Workshop 10: Solving the diffusion equation

In this workshop we will see two methods for solving the diffusion equation. Both methods use finite differences to calculate spatial derivatives but the time stepping is treated differently. The first method uses a forward Euler time step, which was the method used to time step the advection equation in workshop 3. The second method uses an implicit time step which requires solving a system of linear equations. The linear system can be represented as a matrix equation and we can use routines from the Lapack library to solve this linear system.

The programs in this workshop are not resource intensive so the workshop has been designed to be carried out on the Lovelace Lab PCs.

1 Solving the diffusion equation

This section outlines the explicit and implicit time stepping methods. Explicit time stepping was used previously to solve the advection equation but we have not previously seen implicit time stepping in this module. Implicit time stepping generates a system of linear equations which can be represented as a sparse matrix equation.

1.1 Explicit time stepping

The diffusion of a one-dimensional scalar field $u(x,t)$ is described by the diffusion equation

$$\frac{\partial u}{\partial t} = K\frac{\partial^2 u}{dx^2} \tag{1}$$

where x is position, t is time and K is the diffusivity. This equation can be solved numerically by using finite difference expressions to approximate the derivatives

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = K\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} \tag{2}$$

where u_i^n is the solution at grid point i at time step n, and u_i^{n+1} is the solution at grid point i at the next time step (time step $n+1$). This expression can be re-arranged to give

$$u_i^{n+1} = u_i^n + \frac{K\Delta t}{(\Delta x)^2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right) \tag{3}$$

This method of time stepping is known as explicit time stepping and equation 3 allows us to update the solution to next time step using only values from the current time step.

1.2 Implicit time stepping

An alternative approach is to calculate the finite differences on the right hand side using the values at the next time step (time step $n+1$)

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = K\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{(\Delta x)^2} \tag{4}$$

We can re-arrange equation 4 to give

$$u_i^{n+1} = u_i^n + \alpha\left(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}\right) \tag{5}$$

where $\alpha = \frac{K\Delta t}{(\Delta x)^2}$. Equation 5 is like equation 3 but with the spatial derivative evaluated at time step $n+1$ instead of time step n. Equation 5 can be re-arranged to give

$$u_i^{n+1} - \alpha\left(u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}\right) = u_i^n \tag{6}$$

$$\Rightarrow -\alpha u_{i+1}^{n+1} + (1+2\alpha)\, u_i^{n+1} - \alpha u_{i-1}^{n+1} = u_i^n \tag{7}$$

By applying equation 7 to each grid point we can set up a system of linear equations which describe the solution at time step $n+1$ in terms of the solution at time step n. This is known as implicit time stepping.

We can write the linear system which needs to be solved as a matrix equation

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ -\alpha & (1+2\alpha) & -\alpha & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & -\alpha & (1+2\alpha) & -\alpha & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & -\alpha & (1+2\alpha) & -\alpha & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & -\alpha & (1+2\alpha) & -\alpha \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_0^{n+1} \\ u_1^{n+1} \\ u_2^{n+1} \\ u_3^{n+1} \\ \vdots \\ u_{N-3}^{n+1} \\ u_{N-2}^{n+1} \\ u_{N-1}^{n+1} \\ u_N^{n+1} \end{pmatrix} = \begin{pmatrix} u_0^n \\ u_1^n \\ u_2^n \\ u_3^n \\ \vdots \\ u_{N-3}^n \\ u_{N-2}^n \\ u_{N-1}^n \\ u_N^n \end{pmatrix}$$

where the values u_0 and u_N are the boundary conditions (which are assumed to be constant values in this case). This is a sparse matrix with non-zero elements on the main diagonal and the first diagonals above and below. This is a type of band matrix known as a tridiagonal matrix[1].

The Lapack linear algebra library can be used to solve systems of linear equations, and Lapack has specific routines which can be used when the matrix is tridiagonal. The function `DGTTRF`[2] factorises a double precision, tridiagonal matrix so that all the non-zero elements are on or above the diagonal. The output from the factorisation can then be used as input to the function `DGTTRS` which solves the linear system (there is more information about solving linear systems with Lapack in the Lapack user guide).

