative.

   (b) In each case, you should produce a log-log plot of the error $E(\Delta x)$ made in approximating the first-order derivative. Make sure that you use enough values of Δx to get a good estimate of the slope which should be given by the order of accuracy of the differentiation scheme. For instance, you may want to use increasing powers of 2 as the number of grid points.

   [Hint: note that when your grid points are close to the boundary points $x = 0$ and $x = \pi$, the formula for the derivative may take you outside of the domain. To solve this problem, you may use the fact that the function $\sin(x)$ is 2π-periodic and so $\sin(-\Delta x) = \sin(2\pi - \Delta x)$. Alternatively, you could start and end at points that do not take you outside the domain (This is not the preferred method).]

2. ✦ In this problem, we will learn how to derive higher-order finite difference schemes. Here, we want to derive the fourth-order central difference scheme for first-order derivatives (given in the lecture notes). To do so, write down Taylor expansions for $u(x \pm \Delta x)$ and $u(x \pm 2\Delta x)$; this will give you 4 expressions. By considering a linear combination of the form

$$\alpha u(x + 2\Delta x) + \beta u(x + \Delta x) + \delta u(x - \Delta x) + \gamma u(x - 2\Delta x)$$

   write down the system of linear equations for $(\alpha, \beta, \delta, \gamma)$ that one needs to satisfy to have

$$\alpha u(x + 2\Delta x) + \beta u(x + \Delta x) + \delta u(x - \Delta x) + \gamma u(x - 2\Delta x) = u'(x) + \mathcal{O}\left((\Delta x)^4\right)$$

   Solve this system of equations to obtain the fourth-order central difference scheme for first-order derivatives.

3. ✦✦ Now, let's use finite differences to solve a diffusion problem. Here, we will consider the following 1D Dirichlet problem

$$\begin{aligned} \frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2}, \quad 0 < x < 1, t > 0 \\ u(0,t) &= u(1,t) = 0 \\ u(x,t) &= 4x(1-x) \end{aligned}$$

(a) What is the steady-state solution of this problem?

(b) Implement a FTCS scheme to solve this problem over time. To do so, you will write a the FTCS scheme with Dirichlet boundary conditions in matrix form; note that the `diag()` function from the `numpy` module in Python or the `diag()` function in MATLAB may prove useful in writing your numerical scheme in matrix form. After having initialized your solution using the initial conditions provided, you may advance your solution in time using a `for` or a `while` loop. In the first instance, you will solve your problem using $\Delta x = 0.01$ over a time interval $[0, T]$ where $T = 0.3$. How should you pick the value of the time step Δt? Your code should store the value of your solution over time.

(c) Represent the approximate solution as a surface in the domain $[0, 1] \times [0, T]$, with $T = 0.3$. (You may want to use the `surf()` or `contourf()` functions in MATLAB or their equivalent in the `matplotlib` module in Python).

(d) Check that your solution does indeed converge to the expected steady-state solution at long times.

4. ✦✦ In this question, we will implement the BTCS scheme for the 1D diffusion equation.

(a) Following what we did in the lecture notes, devise a strategy to implement Dirichlet boundary conditions in a BTCS scheme.

(b) Modify the program you wrote in Q3 to now use a BTCS scheme instead of the FTCS you previously implemented. You will first solve the same problem as that of Q3. Using $\Delta x = 0.01$, use different values of Δt to confirm that your numerical scheme is unconditionally stable. Implementing this scheme will require you to invert a matrix, to do so you can use the `inv()` function which is part of the `numpy.linalg` module in Python or the `inv()` function in MATLAB.

(c) Represent the approximate solution as a surface in the domain $[0, 1] \times [0, T]$, with $T = 0.3$. Compare with the solution you obtained in Q3.

(d) You should now devise a strategy to impose Neumann boundary conditions in a BTCS scheme.

(e) Write a new program to solve the following 1D diffusion problem with Neumann boundary conditions

$$\begin{aligned}
&\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \quad -L < x < L, t > 0 \\
&\frac{\partial u}{\partial x}(-L, t) = \frac{\partial u}{\partial x}(L, t) = 0 \\
&u(x, t) = \exp(-x^2)
\end{aligned}$$

i.e. a problem in which the initial conditions are given by a Gaussian pulse and where we have imposed no-flux boundary conditions. You will use $L = 5$, $\Delta x = 0.1$, $T = 10$ and $\Delta t = 0.05$. Represent the approximate solution as a surface in the domain $[-5, 5] \times [0, 10]$.

5. ✦✦✦ **Bonus:** Solve the following Dirichlet problem

$$\begin{aligned}
&\frac{\partial u}{\partial t} = \sqrt{2}\frac{\partial^2 u}{\partial x^2}, \quad 0 < x < 1, t > 0 \\
&u(0, t) = 2, \ u(1, t) = 0.5 \\
&u(x, t) = 2 - 1.5x + \sin(\pi x)
\end{aligned}$$

using the Crank-Nicolson scheme. You will use reasonable values for the grid spacing Δx and time step Δt. First, represent the initial conditions. What is the steady-state solution that you expect? Confirm that your numerical solution converges to the expected steady-state solution by representing both the solution surface and the profile of u at long time. You may want to solve your problem on a short time interval $[0, T]$ first and progressively increase the value of T.

6. ✦✦✦ Reaction-diffusion systems are commonly used to model systems of diffusing chemical species which can interact with each other. So far, we have seen in lectures that diffusion tends to smooth fluctuations. However, in a system of diffusing and interacting chemical species, under the right conditions diffusion can surprisingly lead to the emergence of patterns (i.e. stable, time-independent, spatially heterogeneous solutions) in a system which would be in a stable uniform steady-state in the absence of diffusion! This is exemplified even in simple reaction-diffusion systems like the Gierer-Meinhardt model (1972) which has been used to explain morphogenesis and patterns in developmental biology. The Gierer-Meinhardt model is an activator-inhibitor system (you do not need to know what this means at this point) in which two chemical species with concentration u and v diffuse and interact according to the following system of PDEs

$$\frac{\partial u}{\partial t} = D_u \frac{\partial^2 u}{\partial x^2} + \frac{u^2}{v} - bu \tag{1}$$

$$\frac{\partial v}{\partial t} = D_v \frac{\partial^2 v}{\partial x^2} + u^2 - v \tag{2}$$

where D_u and D_v are the respective diffusivities and b is a free parameter of the model. We will solve this problem on the interval $[0, L]$. We will use periodic boundary conditions, i.e. that $u(0,t) = u(L,t)$.

(a) Based on your answer to Q4, try to implement a numerical scheme centered in space and implicit in time to solve this reaction-diffusion equation. What is the problem? What assumption can you make to solve it? Note that here we assume periodic boundary conditions.

(b) Solve the Gierer-Meinhardt model with $b = 0.35$, $D_u = 1$ and $D_v = 50$. You will use the following initial conditions $u(x,0) = 1/b + \eta(x)$ and $v(x,0) = 1/b^2$, where η is a noise with amplitude 0.01 (to generate this noise, you may want to use the `rand()` function in MATLAB or the `rand()` function from the `numpy.random` module in Python to generate uniformly distributed random numbers in the interval $[0, 0.01]$ for instance). You should use a domain size $L = 80$ and a number of grid points $N = 500$; you will use a time step $\Delta t = 1$. Show that a stable pattern of stripes forms by running this simulation until $T = 300$. This is for instance reminiscent of stripe patterns observed on certain seashells.

(c) Explore the Gierer-Meinhardt model by simulating it for different values of the free parameter b as well as different initial conditions.