MATH 353: Engineering Mathematics III – Section 012

Spring 2013 (F.–J. Sayas) Lab # 11 May 3

Open Matlab and *move to the Desktop or to a folder where you can find your work* at the end of the session. Type these lines

```
>> diary myworkMay3
>> format long
>> format compact
```

Download `heatForwardFD.m` and `scriptMay3.m` from the website.

To review. We want to approximate the following initial and boundary value problem for the heat equation: $u(x,t)$ with $a \leq x \leq b$ and $0 \leq t \leq T$ satisfies the partial differential equation

$$\frac{\partial u}{\partial t} = D\,\frac{\partial^2 u}{\partial x^2}, \qquad a < x < b, \quad 0 < t \leq T, \qquad (D > 0 \text{ is the diffusivity}),$$

and initial condition at time $t = 0$

$$u(x,0) = u_0(x) \qquad a \leq x \leq b,$$

and two boundary conditions at $x = a$ and $x = b$ for all times

$$u(a,t) = l(t), \qquad u(b,t) = r(t) \qquad 0 < t \leq T.$$

To discretize we choose integers N and M and define

$$h = \frac{b-a}{N+1} \qquad x_i = a + i\,h, \qquad i = 0, 1, \ldots, N, N+1,$$

and

$$k = \frac{T}{M} \qquad t_n = n\,k, \quad n = 0, \ldots, M.$$

The forward finite difference method approximates

$$U_i^n \approx u(x_i, t_n)$$

using the following time-stepping formula for $n \geq 0$:

$$\frac{U_i^{n+1} - U_i^{n+1}}{k} = D\frac{U_{i-1}^n - 2U_t^n + U_{i+1}^n}{h^2} \qquad i = 1, \ldots, N$$

The quantities corresponding to $n = 0$ are taken from the initial condition

$$U_i^0 = u_0(x_i) \qquad i = 0, \ldots, N+1,$$

whereas the equantities corresponding to boundary points $i = 0$ and $i = N+1$ are taken from the boundary conditions

$$U_0^n = l(t_n) \qquad U_{N+1}^n = r(t_n), \qquad n \geq 1.$$

To understand. The time-stepping process can be organized in the following way. We take a column vector $\mathbf{u}^n = (U_1^n, \ldots, U_N^n)^\top$ collecting all the unknowns at time t_n on interior points (the boundary points are excluded). Then the method can be written as

$$\mathbf{u}^{n+1} = \mathbf{u}^n - \mathrm{A}\mathbf{u}^n + \mathbf{b}(t_n) \qquad n = 1, \ldots, M,$$

where

$$\mathrm{A} = \frac{k\,D}{h^2}\begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix} \qquad \mathbf{b}(t) = \frac{k\,D}{h^2}\begin{bmatrix} l(t) \\ 0 \\ \vdots \\ 0 \\ r(t) \end{bmatrix}.$$

In this way, the unknowns U_0^n and U_{N+1}^n are never defined explicitly (but they are used in the boundary conditions at the time-stepping level). In the code, they are also imposed in order to have a clear view of the entire solution.

To test. Go now to the code provided.

1. Have a look at the function `heatForwardFD.m`. Is there anything you don't understand? I'll ask you for two easy modifications, so you'll need to know what's in it.
2. Open the script. It solves the equation with the following parameters: $D = \frac{1}{2}$, $(a, b) = (0, 1)$, $T = 1$, with initial condition

   $$u_0(x) = x^2(1-x)^2.$$

   What are the boundary conditions? Write the mathematical expression for both of them.
3. Run the script and check how the solution evolves with time. Can you see it getting to a steady-state?
4. There's a serious stability requirement:

   $$\frac{k\,D}{h^2} < \frac{1}{2}, \qquad \text{that is} \qquad k < \frac{h^2}{2\,D}.$$

   If $N = 20$, what is the longest time-step that you can use ensuring stability of the method?
5. Check now the other values of M (number of time steps) in the script. Can you see the effect on the numerical approximation?

To modify. The stability requirment means that when we want good spatial resolution (large N, small h) and diffusivity D is large, the time steps k have to be very short, even if the solution does not evolve in any dramatic way. To avoid that, we can change the method to a backward (implicit method), computing for $n \geq 0$:

$$\frac{U_i^{n+1} - U_i^{n+1}}{k} = D\frac{U_{i-1}^{n+1} - 2U_t^{n+1} + U_{i+1}^{n+1}}{h^2} \qquad i = 1, \ldots, N$$

which, after some algebra and the substitution of the boundary conditions, can be written as

$$(\mathrm{I} + \mathrm{A})\mathbf{u}^{n+1} = \mathbf{u}^n + \mathbf{b}(t_{n+1}) \qquad n = 1, \ldots, M.$$

Your goal now is:

1. Create a function `heatBackwardFD.m` that implements this new method. The modifications from the forward method are minimal, so you can just go ahead and copy-paste everything and just change what you need to change.

2. Use the same script with the new method and check ow there are no stability conditions, that is, there are no restrictions on the time step depending on diffusivity and on the grid size in the space variable.