Numerical Treatment of Differential Equations II
Exercise 1

January 15, 2009

Topics: Heat equation, finite volume method, conservation, variable coefficients, boundary conditions.

Purpose: This exercise builds on the basic course in numerical treatment of differential equations. We will compute approximate solutions to a time-dependent PDE on a 2D domain. Of particular interest is the derivation of a basic finite volume method and how to represent the discrete problem in a way that is practical for analysis and implementation.

Instructions:

- Hand in a written report on the date indicated in the lecture and/or homepage. Report should contain answers to all questions stated and proper motivation for each, e.g. derivations. Submit code via e-mail.
- If you work alone, choose either to answer section 4.1 (variable coefficients) or section 4.2 (boundary conditions). If you work in pairs, you must answer both.

1 Heat equation

A classical example of an parabolic PDE is the heat equation on a square. Let $q(x, y, t)$ denote temperature. Then,

$$q_t = \nabla \cdot (\nabla q) + S, \quad (x, y) \in [0\ 1] \times [0\ 1], t \geq 0 \tag{1}$$

$$q(x, y, 0) = 0 \tag{2}$$

$$\hat{n} \cdot \nabla q = 0, \quad (x, y) \text{ on boundary} \tag{3}$$

We have a heat source at the point, (1/2,1/2),

$$S(x, y, t) = \delta(x - 1/2)\delta(y - 1/2)g(t)$$

$$g(t) = \begin{cases} 2 & \text{if } 0 \leq t \leq 1/4 \\ 0 & \text{otherwise.} \end{cases}$$

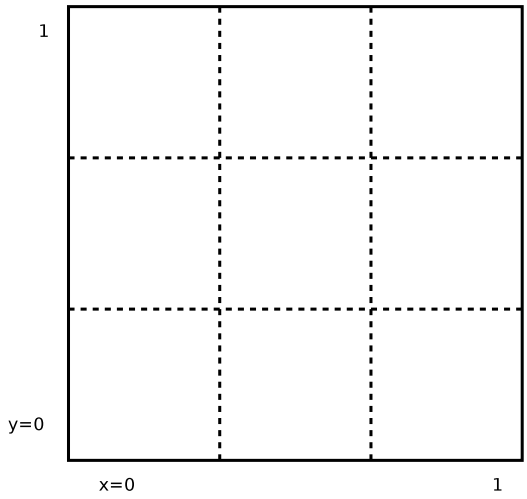

Figure 1: Finite volume grid

The boundary conditions (3) state that there is no heat flux across boundaries. Physically, we have a plate that is insulated from its surroundings and initially at zero temperature. It is heated at a point for 0.25 seconds and then the source is turned off.

1.1 Analytical preamble

1. Determine $\tilde{Q}(t) = \int_0^1 \int_0^1 q(x, y, t) dx dy$ as a function of t.

2 Discretization and implementation

Let Q_{ij} denote the cell average of q over cell (i, j) (see Figure 1) and introduce cell sizes Δx and Δy such that $m\Delta x = n\Delta y = 1$.

1. Derive a conservative finite volume method for the spatial part of (1) by integrating and forming cell averages. Take care that the source term gets included correctly. Show that you obtain an expression of the form

$$\frac{d}{dt} Q_{ij} = \Delta_5 Q_{ij} + S_{ij}, \quad (i, j) \text{ interior indices}$$

   where Δ_5 is the five-point Laplacian stencil familiar from finite difference methods. What stencils do you get at the boundaries?

2. Integrate in time using the first order implicit Euler scheme. Why is this more appropriate than explicit? *Hint:* time-step restriction. State the fully discrete problem.

3. Let $\mathcal{Q}$ be an $m \times n$ array that contains the Q_{ij} values. The Laplacian can be expressed as[1]

$$\Delta_5 \mathcal{Q} = \mathcal{Q} T_x + T_y \mathcal{Q}$$

where T represents the Laplacian difference operator in 1D for each dimension respectively. Why is this convenient? *Hint:* Boundary conditions. Use this idea to state the fully discrete problem in matrix form.

Prove that the finite volume scheme is *exactly* conservative.

4. Implement the finite volume method, e.g. in MATLAB.

A linear system has to be solved in each time-step. Due to the boundary conditions, this matrix has a block diagonal structure (it is **not** simply diagonal). Constructing it can be fairly hard in a Matlab program, and possibly *very* computationally inefficient (see example in "Notes on Efficient Matlab Programming").

