MACSI One Day Graduate Course:
Numerical Solution to Differential Equations using Matlab

Part 6: Partial Differential Equations

Niall Madden and Meghan Stephens
Department of Mathematics
National University of Ireland, Galway
Ollscoil na hÉireann, Gaillimh

3rd of April 2007.

A parabolic problem

Our model problem is

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial t^2} + ru = f$$

subject to the initial condition

$$u(x, 0) = 0$$

and the boundary conditions

$$u(0, t) = u(1, t) = 0.$$

Discretization

The numerical scheme that we will use is central differencing in space, and backward-differencing is time.

Take the spacial mesh $\omega^N = \{x_0, x_1, \ldots, x_N\}$,
and temporal mesh $\{0, \tau, 2\tau, 3\tau, \ldots, M\tau = T\}$

Denote by $U_{i,j}$ the numerical solution at the point $x = x_i$ and time $t = j\tau$.

Then the numerical method is

$$\frac{U_{i,j} - U_{i,j-1}}{\tau} - \delta^2 U_{i,j} + r_{i,j} U_{i,j} = f_{i,j}$$

for $i = 0, 1, \ldots, N+1$, and $j = 1, 2, \ldots, M$.

Discretization

This can be rearranged to get

$$-\tau\delta^2 U_{i,j} + (\tau + r_{i,j})U_{i,j} = \tau f_{i,j} + u_{i,j-1}.$$

That is, at every time-step, we just solve a (stationary) boundary value problem.

Sample code for this is given in `Parabolic.m`

Our only new Matlab function is `meshgrid`:

```
[X,Y] = meshgrid(x,y)
```

returns matrices `X` and `Y` so that the rows of `X` are copies of the vector `x`; columns of the output array `Y` are copies of the vector `y`

Discretization

Exercise

- Rewrite this script as a function file.
- Extend it so that there is a convective term present, and so that the boundary conditions are not necessarily homogeneous.

An Elliptic Problem

Our model problem is:

find $u(x, y)$ = that satisfies

$$Lu := -\varepsilon^2 \Delta u + ru = f \quad \text{on } \Omega := (0,1) \times (0,1),$$

$$u = 0 \quad \text{on } \partial\Omega,$$

where Δu is the Laplacian:

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

The numerical method

We approximate the solution to this problem by applying a standard finite difference method on a tensor-product mesh.

Choose one-dimensional meshes ω_x and ω_y and let $\bar{\Omega}^N = \{(x_i, y_j)\}_{i,j=0}^N$ be their tensor product.

Set $h_i = x_i - x_{i-1}$ and $k_i = y_i - y_{i-1}$ for each i. Given a mesh function $\{v_{i,j}\}_{i,j=0}^N$, define the standard second-order central differencing operators

$$\delta_x^2 v_{i,j} := \frac{1}{\bar{h}_i}\left(\frac{v_{i+1,j} - v_{i,j}}{h_{i+1}} - \frac{v_{i,j} - v_{i-1,j}}{h_i}\right) \quad \text{for } i = 1, \dots, N-1,$$

$$\delta_y^2 v_{i,j} := \frac{1}{\bar{k}_i}\left(\frac{v_{i,j+1} - v_{i,j}}{k_{i+1}} - \frac{v_{i,j} - v_{i,j-1}}{k_i}\right) \quad \text{for } j = 1, \dots, N-1,$$

where $\bar{h}_i = (h_{i+1} + h_i)/2$ and $\bar{k}_i = (k_{i+1} + k_i)/2$.

The numerical method

Set $\Delta^N v_{i,j} := (\delta_x^N + \delta_y^N) v_{i,j}$. Then we define the difference operator as

$$(L^N U)_{i,j} = -\varepsilon^2 \Delta^N U_{i,j} + r(x_i, y_j) U_{i,j}, \quad \text{for } i = 1, \dots N-1, j = 1, \dots N-1.$$

To generate a numerical approximation, solve the system of $N+1$ linear equations

$$(L^N U)_{i,j} = f(x_i, y_j) \quad \text{for } (x_i, y_j) \in \Omega^N,$$

$$U_{i,j} = 0 \quad \text{for } (x_i, y_j) \in \partial\Omega^N.$$

The Matlab Implementation

See `Elliptic.m` and `RunElliptic.m`

Where the code appears complicated, it is because we are dealing with a two-dimensional mesh. (See notes on the black-board.)

New Matlab features include

- `reshape`
- Use of structures.