

Mat1062: Computational Methods for PDE

Problem Set 2

Thursday, January 29, 2008

due: Thursday February 12 in class

1. Revisit the local truncation error analysis for the Forward Euler method. For general λ , we found that the local truncation error is $\mathcal{O}(k^2 + kh^2)$. However, there's a special value of $\lambda \in (0, 1/2]$ for which the local truncation is like that of Crank-Nicolson — Forward Euler with this choice of refinement path will be much more accurate. Find this value of λ via the local truncation error analysis.
2. Consider the θ -method for the diffusion equation $u_t = u_{xx}$ on the line:

$$\frac{u_j^{n+1} - u_j^n}{k} = \theta \frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{h^2} + (1 - \theta) \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{h^2}$$

where $\theta \in [0, 1]$, $n \in \mathbb{N}$, $j \in \mathbb{Z}$.

- (a) Show that if $\lambda(1 - \theta) \leq 1/2$ then the θ -method satisfies the maximum principle. *Hint: assume that the value u_j^{n+1} is greater than or equal to the values at the other five mesh-points in the stencil. What does this mean about the values at those mesh-points?*
- (b) In the convergence proof for Forward Euler, a key issue was considering one step of Forward Euler and from that showing that the errors at the n and $n + 1$ timesteps satisfied:

$$E^{n+1} \leq E^n + \max_{j \in \mathbb{Z}} |\epsilon_j^n|$$

For the θ -method, prove that the same inequality holds if $\lambda(1 - \theta) \leq 1/2$. (As a result, the method will be convergent if you choose h and k that are on such a refinement path.)

3. In the January 29 lecture notes, I gave a table which showed how the sup-norm error decreased as h decreased.
 - (a) Present this table where instead of the sup-norm error you look at the l^1 error and the l^2 error. If f is a vector of length $N + 1$ these norms are

$$\|f\|_{l^1} := h \sum_{j=0}^N |f_j|, \quad \|f\|_{l^2} := \sqrt{h \sum_{j=0}^N |f_j|^2}$$

We're talking about the first table in the notes: the one having based on the correct implementation of the Euler method w/ Neumann BCs.

- (b) In the code, if the initial data comes from the function $u_0(x)$, this function is used to create initial values u_j^0 via direct sampling: $u_j^0 = u_0(x_j)$. Rather than doing direct sampling modify the code to use the average value of u_0 over the interval $[x_j - h/2, x_j + h/2]$:

$$u_j^0 = \frac{1}{h} \int_{x_j - h/2}^{x_j + h/2} u_0(x) dx.$$

Reproduce the table of errors, discuss what you observe and why it makes sense.

4. Write a code for the θ -method (any $0 \leq \theta \leq 1$) for $u_t = Du_{xx}$ on $X_L < x < X_R$, that can accept either Dirichlet or homogeneous Neumann conditions at $x = X_L, X_R$. To write this code, you should write a subroutine which will solve a tri-diagonal system quickly. To spare you this, create the full matrix that has to be inverted and then solve the required problem $AX = b$ via matlab's "slash" command (via $X = A \setminus b$). The slash command is quite robust and will solve the problem even if A is not invertible.

Pick some simple Fourier initial data (such $\sin(\kappa x)$, $\cos(\kappa x)$, where κ has been chosen so that the initial data satisfies the boundary conditions) and compare your computed solution to the exact solution at a fixed time. Run your code in each of the following three scenarios

$$(a) \quad \theta = 0 : \quad k = \lambda h^2$$

$$(b) \quad \theta = 0 : \quad k = \lambda h^2$$

$$(c) \quad \theta = 1/2 : \quad k = \mu h$$

$$(d) \quad \theta = 1 : \quad k = \mu h$$

Here h = space step, k = time step, and λ, μ are fixed as $h, k \rightarrow 0$. In the (b) run, please use the special value of λ that you found in Problem 1.

The result will be a set of eight graphs (which can be superimposed): error E as a function of h and of k in each of the three scenarios. In a log-log plot, you should see straight lines; that is, $E \sim Ch^p$ and $E \sim Ck^q$ with six exponents (slopes) $p_0, p_{1/2}, p_1, q_0, q_{1/2}, q_1$. Explain the observed slopes in terms of truncation error analysis.

Also, keep track of the runtimes for the four methods. For each value of one, find the amount of time it would take to reach an error of 10^{-10} . To fit the exponents, have a look at how I implemented the least-squares approximation in class on Jan 29.