2 Getting set up

- Start by logging in to a Lovelace lab Linux PC, using the Linux computer availability dashboard (`http://students.emps.ex.ac.uk/dashboard/`) to choose a computer where the system load is low (you may need to use the VPN to view the dashboard).
- The example programs for this workshop are provided as a tar file. Take a copy of the tar file for this workshop by running the command:

  ```
  cp /secamfs/userspace/ug/shared/ecm3446/workshop10.tar .
  ```

  where the dot at the end of the command causes the file to be copied into the current working directory.
- Unpack the tar file by running the following command:

  ```
  tar xvf workshop10.tar
  ```

  This will create a directory called `workshop10` which contains the example programs for this workshop.
- Load the `intel` module so that you have access to the Math Kernel Library which provides Lapack:

  ```
  module load intel-oneapi/2022.3.0
  ```

[1] `https://mathworld.wolfram.com/TridiagonalMatrix.html`

[2] The naming system for Linpack is similar to BLAS so the `D` means double precision and `GT` means a general tridiagonal matrix.

3 Explicit time stepping

We will start by running a program which solves the diffusion equation using explicit time stepping. The program calculates a stable time step using the condition

$$\Delta t = \nu \frac{(\Delta x)^2}{K} \tag{8}$$

where $\nu = 0.5$[3] The calculation continues until a specified end time is reached (`end_time=0.01`).

- Use the following commands to build and run the explicit time stepping program

  ```
  cd workshop10/explicit
  icc -std=c99 -o diffusion_1D diffusion_1D.c
  ./diffusion_1D
  ```

- The results can be plotted using the supplied `gnuplot` script

  ```
  gnuplot plot_diffusion
  ```

  This will generate a PNG file which can be copied to your local computer for viewing, for example run the following command on your local computer

  ```
  scp abc123@blue01.ex.ac.uk:/home/links/abc123/workshop10/explicit/diffusion.png .
  ```

  where you should replace `abc123` with your username on the Lovelace lab PCs.

Question 1: How many time steps are required to compute the solution?

- Increase the number of grid points from 1000 to 2000 and re-run the program.

Question 2: How many time steps are required to compute the solution with the new spatial resolution and how is the time step scaling with Δx?

- We will now check the time step constraint by increasing the time step slightly. Set `nu=0.501` and re-run the program.

Question 3: Does the calculation produce valid results with the increased time step?

4 Implicit time stepping

We will now look at the version of the program which used implicit time stepping.

- Use the following commands to build and run the implicit time stepping program

  ```
  cd ../implicit
  icc -mkl -std=c99 -o diffusion_1D_im  diffusion_1D_im.c -lm
  ./diffusion_1D_im
  ```

- The results can be plotted using the supplied `gnuplot` script

  ```
  gnuplot plot_diffusion
  ```

  This will generate a PNG file which can be copied to your local computer for viewing, for example run the following command on your local computer

[3] This condition ensures stability but you may see also $\nu = 0.25$ which is required to ensure accurate representation of the $2\Delta x$ mode.

```
scp abc123@blue01.ex.ac.uk:/home/links/abc123/workshop10/explicit/diffusion.png .
```

where you should replace `abc123` with your username on the Lovelace lab PCs.

Check that the results match those from the implicit solution. There is a second `gnuplot` script called `plot_ex_im` which plots the explicit and implicit solutions on the same graph but you should find that the results are indistinguishable.

The implicit solution is unconditionally stable so we should be able to increase the time step without the solution becoming unstable.

- Try increasing the time step by setting `nu=5.0` and re-run the program.

Question 4: How many time steps are now required and has this change noticeably affected the solution?

- Try increasing the time step even more by setting `nu=500.0`.

Question 5: How many time steps are now required and how has this change affected the solution?