A good option is to use Kronecker products [2] to construct this difference matrix. The corresponding function is called `kron` in Matlab. It is highly recommended to use the function `reshape` to rearrange a $m \times n$ array into a $mn \times 1$ column vector and vice versa.

When you think the program works, compare your solutions to the reference plots posted on the web page.

You are strongly encouraged to write an efficient program – one that can handle fine resolutions in reasonable time. Take care to make the code clean and readable. On an average workstation the solver may be able to handle $m = n = 800$ or more without much trouble (aside from plotting such a large data set). To get this efficiency, try to move as much work as possible out of the time loop. Use the profiler in Matlab! *Hint:* Is the matrix in the linear system constant in time? Consider LU-factorizing (using the function `lu` in Matlab).

3 Numerical results

In your report, please include the following computational results:

Solution plots for some time levels before and after $t = 1/4$.

Convergence Choose a point (x_0, y_0) and compile a table which shows that the error behaves like $\mathcal{O}(\Delta t^p) + \mathcal{O}(h^r)$, where $h = \Delta x = \Delta y$. Determine p and r. What would one anticipate from theory? Why is it a bad idea to look at the error in $\infty-$ or L_2-norm?

[1] cf. supplementary material from Demmel

[2] cf. http://en.wikipedia.org/wiki/Kronecker_product

Numerical conservation Demonstrate that the method is numerically conservative by looking at $\int q dx dy = \Delta x \Delta y \sum_{ij} Q_{ij}$ for all t. Compare it to the expression computed in Section 1.1. Conservation in "eye norm" is not enough!

4 Refinements

Now we move on to slightly more advanced problems. The framework developed thus far in this lab should be very helpful when you tackle these problems. Do **not** proceed with these tasks until the program above works as expected!

4.1 Variable coefficients

Consider

$$q_t = a(y)q_{xx} + b(x)q_{yy} + S \tag{4}$$

instead of the PDE (1). Choose $a(y)$ and $b(x)$ as smooth and positive functions.

1. Formulate the fully discrete problem for the variable coefficient case, preferably in the Kronecker notation. *Hint:* Multiplication from the left with a diagonal matrix scale each row of a matrix. How do you scale the columns?

2. Implement the variable coefficient problem. With the Kronecker product construction, this should be fairly simple.

   Present convergence and conservation results as in section 3

4.2 Boundary conditions

Change the boundary conditions (3) to

$$\begin{aligned} q(0, y, t) &= \frac{1}{\pi}\sin(\pi y) \\ q(1, y, t) &= \frac{1}{3\pi}\sin(3\pi y) + 1 \\ q_y(x, 0, t) &= -1 \\ q_y(x, 1, t) &= -1 \end{aligned}$$

You may choose a different set of boundary conditions if you want to, as long as you include at leas one non-homogeneous Neuman and Dirichlet condition.

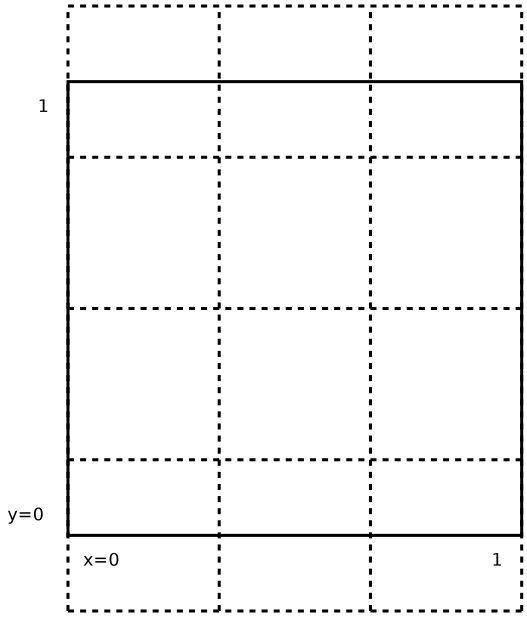

Figure 2: Staggered finite volume grid

1. Implement this new boundary condition.

   The approach is as follows: Construct 1D difference matrices for each dimension and make sure they express the right boundary conditions. Then assemble the 2D difference matrix from the 1D matrices using Kronecker products. *Hint:* In the matrix form of of the fully discrete problem only one of the T operators needs to change (in the case of the suggested new BC).

   There are different options for how to enforce Dirichlet BCs in a finite volume method. For the suggested boundary conditions, you may choose the grid given in Figure 2.

   Present convergence results as in section 3 and discuss conservation in the context of non-homogenous boundary conditions.